

# nehta

---

## **Notification**

### **Endpoint Specification**

Version 1.0 draft — 1 September 2008

Draft for comment

---

**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 © 2008, 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.

# Document Information

## Change History

Version	Date	Comments
1.0 draft	2008-09-01	Draft for comment

# Table of Contents

<b>Document Information</b> .....	<b>iii</b>
Change History .....	iii
<b>Table of Contents</b> .....	<b>iv</b>
<b>Preface</b> .....	<b>1</b>
Document Purpose .....	1
Scope .....	1
Intended Audience.....	1
Document status .....	1
References and Related Documents .....	1
Definitions, Acronyms and Abbreviations.....	2
Conformance .....	2
Overview .....	2
<b>1 WSDL: Notification Consumer</b> .....	<b>3</b>
1.1 Introduction.....	3
1.1.1 Purpose .....	3
1.1.2 Identity .....	3
1.2 Overview.....	3
1.3 Operations.....	3
1.3.1 deliverNotification .....	4
1.4 Schema components .....	4
1.4.1 element: deliverNotification .....	4
1.4.2 element: deliverNotificationResponse .....	4
1.4.3 element: deliverNotificationError .....	5
1.4.4 simpleType: DeliverNotificationErrorCode .....	5
1.4.5 simpleType: DeliverNotificationStatus .....	5
1.4.6 complexType: DeliverNotificationError.....	5
1.5 Implementation.....	5
1.5.1 Web services .....	5
1.5.2 Authorisation.....	5
<b>2 WSDL: Notification Supplier</b> .....	<b>6</b>
2.1 Introduction.....	6
2.1.1 Purpose .....	6
2.1.2 Identity .....	6
2.1.3 Overview .....	6
2.2 Operations.....	7
2.2.1 retrieveNotifications .....	7
2.2.2 removeNotifications.....	8
2.3 Schema components .....	9
2.3.1 element: removeNotifications .....	9
2.3.2 element: removeNotificationsResponse .....	9
2.3.3 element: retrieveNotifications .....	9
2.3.4 element: retrieveNotificationsResponse .....	9
2.3.5 element: removeNotificationsError .....	9
2.3.6 element: retrieveNotificationsError .....	9
2.3.7 simpleType: RemoveNotificationStatus.....	9
2.3.8 simpleType: RemoveNotificationsErrorCode .....	9
2.3.9 simpleType: RetrieveNotificationsErrorCode .....	10
2.3.10 complexType: NotificationsList .....	10
2.3.11 complexType: RemoveNotificationsResult .....	10
2.3.12 complexType: RemoveNotificationsError .....	10
2.3.13 complexType: RetrieveNotificationsError .....	10
2.4 Implementation.....	10

---

2.4.1	Web services .....	10
2.4.2	Authorisation .....	10
<b>3</b>	<b>XSD: Notification .....</b>	<b>11</b>
3.1	Introduction .....	11
3.1.1	Purpose .....	11
3.1.2	Identity .....	11
3.1.3	Overview .....	11
3.2	Schema components .....	11
3.2.1	complexType: Notification .....	11
	<b>Definitions .....</b>	<b>13</b>
	Shortened Terms .....	13
	Glossary .....	13
	Namespaces .....	13
	<b>References .....</b>	<b>14</b>
	Normative references .....	14
	<b>Appendix A: Quick reference .....</b>	<b>15</b>
	A.1 Notification Consumer .....	15
	A.2 Notification Supplier .....	15

This page is intentionally left blank.

# Preface

## Document Purpose

The purpose of this document is to partially define the service interfaces for the *Notification* services.

The complete service interface specification is made up of:

- this document; and
- the files in the *Notification: Endpoint Specification: WSDL and XML Schema files v1.0* [NESWX2008]

## Scope

This service interface specification only includes services that are invoked by external parties. Services that are invoked by internal parties are not covered.

For example, management interfaces are not defined. In the process of delivering notification, one organisation does not invoke another organisation's management services. If required, products can define their own management interfaces.

This service interface specification does not define how these service interfaces are to be implemented and which business processes they participate in.

## Intended Audience

This is a technical document.

This document should be read and understood by:

- Solution Architect:
  - To understand the service interfaces specifications to incorporate them into their designs.
- Developer:
  - To implement the service and/or clients which use the service interface specifications.
- Tester:
  - To evaluate whether an implementation conforms to the service interface specifications.

The reader is expected to understand XML, XML Schema, Web services, and WSDL.

