



**Pathology Result Reporting Package
(v1.0 Draft)**

Endpoint Specification v2.1

4 September 2008

Draft for comment - Commercial-in-confidence

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 © 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	Author	Comments
2.1	4/09/2008	NEHTA	Correction
2.0	1/09/2008	NEHTA	Package v1.0 Draft update
1.0 draft	27/8/2008	NEHTA	Internal Draft

Table of Contents

Document Information	iii
Change History	iii
Table of Contents	iv
Preface	1
Document Purpose	1
Scope	1
Intended Audience.....	1
Document status	1
Document Map.....	2
References and Related Documents	2
Definitions, Acronyms and Abbreviations.....	3
Conformance	3
Overview	3
1 WSDL: Pathology Sealed Report Consumer	4
1.1 Introduction.....	4
1.1.1 Purpose	4
1.1.2 Identity	4
1.1.3 Overview	4
1.2 Operations.....	4
1.2.1 deliverSealedReport	5
1.3 Schema components	6
1.3.1 element: deliverSealedReport	6
1.3.2 element: deliverSealedReportResponse	6
1.3.3 element: deliverSealedReportError	6
1.3.4 simpleType: DeliverSealedReportErrorCode	6
1.3.5 simpleType: DeliverSealedReportStatus	6
1.3.6 complexType: DeliverSealedReportError.....	6
1.4 Implementation.....	6
1.4.1 Web services.....	6
1.4.2 Authorisation.....	6
2 WSDL: Pathology Sealed Report Supplier	8
2.1 Introduction.....	8
2.1.1 Purpose	8
2.1.2 Identity	8
2.1.3 Overview	8
2.2 Operations.....	9
2.2.1 listSealedReports	9
2.2.2 retrieveSealedReport	10
2.2.3 removeSealedReport	11
2.3 Schema components	12
2.3.1 element: listSealedReports	12
2.3.2 element: listSealedReportsResponse.....	12
2.3.3 element: removeSealedReport	12
2.3.4 element: removeSealedReportResponse.....	12
2.3.5 element: retrieveSealedReport.....	12
2.3.6 element: retrieveSealedReportResponse.....	12
2.3.7 element: listSealedReportsError	12
2.3.8 element: removeSealedReportError	12
2.3.9 element: retrieveSealedReportError.....	13
2.3.10 simpleType: ListSealedReportsErrorCode	13
2.3.11 simpleType: RemoveSealedReportErrorCode	13
2.3.12 simpleType: RemoveSealedReportStatus	13

2.3.13	simpleType: RetrieveSealedReportErrorCode	13
2.3.14	complexType: ListSealedReportsError	13
2.3.15	complexType: RemoveSealedReportError	13
2.3.16	complexType: RetrieveSealedReportError	14
2.3.17	complexType: SealedReportReferenceList	14
2.4	Implementation	14
2.4.1	Web services	14
2.4.2	Authorisation	14
3	WSDL: Pathology Sealed Report Acknowledgement Consumer	15
3.1	Introduction	15
3.1.1	Purpose	15
3.1.2	Identity	15
3.1.3	Overview	15
3.2	Operations	15
3.2.1	deliverAck	16
3.3	Schema components	17
3.3.1	element: deliverAck	17
3.3.2	element: deliverAckResponse	17
3.3.3	element: deliverAckError	17
3.3.4	simpleType: DeliverAckErrorCode	17
3.3.5	simpleType: DeliverAckStatus	17
3.3.6	complexType: DeliverAckError	17
3.4	Implementation	17
3.4.1	Web services	17
3.4.2	Authorisation	17
4	WSDL: Pathology Sealed Report Acknowledgement Supplier	19
4.1	Introduction	19
4.1.1	Purpose	19
4.1.2	Identity	19
4.1.3	Overview	19
4.2	Operations	20
4.2.1	retrieveAcks	20
4.2.2	removeAcks	21
4.3	Schema components	22
4.3.1	element: removeAcks	22
4.3.2	element: removeAcksResponse	22
4.3.3	element: retrieveAcks	22
4.3.4	element: retrieveAcksResponse	22
4.3.5	element: removeAcksError	22
4.3.6	element: retrieveAcksError	22
4.3.7	simpleType: RemoveAckStatus	22
4.3.8	simpleType: RemoveAcksErrorCode	23
4.3.9	simpleType: RetrieveAcksErrorCode	23
4.3.10	complexType: AckList	23
4.3.11	complexType: RemoveAckResult	23
4.3.12	complexType: RemoveAcksError	23
4.3.13	complexType: RetrieveAcksError	23
4.4	Implementation	23
4.4.1	Web services	23
4.4.2	Authorisation	23
5	XSD: Pathology Report	25
5.1	Introduction	25
5.1.1	Purpose	25
5.1.2	Identity	25
5.1.3	Overview	25
5.2	Schema components	25
5.2.1	element: pathologyReport	25
5.2.2	complexType: PathologyReport	25

6	XSD: Pathology Sealed Report Instance	27
6.1	Introduction.....	27
6.1.1	Purpose.....	27
6.1.2	Identity	27
6.1.3	Overview	27
6.2	Schema components	27
6.2.1	complexType: SealedReportInstance	27
6.2.2	complexType: SealedReportInstanceMetadata.....	28
7	XSD: Pathology Sealed Report Acknowledgement	30
7.1	Introduction.....	30
7.1.1	Purpose.....	30
7.1.2	Identity	30
7.1.3	Overview	30
7.2	Schema components	30
7.2.1	complexType: Ack.....	30
7.2.2	complexType: AckInstance.....	31
7.2.3	complexType: AckMetadata	31
8	XSD: Pathology Sealed Report Notification	33
8.1	Introduction.....	33
8.1.1	Purpose.....	33
8.1.2	Identity	33
8.1.3	Overview	33
8.2	Schema components	33
8.2.1	element: pathologySealedReportNotification	33
8.2.2	complexType: PathologySealedReportNotification	33
	Definitions	35
	Shortened Terms.....	35
	Glossary	35
	Namespaces	35
	References	36
	Normative references.....	36
	Informative references	36
	Appendix A: HL7 V2.4 Comparisons	37
A.1	Concepts	37
A.1.1	Messages and Operations.....	37
A.1.2	Acknowledgements	37
A.1.3	Batching	37
A.2	Messages	38
A.2.1	Message: ACK_ALL.....	38
A.2.2	Message: OUL_R21	38
A.2.3	Segment: ERR	38
	Appendix B: Quick reference	39
B.1	Pathology Sealed Report Consumer	39
B.2	Pathology Sealed Report Supplier.....	39
B.3	Pathology Sealed Report Acknowledgement Consumer	39
B.4	Pathology Sealed Report Acknowledgement Supplier.....	39

Preface

Document Purpose

The purpose of this document is to partially define the service interfaces for the *Pathology Result Reporting Package*.

The complete service interface specification is made up of:

- this document; and
- the files in the *Pathology Result Reporting Package: Endpoint Specification: WSDL and XML Schema files v1.0* [PRRSISWX2008].

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 pathology result report delivery, 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. For that information, please see the *Pathology Result Reporting Package: Technical Architecture* [PRRPATA2008]

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.

