



Australian Government

Australian Digital Health Agency



## **HIPS**

### **Module Guide (P2P)**

7 November 2016

V6.1

Approved for external use

**Australian Digital Health Agency**

Level 25, 56 Pitt Street

Sydney, NSW 2000

Australia

[www.digitalhealth.gov.au](http://www.digitalhealth.gov.au)

**Acknowledgements****Council of Australian Governments**

The Australian Digital Health Agency is jointly funded by the Australian Government and all state and territory governments.

**HL7 International**

This document includes excerpts of HL7™ International standards and other HL7 International material. HL7 International is the publisher and holder of copyright in the excerpts. The publication, reproduction and use of such excerpts is governed by the HL7 IP Policy (see <http://www.hl7.org/legal/ippolicy.cfm>) and the HL7 International License Agreement. HL7 and CDA are trademarks of Health Level Seven International and are registered with the United States Patent and Trademark Office.

**Disclaimer**

The Australian Digital Health Agency ("the Agency") 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. The Agency 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 and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is the latest revision.

**Copyright © 2016 Australian Digital Health Agency**

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 the Australian Digital Health Agency. All copies of this document must include the copyright and other information contained on this page.

# Document information

## Key information

<b>Owner</b>	Executive General Manager, Innovation & Development
<b>Contact for enquiries</b>	Australian Digital Health Agency Help Centre
t:	1300 901 001
e:	<a href="mailto:help@digitalhealth.gov.au">help@digitalhealth.gov.au</a>

## Product version history

Product version	Date	Release comments
5.0	July 2015	Initial release (HIPS v5.0)
6.1	November 2016	See release note (DH-2445:2016) for details of changes and bug fixes

# Table of contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
1.1	Purpose .....	5
1.2	Scope.....	5
1.3	Assumptions .....	5
1.4	Definitions and Acronyms .....	6
<b>2</b>	<b>Architectural Detail.....</b>	<b>7</b>
2.1	Component Model .....	8
2.1.1	Existing Functionality for IHI and My Health Record Components .....	8
2.1.2	P2P Addressing Components .....	10
2.1.3	P2P Document Delivery Components .....	12
2.1.4	P2P Document Receiving Components .....	13
2.2	Business Logic.....	15
2.2.1	P2P Messaging Services .....	15
2.2.2	Secure Message Delivery Scheduled Services .....	26
2.2.3	Secure Message Receipt Scheduled Services .....	29
2.2.4	Secure Message Delivery Web Services.....	34
2.2.5	LHSD Services .....	38
2.3	Database Resource Access Layer .....	61
2.3.1	Secure Message Delivery Management .....	61
2.3.2	NEPS Subscription.....	65
2.3.3	Secure Message Endpoint Location .....	67
2.3.4	Provider Individual .....	69
2.3.5	Provider Organisation .....	71
2.3.6	Provider Link .....	74
2.3.7	Change Tracking .....	75

# 1 Introduction

## 1.1 Purpose

Adding the P2P functionality into HIPS will allow states and territories who wish to use HIPS, to have P2P functionality, which will enable the sending and receiving of clinical documents both interstate and intrastate once upgraded to the newest version of HIPS.

HIPS P2P extends the Core and the UI components of HIPS by providing the following key capabilities:

- Directory maintenance, providing a Local Health Service Directory (LHSD) used for the purposes of addressing during P2P messaging, and a mechanism for maintaining and synchronising this LHSD from the National Health Service Directory (NHSD) and associated data sources exposed by the National Endpoint Proxy Service (NEPS).
- Secure message delivery (SMD), providing mechanisms to address, send and receive messages securely between providers, supported by the information maintained within the LHSD.

Throughout the document the term **"Participating Organisation"** is used which related to one of the following system implementers of HIPS:

- Health Provider Organisation (HPO),
- Contracted Service Provider (CSP) or
- Jurisdiction

## 1.2 Scope

This document describes the HIPS P2P Module functions including:

- P2P Messaging Service – point-to-point messaging services
- SMD Services – secure message delivery services
- LHSD Services - local health service directory used for addressing

This document does not list the web services but rather the business logic. The web service methods covered in the HIPS Release v6.1 - Service Catalogue – P2P.

This document does not describe any details of the other HIPS modules as these are covered by other documentation.

## 1.3 Assumptions

During the development of this document the following assumptions have been made:

- The document audience have a high level understanding of health information systems and the terminology used.

## 1.4 Definitions and Acronyms

Throughout the document the term "**Participating Organisation**" is used which related to one of the following system implementers of HIPS:

- Health Provider Organisation (HPO);
- Contracted Service Provider (CSP); or
- Jurisdiction.

The following acronyms have been used through the document:

Item	Definition
ESB	Enterprise Service Bus – integration hub for routing and transforming messages within and between healthcare facilities.
HL7	Health Level Seven
HIPS	Healthcare Identifiers and PCEHR Systems
MRN	Medical Record Number, identified by the code "MR" in PID-3. Ideally one MRN is allocated by the hospital for each patient, though it is common to temporarily allocate a new MRN for emergency patients until their identity is confirmed. These temporary MRNs should be merged back to the original MRN for the patient using an A36 Merge MRN message. This number stored in HospitalPatient.Mrn and is the primary identifier used to find the existing patient records in the HIPS database.
LHSD	Local Health Service Directory, used to lookup addressing

## 2 Architectural Detail

The main elements within the architecture include:

- The Enterprise Service Bus (ESB) is a key part of health jurisdiction enterprise architecture. The ESB brokers messages between numerous jurisdictional systems.
- HIPS, an application server that hosts internally accessible web services and middleware for communicating between internal systems and the HI Service, My Health Record, NASH, NEPS and the DMZ components.
- The DMZ, which encapsulates two secured network zones containing Windows Server 2008 R2 environments, denoted as DMZ Internal and DMZ External. These components are separated from the main HIPS application server to allow for jurisdictions to meet their specific security requirements.
  - DMZ External zone contains the Front-End Server, a Windows Server 2008 R2 environment that hosts a number of publicly accessible web services, which allow external endpoints to deliver messages into the jurisdiction. A number of internal web services are also hosted in this environment that enable internal systems to poll for and remove messages that have been received.
  - DMZ Internal zone contains the Decrypter, a Windows Server 2008 R2 environment, which hosts an internally accessible web service to retrieve encrypted messages received by the Front-End Server, decrypt the messages, and return the decrypted messages to the web service invoker.
- The architecture employs a 'pull' model, in transferring data from a lesser trusted systems to more highly trusted systems. This involves the systems with higher trust being responsible to invoke web services to obtain data from lesser trusted systems.
- Communication between the ESB and HIPS systems occur using SOAP web services over a secure channel (TLS).
- Invocations of web services hosted within the DMZ Internal environment occur over standard (unsecured) HTTP. This enables the firewall between the DMZ and Internal Network to scan incoming messages for viruses and other threats.
- Invocations of web services hosted within the DMZ External environment will occur over a secure channel (TLS), where connecting applications will need to present a client certificate to be authenticated.
- Verification of signing certificates, used by senders to sign payloads within Sealed Messages, will be performed using Certificate Revocation Lists (CRL). HIPS is responsible for verifying signatures and is also responsible for periodically initiating the process to download CRLs from trusted Certificate Authorities (CA).

## 2.1 Component Model

### 2.1.1 Existing Functionality for IHI and My Health Record Components

- Hospital Patient Administration System (PAS). The PAS Records patient identifiers, demographic information, admissions, discharges and transfers. Sends HL7 messages to downstream systems whenever a patient record is created or updated, and whenever a patient is admitted, discharged or transferred. Each hospital may have a separate PAS. These are simulated in development and test using de-identified production messages.
- Enterprise Master Patient Index (EMPI). The EMPI records patient identifiers and demographic information and assigns enterprise identifiers to patients. Sends HL7 messages to downstream systems whenever a hospital patient MRN is moved from one enterprise ID to another. This is an existing enterprise system. Simulated in development and test using de-identified production messages.
- Discharge Summary Authoring. An application in which users create discharge summary documents. Allows creation of discharge summary documents and distribution to the National eHealth Record System and health provider organisations or individuals selected from a provider directory. Recipients may be selected from a local copy of the healthcare provider directory. This system may include a tool for monitoring and reprocessing failed distributions of clinical documents. Jurisdictions may have one or more systems that perform this function.
- Clinical Document Viewing. An application in which users view clinical documents. Allows viewing of clinical documents, whether locally created, downloaded from the National eHealth Record, or received via P2P delivery. Jurisdictions may have one or more systems that perform this function.
- Patient Integration. Integration components for HL7 messages from PAS and EMPI systems. Integrates the PAS, EMPI, CIS and HIPS. Inserts the state/territory healthcare identifier and transforms PAS messages into the standardised format for HIPS. May allow EMPI and/or CIS to obtain the IHI from HIPS. Preferably implemented in an enterprise service bus.
- My Health Record Integration. Integration components for accessing My Health Record from clinical systems. Integrates the CIS with HIPS. May transform clinical documents to or from CDA. Preferably implemented in an enterprise service bus.
- PAS Loader Service. HIPS web service that accepts PAS HL7 messages. Stores patients and episodes in the HIPS Core database. This is an existing component of HIPS.
- Patient Participation Service. HIPS web services for determining which patients have a My Health Record. Allows clinical systems to query or subscribe to a feed of patients who are found to have a My Health Record. Also allows the disclosure of the existence of a hidden My Health Record. This is an existing component of HIPS.
- Consent Withdrawal Service. HIPS web services for managing the withdrawal of consent to upload documents to the My Health Record. Store and query a patient's withdrawal of consent to upload a document. This is an existing component of HIPS.



- Validated IHI Retrieval. Web service that returns the IHI for a specified patient. Checks whether the IHI was last validated more than the configured period of time in the past, and if so, revalidates the IHI before returning it. This is an existing component of HIPS.
- IHI Search & Validation. Business logic and service invoker for IHI Lookup Service. Searches for an IHI for each patient and revalidates the IHI when required. This is an existing component of HIPS.
- Digital Health Record Check Invoker. Service invoker for the 'Does PCEHR Exist' check. Each time a patient's IHI is obtained from the HI Service, or a new episode is created, invokes the My Health Record web service to check whether the patient has a My Health Record. This is an existing component of HIPS.
- Digital Health Record Access Service. HIPS web service that establishes access to the My Health Record for a specified patient. Checks whether the IHI was last validated more than the configured period of time in the past, and if so, revalidates the IHI before calling the My Health Record Gain Access service. This is an existing component of HIPS.
- Upload CDA Document Service. HIPS web service that accepts a document for upload to My Health Record. After checking the user, patient, episode, consent, age and IHI, sends document for CDA packaging and onto the operation queue to be uploaded. This is an existing component of HIPS.
- CDA Packaging. Business logic for packaging a CDA document. Creates an electronic signature for the document, using the NASH PKI Certificate for the Health Provider Organisation that authored the document, and creates the CDA package ZIP file using the attachments supplied and the logo image supplied or configured. This is an existing component of HIPS. Supports wrapping the CDA package in an HL7 v2 MDM wrapper to support sending to receivers who publish interaction records with an MDM service category.
- Operation Queue. Queue for operations that HIPS will take care of completing. Process queued upload or remove operations. Retry operations that encountered a temporary failure on the external system. Maintain the original order of operations that relate to the same document set. This is an existing component of HIPS.
- Remove Document from My Health Record service. HIPS web service that accepts a request to remove a document from the My Health Record. After checking the user, patient, episode and IHI, adds the request onto the operation queue. This is an existing component of HIPS.
- Document Upload Invoker. HIPS business logic and service invoker for My Health Record upload document service. Processes an upload operation from the queue. This is an existing component of HIPS.
- Remove Document Invoker. HIPS business logic and service invoker for My Health Record remove document service. Processes a remove operation from the queue. This is an existing component of HIPS.
- List Documents on Digital Health Record Service. HIPS web service that executes a document query against the My Health Record system. After checking the user, patient, and IHI, executes the query against the My Health Record system and returns the list of matching documents. This is an existing component of HIPS.

- Download Document from Digital Health Record Service. HIPS web service that downloads a document from the My Health Record system. After checking the user, patient, and IHI, requests the document from the My Health Record system, validates the CDA package, checks the patient demographics, and returns the document contents. This is an existing component of HIPS.
- Digital Health Record Change History Service. HIPS web service that obtains a list of all versions of a document from the My Health Record system. After checking the user, patient, and IHI, requests the change history view from the My Health Record system, and returns the list of document versions. This is an existing component of HIPS.
- HI Service Search for IHI Service. Web service for looking up a consumer's IHI given certain demographic details. Allows the search and validation of IHI numbers. This is an existing component of the HI Service.
- 'Does PCEHR Exist' Service. Web service for checking whether a consumer has a My Health Record that is visible to the accessing organisation. Determine the access permission for the organisation – no My Health Record or hidden, already gained access, can gain access with code, or can gain access without code. This is an existing component of the My Health Record B2B Gateway.
- My Health Record Gain Access Service. Web service for gaining access to a consumer's My Health Record. Adds the accessing organisation to the Provider Access List of the consumer's My Health Record. Sets the access permission according to the code that was supplied. This is an existing component of the My Health Record B2B Gateway.
- My Health Record Upload Document Service. My Health Record web service for uploading a document to a consumer's My Health Record. Validates and stores a document, optionally superseding an existing document. This is an existing component of the My Health Record B2B.
- My Health Record Remove Document Service. Web service for effectively removing a document that was previously uploaded to a consumer's My Health Record. Mark the document so that it is hidden from the consumer and other healthcare providers. This is an existing component of the My Health Record B2B.
- My Health Record Find Documents Service. Web service for listing the documents that match a query. List the documents that match the given query. This is an existing component of the My Health Record B2B.
- My Health Record Retrieve Document Service. Web service for retrieving a document from a patient's My Health Record. Retrieves the document with the specified unique ID and repository ID. This is an existing component of the My Health Record B2B Gateway.
- My Health Record Change History Service. Web service for listing the versions of a specified document. List the versions of the document with the specified entry UUID. This is an existing component of the My Health Record B2B Gateway.

### **2.1.2 P2P Addressing Components**

The P2P Addressing components provide a Local Health Service Directory (LHSD) for the purpose of addressing during the P2P messaging. A mechanism is provided

for maintaining and synchronising the LHSD from the National Health Service Directory (NHSD) and associated data sources exposed by the National Endpoint Proxy Service (NEPS).