## Document status

This document is a draft and has been released for comment and feedback purposes.

## References and Related Documents

For a list of all referenced documents, see the [References](#) at the end of the document, on page 14.

## Definitions, Acronyms and Abbreviations

For a lists of abbreviations, acronyms and abbreviations, see the [Definitions section](#) at the end of the document, on page 13.

## Conformance

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

## Overview

The notification defines two service interfaces:

- Notification consumer (chapter 1); and
- Notification supplier (chapter 2)

It also defines one XML Schema:

- Notification (chapter3)

As previously stated, this document forms one part of the service interface specifications. It must be read in conjunction with the other part: the WSDL and XML Schema files from [NESWX2008].

# 1 WSDL: Notification Consumer

## 1.1 Introduction

### 1.1.1 Purpose

This service interface is implemented by parties that receive notifications from other parties.

For example, this service interface could be implemented by the receiver of the notification. It could also be implemented by intermediaries to receive notifications and then deliver them to the receiver.

### 1.1.2 Identity

This service interface is identified by the service namespace:

```
urn:xml-gov-au:nehta:service:NotificationConsumer:1.0-draft-20080901
```

XML Schema components defined and declared in this interface belong to the following namespace (which is also the namespace used for unprefix names in this chapter):

```
urn:xml-gov-au:nehta:service:NotificationConsumer:1.0-draft-20080901
```

## 1.2 Overview

This service interface is illustrated in Figure 1.

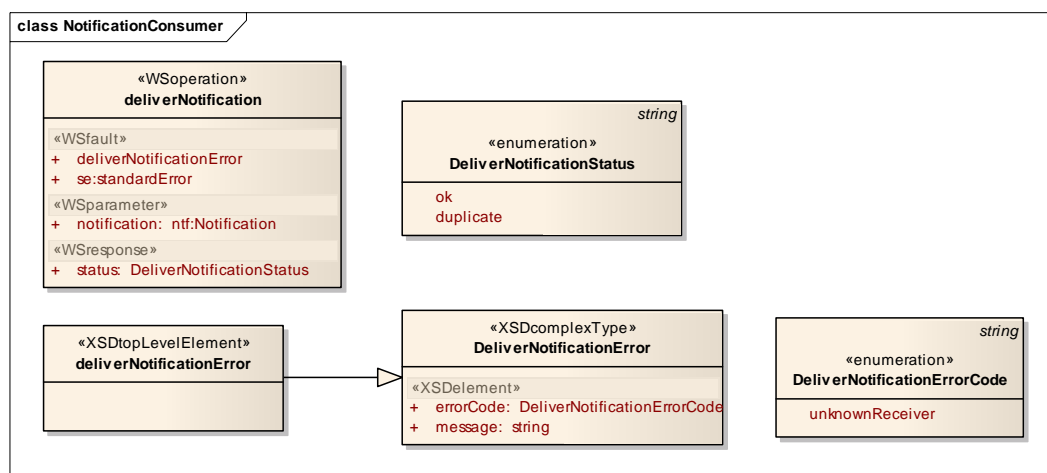


Figure 1: Components of the Notification Consumer service interface

## 1.3 Operations

This service interface defines one operation:

- `deliverNotification`

A client would invoke `deliverNotification` with each notification being sent. Usually, the service will be offered by the receiver of the notification, unless there is some prior agreement with the service provider to ensure that the notifications will get delivered to the receiver.

### 1.3.1 deliverNotification

#### 1.3.1.1 Purpose

This operation is used to deliver a notification to the service provider. It is invoked by the client that has a notification to deliver.

It is possible to invoke this operation multiple times with the same notification identifier. Subsequent invocations will be rejected as duplicates. This is useful for retrying the operation if the client does not know whether it succeeded or not.

#### 1.3.1.2 Request

See the WSDL and XML Schema files for full details about the request. Information in this section assumes knowledge of the WSDL and XML Schema files.

The `notificationId` MUST be a globally unique identifier for each new notification it sends.

Clients which are not the sender MUST use the same `notificationId` that was used by the sender for this notification.

The `notificationId` SHOULD be a UUID formatted as a URI as defined in [RFC4122].

The `xsd:any` element contains the notification data.

#### 1.3.1.3 Response

See the WSDL and XML Schema files for full details about the response. Information in this section assumes knowledge of the WSDL and XML Schema files.

The response indicates whether the operation succeeds or not using the enumerated values in the `deliverNotificationStatus` simpleType (see section 1.4.5).