Document Map

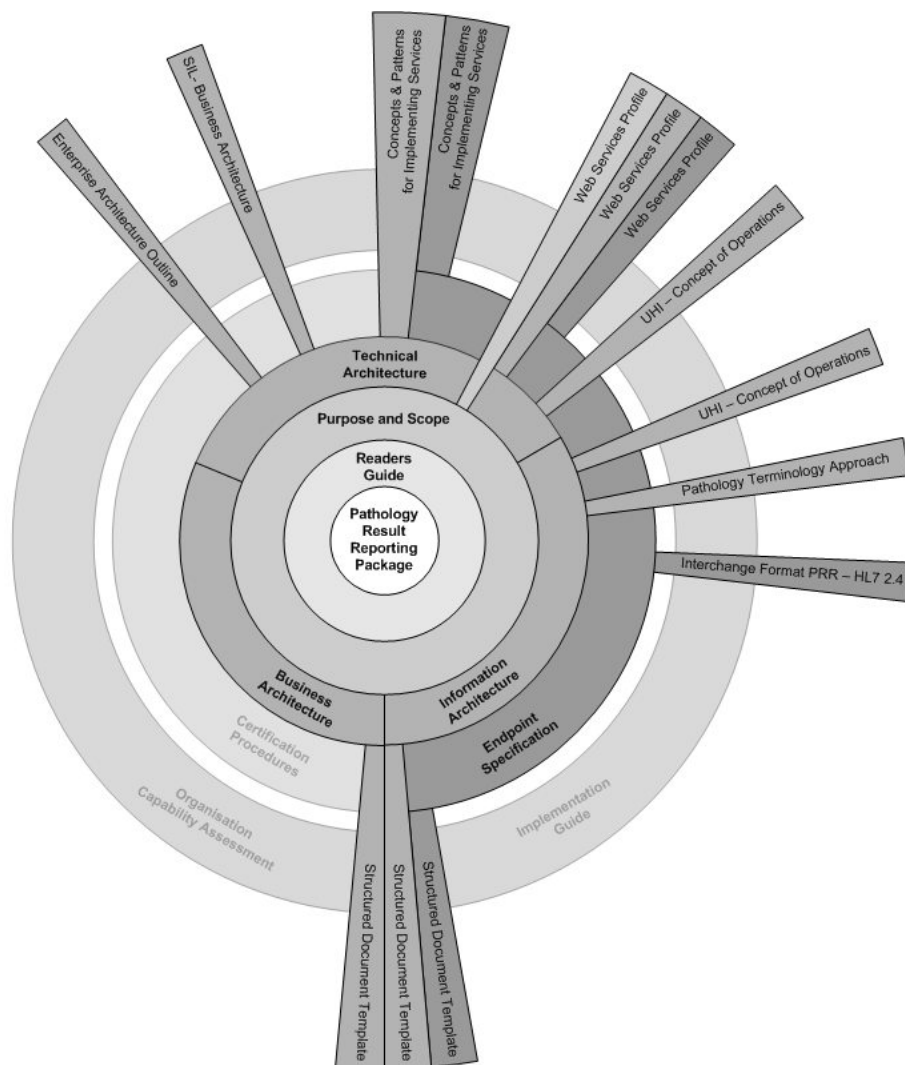


Figure 1: Pathology Result Reporting Package document map

The Package Document Map is designed to show the hierarchy of core documents within the package, and their relationships to ancillary documents. Core package documents are represented as arcs, while ancillary documents (or references to such) appear as radiating spokes. Note that, due to the 'many-to-many' relationships within the package, some ancillary documents appear more than once, and have typically been grouped for clarity.

It is recommended that readers commence with documents at the centre of the map (i.e. the 'Readers' Guide', and 'Purpose and Scope'), working outwards to the detailed, technical documents as needed. Business sponsors may wish to focus upon core documentation, while technical implementers will also likely include ancillary documents.

Core documents are explained in the Readers' Guide [PATH-PRR-RG].

References and Related Documents

For a list of all referenced documents, see the [References](#) at the end of the document, on page 36.

Definitions, Acronyms and Abbreviations

For a lists of abbreviations, acronyms and abbreviations, see the [Definitions section](#) at the end of the document, on page 35.

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 Pathology Result Reporting Package defines four service interfaces:

- Pathology sealed report consumer (chapter 1);
- Pathology sealed report supplier (chapter 2);
- Pathology sealed report acknowledgement consumer (chapter 3); and
- Pathology sealed report acknowledgement supplier (chapter 4).

It also defines four XML Schemas:

- Pathology report (chapter 5)
- Pathology sealed report instance (chapter 6)
- Pathology sealed report acknowledgements (chapter 7)
- Pathology sealed report notification (chapter 8).

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 [PRRSISWX2008].

1 WSDL: Pathology Sealed Report Consumer

1.1 Introduction

1.1.1 Purpose

This service interface is implemented by parties that receive pathology sealed reports from other parties.

For example, this service interface could be implemented by Practice Management Software that receives pathology reports directly from pathology laboratories. It could also be implemented by intermediaries to receive pathology reports from pathology laboratories, which the intermediary then delivers to the GP.