The P2P Addressing components consist of:

- **Provider Directory Integration.** Integration components for the provider directory. Provides a search capability or a feed of updates to the provider directory in the CIS, so that the CIS user may select document recipients. Jurisdictions will develop integration components if required to integrate their clinical systems with HIPS.
- **Validated HPI-I Retrieval Service.** Web service that returns the HPI-I for a specified provider individual. Check whether the HPI-I was last validated more than the configured period of time in the past, and if so, revalidates the HPI-I before returning it.
- **HPI-I Search & Validation.** Business logic and service invoker for HPI-I Lookup Service. Allows the search and validation of HPI-I numbers.
- **Validated HPI-O Retrieval Service.** Web service that returns the HPI-O for a specified provider organisation. Check whether the HPI-O was last validated more than the configured period of time in the past, and if so, revalidates the HPI-O before returning it.
- **HPI-O Validation.** Business logic and service invoker for HPI-O validation in the NEPS. Allows the validation of HPI-O numbers.
- **LHSD Update Retrieval Service.** HIPS web service for internal systems to retrieve updates to the local provider directory (LHSD). Query the local provider directory database and return all records that have changed since a given point in time.
- **LHSD Search Service.** HIPS web service for internal systems to search the local provider directory (LHSD). Query the local provider directory database and return all records that match the query criteria.
- **NHSD Update Retrieval.** Service host to periodically query the NHSD to obtain any updates. Obtain the list of updated records from NHSD and apply the updates in the LHSD.
- **HIPS Directory Maintenance Web Interface.** A web interface provided with HIPS for maintaining the provider directory, both via manual configuration and via searching and subscribing to records in NEPS. Allows system administrators to add, modify, enable and disable items in the local provider directory (LHSD), including:
  - provider organisations
  - provider individuals
  - provider links
  - endpoint location services
  - interaction records
  - service providers
  - service categories
  - service interfaces

- Allows clinical support staff to add, modify or remove items from a NEPS subscription, to search the NEPS for individuals or organisations, and to add selected items from the result set to a current subscription.
- HIPS Directory Maintenance Services. HIPS web services for maintaining the provider directory, both via manual configuration and via searching and subscribing to records in NEPS. Add, modify, enable and disable items in the local provider directory (LHSD). Add, modify or remove items from a NEPS subscription. Search the NEPS for individuals or organisations. Add specific organisations to a NEPS subscription.
- NEPS Search. HIPS business logic that sends changes to NEPS that will affect the set of provider records that is included in the feed of updates from the NEPS to HIPS. Sends a request to NEPS to add, modify or remove a subscription item. Subscription items may be defined by geographical locations and specialties, or may be specific provider organisations.
- NEPS Subscription Maintenance. HIPS business logic that sends changes to NEPS that will affect the set of provider records that is included in the feed of updates from the NEPS to HIPS. Sends a request to NEPS to add, modify or remove a subscription item. Subscription items may be defined by geographical locations and specialties, or may be specific provider organisations.

### **2.1.3 P2P Document Delivery Components**

The P2P Document Delivery components provide data stores and services to support the secure delivery of messages to providers. Provide services and an administration user interface to support the management of messages sent via Secure Message Delivery (SMD).

The P2P Document Delivery components consist of:

- Send Integration. Integration components for sending discharge summaries from the CIS. Passes discharge summaries and other types of document into HIPS for delivery to the intended recipient. Notify the source system in case of failure to deliver. Inserts the national healthcare identifiers (IHI, HPI-I and HPI-O) if these are not handled by the CIS. Transforms discharge summary documents into the specification-conformant CDA format. Jurisdictions will develop integration components if required to facilitate the delivery of documents from clinical systems into HIPS
- Send CDA Document. HIPS web service that accepts a CDA document to be delivered using SMD. Validates the user, patient, episode, IHI, HPI-Is and HPI-Os in the document. Passes the document to the CDA packaging component.
- Send Any File Type Service. HIPS web service that accepts any file to be delivered using SMD. Responsibilities Passes the file to the SMD Addressing component.
- SMD Addressing. Business logic for addressing a sealed message. Looks up the preferred interaction record for delivering the message, using the priorities assigned to the service provider (message vendor) to decide which record to use. In the case of a retry, invokes the validate interaction record before continuing. Passes the message and interaction record to the signing and encryption component.

- **Interaction Record Validation.** Business logic and service invoker for validating an interaction record using the NEPS interface. Allows the validation of interaction records. Interaction records are considered valid until their use fails.
- **Signing and Encryption.** Business logic for signing and encrypting a message. Responsibilities Signs the message payload using the sender organisation certificate. Invokes the certificate validation component for the receiving organisation certificate. Encrypts the signed payload and sends the encrypted payload and interaction record to the Send Message component.
- **Certificate Validation.** Business logic for validating the NASH PKI Certificate for a Healthcare Provider Organisation. If the CRL has expired, retrieves the latest CRL from the CRL distribution point. Validates the certificate using the current CRL. Checks the identity of the certificate matches the HPI-O of the organisation to which it is expected to belong.
- **Message Delivery.** Service host to periodically query for incoming transport responses. Invokes the retrieve method of the TRR service on the Sender Intermediary component in the DMZ. Passes the transport response to the Response Validation component. After processing of the transport response is complete, invokes the remove method of the TRR service.
- **Response Validation.** Business logic to validate transport responses. Stores the transport response, checks that the digest matches the digest of the sent message, and updates the status of the sent message.
- **Delivery Retry Controller.** HIPS service host that periodically retries the delivery of messages. Checks for expired sealed messages for which no final transport response has been received. Passes these to the SMD addressing component which will validate the interaction record before trying to send again.
- **Delivery Status Query.** Web service for internal systems to check the delivery status, and to abort delivery of, any messages that HIPS has been directed to send. Responsibilities Lists or aborts the messages matching the given query.
- **Delivery Status Web Interface.** Monitoring interface that shows delivery status of outbound messages. Allows system administrators to query the status of any sent messages, retry the delivery or abort the delivery.
- **HIPS Sender Intermediary.** Internet-facing secure messaging web service components for sending messages. Hosts an inward facing SMD service for HIPS to provide a sealed message, and forwards the message to its external destination. Hosts an outward facing TRD service that accepts authenticated TLS connections from external receivers, and stores the incoming transport responses on the local file system. Hosts an inward facing TRR service for HIPS to retrieve and remove the stored transport responses. This is an existing component of the Agency's Secure Message Delivery B2B Client Library (SMD Library) sample code which will be provided as a component to be installed with HIPS.

#### **2.1.4 P2P Document Receiving Components**

The P2P Document Receiving Components provide data stores and services to support the secure receipt of messages from providers. Provide services and an

administration user interface to support the management of messages received via SMD.

The P2P Document Receiving components consist of:

- **HIPS Receiver Intermediary.** Internet-facing secure messaging web service components for receiving messages. Hosts an outward facing SMD service that accepts authenticated TLS connections from external senders, and stores the incoming encrypted sealed messages on the local file system. Hosts an inward facing SMR service for the Decrypter to retrieve the stored sealed messages. Hosts an inward facing TRD service for HIPS to remove a stored sealed message and forward the transport response to its external destination. This is an existing component of the SMD Library.
- **HIPS Message Decryption.** Single-purpose web service for decrypting incoming sealed messages. When HIPS makes a request to retrieve messages, this component makes a request to the Receiver Intermediary to retrieve encrypted messages, decrypts those messages, and returns the decrypted messages. This is an existing component of the SMD Library.
- **Message Retrieval.** Service host to periodically query for incoming messages. Invokes the retrieve method of the UMR service on the Decrypter component in the DMZ. Passes the decrypted message to the signature validation component.
- **Signature Validation.** Business logic to validate the signature on received messages. Invokes the certificate validation component for the sending organisation certificate. Checks that the signature was created correctly. Passes the message metadata and digest to the Send Response component.
- **Message Storage.** Business logic to store the received messages. Links the received message to a patient record where possible. Stores the message into the HIPS database.
- **Response Delivery.** Business logic and service invoker to deliver the transport response via the Receiver Intermediary component. Invokes the inward facing TRD service on the Receiver Intermediary component in the DMZ, which will remove the message and forward the response on to the message sender.
- **Received Message Push Invoker.** Business logic and service invoker to deliver received messages to the ESB. Delivers all received messages to a single web service interface that is hosted by the jurisdiction's Enterprise Service Bus and conforms to the HIPS web service interface specification.
- **Received Message Query Service.** Web service for internal systems to list, retrieve and remove any messages that have been received by HIPS. Lists, retrieves or removes the messages matching the given query.
- **Receive Monitoring Web Interface.** Monitoring interface that shows inbound messages. Allows system administrators to query the status of any received messages, retry or abort the delivery of the transport response, and remove the messages from the system.
- **Receive Integration.** Integration components for routing received messages to the CIS. Regularly invokes the received message query to obtain recently received messages and pass them to the clinical document viewing system. Jurisdictions may consider implementing this component to assist

with routing received documents to the systems where they may be viewed.

## 2.2 Business Logic

### 2.2.1 P2P Messaging Services

The P2P Messaging Services support the delivery and management of messages sent via Secure Message Delivery (SMD) or the receipt and management of messages received via SMD.

#### 2.2.1.1 Secure Message Delivery Service

HIPS P2P provides web services to support the sending of messages via Secure Message Delivery (SMD). The *MessageDeliveryService* provides services to support retrieving and actioning messages sent via SMD. The following describes the operations for the *MessageDeliveryService* in more detail

##### ***SendMessage***

This service accepts a new binary or CDA message to be delivered via SMD.

When provided with a binary payload (e.g. legacy HL7v2 message) HIPS can forward this to a SMD compliant provider. It will perform validation of the sending and receiving organisation, but it cannot validate the contents of the payload, so there will be no validation of the patient or the author.

When provided with a CDA payload, HIPS will not send the raw XML document, but according to the payload scheme (service category) in the receiver's interaction record, it will first perform packaging to either XDM-ZIP format and then if required add further wrapping to the HL7 MDM format that is required by most GP desktop software. HIPS can perform further validation if the payload format is CDA, including validation of the patient's identity and the author's identity.

Most software that is SMD compliant is also CDA compliant, and can handle CDA documents, so you may choose to send all documents in CDA format. Legacy file formats, such as HL7v2 or PIT can also be used.

The technical limit for the final encrypted message size in HIPS P2P will be approximately 1 GB. Due to the overhead of the XML representation, this may limit the raw payload to approximately 700 MB. The receiving clinical software and the messaging vendors (intermediaries) in the path between HIPS P2P and the receiving organisation may impose their own limits on message size.

##### ***Logic:***

- 1 Validate the request message. If the request message is not valid, return an error.
- 2 Locate and validate the Sender:
  - a If the *MessageAddressee.Organisation.Identifier.Type.Code* is not "HPIO", retrieve the mapped HPI-O value from the *p2p.ProviderOrganisationIdentifier* table in the LHSD data store based on the *Type.Code* and *Value* provided in the request. If the HPI-O value cannot be obtained, return a fault.
  - b If provided, and the *MessageAddressee.Individual.Identifier.Type.Code* is not "HPII", retrieve the mapped HPI-I value from the

- p2p.ProviderIndividualIdentifier* table in the LHSD data store based on the *Type.Code* and *Value* provided in the request. If the HPI-I value cannot be obtained, return a fault.
- c Return a fault if the Sender HPI-O is not found in the response from HIPS Core ListHospitals.
- d Return a fault if unable to locate the Sender HPI-O certificate from the machine store using the PcehrCertSerial in the response from HIPS Core ListHospitals.
- e Return a fault if the certificate subject does not identify the Sender Organisation HPI-O.
- f Return a fault if the certificate policy OID is not that of a NASH PKI Certificate for Healthcare Provider Organisations.
- g Return a fault if the certificate has an invalid chain of trust, is expired or revoked.
- h Return a fault if the certificate private key is not available.
- i Invoke HIPS P2P GetValidatedHpII for the Sender HPI-I if provided.
- j Invoke HIPS P2P GetValidatedHpio for the Sender HPI-O.
- k Return a fault if there are unresolved alerts on either the HPI-I or HPI-O.
- 3 Locate and validate the Receiver:
  - a If the MessageAddressee.Organisation.Identifier.Type.Code is not "HPIO", retrieve the mapped HPI-O value from the *p2p.ProviderOrganisationIdentifier* table in the LHSD data store based on the *Type.Code* and *Value* provided in the request. If the HPI-O value cannot be obtained, return a fault.
  - b If provided, and the MessageAddressee.Individual.Identifier.Type.Code is not "HPII", retrieve the mapped HPI-I value from the *p2p.ProviderIndividualIdentifier* table in the LHSD data store based on the *Type.Code* and *Value* provided in the request. If the HPI-I value cannot be obtained, return a fault.
  - c Return a fault if the Receiver HPI-O is not found in the LHSD or not active.
  - d Return a fault if the Receiver Organisation is not active.
  - e Invoke HIPS P2P GetValidatedHpII for the Receiver HPI-I if provided.
  - f Invoke HIPS P2P GetValidatedHpio for the Receiver HPI-O.
  - g Return a fault if there are unresolved alerts on either the HPI-I or HPI-O.
- 4 If the ContentPackage is a BinaryMessageContentPackage:
  - a Return a fault if unable to locate a Payload Scheme using the qualified identifier.
  - b Return a fault if unable to locate any active (see common logic) Interaction Records using the Receiver Organisation and Payload Scheme, where the Interface is SMD.
  - c Choose the Interaction Record with the highest priority Delegate.
- 5 If the ContentPackage is a CdaMessageContentPackage:
  - a Return a fault if unable to parse the Payload as an XML document.



- b If the IHI Validation Flag is true, validate the patient demographics and IHI in the document via a new HIPS *IHIService* operation *ValidateSendingPatient* that implements applicable IHI conformance requirements for use case UC.330 "Sending an eHealth Message".
  - c Invoke HIPS P2P *GetValidatedHpii* for the Author HPI-I if provided in the document.
  - d Invoke HIPS P2P *GetValidatedHpio* for the Author HPI-O in the document.
  - e Return a fault if there are unresolved alerts on the IHI, HPI-I or HPI-O.
  - f Return a fault if unable to locate a Document Type using the code in the document.
  - g Return a fault if unable to locate any active (see common logic) Interaction Records using the Receiver Organisation and Document Type, where the Interface is SMD and the Packaging Format is either XDM-ZIP or HL7-MDM.
  - h Choose the Interaction Record with the highest priority Delegate and highest priority Payload Scheme, with Delegate taking precedence.
  - i Package the document and attachments using the Packaging Format associated with the Payload Scheme of the chosen Interaction Record. Packaging will create a signature using the NASH certificate of the Author HPI-O in the document.
- 6 Allocate a new invocation identifier UUID URN for the first delivery attempt.
  - 7 Create and persist the *MessagePayload* and *SealedMessage* records.
  - 8 Return the summary of the message that has been queued for delivery.

*Common Logic:*

- 1 An interaction record is considered active if all of the following conditions hold:
  - a The delegate is active.
  - b The payload scheme is active.
  - c The target organisation's HPI-O is active.
  - d The certificate can be parsed as an X.509 v3 certificate.
  - e The certificate subject identifies the HPI-O of the target organisation.
  - f The certificate policy OID is that of a NASH PKI Certificate for Healthcare Provider Organisations.
  - g The certificate has a valid chain of trust, and is neither expired nor revoked.

**QueryMessages**

Lists outbound messages delivered via SMD matching specified criteria.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return an error.
- 2 Identify results matching the specified criteria. Each top-level criterion is combined with AND – to be included, the result must match every top-level criterion specified. Each criterion within a top-level criterion is combined with OR – to be included, the result must match any of the specified criteria.