The `deliverNotification` operation must detect subsequent invocations that have the same `notificationId` value and not deliver duplicate notifications.

This permits the client to retry the operation if it might have failed. This situation may occur when the operation has been processed completely by the service, but the client did not receive the response. Therefore, it can retry the operation with the same `notificationId`. If the earlier invocation was successful, it will not result in a duplicate notification. If the earlier invocation did not succeed, then the retry can succeed.

##### 1.3.1.3.1 Faults

The `deliverNotification` operation MAY respond with a fault containing a `se:standardError` element as defined by [WSP2008].

The `deliverNotification` operation MAY respond with a fault containing a `deliverNotificationError` element.

## 1.4 Schema components

### 1.4.1 element: deliverNotification

This element is used in requests to the `deliverNotification` operation.

### 1.4.2 element: deliverNotificationResponse

This element is used in responses from the `deliverNotification` operation.

### 1.4.3 element: `deliverNotificationError`

This element is used in faults from the `deliverAck` operation.

### 1.4.4 simpleType: `DeliverNotificationErrorCode`

This simpleType is used in faults from the `deliverNotification` operation.

The enumerated value is:

`unknownReceiver`

The receiver is not supported by the service provider.

### 1.4.5 simpleType: `DeliverNotificationStatus`

This simpleType is used in responses from the `deliverNotification` operation.

The enumerated values are:

`ok`

notification successfully delivered to service provider.

`duplicate`

the `notificationId` is the same as another notification that has been processed by the service. Typically (though not always) it indicates that `deliverNotification` on the service was previously invoked with the same `notificationId`.

### 1.4.6 complexType: `DeliverNotificationError`

This complexType is used in the faults from the `deliverNotification` operation.

## 1.5 Implementation

### 1.5.1 Web services

An implementation of this service interface **MUST** conform to the *End-to-end security profile*, as defined in the *Web Services Profile v3.0* [WSP2008].

### 1.5.2 Authorisation

An implementation of this service interface **MUST** provide appropriate access control mechanisms.

What is appropriate will depend on the particular implementation, the party operating the service, and the parties that use it.

Implementations **MUST** allow all parties that it expects to receive notifications from to invoke this interface.

Implementations **SHOULD** allow parties that have been authenticated as the notification's sender to invoke this interface.

Implementations **MAY** allow other parties to invoke this interface by prior agreement and authorisation.

## 2 WSDL: Notification Supplier

### 2.1 Introduction

#### 2.1.1 Purpose

This service interface is implemented by parties that make notifications available for other parties to retrieve.

For example, this service interface could be implemented the sender of notifications. It could also be implemented by intermediaries (who have had notifications delivered to them) for the receiver to retrieve from.

#### 2.1.2 Identity

This service interface is identified by the service namespace:

```
urn:xml-gov-au:nehta:service:NotificationSupplier:1.0-draft-20080901
```

XML Schema components identified and declared in this interface belong to the following namespace (which is also the namespace used for unprefix names in this chapter):

```
urn:xml-gov-au:nehta:service:NotificationSupplier:1.0-draft-20080901
```

#### 2.1.3 Overview

This service interface is illustrated in Figure 2.