1.1.2 Identity

This service interface is identified by the service namespace:

```
urn:xml-gov-au:nehta:service:PathologySealedReportConsumer: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 unprefixed names in this chapter):

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

1.1.3 Overview

This service interface is illustrated in Figure 2.

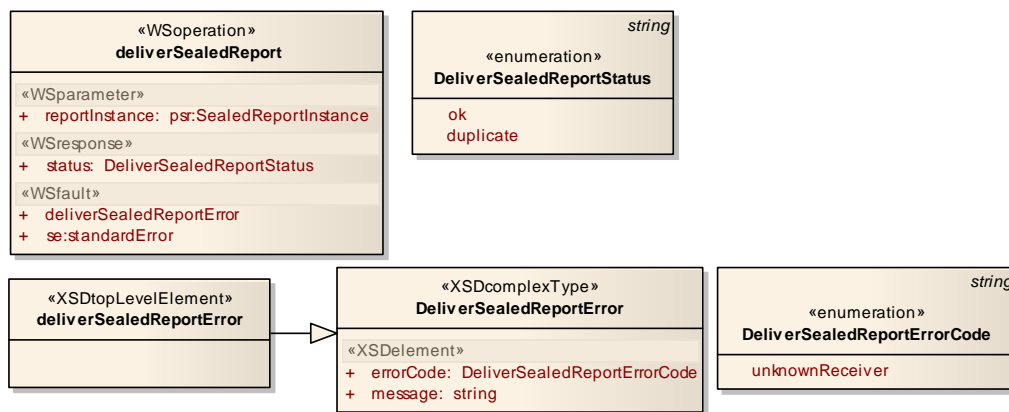


Figure 2: Components of the Pathology Sealed Report Consumer service interface

1.2 Operations

This service interface defines one operation:

- deliverSealedReport

A client would invoke `deliverSealedReport` with each pathology report being sent. Usually, the service will be provided by the receiver of the pathology report (e.g. the General Practice), unless there is some prior agreement with the service provider to ensure that the pathology report instance will get delivered to the receiver.

1.2.1 deliverSealedReport

1.2.1.1 Purpose

This operation is used to deliver a pathology sealed report to the service provider. It is invoked by the client that has a pathology report instance to deliver.

It is possible to invoke this operation multiple times with the same report instance. Subsequent invocations will be detected as duplicates, and will not result in a duplicate report instance being delivered. This is useful for retrying the operation if the client does not know whether it succeeded or not.

1.2.1.2 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 contains a report instance, which includes the metadata about the report instance (basically, a report instance identifier and where it is being delivered to and from), and the pathology report (signed and then encrypted).

The `reportInstanceId` MUST be a globally unique identifier for the report instance.

The sender MUST generate a new `reportInstanceId` for each new report instance it sends.

Clients which are not the sender MUST use the same `reportInstanceId` that was used by the sender for this report instance.

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

1.2.1.3 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 indicates whether the operation succeeded or not using the enumerated values in the `DeliverSealedReportStatus` simpleType (see section 1.3.3).

The `deliverSealedReport` operation MUST detect subsequent invocations that have the same `reportInstanceId` value and not deliver duplicate report instances.

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 `reportInstanceId`. If the earlier invocation was successful, it will not result in a duplicate report. If the earlier invocation did not succeed, then the retry can succeed.

1.2.1.3.1 Faults

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

The `deliverSealedReport` operation MAY respond with a fault containing a `deliverSealedReportError` element.

1.3 Schema components

1.3.1 element: deliverSealedReport

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

1.3.2 element: deliverSealedReportResponse

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

1.3.3 element: deliverSealedReportError

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

1.3.4 simpleType: DeliverSealedReportErrorCode

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

The enumerated value is:

`unknownReceiver`

the service could not handle sealed reports for the requested receiver.

1.3.5 simpleType: DeliverSealedReportStatus

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

The enumerated values are:

`ok`

report instance successfully delivered to service provider.

`duplicate`

the `reportInstanceId` is the same as another sealed report that has been processed by the service. This is not considered an error because the actual delivery of the report instance has been successful.

1.3.6 complexType: DeliverSealedReportError

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

1.4 Implementation

1.4.1 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].

1.4.2 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 **MUST** allow all parties that it expects to receive pathology sealed report instances from to invoke operations from this interface.

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

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

For example, if this interface is implemented by a Practice Management Software, it could allow access to all the pathology laboratories that it expects to get reports from. Since unsolicited copy reports could be delivered to the practice, it would usually allow all pathology laboratories to have access. If the practice has established a relationship with a third party intermediary (to take care of delivery) it would allow them access too.

2 WSDL: Pathology Sealed Report Supplier

2.1 Introduction

2.1.1 Purpose

This service interface is implemented by parties that make pathology sealed reports available for other parties to retrieve.

For example, this service interface could be implemented by a pathology laboratory for GPs to retrieve pathology reports from. It could also be implemented by intermediaries (who have had pathology reports delivered to them by the pathology laboratories) for GPs to then retrieve.

2.1.2 Identity

This service interface is identified by the service namespace:

`urn:xml-gov-au:nehta:service:PathologySealedReportSupplier: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 unprefixed names in this chapter):

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

2.1.3 Overview

This service interface is illustrated in Figure 3 and Figure 4.

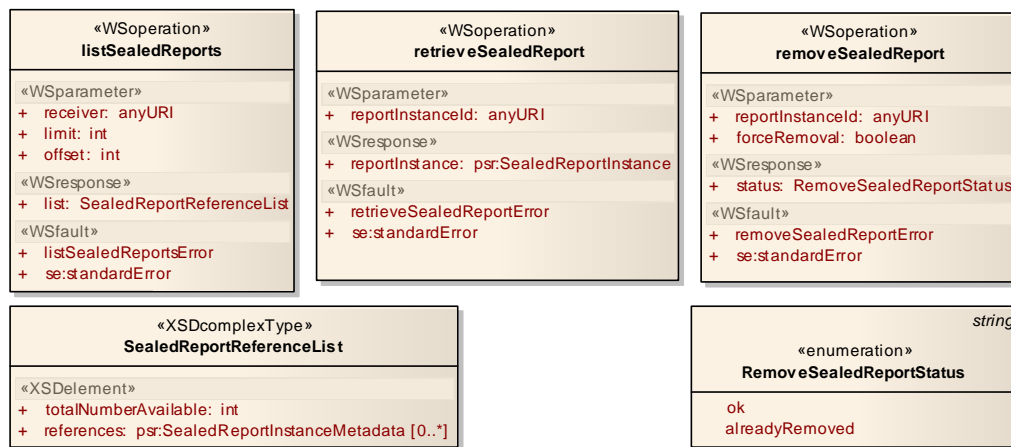


Figure 3: Components of the Pathology Sealed Report Supplier service interface (1)

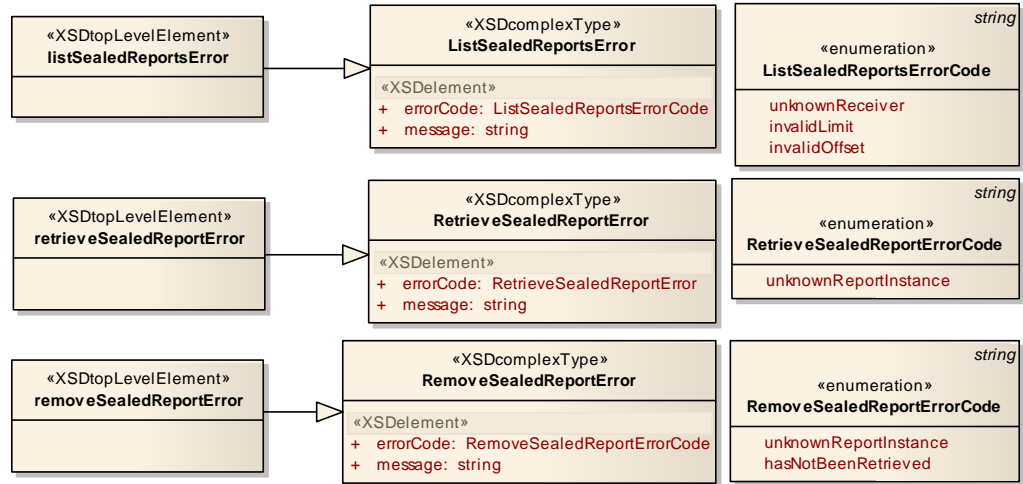


Figure 4: Components of the Pathology Sealed Report Supplier service interface (2)

2.2 Operations

This service interface defines three operations:

- `listSealedReports`;
- `retrieveSealedReport`; and
- `removeSealedReport`.

A client would invoke `listSealedReports` to obtain a list of reports that are available for it to retrieve. For each report in the list, they can invoke `retrieveSealedReport` to retrieve it, and invoke `removeSealedReport` to indicate that they have retrieved it. Later, additional invocations to `listSealedReport` can be made to obtain any new reports.

This service interface is designed to support both targeted collection and polling.

- With targeted collection, the client already knows which report instance they are retrieving. This might be because the client has received a notification indicating which report instance is available for retrieval. The `listSealedReports` operation does not need to be used.
- With polling, the client periodically invokes `listSealedReports` to discover if there are any report instances to retrieve.

2.2.1 listSealedReports

2.2.1.1 Purpose

This operation is used to indicate which reports are available for retrieval. It is invoked by clients that poll the service to retrieve reports.

It is possible to invoke this operation multiple times with the same request. The response will be the same, unless some report instances have been removed (by invoking `removeSealedReport`) 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 report instances and includes a list of references to some (or all) of them. A reference is the report instance metadata, which includes the `reportInstanceId` that can be used to invoke the `retrieveSealedReport` and `removeSealedReport` operations.