- a Include messages from the *p2p.SealedMessage* table with a value for the *MessageDirectionId* matching the "Outbound" member of the *MessageDirection* enumeration.
- b If a value is provided for the *InvocationIdentifiers* property of the request, include messages from the *p2p.SealedMessage* table with a value for the *InvocationIdentifier* column matching any of the values specified in the request.
- c If a value is provided for the *Senders* property of the request, include messages from the *p2p.MessagePayload* table where:
  - i If a value has been provided for the *MessageAddressee.Organisation.Identifier* property of the request, the record in the *p2p.ProviderOrganisationIdentifier* table associated with the *SenderOrganisationId* in the *p2p.MessagePayload* table or the de-qualified *SenderOrganisation* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.Identifier.Value* (and *Type.Code*) of the *Sender* specified in the request.
  - ii If a value has been provided for the *MessageAddressee.Organisation.OrganisationName* property of the request, the *p2p.ProviderOrganisation.OrganisationName* associated with the *SenderOrganisationId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.OrganisationName* of the *Sender* specified in the request.
  - iii If a value has been provided for the *MessageAddressee.Individual.Identifier* property of the request, the record in the *p2p.ProviderIndividualIdentifier* table associated with the *SenderIndividualId* in the *p2p.MessagePayload* table or the de-qualified *SenderIndividual* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Identifier.Value* (and *Type.Code*) of the *Sender* specified in the request.
  - iv If a value has been provided for the *MessageAddressee.Individual.Name* property of the request, the *p2p.ProviderIndividualName.FamilyName* associated with the *SenderIndividualId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Name.FamilyName* of the *Sender* specified in the request.
- d If a value is provided for the *Receivers* property of the request, include messages from the *p2p.MessagePayload* table where:
  - i If a value has been provided for the *MessageAddressee.Organisation.Identifier* property of the request, the record in the *p2p.ProviderOrganisationIdentifier* table associated with the *ReceiverOrganisationId* in the *p2p.MessagePayload* table or the de-qualified *ReceiverOrganisation* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.Identifier.Value* (and *Type.Code*) of the *Receiver* specified in the request.
  - ii If a value has been provided for the *MessageAddressee.Organisation.OrganisationName* property of the request, the *p2p.ProviderOrganisation.OrganisationName* associated with the *ReceiverOrganisationId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.OrganisationName* of the *Receiver* specified in the request.

- iii If a value has been provided for the *MessageAddressee.Individual.Identifier* property of the request, the record in the *p2p.ProviderIndividualIdentifier* table associated with the *ReceiverIndividualId* in the *p2p.MessagePayload* table or the de-qualified *ReceiverIndividual* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Identifier.Value* (and *Type.Code*) of the *Receiver* specified in the request.
    - iv If a value has been provided for the *MessageAddressee.Individual.Name* property of the request, the *p2p.ProviderIndividualName.FamilyName* associated with the *ReceiverIndividualId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Name.FamilyName* of the *Receiver* specified in the request.
  - e If a value is provided for the *MessageStatuses* property of the request, include messages from the *p2p.SealedMessage* table with a value for the *MessageStatusId* column matching any of the values specified in the request.
  - f If a value is provided for the *DeliveryPeriod* property of the request:
    - i If a value is not specified for the *Start* property of the *DeliveryPeriod*, default the *Start* to the start of time.
    - ii If a value is not specified for the *End* property of the *DeliveryPeriod*, default the *End* to the end of time.
    - iii Include messages from the *p2p.SealedMessage* table where the *CreationTime* is between the *Start* and *End* values specified.
  - g If a value is provided for the *PayloadSchemes* property of the request, include messages from the *p2p.SealedMessage* table with an associated record in the *p2p.InteractionRecord* table that has a payload scheme *QualifiedIdentifier* (via *p2p.PayloadScheme*) matching any of the values specified in the request.
  - h If a value is provided for the *DocumentTypes* property of the request, include messages from the *p2p.SealedMessage* table with an associated record in the *p2p.InteractionRecord* table that has a document type *Code* (via *p2p.PayloadScheme* and *p2p.DocumentType*) matching any of the values specified in the request.
- 3 Construct and return a *QueryMessagesResponse* containing the results.

### **RetryMessage**

Retries the delivery of a specified outbound message that has failed to be delivered.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return an error.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - i. Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - ii. Return a fault if the *MessageDirectionId* does not match the *Outbound* member of the *MessageDirection* enumeration.

- 3 Retrieve all Sealed Messages for the same *MessagePayloadId* from the *p2p.SealedMessage* table.
  - a Return a fault if the Message Status of any of these Sealed Messages is not "Removed", "Expired", "Failed", "Undeliverable – Exceeded Retry Count" or "Undeliverable – Expired". It should not be possible to request retry if delivery was completed, or is still in progress.
- 4 Create a new Sealed Message that references the existing Message Payload.
  - a Set the Invocation Identifier in the new Sealed Message to a new UUID URN.
  - b Set the Message Status in the new Sealed Message to Undelivered.
  - c Set the Expiry Time in the new Sealed Message to the *Expiry* property of the request.
  - d Persist the new Sealed Message.
- 5 Construct and return a *RetryMessageResponse* containing a summary of the new Sealed Message.

### **AbortMessage**

Aborts the delivery of a specified outbound message. Sets the status of all of the sealed messages for the message payload to "Removed".

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Outbound" member of the *MessageDirection* enumeration.
- 3 Retrieve all Sealed Messages for the same *MessagePayloadId* from the *p2p.SealedMessage* table.
  - a Return a fault if all the messages are already aborted, that is, the *MessageStatusId* of all retrieved Sealed Messages matches the "Removed" member of the *MessageStatus* enumeration.
- 4 Update the *MessageStatusId* of all the retrieved Sealed Messages to match the "Removed" member of the *MessageStatus* enumeration.
- 5 Construct and return an *AbortMessageResponse*.

### **GetMessage**

Retrieves the binary content of a specified outbound message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.

- a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Outbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the message contents from the corresponding records in the *p2p.MessagePayload* and *p2p.PayloadScheme* tables.
- 4 Construct and return a *GetMessageResponse* containing a *BinaryMessageContent* instance based on the *Contents* column in the database.

### **GetCdaMessage**

Retrieves the CDA content of a specified outbound message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Outbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the message contents from the corresponding records in the *p2p.MessagePayload* and *p2p.PayloadScheme* tables.
- 4 If the *PayloadPackagingId* of the Payload Scheme associated with the message is not either the "xdmZip" or "hl7Mdm" member of the *PayloadPackaging* enumeration, return an error.
- 5 Invoke the HIPS *UnpackageCdaDocument* service to extract the CDA document and attachments from the package.
- 6 Construct and return a *GetCdaMessageResponse*.

### **2.2.1.2 Secure Message Receipt Service**

HIPS P2P provides web services to support the retrieving and actioning of messages received via SMD. The following describes the operation that are provided in the *MessageReceiptService* service.

### **QueryMessages**

Lists inbound messages received via SMD matching specified criteria.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return an error.
- 2 Identify results matching the specified criteria. Each top-level criterion is combined with AND – to be included, the result must match every top-level criterion specified. Each criterion within a top-level criterion is combined with OR – to be included, the result must match any of the specified criteria.

- a Include messages from the *p2p.SealedMessage* table with a value for the *MessageDirectionId* matching the "Inbound" member of the *MessageDirection* enumeration.
- b If a value is provided for the *Senders* property of the request, include messages from the *p2p.MessagePayload* table where:
  - i If a value has been provided for the *MessageAddressee.Organisation.Identifier* property of the request, the record in the *p2p.ProviderOrganisationIdentifier* table associated with the *SenderOrganisationId* in the *p2p.MessagePayload* table or the de-qualified *SenderOrganisation* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.Identifier.Value* (and *Type.Code*) of the *Sender* specified in the request.
  - ii If a value has been provided for the *MessageAddressee.Organisation.OrganisationName* property of the request, the *p2p.ProviderOrganisation.OrganisationName* associated with the *SenderOrganisationId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.OrganisationName* of the *Sender* specified in the request.
  - iii If a value has been provided for the *MessageAddressee.Individual.Identifier* property of the request, the record in the *p2p.ProviderIndividualIdentifier* table associated with the *SenderIndividualId* in the *p2p.MessagePayload* table or the de-qualified *SenderIndividual* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Identifier.Value* (and *Type.Code*) of the *Sender* specified in the request.
  - iv If a value has been provided for the *MessageAddressee.Individual.Name* property of the request, the *p2p.ProviderIndividualName.FamilyName* associated with the *SenderIndividualId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Name.FamilyName* of the *Sender* specified in the request.
- c If a value is provided for the *Receivers* property of the request, include messages from the *p2p.MessagePayload* table where:
  - i If a value has been provided for the *MessageAddressee.Organisation.Identifier* property of the request, the record in the *p2p.ProviderOrganisationIdentifier* table associated with the *ReceiverOrganisationId* in the *p2p.MessagePayload* table or the de-qualified *ReceiverOrganisation* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.Identifier.Value* (and *Type.Code*) of the *Receiver* specified in the request.
  - ii If a value has been provided for the *MessageAddressee.Organisation.OrganisationName* property of the request, the *p2p.ProviderOrganisation.OrganisationName* associated with the *ReceiverOrganisationId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Organisation.OrganisationName* of the *Receiver* specified in the request.
  - iii If a value has been provided for the *MessageAddressee.Individual.Identifier* property of the request, the record in the *p2p.ProviderIndividualIdentifier* table associated with the *ReceiverIndividualId* in the *p2p.MessagePayload* table or the de-

- qualified *ReceiverIndividual* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Identifier.Value* (and *Type.Code*) of the *Receiver* specified in the request.
- iv If a value has been provided for the *MessageAddressee.Individual.Name* property of the request, the *p2p.ProviderIndividualName.FamilyName* associated with the *ReceiverIndividualId* in the *p2p.MessagePayload* table matches the *MessageAddressee.Individual.Name.FamilyName* of the *Receiver* specified in the request.
  - d If a value is provided for the *MessageStatuses* property of the request, include messages from the *p2p.SealedMessage* table with a value for the *MessageStatusId* column matching any of the values specified in the request.
  - e If a value is provided for the *ReceiptPeriod* property of the request:
    - i If a value is not specified for the *Start* property of the *ReceiptPeriod*, default the *Start* to the start of time.
    - ii If a value is not specified for the *End* property of the *ReceiptPeriod*, default the *End* to the end of time.
    - iii Include messages from the *p2p.SealedMessage* table where the *CreationTime* is between the *Start* and *End* values specified.
  - f If a value is provided for the *PayloadSchemes* property of the request, include messages from the *p2p.MessagePayload* table with an associated record in the *p2p.PayloadScheme* table that has a payload scheme *QualifiedIdentifier* matching any of the values specified in the request.
  - g If a value is provided for the *DocumentTypes* property of the request, include messages from the *p2p.MessagePayload* table with an associated record in the *p2p.PayloadScheme* table that has an associated record in the *p2p.DocumentType* table that has a document type *Code* matching any of the values specified in the request.
- 3 Construct and return a *QueryMessagesResponse* containing the results.

### **GetMessage**

Retrieves the binary content of a specified inbound message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Inbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the message contents from the corresponding records in the *p2p.MessagePayload* and *p2p.PayloadScheme* tables.
- 4 If the *MarkAsRead* parameter is set to "true", set the *MessageStatus* to "Retrieved".

- 5 Construct and return a *GetMessageResponse* containing a *BinaryMessageContent* instance based on the *Contents* column in the database.

### **GetCdaMessage**

Retrieves the CDA content of the specified inbound message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Inbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the message contents from the corresponding records in the *p2p.MessagePayload* and *p2p.PayloadScheme* tables.
- 4 If the *PayloadPackagingId* of the Payload Scheme associated with the message is not either the "xdmZip" or "hl7Mdm" member of the *PayloadPackaging* enumeration, return an error.
- 5 Invoke the HIPS *UnpackageCdaDocument* service to extract the CDA document and attachments from the package.
- 6 If the *MarkAsRead* parameter is set to "true", set the *MessageStatus* to "Inbound Retrieved".
- 7 Construct and return a *GetCdaMessageResponse*.

### **RetryResponse**

Retries the delivery of the transport response for a specified inbound message that has failed to be delivered.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Inbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the Transport Response associated with the Sealed Message from the *p2p.TransportResponse* table.
  - a Return a fault if there is not exactly one transport response.
  - b Return a fault if the *TransportResponseStatusId* does not match the "Inbound Expired" or "Inbound Aborted" member of the *TransportResponseStatus* enumeration.



- 4 Update the *p2p.TransportResponse* table to set the *TransportResponseStatusId* of the transport response to the value of the "Inbound Undelivered" member of the *TransportResponseStatus* enumeration.
- 5 Construct and return a *RetryResponseResponse*.

### **AbortResponse**

Aborts the delivery of the transport response for a specified inbound message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Inbound" member of the *MessageDirection* enumeration.
- 3 Retrieve the undelivered Transport Response associated with the sealed message from the *p2p.TransportResponse* table.
  - a Return a fault if there is not exactly one transport response associated with the sealed message.
  - b Update the record in the *p2p.TransportResponse* table to set its *TransportResponseStatusId* to the value of the "Inbound Aborted" member of the *TransportResponseStatus* enumeration.
  - c Determine if the transport response is for a valid message based on *ResponseClass* = "Successful".
  - d If *p2p.SealedMessage.MessageStatus* = "Unacknowledged":
    - i If valid, set *p2p.SealedMessage.MessageStatus* = "Unpublished".
    - ii Otherwise, set *p2p.SealedMessage.MessageStatus* = "Failed".
- 4 Construct and return an *AbortResponseResponse*.

### **RemoveMessage**

Removes a specified inbound message. This removal is an acknowledgement by internal systems that HIPS need not perform any further processing of the message.

*Logic:*

- 1 Validate the request message. If the request message is not valid, return a fault.
- 2 Retrieve the Sealed Message from the *p2p.SealedMessage* table based on the *InvocationIdentifier* in the request.
  - a Return a fault if no record representing the message exists in the *p2p.SealedMessage* table.
  - b Return a fault if the *MessageDirectionId* does not match the "Inbound" member of the *MessageDirection* enumeration.

- c Return a fault if the *MessageStatusId* does not match the "Expired", "Unpublished", "Retrieved", "Published" or "Failed" member of the *MessageStatus* enumeration.
- 3 Modify the existing record in the *p2p.SealedMessage* table to set its *MessageStatusId* to the value of the "Removed" member of the *MessageStatus* enumeration.
- 4 Construct and return a *RemoveMessageResponse*.

## 2.2.2 Secure Message Delivery Scheduled Services

Scheduled services operate on a configurable timer interval to perform various tasks that need to be performed at regular intervals. They are implemented using Windows Communication Foundation (WCF) service hosts so that the scheduled services and web services will execute within the same application. Scheduled services are typically consumers of web services that are hosted by other applications. Scheduled services invoke services on the HIPS Core, NEPS, HIPS SMD Server, HIPS SMD Decrypter and jurisdiction's other applications or ESB that accept message publishing

### 2.2.2.1 Scheduled Message Delivery Service

The *ScheduledMessageDeliveryService* provides a scheduled service operation in HIPS Core to periodically assess and action messages to be delivered via SMD.

It is implemented as a scheduled service that executes at a configurable interval to retrieve messages from the P2P database, validate interaction records via the NEPS, attempt delivery via the HIPS SMD Server, and update the message in the P2P database.

The *ScheduledMessageDeliveryService* is implemented as follows:

- 1 Commence processing after configured interval.
- 2 Retrieve outbound messages with status "Undelivered" or "Delivered" from P2P database.
- 3 For each message retrieved:
  - a If the message status is "Undelivered" and the current time is after the *ExpiryTime* of the message, change its status to "Expired", log that the message was not delivered by the expiry time, and do not proceed further for this message.
  - b If the message status is "Delivered":
    - i If the current time is after the *ExpiryTime* of the message, change its status to "Expired" and log that a final transport response for the message was not received by the expiry time.
    - ii Do not proceed further for this message.
  - c Validate the sender and receiver organisation's HPI-O via NEPS unless the HPI-O is present in the LHSD and marked as validated within the past 24 hours.
  - d If the sender or receiver individual is identified by an HPI-I, validate the HPI-I via HIPS HPI-I search unless the HPI-I is present in the LHSD and marked as validated within the past 24 hours.