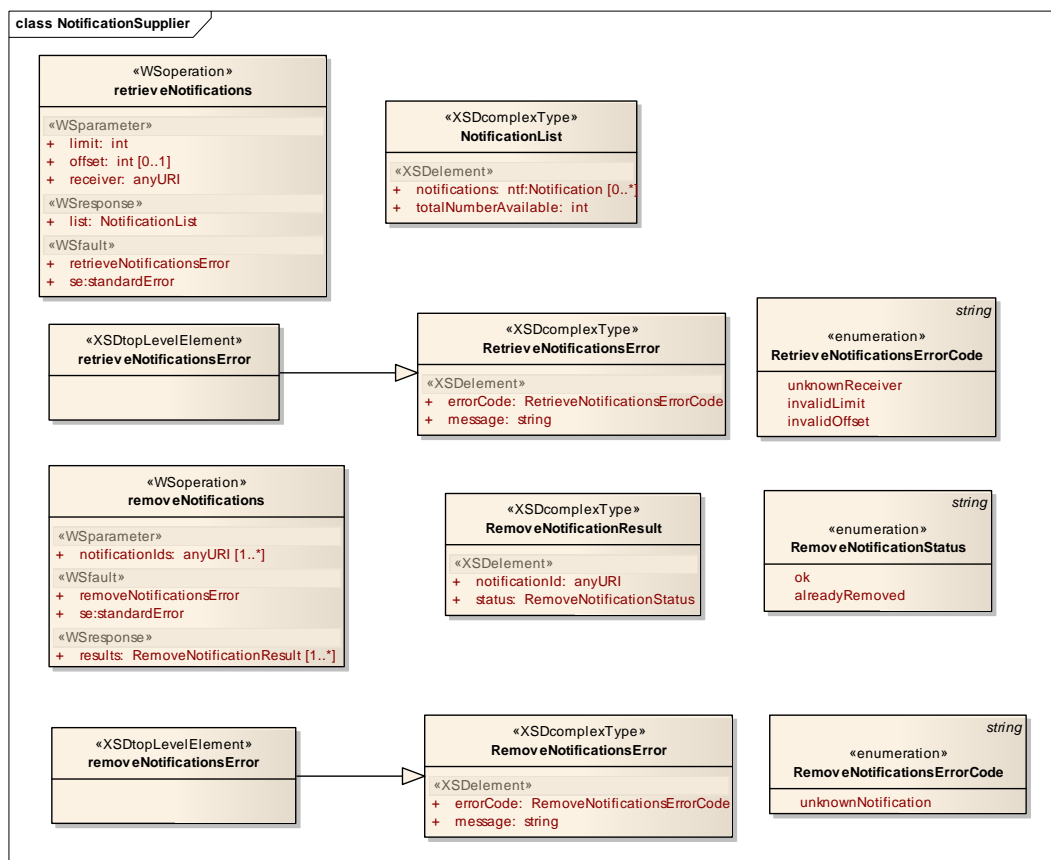


Figure 2: Components of the Notification Supplier service interface

## 2.2 Operations

This service interface defines two operations:

- `retrieveNotifications`; and
- `removeNotifications`.

A client would invoke `retrieveNotifications` to obtain a list of notifications. It can then invoke `removeNotifications` to indicate that those notifications have been retrieved. Later, additional invocations to `retrieveNotifications` can be made to obtain any new notifications.

This service interface is designed to support polling, where the operation is periodically invoked to obtain new notifications.

### 2.2.1 `retrieveNotifications`

#### 2.2.1.1 Purpose

This operation is used to obtain notifications. It is invoked by clients that poll the service to retrieve new notifications.

It is possible to invoke this operation multiple times with the same request. The response will be the same, unless some notifications have been removed (by invoking `removeNotifications`) or new ones added (by a mechanism external to this interface).

#### 2.2.1.2 Request

See the WSDL and XML Schema files for full details about the request. Information in this section assumes knowledge of the WSDL and XML Schema files.

The request identifies a `receiver`, `limit` and `offset`.

The `limit` MUST be a positive integer or zero.

The `offset` MUST be a positive integer or zero.

#### 2.2.1.3 Response

See the WSDL and XML Schema files for full details about the response. Information in this section assumes knowledge of the WSDL and XML Schema files.

The response contains the total number of available notifications and includes a list of some (or all) of them.

The `totalNumberAvailable` element MUST indicate the total number of available notifications that have been sent to the receiver identified in the request.

The number of notifications in the response MUST be less than or equal to the `limit` indicated in the request.

The `limit` is a mechanism for the client to indicate the maximum number of notifications they can process, since a large list will take longer to retrieve and more storage resources to process.

A `limit` of zero is permitted. This would be used if the client only wants to find out how many notifications are available and not retrieve any of them.

The available notifications are treated as an ordered list, with the `offset` indicating where in that list to start returning notifications from. An `offset` of zero means to start returning notifications from the beginning of the list. A non-zero value means to skip that many notifications before starting to return them.

The number of notifications in the response MAY be less than the maximum number possible. In this situation, the client would need to invoke the operation again (with different offset values) to obtain the remaining notifications. The service can deliberately respond with fewer notifications (maybe to reduce bandwidth and processing overheads). This is like a limit for the service provider.

If the `offset` is greater than or equal to `totalNumberAvailable` then there are no notifications returned: this is not an error condition.

If there are notifications it could return (and the `limit` was greater than zero) it MUST return at least one notification.

#### 2.2.1.3.1 *Faults*

The `retrieveNotifications` operation MAY respond with a fault containing a `se:standardError` element as defined by [WSP2008].

The `retrieveNotifications` operation MAY respond with a fault containing a `retrieveNotificationsError` element.

## 2.2.2 **removeNotifications**

### 2.2.2.1 Purpose

This operation is used to indicate that some notifications are no longer available for retrieval.