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

The number of references 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 references it 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 report instances are available and not retrieve any of them.

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

The number of references returned 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 references. The service can deliberately return fewer references (maybe to reduce bandwidth and processing overheads). This is like a limit for the service provider.

The `offset` can be greater than or equal to the `totalNumberAvailable`. In this situation, no references are returned.

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

2.2.1.3.1 Faults

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

The `listSealedReports` operation MAY respond with a fault containing a `listSealedReportsError` element.

2.2.2 retrieveSealedReport

2.2.2.1 Purpose

This operation is used to make a report instance available. It is invoked by a client to obtain the report instance.

It is possible to invoke this operation multiple times with the same `reportInstanceId`. The same report instance will be returned until it is removed using the `removeSealedReport` operation.

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 the `reportInstanceId` that identifies the report instance to retrieve.

2.2.2.3 Response

The response contains the report instance. It includes the instance metadata (i.e. `reportInstanceId`, `sender`, `origin`, `receiver`, `destination`, and `size`) and the signed and encrypted pathology report.

2.2.2.3.1 Faults

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

The `retrieveSealedReport` operation MAY respond with a fault containing a `retrieveSealedReportError` element.

2.2.3 removeSealedReport

2.2.3.1 Purpose

This operation is used to indicate that the report instance is no longer available for retrieval.

It is usually invoked by clients that have successfully retrieved the report instance (by using `retrieveSealedReport`). It is also possible to remove a report instance without first retrieving it.

It is possible to invoke this operation multiple times with the same `reportInstanceId`. Subsequent invocations will indicate that the report instance 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.

2.2.3.2 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 two things:

- `reportInstanceId` which identifies which report instance to remove; and
- `forceRemoval` which indicates whether a report instance can be removed if it has not been retrieved.

2.2.3.2.1 *reportInstanceId*

The `reportInstanceId` element MUST identify the report instance to remove.

2.2.3.2.2 *forceRemoval*

If the `forceRemoval` element is set to true, the service MUST remove the report instance (even if it has not been retrieved).

If the `forceRemoval` element is set to false and the report instance has not been retrieved, the service MUST respond with `hasNotBeenRetrieved` error code and not remove the report instance.

The `forceRemoval` provides a safety mechanism to prevent accidental removal of report instances that have not been retrieved. Removal of report instances that have not been retrieved is not expected to occur under normal operation. It might be used for maintenance operations, such as removing report instances that are never going to be retrieved.

Clients SHOULD invoke this operation with `forceRemoval` set to false unless they know they are deliberately removing a report that has not been retrieved.

2.2.3.3 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 `RemoveSealedReportStatus` simpleType (see section 2.3.12).

2.2.3.3.1 Faults

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

The `removeSealedReport` operation MAY respond with a fault containing a `removeSealedReportError` element.

2.3 Schema components

2.3.1 element: listSealedReports

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

2.3.2 element: listSealedReportsResponse

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

2.3.3 element: removeSealedReport

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

2.3.4 element: removeSealedReportResponse

This element is used in responses to the `removeSealedReport` operation.

2.3.5 element: retrieveSealedReport

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

2.3.6 element: retrieveSealedReportResponse

This element is used in responses to the `retrieveSealedReport` operation.

2.3.7 element: listSealedReportsError

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

2.3.8 element: removeSealedReportError

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

2.3.9 element: retrieveSealedReportError

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

2.3.10 simpleType: ListSealedReportsErrorCode

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

The enumerated values are:

`unknownReceiver`

The receiver cannot be accepted by the service provider.

`invalidLimit`

The limit is not a positive integer or zero.

`invalidOffset`

The offset is not a positive integer or zero.

2.3.11 simpleType: RemoveSealedReportErrorCode

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

The enumerated values are:

`unknownReportInstance`

The `reportInstanceId` is unknown to the service provider.

`hasNotBeenRetrieved`

The report instance has not been retrieved and `forceRemoval` has not been set to true.

2.3.12 simpleType: RemoveSealedReportStatus

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

The enumerated values are:

`ok`

Operation was successful. The report instance has been removed by this invocation of the operation.

`alreadyRemoved`

Operation was successful. The report instance was previously removed.

2.3.13 simpleType: RetrieveSealedReportErrorCode

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

The enumerated value is:

`unknownReportInstance`

The `reportInstanceId` is unknown to the service provider.

2.3.14 complexType: ListSealedReportsError

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

2.3.15 complexType: RemoveSealedReportError

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

2.3.16 complexType: RetrieveSealedReportError

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

2.3.17 complexType: SealedReportReferenceList

This complexType is use in responses from the `listSealedReports` 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 **MUST** allow all parties that it expects to retrieve pathology sealed report instances from it to invoke operations from this interface.

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

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

For example, if this interface is implemented by a pathology laboratory, it could allow all general practices that are its customers to invoke operations from this interface. It could prohibit access from other parties (including other general practices) from access. If the pathology laboratory has established a relationship with a third party intermediary (to collect reports and then deliver them to their receivers) it would allow them access too.

3 WSDL: Pathology Sealed Report Acknowledgement Consumer

3.1 Introduction

3.1.1 Purpose

This service interface is implemented by parties that wish to receive acknowledgements that a sealed pathology report has been successfully delivered.