- e If any of the above HPI-O or HPI-I identifiers have unresolved alerts or fail validation, persist a warning transport response for this message, with a detailed description of the issue, and do not proceed further for this message.
- f If a warning transport response for this message exists, this is not the first attempt to deliver the message, so validate and refresh the interaction records for the receiver organisation via NEPS and persist the new interaction records into the LHSD database.
- g Look up the preferred interaction record for delivering the message, using the priorities assigned to the delegates (message vendors) and secondarily payload schemes to decide which record to use.
- h Load the receiver organisation certificate designated for payload encryption usage from the preferred interaction record.
- i Load the sender organisation certificate and ensure that the corresponding private key is available. This certificate can be obtained from the machine certificate store using the cached results from calling the GetHospitals operation on the HIPS ReferenceService and selecting the PcehrCertSerial property from a hospital whose HPI-O matches the sender organisation's HPI-O.
- j If the cached CRL has expired, retrieve the latest CRL from the CRL distribution point. Check that the sender organisation certificate and the receiver organisation certificate are valid through chain trust to a trusted root certificate, contain the policy OID for "NASH PKI Certificate for Healthcare Provider Organisations", have not expired and are not revoked, and that the identity of each certificate matches the HPI-O of the organisation to which it is expected to belong.
- k If either of the certificates fail the above validation, persist a warning transport response for this message, with a detailed description of the issue, and do not proceed further for this message.
- l If the packaging format of the preferred interaction record is different to the packaging format of the interaction record currently associated with the message, un-package the message using the packaging format of the interaction record currently associated with the message and re-package the message using the packaging format of the newly selected interaction record. Set the InteractionRecordId, PayloadSchemeId and Content properties of the SealedMessage and MessagePayload records.
- m Use the SMD Library sample code to construct the sealed message metadata, sign the secure message payload using the sending organisation's certificate and encrypt the signed payload using the receiving organisation's certificate. Set the DigestValue property in the SealedMessage record.
- n If there is an external Sender Intermediary configured, deliver the sealed message via the HIPS SMD Server to the endpoint address of the external sender intermediary, otherwise, deliver the sealed message via the HIPS SMD Server to the endpoint address in the chosen interaction record.
- o If an "ok" or "duplicate" response is received, set the status of the message to "Delivered". Otherwise, persist a warning transport response with a detailed description of the issue.

- p Update the sealed message and message payload tables in the P2P database with the newly selected interaction record, payload scheme, payload, digest value and status.
- 4 After all messages have been processed, sleep until the next configured interval has elapsed.

### 2.2.2.2 Scheduled Response Receipt Service

The *ScheduledResponseReceiptService* provides a scheduled service operation in HIPS Core to periodically retrieve and action received transport responses for messages sent via SMD.

It is implemented as a custom WCF service host that executes at a configurable interval to retrieve transport responses from the HIPS SMD Server, validate and match them with the outbound message to which they relate in the P2P database, and subsequently remove them from the HIPS SMD Server.

The *ScheduledResponseReceiptService* is implemented as follows:

- 1 Commence processing after configured interval.
- 2 Invoke the [retrieve](#) service operation of the Transport Response Retrieval service provided by the HIPS SMD Server with a value of 0 for the *limit* parameter to retrieve a count of all available transport responses.
- 3 Invoke the [retrieve](#) service operation of the Transport Response Retrieval service provided by the HIPS SMD Server with the value of the *limit* parameter set to the *totalNumberAvailable* property in the response from the preceding invocation to retrieve all available transport responses.
- 4 For each transport response retrieved:
  - a Insert data into the *p2p.TransportResponse* and *p2p.TransportResponseMetadata* tables based on the *deliveryResponse* and *metadata* properties of the transport response.
  - b Identify the corresponding record representing the delivered sealed message in the *p2p.SealedMessage* table based on the following criteria:
    - i InvocationIdentifier = metadata.invocationId
  - c If a record is found:
    - i Compare the digest value in the *deliveryResponse.digestValue* property of the transport response to the value of the *p2p.SealedMessage.DigestValue* column.
    - ii Identify whether there are any *p2p.TransportResponse* records associated with the *p2p.SealedMessage* record with *final* = "true" (apart from the current transport response).
    - iii If not, update the value of the *p2p.SealedMessage.MessageStatusId* column based on the contents of the transport response:
      - If *final* = "false", set to "Delivered"
      - If *deliveryResponse.responseClass* = "Information" or "Warning", set to "Delivered"
      - If *deliveryResponse.responseClass* = "Error" or the digest values do not match, set to "Failed"

- If *deliveryResponse.responseClass* = "Success" and digest values match and *final* = "true", set to "Acknowledged"
  - d Invoke the [remove](#) service operation of the Transport Response Retrieval service provided by the HIPS SMD Server to remove the current transport response based on the following:
    - i *responseId* = *metadata.responseId*
  - e Move to the next transport response.
- 5 After all transport responses have been processed, sleep until the next configured interval has elapsed.

### 2.2.3 Secure Message Receipt Scheduled Services

The Secure Message Receipt scheduled services in HIPS -Core provides services to manage messages received via SMD. There are three scheduled services, a Message Receipt Service that manages messages received via SMD; Response Delivery Service that manages transport responses for messages received via SMD and the Message Publisher Service that publishes messages received via SMD.

#### 2.2.3.1 Message Receipt Service

The *ScheduledMessageReceiptService* provides a scheduled service operation in HIPS Core to periodically retrieve and action messages received via SMD.

It is implemented as a custom WCF service host that executes at a configurable interval to retrieve messages from the HIPS SMD Decrypter Server, validate the received message and create a transport response to be delivered by the response delivery service.

The *ScheduledMessageReceiptService* is implemented as follows:

- 1 Commence processing after configured interval.
- 2 Invoke the [retrieve](#) service operation of the Unsealed Message Retrieval service provided by the HIPS SMD Decrypter Server to retrieve all available messages up to the configured Message Retrieval Limit.
- 3 For each message retrieved:
  - a Load the receiver organisation certificate from the machine certificate store using the results from calling the GetHospitals operation on the HIPS ReferenceService and selecting the PcehrCertSerial property from a hospital whose HPI-O matches the receiver organisation's HPI-O. The results may be cached in the resource access layer with a configurable cache expiry timeout.
  - b Load the sender organisation certificate from the signedPayload XML path `"/sp:signedPayload/sp:signatures/ds:Signature/ds:KeyInfo/ds:X509Data/ds:X509Certificate"`.
  - c Use the [SignedXml](#) class to validate the electronic signature in the signedPayload of the message.
  - d If the cached CRL has expired, retrieve the latest CRL from the CRL distribution point. Check that the sender organisation certificate and the receiver organisation certificate are valid through chain trust to a trusted root certificate, contain the policy OID for "NASH PKI Certificate for Healthcare Provider Organisations", have not expired and are not revoked,

- and that the identity of each certificate matches the HPI-O of the organisation to which it is expected to belong.
- e Validate the sender and receiver organisation's HPI-O via NEPS unless the HPI-O is present in the LHSD and marked as validated within the past 24 hours.
- f If the sender or receiver individual are identified by an HPI-I, validate the HPI-I via HIPS HPI-I search unless the HPI-I is present in the LHSD and marked as validated within the past 24 hours.
- g Look up the Payload Scheme whose *QualifiedIdentifier* matches the *metadata.serviceCategory* of the message. If the *PayloadPackagingId* matches the "XDM-ZIP" or "HL7-MDM" member of the *PayloadPackaging* enumeration:
  - i Invoke the HIPS Core *UnpackageCdaDocument* method to un-package the CDA document.
  - ii Extract the author and patient information from the CDA document.
  - iii Validate the author organisation's HPI-O via NEPS unless the HPI-O is present in the LHSD and marked as validated within the past 24 hours.
  - iv If the author individual is identified by an HPI-I, validate the HPI-I via HIPS HPI-I search unless the HPI-I is present in the LHSD and marked as validated within the past 24 hours.
  - v If the IHI Validation Flag is true, validate the patient demographics and IHI via a new HIPS *IHIService* operation *ValidateReceivedPatient* that implements applicable IHI conformance requirements for use case UC.325 "Receiving an eHealth message".
- h Create a record in the *p2p.MessagePayload* table with properties as follows:
  - i Set *PayloadSchemeId* to the *PayloadSchemeId* of the payload scheme obtained above.
  - ii Set *ReceiverOrganisationUri* to the *metadata.receiverOrganisation* of the message.
  - iii Set *SenderOrganisationUri* to the *metadata.senderOrganisation* of the message.
  - iv Set *ReceiverIndividualUri* to the *metadata.receiverIndividual* of the message.
  - v Set *SenderIndividualUri* to the *metadata.senderIndividual* of the message.
  - vi Set *ReceiverOrganisationId* to the ID of the receiver organisation if it exists in the LHSD, otherwise null.
  - vii Set *SenderOrganisationId* to the ID of the sender organisation if it exists in the LHSD, otherwise null.
  - viii Set *ReceiverIndividualId* to the ID of the receiver individual if it exists in the LHSD, otherwise null.
  - ix Set *SenderIndividualId* to the ID of the sender individual if it exists in the LHSD, otherwise null.
  - x Set *Contents* to the base-64 decoded value of the signedPayload XML path

- "/sp:signedPayload/sp:signedPayloadData/msg:message/msg:data" if the "msg:message" element exists, otherwise to the inner XML of the "sp:signedPayload/sp:signedPayloadData" element.
- xi Set *SexId* to the value of the member of the Sex enumeration corresponding to the patient sex if the message was unpackaged to a CDA document, otherwise null.
  - xii Set *Ihi* to the patient IHI if the message was unpackaged to a CDA document, otherwise null.
  - xiii Set *FamilyName* to the patient family name if the message was unpackaged to a CDA document, otherwise null.
  - xiv Set *GivenNames* to the patient given names joined by spaces if the message was unpackaged to a CDA document, otherwise null.
- i Create a record in the *p2p.SealedMessage* table with properties as follows:
    - i Set *MessageDirectionId* to the value of the "Inbound" member of the *MessageDirection* enumeration.
    - ii Set *MessageStatusId* to the value of the "Unacknowledged" member of the *MessageStatus* enumeration.
    - iii Set *MessagePayloadId* to the ID of the *MessagePayload* record.
    - iv Set *InteractionRecordId* to null.
    - v Set *DigestValue* to the value of the signedPayload XML path "/sp:signedPayload/sp:signatures/ds:Signature/ds:SignedInfo/ds:Reference/ds:DigestValue".
    - vi Set *CreationTime* to the *metadata.creationTime* of the message.
    - vii Set *ExpiryTime* to the *metadata.expiryTime* of the message.
    - viii Set *InvocationIdentifier* to the *metadata.invocationId* of the message.
  - j Create records in the *p2p.SealedMessageMetadata* table for each *metadata.otherTransportMetadata* element with properties as follows:
    - i Set *MetadataType* to the *metadataType* element of *otherTransportMetadata*.
    - ii Set *MetadataValue* to the UTF-8 encoding of the outer XML of the *metadataValue* element of *otherTransportMetadata*.
    - iii Set *SealedMessageId* to the ID of the *SealedMessage* record.
  - k Create a record in the *p2p.TransportResponse* table with properties as follows:
    - i Set *TransportResponseClassId* to the value of the "Error" member of the *TransportResponseClass* enumeration if any validation steps failed, otherwise the "Success" member.
    - ii Set *TransportResponseStatusId* to the value of the "Undelivered" member of the *TransportResponseStatus* enumeration.
    - iii Set *ReceiverOrganisationUri* to the *metadata.receiverOrganisation* of the message.
    - iv Set *SenderOrganisationUri* to the *metadata.senderOrganisation* of the message.

- v Set *ReceiverOrganisationId* to the ID of the receiver organisation if it exists in the LHSD, otherwise null.
  - vi Set *SenderOrganisationId* to the ID of the sender organisation if it exists in the LHSD, otherwise null.
  - vii Set *ResponseIdentifier* to a new UUID prefixed with "urn:uuid:".
  - viii Set *InvocationIdentifier* to the *metadata.invocationId* of the message.
  - ix Set *SealedMessageId* to the ID of the *SealedMessage* record.
  - x Set *ResponseTime* to the current date and time.
  - xi Set *Final* to true.
  - xii Set *ResponseCode* to  
"http://ns.electronichealth.net.au/smd/codes/Error" if any validation steps failed, otherwise  
"http://ns.electronichealth.net.au/smd/codes/Success".
  - xiii Set *Message* to a description of the validation issue that does not include any sensitive information, or "OK" if there is no issue.
  - xiv Set *DigestValue* to the value of the *signedPayload* XML path  
"/sp:signedPayload/sp:signatures/ds:Signature/ds:SignedInfo/ds:Reference/ds:DigestValue".
  - xv Set *DeliveryEndpoint* to the *interaction.serviceEndpoint* of the last *metadata.routeRecord* whose *interaction.serviceInterface* is  
"http://ns.electronichealth.net.au/smd/intf/TransportResponseDelivery/TLS/2010".
- l Move to the next message.
- 4 If there were no messages retrieved, then sleep until the configured interval has elapsed, otherwise repeat from step 2 immediately.

### 2.2.3.2 Response Delivery Service

The *ScheduledResponseDeliveryService* provides a scheduled service operation in HIPS Core to periodically assess and action transport responses.

It is implemented as a custom WCF service host that executes at a configurable interval to retrieve unacknowledged received messages from the P2P data store and attempt delivery of the required transport response.

The *ScheduledResponseDeliveryService* is implemented as follows:

- 1 Commence processing after configured interval.
- 2 Query the *p2p.TransportResponse* table for records meeting the following criteria:
  - a *TransportResponseStatus* = "Undelivered"
  - b *p2p.SealedMessage.MessageDirection* = "Inbound"
- 3 For each transport response:
  - a If [Current Date/Time] > *ResponseTime* + [Timeout Period]
    - i Set *TransportResponseStatus* = "Expired".
    - ii Ensure this event is logged via the configured P2P log provider.
  - b Otherwise:



- i Invoke the [deliver](#) service operation of the Transport Response Delivery service provided by the HIPS SMD Server to attempt delivery of the transport response for the current message. Provide the URL from *DeliveryEndpoint* in the *metadata.otherTransportMetadata* property of the request, as described in [TransportResponseMetadataType](#).
    - ii If successful, set *TransportResponseStatus* = "Delivered".
    - iii Otherwise, do not proceed further for this transport response.
  - c Determine if the transport response is for a valid message based on *ResponseClass* = "Successful".
  - d If *p2p.SealedMessage.MessageStatus* = "Unacknowledged":
    - i If valid, set *p2p.SealedMessage.MessageStatus* = "Unpublished".
    - ii Otherwise, set *p2p.SealedMessage.MessageStatus* = "Failed".
  - e Move to the next transport response.
- 4 After all transport responses have been processed, sleep until the next configured interval has elapsed.

### 2.2.3.3 Message Publisher Service

The *ScheduledPublishingService* provides a scheduled service operation to periodically publish messages received via SMD.

It is implemented as a custom WCF service host that executes at a configurable interval to identify and retrieve messages within the P2P data store that have been received (and are ready to be published) and publish them to a configurable web services-based endpoint provided by the implementer according to a defined message publishing contract.