It is invoked by clients that have successfully retrieved the notification (by using `retrieveNotifications`).

It is possible to invoke this operation multiple times with the same `notificationId`. Subsequent invocations will indicate that the notification has already been removed and take no further action. This is useful for retrying the operation if the client does not know whether it succeeded or not.

### 2.2.2.2 Request

See the WSDL and XML Schema files for full details about the request. Information in this section assumes knowledge of the WSDL and XML Schema files.

The request contains a sequence of one or more `notificationId` elements which identifies the notifications to remove.

Each `notificationId` element MUST identify a notification to remove.

### 2.2.2.3 Response

See the WSDL and XML Schema files for full details about the response. Information in this section assumes knowledge of the WSDL and XML Schema files.

The response indicates whether the operation succeeded or not using the enumerated values in the `RemoveNotificationStatus` simpleType (see section 2.3.7).

It is expected that the service provider will keep track of removed notifications for a period of time, so that it can respond to duplicate removes with an `alreadyRemoved` status instead of an `unknownNotification` error. However, this period of time is left to the implementation to choose. Ideally, implementations should keep track of removed notifications forever. In practice, implementations should keep track of them for the period where it could reasonably expect retries to occur. At one extreme, an implementation could forget about a notification as soon as it is removed: in this situation duplicate removals will always result in an `unknownNotification` error—this is not as useful for retries, but it is permitted by this specification.

### 2.2.2.3.1 *Faults*

The `removeNotifications` operation MAY respond with a fault containing a `se:standardError` element as defined by [WSP2008].

The `removeNotifications` operation MAY respond with a fault containing a `removeNotificationsError` element.

If a fault occurs, the operation MUST NOT remove any notifications.

The `notificationId` in the `removeNotificationsError` element indicates which of the notifications were unknown.

## 2.3 Schema components

### 2.3.1 **element: `removeNotifications`**

This element is used in requests to the `removeNotifications` operation.

### 2.3.2 **element: `removeNotificationsResponse`**

This element is used in responses from the `removeNotifications` operation.

### 2.3.3 **element: `retrieveNotifications`**

This element is used in requests to the `retrieveNotifications` operation.

### 2.3.4 **element: `retrieveNotificationsResponse`**

This element is used in responses from the `retrieveNotifications` operation.

### 2.3.5 **element: `removeNotificationsError`**

This element is used in faults from the `removeNotifications` operation.

### 2.3.6 **element: `retrieveNotificationsError`**

This element is used in faults from the `retrieveNotifications` operation.

### 2.3.7 **simpleType: `RemoveNotificationStatus`**

This simpleType is used in responses from the `removeNotifications` operation.

The enumerated values are:

`ok`

the notification was successfully removed.

`alreadyRemoved`

the notification was previously removed.

### 2.3.8 **simpleType: `RemoveNotificationsErrorCode`**

This simpleType is used in faults from the `removeNotifications` operation.

The enumerated value is:

`unknownNotification`

the notification identifier is not known.

### 2.3.9 simpleType: RetrieveNotificationsErrorCode

This simpleType is used in faults from the `retrieveNotifications` operation.

The enumerated values are:

`unknownReceiver`

The receiver cannot be accepted by the service provider.

`invalidLimit`

The limit value is not a positive integer or zero.

`invalidOffset`

The offset is not a positive integer or zero.

### 2.3.10 complexType: NotificationsList

This complexType is used in responses from the `listNotifications` operation.

### 2.3.11 complexType: RemoveNotificationsResult

This complexType is used in responses from the `removeNotifications` operation.

### 2.3.12 complexType: RemoveNotificationsError

This complexType is used in faults from the `removeNotifications` operation.

### 2.3.13 complexType: RetrieveNotificationsError

This complexType is used in faults from the `retrieveNotifications` operation.

## 2.4 Implementation

### 2.4.1 Web services

An implementation of this service interface MUST conform to the *End-to-end security profile*, as defined in the *Web Services Profile v3.0* [WSP2008].

### 2.4.2 Authorisation

An implementation of this service interface MUST provide appropriate access control mechanisms.

What is appropriate will depend on the particular implementation, the party operating the service, and the parties that use it.

Implementations SHOULD allow parties that have been authenticated as the receiver to invoke this interface. That is, a receiver should always retrieve and remove notifications that are to be delivered to it.

Implementations MAY allow other parties to invoke this interface by prior agreement and authorisation.