For example, this service interface could be implemented by pathology laboratory software that receives acknowledgements. It could also be implemented by intermediaries to receive acknowledgements from practice management software, which they then deliver to the pathology laboratory.

3.1.2 Identity

This service interface is identified by the service namespace:

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

3.1.3 Overview

This service interface is illustrated in Figure 5.

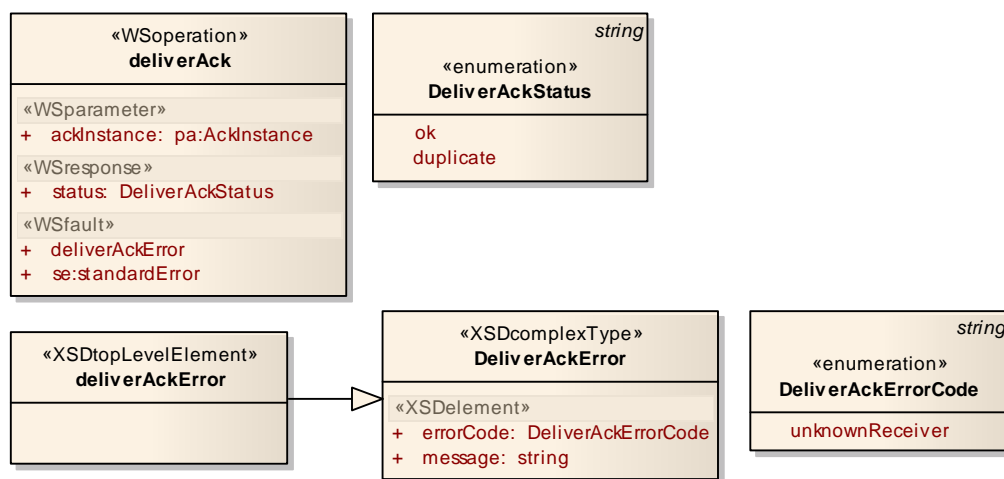


Figure 5: Components of the Pathology Sealed Report Acknowledgement Consumer service interface

3.2 Operations

This service interface defines one operation:

- deliverAck

A client would invoke deliverAck with each pathology sealed report acknowledgement being sent. Usually, the service will be provided by the receiver of the acknowledgement (e.g. the pathology laboratory), unless there

is some prior agreement with the service provider to ensure that the acknowledgement will get delivered to the receiver.

3.2.1 deliverAck

3.2.1.1 Purpose

This operation is used to deliver an acknowledgment of the successful delivery of a pathology sealed report to the service provider. It is invoked by the client, which wants to deliver the acknowledgement.

It is possible to invoke this operation multiple times with the same acknowledgement instance. Subsequent invocations will be detected as duplicates, and will not result in a duplicate report instance being delivered. This is useful for retrying the operation if the client does not know whether it succeeded or not.

3.2.1.2 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 `ackInstanceId` must be a globally unique identifier for each new acknowledgement instance it sends.

Clients which are not the sender MUST use the same `ackInstanceId` that was used by the sender for this acknowledgement instance.

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

The `type` indicates the type of acknowledgement.

The data contains a HL7 version 2.4 acknowledgement message.

3.2.1.3 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 indicates whether the operation succeeded or not using the enumerated values in the `DeliveryAckStatus` simpleType (see section 3.3.5).

The `deliverAck` operation MUST detect subsequent invocations that have the same `ackInstanceId` value and not deliver duplicate acknowledgement instances.

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 `AckInstanceId`. If the earlier invocation was successful, it will not result in a duplicate acknowledgement. If the earlier invocation did not succeed, then the retry can succeed.

3.2.1.3.1 Faults

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

The `deliverAck` operation MAY respond with a fault containing a `deliverAckError` element.

3.3 Schema components

3.3.1 element: deliverAck

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

3.3.2 element: deliverAckResponse

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

3.3.3 element: deliverAckError

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

3.3.4 simpleType: DeliverAckErrorCode

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

The enumerated value is:

`unknownReceiver`

The receiver is not supported by the service provider.

3.3.5 simpleType: DeliverAckStatus

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

The enumerated values are:

`ok`

acknowledgement successfully delivered to service provider.

`duplicate`

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

3.3.6 complexType: DeliverAckError

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

3.4 Implementation

3.4.1 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].

3.4.2 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 MUST allow all parties that it expects to receive acknowledgement instances from to invoke this interface.

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

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

For example, if this interface is implemented by a pathology laboratory, it needs to allow access to all the GPs that it has sent pathology reports to. If it expects acknowledgements to be delivered to it from third party intermediaries it needs to allow them access too.

4 WSDL: Pathology Sealed Report Acknowledgement Supplier

4.1 Introduction

4.1.1 Purpose

This service interface is implemented by parties that make pathology sealed report acknowledgments available for other parties to retrieve.

For example, this service interface could be implemented by an intermediary (who has received acknowledgments from GPs) for the pathology laboratory to retrieve.

This service interface is not used by the ordinary pathology result report delivery scenarios. It is defined here for completeness, so that it is available for products which need to handle acknowledgements in this manner. These would be intermediary products or Practice Management Software that has a special relationship with intermediaries or particular pathology laboratories.

4.1.2 Identity

This service interface is identified by the service namespace:

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

4.1.3 Overview

This service interface is illustrated in Figure 6 and Figure 7.



Figure 6: Components of the Pathology Sealed Report Acknowledgement Supplier service interface (1)

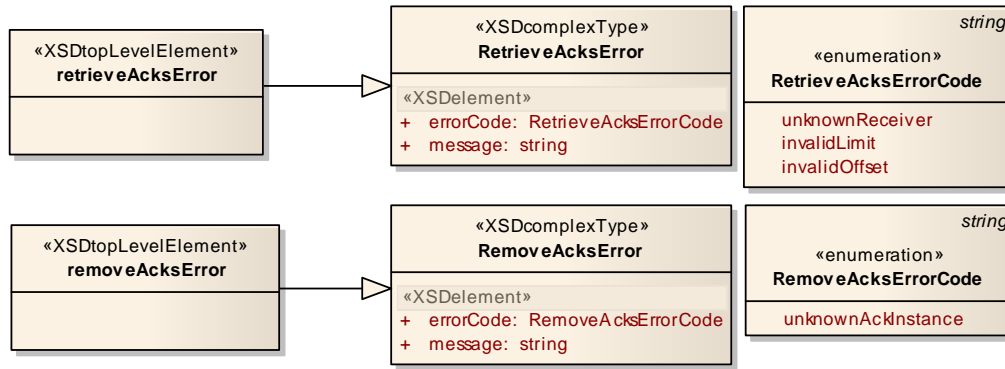


Figure 7: Components of the Pathology Sealed Report Acknowledgement Supplier service interface (2)

4.2 Operations

This service interface defines two operations:

- `retrieveAcks`; and
- `removeAcks`.

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

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

4.2.1 `retrieveAcks`

4.2.1.1 Purpose

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

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

4.2.1.2 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.

4.2.1.3 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 acknowledgement instances and includes a list of some (or all) acknowledgements. An acknowledgement is the acknowledgement metadata, which includes the

`ackInstanceId` that can be used to invoke the `retrieveAcks` and `removeAcks` operations.

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

The number of acknowledgements 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 acknowledgements 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 acknowledgement instances are available and not retrieve any of them.

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

The number of acknowledgements returned 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 acknowledgements. The service can deliberately return fewer acknowledgements (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 acknowledgements returned: this is not an error condition.

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

4.2.1.3.1 *Faults*

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

The `listAcks` operation MAY respond with a fault containing a `retrieveAcksError` element.

4.2.2 **removeAcks**

4.2.2.1 Purpose

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

It is invoked by clients that have successfully retrieved the acknowledgement instance (by using `retrieveAcks`).

It is possible to invoke this operation multiple times with the same `ackInstanceId`. Subsequent invocations will indicate that the acknowledgement instance 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.