The *ScheduledPublishingService* implements the following logic:

- 1 Commence processing after configured interval.
- 2 Query for inbound messages meeting the following criteria using the implementation of the MessageReceiptService/QueryMessages service operation:
  - a Message Status = "Unpublished"
- 3 Create a WCF client proxy based on the "Publishing Endpoint Configuration" defined in the service configuration file.
- 4 For each message:
  - a Retrieve the message content using the implementation of the MessageReceiptService/GetMessage service operation for the Invocation Identifier of the current message.
  - b Construct a new service message conforming to the [Contract for Publishing Messages](#) based on the message metadata contained in the QueryMessages response and the message content contained in the GetMessage response.
  - c Invoke the WCF client proxy created previously to send the constructed service message to the publishing endpoint.
    - i If invocation succeeds:

Modify the status of the current message to be "Published".

ii If invocation fails:

- Log details of the error.
- Do not modify the status of the current message.

d Move to the next message.

5 After all messages have been processed, sleep until the next configured interval has elapsed.

## 2.2.4 Secure Message Delivery Web Services

The Secure Message Delivery web services in the HIPS SMD Gateway provides the HIPS SMD Server that is intended to be hosted in a DMZ environment and hosts SMD services for sending out messages and receiving transport responses for those messages and provides a SMD receiving services in the HIPS SMD Server, intended to be hosted in a DMZ, for receiving messages via SMD.

### 2.2.4.1 Unsealed Message Retrieval

The *UnsealedMessageRetrieval* service is hosted in the HIPS Decrypter App Server with an HTTP binding. It is intended to be accessed only by the P2P App Server in the internal network.

#### ***retrieve***

The *UnsealedMessageRetrieval* service has a single *retrieve* operation that retrieves a batch of inbound messages from the Sealed Message Retrieval service on the SMD App Server and decrypts the message.

*Logic:*

- 1 Invoke the *retrieve* operation of the *SealedMessageRetrieval* service using the *limit* parameter from the request. If a SOAP fault is raised by the service, propagate this fault. If an unexpected exception occurs, raise a *StandardErrorType* fault with error code *serviceTemporaryUnavailable*.
- 2 Iterate through each of the retrieved messages.
  - a Attempt to decrypt the message using the configured certificates and validate the identity of the sender.
  - b If an error occurs during decryption or validation of a message, store the *StandardErrorType* object into the response instead of returning a fault.

### 2.2.4.2 Sealed Message Retrieval

The *SealedMessageRetrieval* service is hosted in the HIPS Front-End SMD Server with an HTTPS binding. It is intended to be accessed only by the HIPS Decrypter in the DMZ.

Despite having the same name, this service is not compatible with the SMD Library's service called Sealed Message Retrieval. The list and retrieve methods have been combined into one method that retrieves messages for all internal organisations.

**retrieve**

The *SealedMessageRetrieval* service has a single *retrieve* operation that retrieves a batch of inbound messages from the file system.

*Logic:*

- 1 If the request is found to be invalid, a *StandardErrorType* fault with error code *badParam* is raised.
- 2 Sort all received message files based upon their creation dates/times. This will ensure that messages are retrieved (and returned) in the order they were received.
- 3 Iterate through each of the message files until the specified limit has been reached or all files in the folder have been processed.

**2.2.4.3 Error Message Retrieval**

The *ErrorMessageRetrieval* Retrieval service is hosted in the HIPS Front-End SMD Server with an HTTPS binding. It is intended to be accessed only by the P2P App Server in the internal network.

**retrieve**

The *ErrorMessageRetrieval* service has a single *retrieve* operation that retrieves errors that have been logged by the SMD Server.

*Logic:*

- 1 If the request is found to be invalid, a *StandardErrorType* fault with error code *badParam* is raised. If an unexpected exception occurs, a *StandardErrorType* fault with error code *serviceTemporaryUnavailable* is raised.
- 2 Sort all error files based upon their creation dates/times. This will ensure that errors are retrieved (and returned) in the order they were created.
- 3 Iterate through each of the error files until the specified limit has been reached or all files in the folder have been processed.
- 4 Move retrieved error files into the archive folder.

**2.2.4.4 Sealed Message Delivery**

The *SealedMessageDelivery* service accepts a message for delivery.

**deliver**

Accepts a message for delivery. Inbound messages are stored into the file system for later retrieval. Outbound messages are forwarded to their destination.

*Logic:*

- 1 If the request is found to be invalid, a *StandardErrorType* fault with error code *badParam* is raised. If an unexpected exception occurs during validation, a *DeliverErrorType* fault is raised.
- 2 Check whether an endpoint URL is defined in the other transport metadata and the sender of the message is configured as an internal organisation. If this is the case, then the Sealed Message needs to be forwarded on to the nominated external endpoint.

- a When forwarding Sealed Messages, check that the organisation identifier embedded within the public key matches that of the receiver organisation identifier in the message metadata. If the two do not match, a *StandardErrorType* fault with error code *badEncryption* is raised. This checking is necessary to ensure a Sealed Message is encrypted with the endpoint public key.
  - b When forwarding Sealed Messages, check whether the client certificate provided is an internal certificate to the HIPS instance. If the certificate is not an internal certificate, a *StandardErrorType* fault with error code *notAuthorized* is raised.
- 3 Otherwise, check whether the receiver of the message is configured as an internal organisation. If so, persist the sealed message to the local file system.
  - a If the message already exists, log an error and return the *duplicate* status.
  - b If an unexpected exception occurs, log detailed information about the error, which will be received by the system administrator. Avoid exposing detailed exception information to an external system, which might be the case if a file system exception is raised. Raise a *DeliverErrorType* fault with a generic error message.
- 4 Otherwise, the Sealed Message that has been received is neither sent by an internal system - to be forwarded - and does not have a valid receiver organisation ID. Raise a *DeliverErrorType* fault indicating the incoming Sealed Message is invalid.

#### 2.2.4.5 Transport Response Delivery

The *TransportResponseDelivery* service accepts one or more transport responses for delivery.

##### ***deliver***

Accepts one or more transport responses for delivery. Inbound responses are stored into the file system for later retrieval. One outbound response at a time may be forwarded to its destination.

*Logic:*

- 1 If the request is found to be invalid, a *StandardErrorType* fault with error code *badParam* is raised. If an unexpected exception occurs during validation, a *StandardErrorType* fault with error code *serviceTemporaryUnavailable* is raised.
- 2 If the request contains only 1 transport response, check whether an endpoint URL is defined in the other transport metadata and the source of the transport response is configured as an internal organisation. If this is the case, then the transport response needs to be forwarded on to the nominated external endpoint.
  - a If a deliver exception occurs, check whether the limited period retry processing of forwarding Transport Responses has been configured. If it has, and the forwarding period has expired, move the corresponding sealed message to the error folder.
  - b If a final transport response has been forwarded, archive the corresponding sealed message.

- 3 Otherwise, check whether the original message sender associated with each transport response is configured as an internal organisation. If so, persist the transport response to the local file system.
  - a If the transport response already exists, log an error and raise a *StandardErrorType* fault with error code *badParam*.
  - b If an unexpected exception occurs, log detailed information about the error, which will be received by the system administrator. Avoid exposing detailed exception information to an external system, which might be the case if a file system exception is raised. Raise a *StandardErrorType* fault with error code *serviceTemporaryUnavailable*.
- 4 Otherwise, a transport response that has been received is neither sent by an internal system - to be forwarded - and does not have a valid sender organisation ID. Raise a *DeliverErrorType* fault with error code *unknownSenderOrganisation*.

#### 2.2.4.6 Transport Response Retrieval

The *TransportResponseRetrieval* service is hosted in the HIPS Front-End SMD Server with an HTTPS binding. It is intended to be accessed only by the P2P App Server in the internal network.

Despite having the same name, this service is not compatible with the SMD Library's service called Transport Response Retrieval. The *retrieve* method retrieves transport responses for all internal organisations, and does not support retrieving responses that have already been retrieved.

##### ***retrieve***

Retrieves a batch of inbound transport responses from the file system.

*Logic:*

- 1 If the request contains null or empty parameters, a *StandardErrorType* fault with error code *badParam* is raised. If the organisation identifier is found to be invalid, a *RetrieveErrorType* fault with error code *unknownOrganisation* is raised. If an unexpected exception occurs, a *StandardErrorType* fault with error code *serviceTemporaryUnavailable* is raised.
- 2 Sort all received transport response files based upon their creation dates/times. This will ensure that transport responses are retrieved (and returned) in the order they were received.
- 3 Iterate through each of the transport response files until the specified limit has been reached or all files in the folder have been processed.

##### ***remove***

Archives inbound transport responses in the file system, so that they are not retrieved by subsequent requests to *retrieve*, unless the *allAvailable* flag is set

*Logic:*

- 1 If any of the given response identifiers are not found, a *RemoveErrorType* fault with error code *unknownResponse* is raised.
- 2 If any of the given response identifiers identify a response that has not yet been retrieved, a *RemoveErrorType* fault with error code *hasNotBeenRetrieved* is raised.

- 3 Archive each of the specified transport responses.
- 4 If an unexpected exception occurs, a *StandardErrorType* fault with error code *serviceTemporaryUnavailable* is raised.

## 2.2.5 LHSD Services

HIPS P2P provides web services to support the maintenance and synchronisation of the LHSD data store as well as validation of HPI-O for provider organisations and HPI-I for provider individuals.

### 2.2.5.1 LHSD Maintenance Service

The LHSD Maintenance *DirectoryConfigurationService* in HIPS -Core provides services to support the maintenance of system configuration data within the LHSD data store.

#### **ListIntermediaries**

Returns a list of all configured intermediaries

*Logic:*

- 1 Retrieve all records from the *p2p.Intermediary* table.
- 2 Transform the set of records to their corresponding representation as a set of instances of the *Intermediary* class.
- 3 Construct and return a *ListIntermediariesResponse* containing the set of *Intermediary* instances.

#### **CreateIntermediary**

Creates an intermediary

*Logic:*

- 1 Determine if a record representing the *Intermediary* exists in the *p2p.Intermediary* table based on the value of the *ServiceEndpointUri* property.
- 2 If it does, return an error.
- 3 Otherwise, create a record in the *p2p.Intermediary* table based on the *Intermediary* provided.
- 4 Construct and return a *CreateIntermediaryResponse*.

#### **DeleteIntermediary**

Deletes a specified intermediary.

*Logic:*

- 1 Determine if a record representing the *Intermediary* exists in the *p2p.Intermediary* table based on the value of the *ServiceEndpointUri* property.
- 2 If it does not, return an error.
- 3 If it does, delete the existing record in the *p2p.Intermediary* table based on the value of the *ServiceEndpointUri* property.
- 4 Construct and return a *DeleteIntermediaryResponse*.

**ListPayloadSchemes**

Returns a list of all configured payload schemes.

*Logic:*

- 1 Retrieve all records from the *p2p.PayloadScheme* table (with associated data from the *hips.DocumentType* and *p2p.PayloadPackaging* tables).
- 2 Transform the set of records to their corresponding representation as a set of instances of the *PayloadScheme* class.
- 3 Construct and return a *ListPayloadSchemesResponse* containing the set of *PayloadScheme* instances.

**ActivatePayloadScheme**

Sets the activation state of a specified payload scheme in order to activate or inactivate it

*Logic:*

- 1 Determine if a record representing the *PayloadScheme* exists in the *p2p.PayloadScheme* table based on the value of the *QualifiedIdentifier* property.
- 2 If it does not, return an error.
- 3 If it does, update the existing record in the *p2p.PayloadScheme* table to set the value for its *IsActive* column based on the value provided for the *IsActive* property in the request.
- 4 Construct and return an *ActivatePayloadSchemeResponse*.

**ListDelegates**

Returns a list of all configured delegates.

*Logic:*

- 1 Retrieve all records from the *p2p.Delegate* table (with associated data from the *p2p.Intermediary* table).
- 2 Transform the set of records to their corresponding representation as a set of instances of the *Delegate* class.
- 3 Construct and return a *ListDelegatesResponse* containing the set of *Delegate* instances.

**ActivateDelegate**

Sets the activation state of a specified delegate in order to activate or inactivate it.

*Logic:*

- 1 Determine if a record representing the *Delegate* exists in the *p2p.Delegate* table based on the value of the *QualifiedIdentifier* property.
- 2 If it does not, return an error.
- 3 If it does, update the existing record in the *p2p.Delegate* table to set the value for its *IsActive* column based on the value provided for the *IsActive* property in the request.
- 4 Construct and return an *ActivateDelegateResponse*

### ***PrioritiseDelegate***

Sets the priority of a specified delegate in relation to another delegate. The specified delegate will be prioritised higher than the other delegate. If no other delegate is specified, the specified delegate will be assigned the lowest priority.

*Logic:*

- 1 Determine if a record representing the *Delegate* exists in the *p2p.Delegate* table based on the value of the *QualifiedIdentifier* property.
- 2 If it does not, return an error.
- 3 If specified, determine if a record representing the relative *Delegate* exists in the *p2p.Delegate* table based on the value of the *RelativeToDelegateQualifiedIdentifier* property.
- 4 If it does not, return an error.
- 5 Update:
  - a The record identified by the *QualifiedIdentifier* property in the *p2p.Delegate* table to assign it a value for the *PriorityNumber* column that is greater than the value of the *PriorityNumber* column for the record identified by the *RelativeToDelegateQualifiedIdentifier* property.
  - b Redistribute the values for the *PriorityNumber* column for all records in the *p2p.Delegate* table to ensure unique values.
- 6 Construct and return a *PrioritiseDelegateResponse*

### ***MediateDelegate***

Sets the intermediary for a specified delegate.

*Logic:*

- 1 Determine if a record representing the *Delegate* exists in the *p2p.Delegate* table based on the value of the *QualifiedIdentifier* property.
- 2 If it does not, return an error.
- 3 If specified, determine if a record representing the *Intermediary* exists in the *p2p.Intermediary* table based on the value of the *IntermediaryServiceEndpointUri* property.
- 4 If it does not, return an error.
- 5 Update the existing record in the *p2p.Delegate* table to set the value for its *IntermediaryId* column based on the intermediary identified by the value provided for the *IntermediaryServiceEndpointUri* property in the request:
  - a For an existing record in the *p2p.Intermediary* table, set to the *IntermediaryId*.
  - b For a non-specified value for the *IntermediaryServiceEndpointUri* property of the request, set to NULL.
- 6 Construct and return a *MediateDelegateResponse*.

### ***ListDocumentTypes***

Returns a list of all configured document types.

*Logic:*

- 1 Retrieve all records from the *p2p.DocumentType* table.



- 2 Transform the set of records to their corresponding representation as a set of instances of the *DocumentType* class.
- 3 Construct and return a *ListDocumentTypesResponse* containing the set of *DocumentType* instances.

### 2.2.5.2 LHSD Change Retrieval Service

The LHSD Change Retrieval *DirectorySynchronisationService* in HIPS -Core provides a service operation to support the retrieval of changes to the LHSD.

Changes to the directory will be tracked at the level of a "logical entity", which is either an individual, an organisation, or a link between an individual and an organisation.

Example 1: The creation of a new individual will result in one creation change for the new individual and one creation change for each new link between that individual and an existing organisation.

Example 2: The addition of a phone number to an organisation will result in one modification change to that organisation.

Example 3: An individual moving from one organisation to another will result in two changes: the deactivation of one link and the creation of a new link.

Each change to a logical entity in the directory will be assigned a sequential integer called "ChangeNumber" as well as a date and time "ChangeTimestamp" at which the change was applied to the directory. Clients may request changes from HIPS using either the change number or timestamp. For the avoidance of issues due to slightly varying clocks between servers, it is recommended to use the ChangeNumber as the primary means of tracking and requesting changes.

#### ***RetrieveChanges***

Returns a list of changes for each of the logical entities in the directory. If neither a ChangeNumber nor a ChangeTimestamp are provided, returns changes starting with the earliest recorded change.