## 3 XSD: Notification

### 3.1 Introduction

#### 3.1.1 Purpose

This schema is used to represent notifications.

#### 3.1.2 Identity

This schema has the namespace of:

```
urn:xml-gov-au:nehta:types:Notification:1.0-draft-20080901
```

#### 3.1.3 Overview

This schema is illustrated in Figure 3.

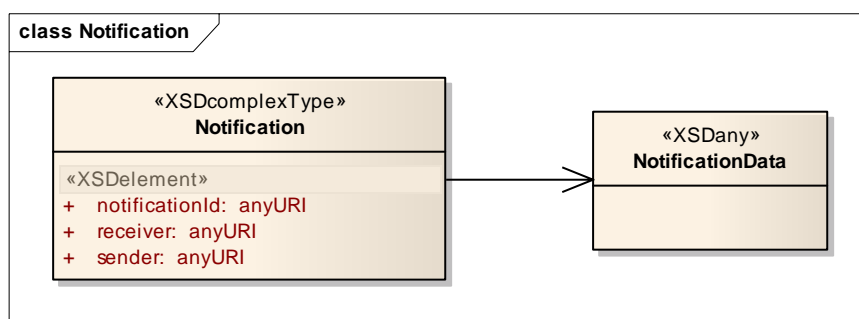


Figure 3: Components of the Notification schema

### 3.2 Schema components

#### 3.2.1 complexType: Notification

The `Notification` complexType is used to represent a notification.

It contains the following elements:

- `notificationId`;
- `receiver`;
- `sender`; and
- the notification data.

##### 3.2.1.1 notificationId

The `notificationId` MUST contain a unique identifier for the notification.

The `notificationId` SHOULD be a UUID formatted as a URN as defined in [RFC4122].

##### 3.2.1.2 receiver

The `receiver` MUST identify the final party that the notification is to be delivered to.

The `receiver` MUST follow the NEHTA scheme for identifiers, as defined by [IDURI2008]. This scheme uses a URI as a unique identifier.

For example, it could contain a URI representing the HPI-O number for the General Practice that the notification is being sent to.

Its value always identifies the final party. Even if this operation is being invoked on an intermediary, its value is still the final party. The identity of the intermediary is not placed in this parameter.

#### 3.2.1.3 sender

The sender **MUST** identify the initial party that sent the notification.

The sender **MUST** follow the NEHTA scheme for identifiers, as defined by [IDURI2008]. This scheme uses a URI as a unique identifier.

For example, it could contain a URI representing the HPI-O number for the organisation that is sending the notification.

#### 3.2.1.4 The notification data

The notification is defined in the domain specific specifications that use notifications.

As far as the notification service interface is concerned, the notification is any XML element.

# Definitions

This section explains the specialised terminology used in this document.

## Shortened Terms

This table lists abbreviations and acronyms in alphabetical order.

Term	Description
HPI-O	Healthcare Provider Identifier for Organisations
URI	Uniform Resource Identifier
URN	Uniform Resource Name
UUID	Universally Unique Identifier
WSDL	Web Services Description Language
XML	Extensible Markup Languages

## Glossary

This table lists specialised terminology in alphabetical order.

Term	Description
Notification	Data that indicates an event has taken place.

## Namespaces

The following XML namespace prefixes are used in this document:

```
se      urn:xml-gov-au:nehta:types:StandardError:1.0-draft-20080901
xsd     http://www.w3.org/2001/05/XMLSchema
```

# References

## Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- [RFC2119] IETF, *RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner, March 1997, <http://ietf.org/rfc/rfc2119.txt>
- [RFC4122] IETF, *RFC 4122: A Universally Unique Identifier (UUID) URN Namespace*, P. Leach, M. Mealling, R. Salz, July 2005, <http://ietf.org/rfc/rfc4122.txt>
- [IDURI2008] NEHTA, *Representation of Identifiers as URIs v1.0*.
- [WSP2008] NEHTA, *Web Services Profile v3.0*.

# Appendix A: Quick reference

## A.1 Notification Consumer

urn:xml-gov-au:nehta:service:NotificationConsumer:1.0-draft-20080901

**deliverNotification** (notification)

→ deliverNotificationStatus

## A.2 Notification Supplier

urn:xml-gov-au:nehta:service:NotificationConsumer:1.0-draft-20080901

**retrieveNotifications** (receiver, limit, offset)

→ totalNumberAvailable, notification\*

**removeNotifications** (notificationId+)

→ removeNotificationsStatus+