4.2.2.2 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 `ackInstanceId` which identifies the acknowledgement instances to remove.

Each `ackInstanceId` element MUST identify an acknowledgement instance to remove.

4.2.2.3 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 `RemoveAckStatus` simpleType (see section 4.3.7).

4.2.2.3.1 Faults

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

The `removeAcks` operation MAY respond with a fault containing a `removeAcksError` element.

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

The `ackInstanceId` in the `removeAcksError` element indicates which of the acknowledgements were unknown.

4.3 Schema components

4.3.1 element: `removeAcks`

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

4.3.2 element: `removeAcksResponse`

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

4.3.3 element: `retrieveAcks`

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

4.3.4 element: `retrieveAcksResponse`

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

4.3.5 element: `removeAcksError`

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

4.3.6 element: `retrieveAcksError`

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

4.3.7 simpleType: `RemoveAckStatus`

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

The enumerated values are:

`ok`

the acknowledgement was successfully removed.

`alreadyRemoved`

the acknowledgement was previously removed.

4.3.8 simpleType: RemoveAcksErrorCode

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

The enumerated value is:

`unknownAckInstance`

the acknowledgement instance identifier is not known.

4.3.9 simpleType: RetrieveAcksErrorCode

This simpleType is used in faults from the `removeAcks` 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.

4.3.10 complexType: AckList

This complexType is use in responses from the `listAcks` operation.

4.3.11 complexType: RemoveAckResult

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

4.3.12 complexType: RemoveAcksError

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

4.3.13 complexType: RetrieveAcksError

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

4.4 Implementation

4.4.1 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].

4.4.2 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 MUST allow all parties that it expects to retrieve acknowledgement instances from it to invoke operations from this interface.

Implementations SHOULD allow parties that have been authenticated as the receiver to invoke this interface. That is, a receiver should always be able to

list, retrieve and remove acknowledgement instances that are to be delivered to it.

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

For example, if this interface is implemented by an intermediary that receives acknowledgements on behalf of a pathology laboratory, then it needs to allow that pathology laboratory to invoke operations from this interface.

5 XSD: Pathology Report

5.1 Introduction

5.1.1 Purpose

This schema is used to represent pathology reports and signed pathology reports.

It is used in the definition of a sealed pathology report, which is an encrypted representation of a signed pathology report.

5.1.2 Identity

This schema has the namespace of:

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

5.1.3 Overview

This schema is illustrated in Figure 8.

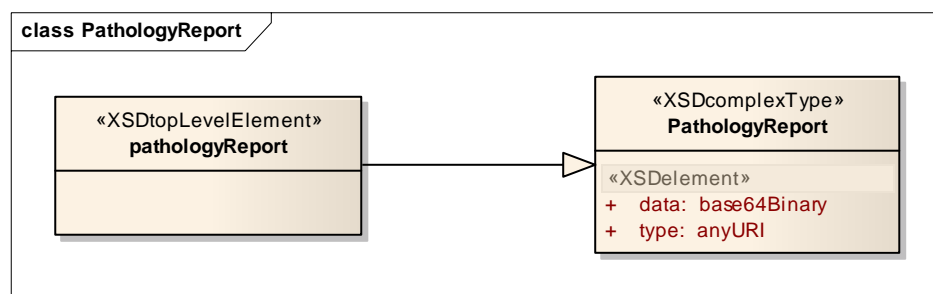


Figure 8: Components of the Pathology Report schema

5.2 Schema components

5.2.1 element: pathologyReport

The report element is of type PathologyReport.

The PathologyReport type is defined in section 5.2.2. In brief, it contains a report type and the report data.

5.2.2 complexType: PathologyReport

The PathologyReport complexType is used to represent a pathology report.

It contains the following elements:

- type; and
- data.

5.2.2.1 type

The type MUST identify the type of report data being used.

The type MUST contain the value of:

- urn:xml-gov-au:nehta:data:PathologyReportTypeH17:2.4

5.2.2.2 data

The data MUST contain the data of the pathology report.

The data MUST contain a HL7 version 2.4 pathology report message.

This element is defined as `xsd:base64Binary` to ensure that the HL7v2 encoding is properly preserved. The HL7v2 syntax can contain binary data which can be lost if it is represented as text in XML.

6 XSD: Pathology Sealed Report Instance

6.1 Introduction

6.1.1 Purpose

This schema is used to represent instances of sealed pathology reports. An *instance* of a sealed pathology report is an occurrence of a sealed pathology report that is being delivered from a sender to a receiver. Each time a pathology sealed report is delivered (e.g. to different receivers or multiple copies of the same report delivered to one receiver) that is a new instance.

This XML Schema is used by the following service interfaces:

- Pathology sealed report consumer (chapter 1); and
- Pathology sealed report provider (chapter 2).

6.1.2 Identity

This schema has the namespace of:

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

6.1.3 Overview

This schema is illustrated in Figure 9.

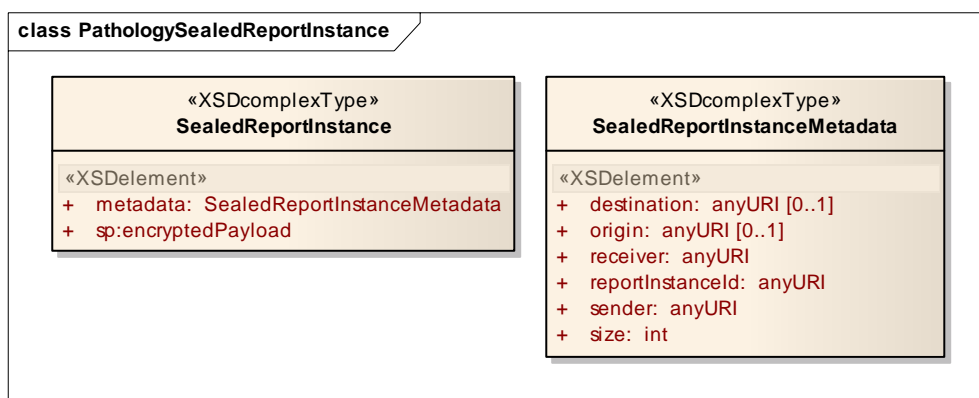


Figure 9: Components of the Pathology Sealed Report Instance schema

6.2 Schema components

6.2.1 complexType: SealedReportInstance

The `SealedReportInstance` complexType is used to represent a report and the associated data for a particular instance being delivered.

It contains the following elements:

- `metadata`; and
- `sp:encryptedPayload`.

6.2.1.1 metadata

The metadata is of type `SealedReportInstanceMetadata` which is described in section 6.2.2.

6.2.1.2 sp:encryptedPayload

This element contains a pathology sealed report, which is constructed from:

- A `sp:encryptedPayload` element where the decrypted data of the encrypted payload contains:
- A `sp:signedPayload` element where the data being signed is:
- A `pr:pathologyReport` element.

The `sp:encryptedPayload` element MUST contain one key that is encrypted for the receiver.

The `sp:signedPayload` element MUST contain at least one signature that was created by the sender.

6.2.2 complexType: SealedReportInstanceMetadata

The `SealedReportInstanceMetadata` complexType is used to represent the metadata that is associated with a report instance being delivered.

It contains the following elements:

- `reportInstanceId`;
- `receiver`;
- `sender`;
- `destination` (optional); and
- `origin` (optional);
- `size`.

The metadata identifies the parties involved in the delivery process according to the model defined in *Concepts and Patterns* [CPIS2008]. In this model, the *origin* and *destination* are the parties that handle the content of the report (e.g. the pathologist and the general practice doctor); and the *sender* and *receiver* are the parties that handle the delivery of the report (e.g. the pathology laboratory and the general practice).

6.2.2.1 reportInstanceId

The `reportInstanceId` MUST contain a unique identifier for the pathology sealed report instance.

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

6.2.2.2 receiver

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

The `receiver` MUST follow the NEHTA scheme for identifiers, as defined by [QI2008]. 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 pathology report 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.

6.2.2.3 sender