*Logic:*

- 1 Identify records in the *p2p.Change* table matching the criteria specified in the request:
  - a If a value has been specified for the ChangeNumber property of the request, include only records with a value for the *ChangeID* column greater than or equal to the ChangeNumber.
  - b If a value has been specified for the ChangeTimestamp property of the request, include only records with a value for the *DateCreated* column greater than or equal to the ChangeTimestamp.
  - c Include only at most a number of records equal to the value specified for the ChangeCount property of the request, with included records ordered in ascending order of the *ChangeID* column.
- 2 For each of the records in the *p2p.Change* table identified as matching the criteria specified in the request, construct an instance of the relevant "\*Change" class corresponding to the type of entity the change is associated with:

- a For a change to a provider organisation, indicated by a non-null value in the *ProviderOrganisationId* column in the *p2p.Change* record, retrieve the current state of the provider organisation from the *nhsd.ProviderOrganisation* table and associated tables and construct an instance of the *OrganisationChange* class containing the required details.
  - b For a change to a provider location, indicated by a non-null value in the *ProviderLocationId* column in the *p2p.Change* record, retrieve the current state of the provider location from the *p2p.ProviderLocation* table and associated tables and construct an instance of the *LocationChange* class containing the required details.
  - c For a change to a provider link, indicated by a non-null value in the *ProviderLinkId* column in the *p2p.Change* record, retrieve the current state of the provider link from the *p2p.ProviderLink* table and associated tables and construct an instance of the *LinkChange* class containing the required details.
  - d For a change to a provider individual, indicated by a non-null value in the *ProviderIndividualId* column in the *p2p.Change* record, retrieve the current state of the provider individual from the *nhsd.ProviderIndividual* table and associated tables and construct an instance of the *IndividualChange* class containing the required details.
- 3 Construct and return a *RetrieveChangesResponse* containing the set of changes.

### 2.2.5.3 NEPS Subscription Maintenance Service

The NEPS Subscription Maintenance *SubscriptionService* in HIPS -Core provides:

- services to support retrieving and maintaining NEPS subscription details for an HPI-O.
- a service operation to retrieve details of the NEPS subscription for a specified HPI-O.
- a service operation to search the NEPS for providers (individuals or organisations): Search the NHSD, Reconcile results with LHSD contents, Indicate which providers are already present in the LHSD (in the area of interest) and which are outside the area of interest.
- service operations to add, modify or remove items (providers, geographical locations, specialties) from the NEPS subscription for a specified HPI-O.

#### **ListSubscriptionDefinitions**

Returns a list of area of interest definitions in the accessing organisation's subscription.

*Logic:*

- 1 Retrieve all records from the *neps.AreaOfInterest* table for the *neps.Subscription* record corresponding to the *p2p.AccessingOrganisation* record for the accessing organisation HPI-O specified in the User property of the request.
- 2 Transform the set of records to their corresponding representation as a set of instances of the *AreaOfInterest* class.

- 3 Construct and return a *ListSubscriptionDefinitionsResponse* containing the set of *AreaOfInterest* instances.

### **FindIndividuals**

Searches the NHSD for provider individuals and returns a list of provider individuals and their linked locations and organisations, and whether or not they are in the accessing organisation's subscription.

*Logic:*

- 1 Look up reference items using the display name, code or sub-code specified in the request. If the user enters anything invalid, they must get a useful message so they know what to fix.
- 2 Invoke the NEPS Lookup *FindProviderIndividuals* service using all the specified criteria. Audit the NEPS call.
- 3 Look up the individual members of the accessing organisation's subscription in the *neps.SubscriptionMemberIndividual* table.
- 4 Mark each individual returned from *FindProviderIndividuals* according to whether it is in the list of subscription members or not.
- 5 Transform the set of records to their corresponding representation as a set of instances of the DTO classes for *ProviderIndividual*, *ProviderLink*, *ProviderLocation* and *ProviderOrganisation* and the DTO classes that they reference.
- 6 Construct and return a *FindProviderIndividualsResponse* containing a *SearchResult* with the sets of DTO instances.

### **FindOrganisations**

Searches the NHSD for provider organisations and returns a list of provider organisations and their linked locations, and whether or not they are in the accessing organisation's subscription.

*Logic:*

- 1 Look up reference items using the display name, code or sub-code specified in the request. If the user enters anything invalid, they must get a useful message so they know what to fix.
- 2 Invoke the NEPS Lookup Service *FindProviderOrganisations* operation using all the specified criteria. Audit the NEPS call.
- 3 Look up the organisation members of the accessing organisation's subscription in the *neps.SubscriptionMemberOrganisation* table.
- 4 Mark each organisation returned from *FindProviderOrganisations* according to whether it is in the list of subscription members or not.
- 5 Transform the set of records to their corresponding representation as a set of instances of the DTO classes for *ProviderLocation* and *ProviderOrganisation* and the DTO classes that they reference.
- 6 Construct and return a *FindProviderOrganisationsResponse* containing a *SearchResult* with the sets of DTO instances.

### **AddAreaOfInterest**

Adds an area of interest definition to the accessing organisation's subscription.

*Logic:*

- 1 For each occupation and classification provided:
  - a If the URI was provided, get the reference item using the URI. Otherwise, invoke the business logic for reference data to search for the reference item using the code and display name.
  - b If the reference item does not exist, or its status is not 'ACTIVE', or more than one is found, throw an *InvalidReferenceDataException*.
  - c If the URI and code were both provided, check that the provided code matches the code of the reference item. If not, throw an *InvalidReferenceDataException*.
  - d If the URI and display name were both provided, check that the provided display name matches the display name of the reference item. If not, throw an *InvalidReferenceDataException*.
- 2 For each geographical area provided:
  - a Look up the state reference item, using the provided state code as the Code.
  - b If the suburb and postcode were provided, get the suburb reference item, using the suburb as the DisplayName and postcode as the SubCode.
- 3 Invoke the NEPS AddAreaOfInterest operation, providing the description, geographical areas, occupations and classification reference items, and the extra member identifiers. Audit the NEPS call.
- 4 Store the Code (which was returned from NEPS) and Description into the *neps.AreaOfInterest* table.
- 5 Store the geographical areas into the *neps.AreaOfInterestGeographicalArea* table.
- 6 Store the occupations and classifications into the *neps.AreaOfInterestService* table.
- 7 Store the extra members into the *neps.AreaOfInterestProviderIdentifier* table.
- 8 Invoke the business logic for NEPS synchronisation to process the added members in the returned NEPS change set.
- 9 Construct and return an *AddAreaOfInterestResponse* containing the added *AreaOfInterest* instance.

### **UpdateAreaOfInterest**

Updates an area of interest definition to the accessing organisation's subscription.

*Logic:*

- 1 Look up the record in the *neps.AreaOfInterest* table using the provided *Code* and the subscription of the accessing organisation. If the record is not found, throw an exception.
- 2 For each new occupation and classification provided:

- a If the URI was provided, get the reference item using the URI. Otherwise, invoke the business logic for reference data to search for the reference item using the code and display name.
  - b If the reference item does not exist, or its status is not 'ACTIVE', or more than one is found, throw an `InvalidReferenceDataException`.
  - c If the URI and code were both provided, check that the provided code matches the code of the reference item. If not, throw an `InvalidReferenceDataException`.
  - d If the URI and display name were both provided, check that the provided display name matches the display name of the reference item. If not, throw an `InvalidReferenceDataException`.
- 3 For each new geographical area provided:
  - a Look up the state reference item, using the provided state code as the Code.
  - b If the suburb and postcode were provided, get the suburb reference item, using the suburb as the `DisplayName` and postcode as the `SubCode`.
- 4 Invoke the NEPS `UpdateAreaOfInterest` operation, providing the description, geographical areas, occupations and classification reference items, and the extra member identifiers. Audit the NEPS call.
- 5 Update the *Description* in the *neps.AreaOfInterest* table.
- 6 Add or remove geographical areas in the *neps.AreaOfInterestGeographicalArea* table.
- 7 Add or remove occupations and classifications in the *neps.AreaOfInterestService* table.
- 8 Add or remove extra members in the *neps.AreaOfInterestProviderIdentifier* table.
- 9 Invoke the business logic for NEPS synchronisation to process the added and removed members in the returned NEPS change set.
- 10 Construct and return an *UpdateAreaOfInterestResponse* containing the updated *AreaOfInterest* instance.

### **DeleteAreaOfInterest**

Deletes an area of interest definition from the accessing organisation's subscription.

*Logic:*

- 1 Look up the record in the *neps.AreaOfInterest* table using the provided *Code* and the subscription of the accessing organisation. If the record is not found, throw an exception.
- 2 Invoke the NEPS `DeleteAreaOfInterest` operation, providing the *Code*. Audit the NEPS call.
- 3 Delete the AOI's records from the *neps.AreaOfInterestGeographicalArea* table.
- 4 Delete the AOI's records from the *neps.AreaOfInterestService* table.
- 5 Delete the AOI's records from the *neps.AreaOfInterestProviderIdentifier* table.
- 6 Delete the AOI's record from the *neps.AreaOfInterest* table.

- 7 Invoke the business logic for NEPS synchronisation to process the removed members in the returned NEPS change set.
- 8 Construct and return a *DeleteAreaOfInterestResponse* containing the deleted *AreaOfInterest* instance.

#### **2.2.5.4 NEPS Subscription Service**

The NEPS Subscription Maintenance *DirectorySynchronisationService* in HIPS -Core provides the following functionality:

- a scheduled service operation to periodically synchronise NEPS subscription results to the LHSD.
  - Retrieve all subscription entities from the LHSD data store
  - For each subscribing HPI-O:
    - Use the authorised employee name and ID for the HPI-O from the subscription entity
    - Retrieve the list of updated entities in the NEPS subscription only for details that have changed since the previous request
    - For each changed individual, retrieve the individual details from NEPS
    - For each changed organisation, retrieve the organisation details, channel preferences and interaction records from NEPS
    - Apply the results of the NEPS subscription changes to the LHSD data store
  - Update the synchronisation timestamp for the HPI-O in the LHSD data store
- Support external (outside of code) configuration of the interval at which synchronisation occurs.
- The information retrieved from NEPS is always considered to be the most current and correct information, and will replace the existing information on each entity within the LHSD data store.

#### ***Synchronise***

Retrieves and applies changes to provider information using the NEPS Subscription Service.

*Logic:*

- 1 Retrieve the *neps.Subscription* record for the accessing organisation. If no subscription exists, throw an exception (mapped to a fault in the service layer).
- 2 Invoke the *RetrieveSubscriptionChanges* business logic operation.

#### ***RetrieveSubscriptionChanges***

Retrieves changes to provider information using the NEPS Subscription Service.

*Logic:*

- 1 Invoke the *GetSubscriptionChanges* operation of the NEPS Subscription Service and audit the request. If no value is provided for *Timestamp* in the request, use the value of the *LastSynchronisationCompleted* in the subscription record as the *Timestamp*.

- 2 Append the added, changed and removed member identifiers to the PendingChange table in the database and the PendingChanges list in the subscription object.
- 3 If the NEPS service failed to retrieve changes, throw an exception (mapped to a fault in the service layer) with details of the errors returned by NEPS.
- 4 If the GetSubscriptionChanges response indicates that there are more members to be processed, call GetSubscriptionChanges again with the same date/time and session ID, adjusting the start and end member numbers to retrieve the next batch of subscription changes, and repeat processing from step 2 onwards.
- 5 Store the current date and time as the *LastSynchronisationCompleted* for the subscription.

### **ProcessPendingChanges**

Applies changes to provider information using the NEPS Lookup Service to obtain the details.

*Logic:*

- 1 Use the *ChangeTypeId* property to divide the list of *PendingChange* records in the subscription into separate lists of removed members, added members and changed members.
- 2 For each removed member:
  - a Look up the individual or organisation using its NHSD URI in the *nhsd.ProviderIndividualIdentifier* or *nhsd.ProviderOrganisationIdentifier* tables respectively.
  - b Delete the record from *neps.SubscriptionMemberIndividual* or *neps.SubscriptionMemberOrganisation* to record that the individual or organisation has been removed from the current subscription.
  - c If the individual or organisation is no longer a member of any subscriptions, set the *IsInSubscription* property of the *ProviderIndividual* or *ProviderOrganisation* table to false, and add a record to *nhsd.Change* to record that the member has been removed from all subscriptions.
- 3 If there are any added members or changed members, invoke the GetProviderIndividual or GetProviderOrganisation operation of the NEPS Lookup Service, to retrieve the details of each added or changed member. Audit each request.

Note: It was originally intended for this step to invoke "GetDetailsForIdentifiers" to retrieve details of added or changed members in batches of up to 50 at a time, however this service operation does not return the service location identifier or document channel preferences, so it will be necessary to call the "Get" operations, one member at a time, to retrieve all the information required for the LHSD.

- 4 If the NEPS service failed to retrieve changes or details, throw an exception (mapped to a fault in the service layer) with details of the errors returned by NEPS.
- 5 For each of the added or changed members:

- a Look up the individual or organisation using the NHSD URI in the `nhsd.ProviderIndividualIdentifier` or `nhsd.ProviderOrganisationIdentifier` tables respectively.
  - b If found, update all the details of the individual, link, location and organisation in the following tables. If not found, create records to store all the details.
    - i `nhsd.ProviderIndividual`
    - ii `nhsd.ProviderIndividualAlias`
    - iii `nhsd.ProviderIndividualIdentifier` (replace national, keep local)
    - iv `nhsd.ProviderIndividualContact`
    - v `nhsd.ProviderIndividualOccupation`
    - vi `nhsd.ProviderLink` (reference existing locations if they exist already)
    - vii `nhsd.ProviderLinkIdentifier` (replace national, keep local)
    - viii `nhsd.ProviderLocation`
    - ix `nhsd.ProviderLocationIdentifier` (replace national, keep local)
    - x `nhsd.ProviderLocationAddress`
    - xi `nhsd.ProviderLocationContact`
    - xii `nhsd.ProviderLocationClassification`
    - xiii `nhsd.ProviderOrganisation`
    - xiv `nhsd.ProviderOrganisationIdentifier` (replace national, keep local)
    - xv `nhsd.ProviderOrganisationName`
    - xvi `nhsd.DocumentChannelPreference`
    - xvii `nhsd.ChannelType`
    - xviii `nhsd.ChannelCarrier`
    - xix `nhsd.ProprietaryValue`
    - xx `nhsd.Change` (keep existing and add one new change record)
  - c Set *IsInSubscription* to true and save the records.
  - d If the member was added, insert a record in `neps.SubscriptionMemberIndividual` or `neps.SubscriptionMemberOrganisation` to record that the member has been added to the subscription.
- 6 Remove all of the pending changes from the database.

#### **2.2.5.5 Provider Organisation Directory Service**

The Provider Organisation Directory *ProviderOrganisationDirectoryService* in HIPS - Core provides the following functionality:

- Support Read, Create or Delete Provider Individual or Organisation identity mappings, specifically for provider organisations
- Validated HPI-O Retrieval



The service supports the notion of “national” and “local” identifiers associated with a provider organisation. “National” identifiers are deemed to be any of the following organisation-related identifier types:

- HPI-O
- NHSD URI for Organisation
- ABN
- ACN

“Local” identifiers are any identifier type that is not a “national” organisation-related identifier type.

### ***ListMappedOrganisations***

Returns a list of provider organisations for whom one or more local identifier mappings exist.