The `sender` MUST identify the initial party that sent the report.

The `sender` MUST follow the NEHTA scheme for identifiers, as defined by [QI2008]. This scheme uses a URI as a unique identifier.

For example, it could contain a URI representing the HPI-O number for the pathology laboratory that is sending the report.

The `sender` also identifies where the delivery acknowledgement will be sent to. See chapters 3 and 4 for the delivery acknowledgement interfaces.

6.2.2.4 destination

The `destination` SHOULD contain the identity of the person the pathology report is intended for at the receiver.

For example, the `destination` can contain the HPI-I number for the General Practitioner.

6.2.2.5 origin

The `origin` SHOULD contain the identity of the person responsible for sending the pathology report.

For example, the `origin` can contain the HPI-I number for the pathologist.

6.2.2.6 size

The `size` MUST contain the size of the sealed pathology report in bytes. This is measured as the size of the binary encrypted data inside the `xenc:CipherValue` element within the `sealedReport` element. It is the size of the binary data—not the size of the base64 encoded text.

The `size` allows clients that retrieve reports (using the pathology sealed report supplier service interface) to choose which reports to retrieve if communications bandwidth is an issue. For example, it could choose to retrieve the smaller reports first.

Editorial note: NEHTA is seeking feedback on whether this size information is useful or not, and whether it justifies the cost needed to produce it. Vendors and users who want this to remain in the interface should inform NEHTA about the scenarios where they intend to use it. If NEHTA does not receive significant support for this feature, it will be removed in the final version of the service interface.

7 XSD: Pathology Sealed Report Acknowledgement

7.1 Introduction

7.1.1 Purpose

This schema is used to represent acknowledgements of the delivery of a pathology sealed report instance.

It is used by the following service interfaces:

- Pathology sealed report acknowledgement consumer (section 3); and
- Pathology sealed report acknowledgement provider (chapter 4).

7.1.2 Identity

This schema has the namespace of:

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

7.1.3 Overview

This schema is illustrated in Figure 10.

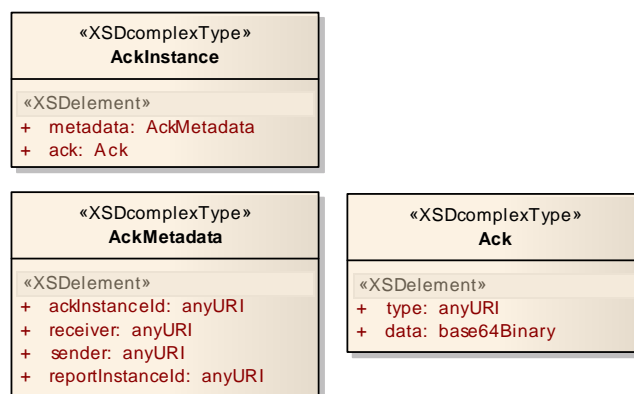


Figure 10: Components of the Pathology Sealed Report Acknowledgement schema

7.2 Schema components

7.2.1 complexType: Ack

The `Ack` complexType is used to represent an acknowledgement.

It contains the following elements:

- `type`; and
- `data`.

7.2.1.1 `type`

The `type` is a value that identifies the type of data in `data`.

The `type` MUST have the following value:

- `urn:xml-gov-au:nehta:data:PathologyAckTypeHL7:2.4`

7.2.1.2 data

The `data` MUST contain the acknowledgement data.

The `data` MUST contain an HL7v2.4 acknowledgement message.

7.2.2 complexType: AckInstance

The `AckInstance` complexType is used to represent a delivery acknowledgement instance for a sealed pathology report instance.

It contains the following elements:

- `metadata`; and
- `ack`.

7.2.2.1.1 metadata

The `metadata` is of type `AckMetadata`, which is described in section 7.2.3.

7.2.2.1.2 ack

The `ack` is of type `Ack`.

The `Ack` complexType is defined in 7.2.1. In brief, it is an acknowledgement type indicating it is an HL7v2.4 acknowledgement and the acknowledgement itself.

7.2.3 complexType: AckMetadata

The `AckMetadata` complexType is used to represent the metadata that is associated with a sealed pathology report acknowledgement.

It contains the following elements:

- `ackInstanceId`;
- `receiver`;
- `sender`; and
- `reportInstanceId`.

7.2.3.1 ackInstanceId

The `ackInstanceId` must contain a unique identifier for the acknowledgement instance.

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

7.2.3.2 receiver

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

The `receiver` MUST follow the NEHTA scheme for identifiers, as defined by [QI2008]. This scheme uses a URI as a unique identifier.

For example, it could contain a URI representing the HPI-O number for the pathology laboratory.

7.2.3.3 sender

The `sender` MUST identify the initial party that sent the acknowledgement.

The `sender` MUST follow the NEHTA scheme for identifiers, as defined by [QI2008]. 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 is sending the acknowledgement.

7.2.3.4 `reportInstancedId`

The `reportInstanceId` MUST contain the report instance identifier of the report instance being acknowledged.

This is the same value that was used for the sealed report instance in either `deliverSealedReport` (section 1.2.1) or `retrieveSealedReport` (section 0). It is usually a UUID formatted as a URN.

8 XSD: Pathology Sealed Report Notification

8.1 Introduction

8.1.1 Purpose

This schema is used to represent notifications to indicate that a sealed pathology report is available for retrieval.

This XML element is used with the following service interfaces:

- Notification consumer; and
- Notification receiver

As defined in the *Notifications: Endpoint Specification* [NSIS2008].

8.1.2 Identity

This schema has the namespace of:

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

8.1.3 Overview

This schema is illustrated in Figure 11.

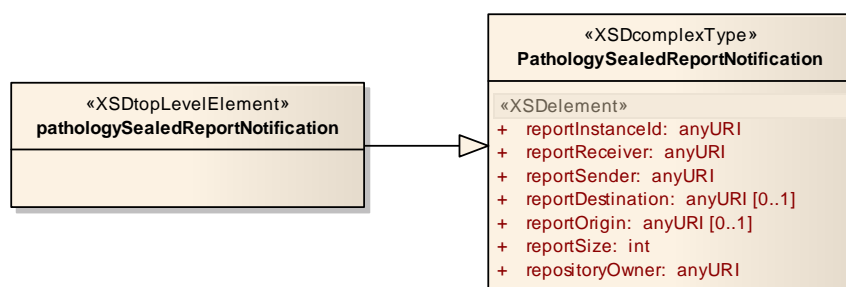


Figure 11: Components of the Pathology Sealed Report Notification schema

8.2 Schema components

8.2.1 element: pathologySealedReportNotification

The `pathologySealedReportNotification` element is used as the notification data.

For more information about the notification service interfaces see *Notifications: Endpoint Specification* [NSIS2008].

8.2.2 complexType: PathologySealedReportNotification

The `PathologySealedReportNotification` complexType is used to represent a notification that a sealed pathology report is available for retrieval.

It contains the following elements:

- `reportInstanceId`.

- `reportReceiver`;
- `reportSender`;
- `reportDestination` (optional);
- `reportOrigin` (optional);
- `reportSize`; and
- `repositoryOwner`.

8.2.2.1 `reportInstanceId`

The `reportInstanceId` MUST contain the report instance identifier for the sealed pathology report instance.

8.2.2.2 `reportReceiver`

The `reportReceiver` MUST identify the sender of the sealed pathology report.

For example, it could contain a URI representing the HPI-O number for the pathology laboratory.

Usually the receiver of the notification will be the same as the receiver of the report, but there are scenarios where notifications are sent to a different receiver.

8.2.2.3 `reportSender`

The `reportSender` MUST identify the receiver of the sealed pathology report.

For example, it could contain a URI representing the HPI-O number for the general practice.

Usually the sender of the notification will be the same as the sender of the report, but there are scenarios where notifications are sent by a different sender.

8.2.2.4 `reportDestination`

The `reportDestination` MUST identify the destination of the sealed pathology report instance, if it is present.

For example, it could contain a URI representing the HPI-I of the general practitioner.

8.2.2.5 `reportOrigin`

The `reportOrigin` must identify the origin of the sealed pathology report, if it is present.

For example, it could contain a URI representing the HPI-I of the pathologist.