*Logic:*

- 1 Retrieve all records from the *nhsd.ProviderOrganisation* table for which a “local” record exists in the *nhsd.ProviderOrganisationIdentifier* table. A “local” record is defined as any record that has an associated *p2p.ProviderIdentifierType.Code* that is not one of the previously identified “national” identifier types.
- 2 Transform the set of records to their corresponding representation as a set of instances of the *ProviderOrganisation* class.
- 3 Construct and return a *ListMappedOrganisationsResponse* containing the set of *ProviderOrganisation* instances.

### ***CreateLocalIdentifier***

Creates a new local identifier mapping for a provider organisation.

*Logic:*

- 1 Locate the record representing the provider organisation in the *nhsd.ProviderOrganisationIdentifier* table based on the values of the *Type.Code* and *Value* properties of the *NationalIdentifier* property of the request.
- 2 If the record does not exist, return an error.
- 3 Locate the corresponding *nhsd.ProviderOrganisation* record and verify the value of the *OrganisationName* column matches the value of the *OrganisationName* property specified in the request.
- 4 If the values do not match, return an error.
- 5 Otherwise, create a record in the *nhsd.ProviderOrganisationIdentifier* table based on the values provided for the *LocalIdentifier* property of the request.
- 6 Construct and return a *CreateLocalIdentifierResponse*.

### ***DeleteLocalIdentifier***

Deletes a local identifier mapping for a provider organisation.

*Logic:*

- 1 Locate the record representing the provider organisation in the *nhsd.ProviderOrganisationIdentifier* table based on the values of the *Type.Code* and *Value* properties of the *NationalIdentifier* property of the request.

- 2 If the record does not exist, return an error.
- 3 Locate the record representing the provider organisation in the *nhsd.ProviderOrganisationIdentifier* table based on the values of the Type.Code and Value properties of the LocalIdentifier property of the request.
- 4 If the record does not exist, return an error.
- 5 Locate the corresponding *nhsd.ProviderOrganisation* record and verify both previously located identifiers are associated with the same provider organisation.
- 6 If the values do not match, return an error.
- 7 Otherwise, delete the existing "local" record from the *nhsd.ProviderOrganisationIdentifier* table.
- 8 Construct and return a *DeleteLocalIdentifierResponse*.

### **GetValidatedHpio**

Attempts to obtain a validated HPI-O for use in a new eHealth message or document to identify the sending or receiving organisation.

*Logic:*

- 1 Identify the *nhsd.ProviderOrganisation* record corresponding to the value of the Identifier property of the request:
  - a Locate the record representing the provider organisation in the *nhsd.ProviderOrganisationIdentifier* table based on the values of the Identifier.Type.Code and Identifier.Value properties of the Identifier property of the request.
  - b Locate the *nhsd.ProviderOrganisation* record based on the shared *ProviderOrganisationId*.
  - c If either record does not exist, return an error.
- 2 Locate the *p2p.ProviderOrganisationHpio* record representing the provider organisation's HPI-O using the shared *ProviderOrganisationId*, and the *AccessingOrganisationNetworkId* corresponding to the current accessing organisation network.
- 3 Retrieve the configured "validity period" (default and maximum is 24 hours) from configuration.
- 4 Determine if the provider organisation's HPI-O needs to be revalidated. It must be revalidated if:
  - a There is no HPI-O record; OR
  - b The status of the HPI-O record indicates there are no unresolved exceptions or alerts associated with the identifier (CCA CR 10613); AND
  - c The value of the *DateLastValidated* column in the HPI-O record is less than the current date/time minus the configured "validity period".
- 5 If the provider organisation's HPI-O needs to be revalidated:
  - a If there is no HPI-O record, invoke the "Get Provider Organisation" service operation from NEPS using the NHSD URI of the provider organisation.
    - i If an error response is received from the NEPS service invocation, ensure relevant information is logged as per CCA CR 5873.



- a Status: p2p.ProviderOrganisationHpio.ProviderIdentifierStatusId
- b Value: p2p.ProviderOrganisationHpio.Value
- c DateLastValidated: p2p.ProviderOrganisationHpio.DateLastValidated
- d Identifiers: nhsd.*ProviderOrganisationIdentifier*
- e LegalName: p2p.*ProviderOrganisationHpio.LegalName*

**NOTE:** If the status of the HPI-O record for the provider organisation in the *nhsd.ProviderOrganisationIdentifier* table is "retired" or "deactivated":

- A warning message must be returned in the Messages property of the response.
- The Status of the response must be set to "Warning".

### **Search**

Searches the LHSD for a list of provider organisations, returning requested entities such as locations, document channel preferences and interaction records.

*Logic:*

- 1 Transform the search criteria to the common model.
- 2 If organisation criteria was supplied, obtain the primary key of organisations that match the following criteria:
  - a Legal name if supplied
  - b Organisation description if supplied
  - c If an organisation identifier is supplied:
    - i identifier type code
    - ii value
  - d If an organisation name is supplied:
    - i name usage code
    - ii value
- 3 If location criteria was supplied, obtain the primary key of locations that match the following criteria:
  - a Location name if supplied
  - b Location description if supplied
  - c If a location identifier is supplied:
    - i type code
    - ii value
  - d If an address is supplied:
    - i contact purpose code
    - ii contact purpose description
    - iii flat type
    - iv flat number
    - v address site name
    - vi level type

- vii level number
- viii street number
- ix lot number
- x street name
- xi street type
- xii street suffix
- xiii postal delivery type
- xiv postal delivery number
- xv unstructured address line
- xvi suburb
- xvii state
- xviii postcode
- xix latitude
- xx longitude
- e If an electronic contact detail supplied:
  - i contact purpose code
  - ii contact type code
  - iii details
- f If a classification is supplied:
  - i organisation type code
  - ii organisation type description
  - iii organisation service type code
  - iv organisation service type description
  - v organisation service unit code
  - vi organisation service unit description
- 4 Retrieve records from ProviderOrganisation based on the primary keys of entities that had criteria, and apply the skip and take limits.
- 5 For each of these organisation records, get the associated entities that are requested by the caller.
- 6 Transform the records to their corresponding representation in the DTO model.
- 7 Construct and return a *SearchResponse* containing the set of DTO instances.

#### **2.2.5.6 Provider Individual Directory Service**

The Provider Individual Directory *ProviderIndividualDirectoryService* in HIPS -Core provides the following functionality:

- Support Read, Create or Delete Provider Individual or Organisation identity mappings, specifically for provider individuals
- Validated HPI-I Retrieval
- LHSD Search

- The service supports the notion of “national” and “local” identifiers associated with a provider organisation. “National” identifiers are deemed to be any of the following individual-related identifier types:
- HPI-I
- AHPRA
- NHSD URI for Individual

“Local” identifiers are any identifier type that is not a “national” individual-related identifier type.

### ***ListMappedIndividuals***

Returns a list of provider individuals for whom one or more local identifier mappings exist.

*Logic:*

- 1 Retrieve all records from the *nhsd.ProviderIndividual* table for which a “local” record exists in the *nhsd.ProviderIndividualIdentifier* table. A “local” record is defined as any record that has an associated *p2p.ProviderIdentifierType.Code* that is not one of the previously identified “national” identifier types.
- 2 Transform the set of records to their corresponding representation as a set of instances of the *ProviderIndividual* class.
- 3 Construct and return a *ListMappedIndividualsResponse* containing the set of *ProviderIndividual* instances.

### ***CreateLocalIdentifier***

Creates a new local identifier mapping for a provider individual.

*Logic:*

- 1 Locate the record representing the provider individual in the *nhsd.ProviderIndividualIdentifier* table based on the values of the *Type.Code* and *Value* properties of the *NationalIdentifier* property of the request.
- 2 If the record does not exist, return an error.
- 3 Locate the corresponding *nhsd.ProviderIndividual* record and verify the values of the associated *p2p.Sex.Code* and *nhsd.ProviderIndividualName.FamilyName* match the values of the *Sex* and *FamilyName* properties specified in the request.
- 4 If the values do not match, return an error.
- 5 Otherwise, create a record in the *nhsd.ProviderIndividualIdentifier* table based on the values provided for the *LocalIdentifier* property of the request.
- 6 Construct and return a *CreateLocalIdentifierResponse*.

### ***DeleteLocalIdentifier***

Deletes a local identifier mapping for a provider individual.

*Logic:*

- 1 Locate the record representing the provider individual in the *nhsd.ProviderIndividualIdentifier* table based on the values of the *Type.Code* and *Value* properties of the *NationalIdentifier* property of the request.
- 2 If the record does not exist, return an error.

- 3 Locate the record representing the provider individual in the *nhsd.ProviderIndividualIdentifier* table based on the values of the Type.Code and Value properties of the LocalIdentifier property of the request.
- 4 If the record does not exist, return an error.
- 5 Locate the corresponding *nhsd.ProviderIndividual* record and verify both previously located identifiers are associated with the same provider individual.
- 6 If the values do not match, return an error.
- 7 Otherwise, delete the existing "local" record from the *nhsd.ProviderIndividualIdentifier* table.
- 8 Construct and return a *DeleteLocalIdentifierResponse*.

### ***GetValidatedHpii***

Attempts to obtain a validated HPI-I for use in a new eHealth message or document to identify the author or recipient individual.

*Logic:*

- 1 Identify the *nhsd.ProviderIndividual* record corresponding to the value of the Identifier property of the request:
  - a Locate the record representing the provider individual in the *nhsd.ProviderIndividualIdentifier* table based on the values of the Identifier.Type.Code and Identifier.Value properties of the Identifier property of the request.
  - b Locate the *nhsd.ProviderIndividual* record based on the shared ProviderIndividualId.
  - c If either record does not exist, return an error.
- 2 Locate the *p2p.ProviderIndividualHpii* record representing the provider individual's HPI-I using the shared *ProviderIndividualId*, and the *AccessingOrganisationNetworkId* for the current accessing organisation context.
- 3 Retrieve the configured "validity period" (default and maximum is 24 hours) from configuration.
- 4 Determine if the provider individual's HPI-I needs to be revalidated. It must be revalidated if:
  - a There is no HPI-I record; OR
  - b The status of the HPI-I record indicates there are NO unresolved exceptions or alerts associated with the identifier (CCA CR 10613); AND
  - c The value of the DateLastValidated column in the record is less than the current date/time minus the configured "validity period".
- 5 If the provider individual's HPI-I needs to be revalidated:
  - a Invoke service operations from the HpiiService in the HIPS \_AppServer with the required parameters until there is a match found or all available search criteria has been tried.
    - i If an HPI-I is assigned to the record, invoke searches with the HPI-I for each family name via the "HpiiValidation" service operation.
      - If the original request included an HPI-I, or there is an existing HPI-I recorded, and the response from the service invocation includes an HPI-I that is different together with a message

- indicating that the HPI-I has been resolved, repeat step (i) with the newly received HPI-I.
    - ii If no HPI-I is assigned but one or more registration IDs are assigned to the record, invoke searches for each combination of registration ID and family name via the "HpIiValidation" service operation.
    - iii If no HPI-I is assigned but the sex and date of birth, and one or more addresses containing street name, suburb, state and postcode are assigned to the record, invoke searches with the sex, date of birth and each combination of address and family name via the "HpIiDemographicSearch" service operation.
  - b If there is no existing HPI-I record and no HPI-I was able to be obtained, return a *ProviderIdentifierAcquisitionFault*.
  - c Save changes:
    - i If there is an existing HPI-I record and no HPI-I was able to be obtained:
      - Set the *ProviderIdentifierStatusId* to the status with code "UA-SEND" to indicate an alert as per CCA CR 17571.
    - ii If there is an existing HPI-I record, and the HPI-I obtained was resolved but could not be validated:
      - Set the *ProviderIdentifierStatusId* to the status with code "UA-RESOLVED" to indicate an alert as per CCA CR 17571.
    - iii Otherwise, create or update the record in the *p2p.ProviderOrganisationHpIi* table with the individual's *ProviderIndividualId* and the *AccessingOrganisationNetworkId* of the current accessing organisation network context.
      - Set *DateLastValidated* to the current date/time
      - Set *Value* to the 16-digit HPI-I from the service response
      - Set *Status* to the status from the service response
  - d Record changes if any were made:
    - i If updates were made to any provider individual attribute except *DateLastValidated*, then insert a record in *p2p.Change* to indicate a modification of the individual.
- 6 If the HPI-I required validation but it could not be validated, return a *ProviderIdentifierValidationFault*.
- 7 If there are unresolved exceptions or alerts associated with the HPI-I, return a *ProviderIdentifierValidationFault*.
- 8 Construct and return a *GetValidatedHpIiResponse* containing a *ValidatedHpIi* instance based on the information in the *nhsd* and *p2p* schemas:
  - a Identifiers: *nhsd.ProviderIndividualIdentifier*
  - b Name: *nhsd.ProviderIndividual*
  - c *DateLastValidated*: *p2p.ProviderIndividualHpIi.DateLastValidated*
  - d *Value*: *p2p.ProviderIndividualHpIi.Value*
  - e *Status*: *p2p.ProviderIndividualHpIi.ProviderIdentifierStatusId*



**NOTE:** If the status of the HPI-I record for the provider individual in the *nhsd.ProviderIndividualIdentifier* table is “retired”, “resolved” or “deactivated”:

- A warning message must be returned in the Messages property of the response.
- The Status of the response must be set to “Warning”.

### **Search**

Searches the LHSD by individual, link, location or organisation criteria, finds matching individuals, and returns associated entities that are specified in a list of entities.

*Logic:*

- 1 Transform the search criteria to the common model.
- 2 If individual criteria was supplied, obtain the primary key of individuals who match the following criteria:
  - a Sex if supplied
  - b Date of Birth if supplied
  - c Family name if supplied
  - d Given name if supplied
  - e Middle name if supplied
  - f Title if supplied
    - i If an individual identifier is supplied:
      - ii identifier type code
    - g value
    - h If a name alias is supplied:
      - i known as type code
      - ii value
    - i If an electronic contact detail is supplied:
      - i contact type code
      - ii contact purpose code
      - iii details
    - j If an individual classification is supplied:
      - i provider type code
      - ii provider type description
      - iii provider specialty code
      - iv provider specialty description
      - v provider specialisation code
      - vi provider specialisation description
- 3 If link criteria was supplied, obtain the primary key of links that match the following criteria:
  - a If a link identifier is supplied:

- i type code
  - ii value
- 4 If location criteria was supplied, obtain the primary key of locations that match the following criteria:
  - a Location name if supplied
  - b Location description if supplied
  - c If a location identifier is supplied:
    - i type code
    - ii value
  - d If an address is supplied:
    - i contact purpose code
    - ii contact purpose description
    - iii flat type
    - iv flat number
    - v address site name
    - vi level type
    - vii level number
    - viii street number
    - ix lot number
    - x street name
    - xi street type
    - xii street suffix
    - xiii postal delivery type
    - xiv postal delivery number
    - xv unstructured address line
    - xvi suburb
    - xvii state
    - xviii postcode
    - xix latitude
    - xx longitude
  - e If an electronic contact detail supplied:
    - i contact purpose code
    - ii contact type code
    - iii details
  - f If a classification is supplied:
    - i organisation type code
    - ii organisation type description
    - iii organisation service type code

- iv organisation service type description
  - v organisation service unit code
  - vi organisation service unit description
- 5 If organisation criteria was supplied, obtain the primary key of organisations that match the following criteria:
  - a Legal name if supplied
  - b Organisation description if supplied
  - c If an organisation identifier is supplied:
    - i identifier type code
    - ii value
  - d If an organisation name is supplied:
    - i name usage code
    - ii value
- 6 Retrieve records from *ProviderIndividual* based on the primary keys of entities that had criteria, and apply the skip and take limits.
- 7 For each of these individual records, get the associated entities that are requested by the caller.
- 8 Transform the records to their corresponding representation in the DTO model.
- 9 Construct and return a *SearchResponse* containing the set of DTO instances.