8.2.2.6 `reportSize`

The `reportSize` MUST contain the size of the sealed pathology report, in bytes.

This has the same value as the size element defined in section 6.2.2.6.

8.2.2.7 `repositoryOwner`

The `repositoryOwner` MUST contain the identity of the party operating the service to collect the sealed pathology report from.

This party MAY be different from the sender. For example, when the report is being stored by an intermediary for collection the intermediary is the owner.

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
HPI-I	Healthcare Provider Identifier for Individuals
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
WSDL	Web Services Description Language
XML	Extensible Markup Language

Glossary

This table lists specialised terminology in alphabetical order.

Term	Description
Pathology report	Data that represents a pathology report.
Pathology sealed report	A pathology report that has been digitally signed and then encrypted. It is sealed in the sense that parties (other than the party or parties that can decrypt it) cannot access the report data.
Pathology sealed report instance	A copy of a sealed pathology report that is being delivered from one party to another.
Report	See "pathology report."
Sealed report	See "pathology sealed report."
Sealed report instance	See "pathology sealed report instance."
Report instance	See "pathology sealed report instance."

Namespaces

The following XML namespace prefixes are used in this document:

pa	urn:xml-gov-au:nehta:types:PathologySealedReportAck:1.0-draft-20080901
pr	urn:xml-gov-au:nehta:types:PathologyReport:1.0-draft-20080901
psr	urn:xml-gov-au:nehta:types:PathologySealedReportInstance:1.0-draft-20080901
se	urn:xml-gov-au:nehta:types:StandardError:1.0-draft-20080901
sp	urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901
xenc	http://www.w3.org/2001/04/xmlenc#
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.

- [CPIS2008] NEHTA, *Concepts and Patterns for Implementing Services v2.0*
- [QI2008] NEHTA, *Qualified Identifiers v1.0*
- [NSIS2008] NEHTA, *Notifications: Endpoint Specification v1.0*
- [PRRPTA2008] NEHTA, *Pathology Result Reporting Package (v1.0 Draft) Technical Architecture v2.0*
- [PRRSISWX2008] NEHTA, *Pathology Result Reporting Package: Endpoint Specification WSDL and XML Schema files v2.1*
- [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>
- [WSP2008] NEHTA, *Web Services Profile v3.0*
- [XSPP2008] NEHTA, *XML Secured Payload Profile v1.0*

Informative references

The documents listed below may provide further information about the issues discussed in this document.

- [IF2007] NEHTA, *Interoperability Framework v2.0*

Appendix A: HL7 V2.4 Comparisons

There is an overlap between the concepts and constructs defined in the HL7 version 2.4 standard and the use of Web services as defined through [WSP2008] and [CPIS2008] and specifically as defined in this document.

It is expected that the HL7 v2.4 messages transported will not include those messages representing transport level concepts. With this in mind, this appendix is provided to explain how the HL7 concepts correspond to the specification described here.

Note that an exhaustive description of the HL7 v2.4 protocol is out of the scope of this document and only those areas which have been identified in early drafts of the documents as causing confusion in the context of Web services are covered here.

A.1 Concepts

A.1.1 Messages and Operations

HL7 version 2.4 has a strong emphasis on the definition of messages, with no reference to a particular transport mechanism. Interoperability is achieved through the exchange of predefined general messages. No formal mechanism is defined for the definition of remote procedure call style operations.

The use of SOA as an architectural style predicates the identification of business level services, and although these do not need to be realised in a particular predefined technology, there is an expectation that a service will be 'invoked'. This in turn implies the use of operations and consequently the use of remote procedure calls instead of message exchange.

The approach taken in this document is to define specific operations, over which particular messages may be exchanged.

A.1.2 Acknowledgements

The HL7 v2.4 standard specifies "original acknowledgement" and "enhanced acknowledgement" as its two supported modes of acknowledging requests.

Operations as defined in [WSP2008] are defined as using the request-response pattern, therefore an "original acknowledgement" is inherent in the definition of an operation that occurs between two business domain points, and these acknowledgements are defined as transport or system layer acknowledgements (response from Web service invocation or report processed).

The "enhanced acknowledgement" corresponds to an action that occurs outside of the context of a called operation, but in response to an incoming operation (request). This action will often be modelled as an asynchronous operation (with respect to the original request/operation). This will then not be identified as an acknowledgement but instead and explicitly as a separate operation.

A.1.3 Batching

The batching of delivery of multiple reports is out of scope of the current version of this specification. Whether or not to introduce batching concerns to the interfaces defined in this specification will be considered at a later date. However, HL7 batching mechanisms will not be used. Consequently, the use of the 'BHS' segment is not used in conjunction with the services and operations defined in this specification.

A.2 Messages

The following sections describe specific messages, segments, elements and how these correspond to the specification described in this document.

Note: this is not an exhaustive list and that only those that are relevant have been listed.

A.2.1 Message: ACK_ALL

The 'ACK' (General Acknowledgement) and 'MCF' (Delayed Acknowledgement) messages are not to be used in conjunction with the services and operations defined in this specification. Instead the operations and implied acknowledgement mechanisms are to be used instead.

A.2.2 Message: OUL_R21

The OUL_R21 (Unsolicited transmission of an observation message) is the expected HL7 v2.4 message to be transmitted as part of this specification.

A.2.3 Segment: ERR

The ERR (Error) segment is used in HL7 v2.4 as a component of the ACK_ALL message. In this specification any operation that occurs between business domain points are defined with 'faults' which may be thrown on invocation of the operation.

Appendix B: Quick reference

B.1 Pathology Sealed Report Consumer

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

deliverSealedReport (reportInstance)

→ deliverSealedReportStatus

B.2 Pathology Sealed Report Supplier

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

listSealedReports (receiver, limit, offset)

→ totalNumberAvailable, metadata*

retrieveSealedReport (reportInstanceId)

→ reportInstance

removeSealedReport (reportInstanceId)

→ removeSealedReportStatus

B.3 Pathology Sealed Report Acknowledgement Consumer

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

deliverAck (ackInstance)

→ deliverAckStatus

B.4 Pathology Sealed Report Acknowledgement Supplier

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

retrieveAcks (receiver, limit, offset)

→ totalNumberAvailable, ackInstance*

removeAcks (ackInstanceId+)

→ removeAckStatus+