### 2.2.5.7 Reference Data Service

The Reference Data *ReferenceDataService* in HIPS -Core provides the following functionality:

- Provide a service operation to list all the reference sets that have been configured.
- Provide a service operation to update the reference data. The service operation must allow a specified reference set to be updated, or all reference sets.
- Provide a service operation to search the reference data. The service operation must allow searching within a specified reference set, using wildcards to match the Code or Display Name attributes, and return all matching reference items.
- Provide a scheduled service to automatically update reference data according to a schedule that is configured for each reference set.

#### **List**

Returns a list of all configured reference sets.

*Logic:*

- 1 Retrieve all records from the *nhsd.ReferenceSet* table.
- 2 Transform the set of records to their corresponding representation as a set of instances of the *ReferenceSet* DTO class.
- 3 Construct and return a *ListResponse* containing the set of *ReferenceSet* instances.

**Search**

Returns a list of reference items in the specified reference set, which match the given search patterns for the code and the display name. Each search pattern may include a wildcard (asterisk) at the beginning, at the end, or both.

*Logic:*

- 1 Convert each search pattern into criteria using the appropriate string operation (*Equals*, *Contains*, *StartsWith* or *EndsWith*) depending on the position of the asterisks, if any.
- 2 Retrieve all records from the *nhsd.ReferenceItem* table using the following criteria:
  - a If *CodePattern* was provided, the *Code* matches the criteria
  - b If *DisplayNamePattern* was provided, the *DisplayName* matches the criteria
  - c The associated *nhsd.ReferenceSet* record's *Uri* is the given *ReferenceSetUri*
- 3 Transform the set of records to their corresponding representation as a set of instances of the *ReferenceItem* DTO class.
- 4 Construct and return a *SearchResponse* containing the set of *ReferenceItem* instances.

**Update**

Downloads and stores the latest reference items using the NEPS Reference Data Service.

*Logic:*

- 1 Retrieve all records from the *nhsd.ReferenceSet* table using the following criteria:
  - a If *ReferenceSetUri* was provided, the *Uri* is the given *ReferenceSetUri*
- 2 For each reference set:
  - a If *ForceRefresh* is false, and the set was updated within the number of days configured in the *UpdateFrequency* column, then skip this set.
  - b Retrieve the current set of reference items from NEPS. Audit the invocation.
  - c If unable to retrieve any items, add the error to a list and skip this set.
  - d Add or update each item in the database using the URI as the lookup key.
  - e Store the current date and time as the last updated for the set.
- 3 If the NEPS service failed to retrieve items for any of the sets, return a *ServiceInvocationFault* with details of the errors returned by NEPS.
- 4 Otherwise construct and return an *UpdateResponse*.

**2.2.5.8 Referenced Data Scheduled Service**

The *ScheduledReferenceDataService* in HIPS Core provides a scheduled service to automatically update reference data according to a schedule that is configured for each reference set. The *ScheduledReferenceDataService* is implemented as a scheduled service that executes at a configurable interval to update reference items in the P2P database.

### **2.2.5.9 Directory Synchronisation Scheduled Service**

The *ScheduledDirectorySynchronisationService* in HIPS Core provides a scheduled service operation to periodically synchronise NEPS subscription results to the LHSD. The *ScheduledDirectorySynchronisationService* is implemented as a scheduled service that executes at a configurable interval to update provider information in the P2P database.

## **2.3 Database Resource Access Layer**

The P2P Data Store is a new database for HIPS P2P that is separate from the existing HIPS Core database. The tables in this database are designed to support addressing, sending and receiving healthcare information.

As with the HIPS Core database, all tables in the P2P Data Store will contain standardised auditing columns DateCreated, UserCreated, DateModified and UserModified; these columns are not shown in the diagrams below.

The key requirements for the design were:

- Support Secure Message Delivery Management.
- Support the NEPS Subscription Maintenance.
- Support the LSHD Maintenance, including Secure Message Endpoints, Provider Individuals and Provider Organisations.
- LHSD Change Retrieval.

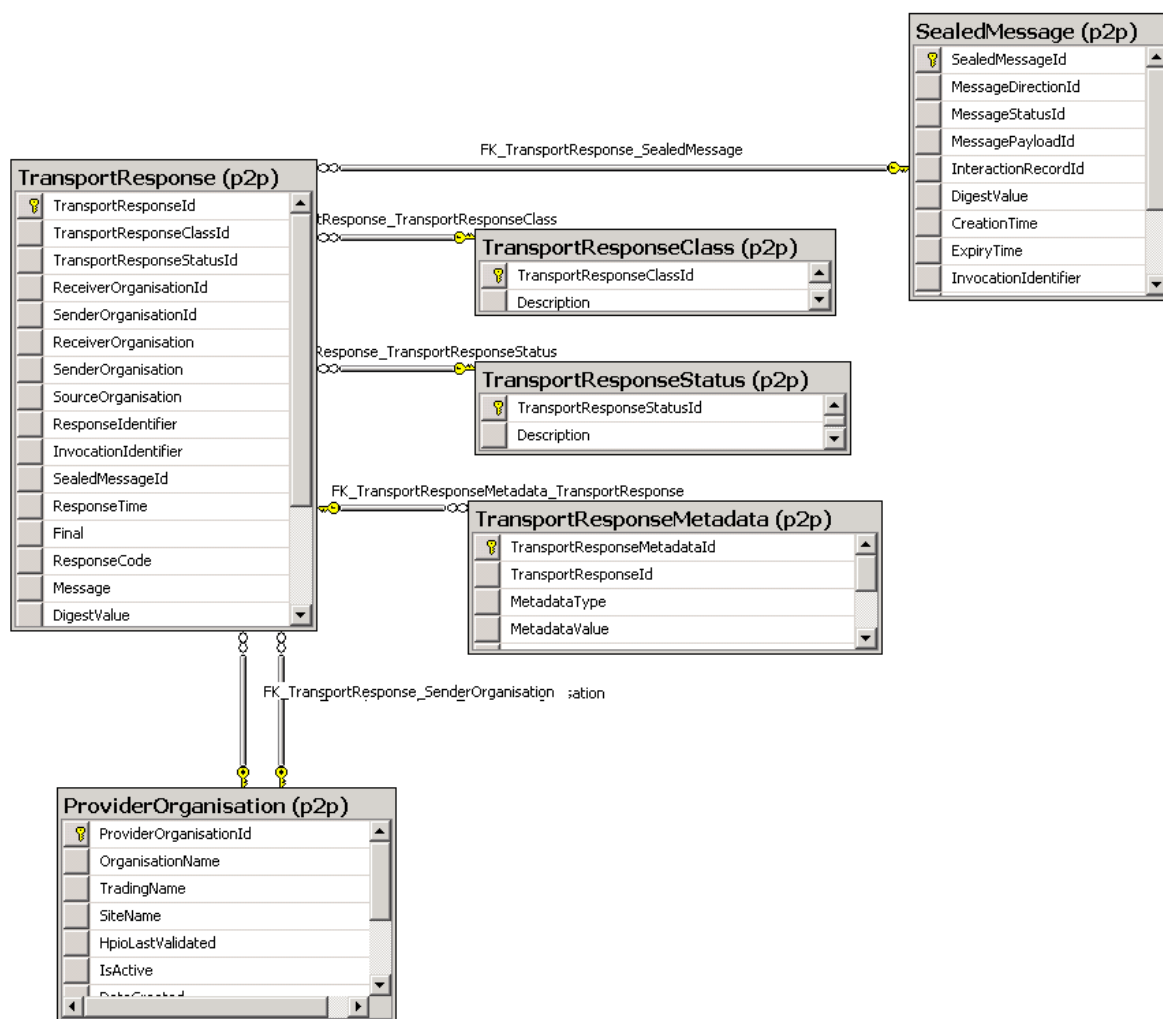
### **2.3.1 Secure Message Delivery Management**

The Message component of the P2P Data Store represents message payloads and the messages in which they are sent and received, supporting Secure Message Delivery and Secure Message Delivery Management.



Schema	Table	Description
p2p	Message Payload	<p>A message payload, which is unchanged regardless of how many sealed messages are created while attempting to deliver the message.</p> <p>The identifiers of the organisation and individual who play the role of sender and receiver of the message are stored in this table because these entities may not exist in the LHSD. Each message payload is linked to the LHSD records for organisations and individuals that represent the sender and receiver of the message, if such LHSD records exist.</p> <p>For message payloads that are CDA documents, HIPS will extract from the message key demographic details of the patient and store these details into this table:</p> <ul style="list-style-type: none"> <li>• IHI</li> <li>• Family Name</li> <li>• Given Names</li> <li>• Sex</li> <li>• Date of Birth</li> </ul>
p2p	Sealed Message	<p>A sealed message, uniquely identified by an invocation identifier. Each sealed message has a specific creation time and expiry time, and is delivered using a specific interaction record.</p> <p>When a user manually retries the sending of a message, a new sealed message with a new invocation identifier is created. The new sealed message references the original message payload.</p>
p2p	Message Direction	Enumeration table used to differentiate outbound and inbound messages.
p2p	Message Status	Enumeration table used to indicate the status of delivery or processing of a message.
p2p	Sealed Message Metadata	Stores a set of name/value pairs with additional metadata associated with a sealed message.
p2p	Interaction Record	<p>Represents the mechanism of delivery for the sealed message, including the service interface, endpoint URL and certificate used to encrypt the message.</p> <p>If the interaction record's delegate has an intermediary defined, then the endpoint URL of the intermediary is used in place of the endpoint URL in this table.</p>
p2p	Payload Scheme	Represents the type of payload, consisting of a document type and a packaging format.
p2p	Document Type	Represents the type of document, such as Discharge Summary, Event Summary, Specialist Letter or eReferral.
p2p	Payload Packaging	Represents the packaging format for the document, such as XDM-ZIP, HL7-MDM or raw binary.
p2p	Provider Organisation	Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS.
p2p	Provider Individual	Contains a record for each Healthcare Provider Individual whose information has been obtained from the NHSD via NEPS.

The Transport Response component of the P2P Data Store also supports Secure Message Delivery and Secure Message Delivery Management.



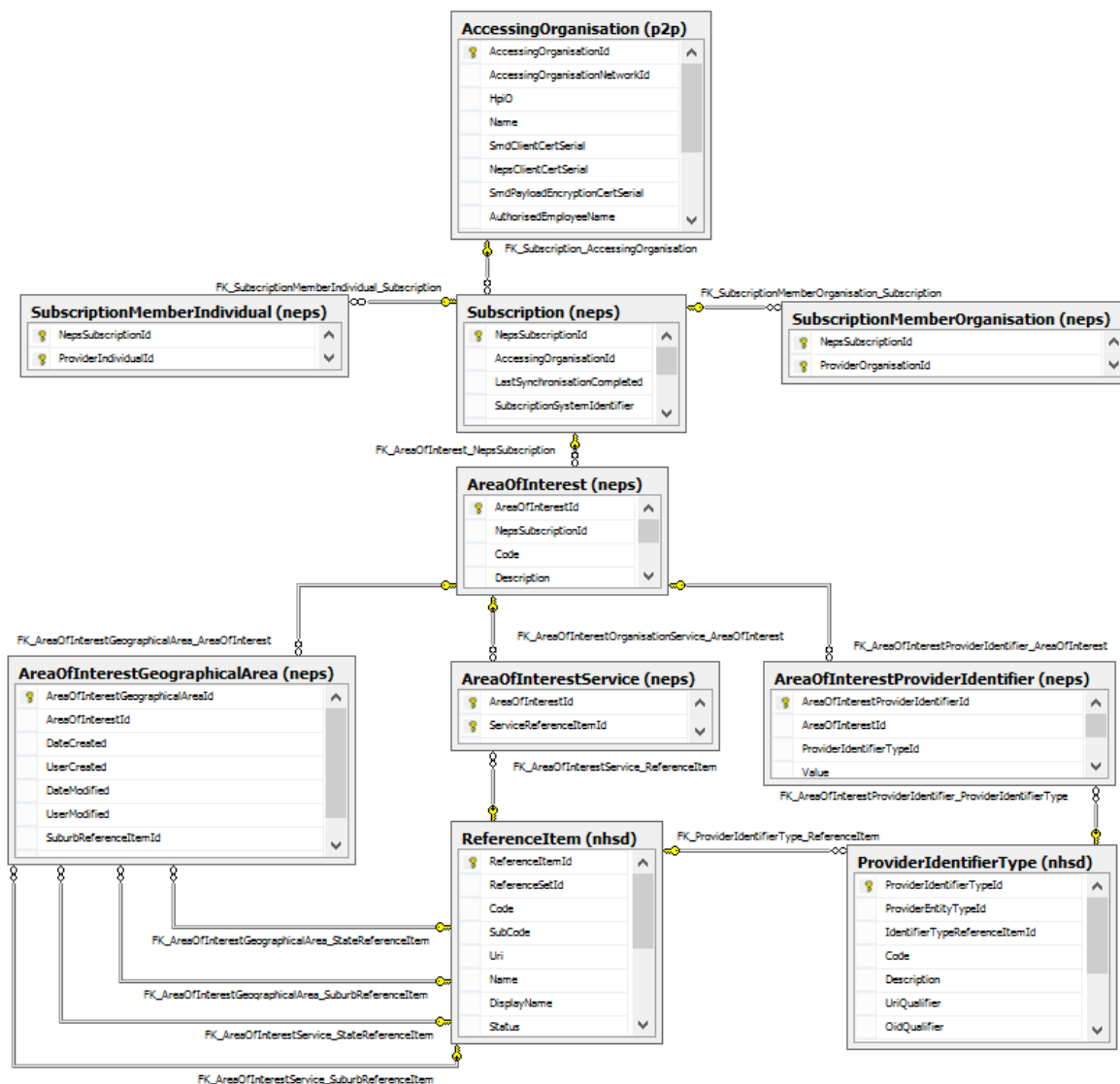
Schema	Table	Description
p2p	Transport Response	<p>A transport response, uniquely identified by a response identifier, and linked to a sealed message via the invocation identifier.</p> <p>Each transport response is linked to the LHSD records for the receiver and sender organisation, if such LHSD records exist.</p> <p>Transport responses for outbound messages may be received from intermediaries or the ultimate receiver, or created locally as a record of a validation failure while attempting to deliver the message.</p> <p>Transport responses for inbound messages are created locally and remain undelivered until delivery is successful, expires or is aborted.</p>
p2p	Transport Response Class	Enumeration table containing entries for Success, Information, Warning and Error as defined in ATS 5822-2010.
p2p	Transport Response Status	Enumeration table used to indicate the status of processing or delivery of a transport response.
p2p	Transport Response Metadata	Stores a set of name/value pairs with additional metadata associated with a transport response.



Schema	Table	Description
p2p	Provider Organisation	Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS.
p2p	Sealed Message	<p>A sealed message, uniquely identified by an invocation identifier. Each sealed message has a specific creation time and expiry time, and is delivered using a specific interaction record.</p> <p>Where the transport response is able to be correlated with a record in the Sealed Message table (based on the shared Invocation Identifier), the Transport Response record will include the primary key of the associated Sealed Message.</p>

### 2.3.2 NEPS Subscription

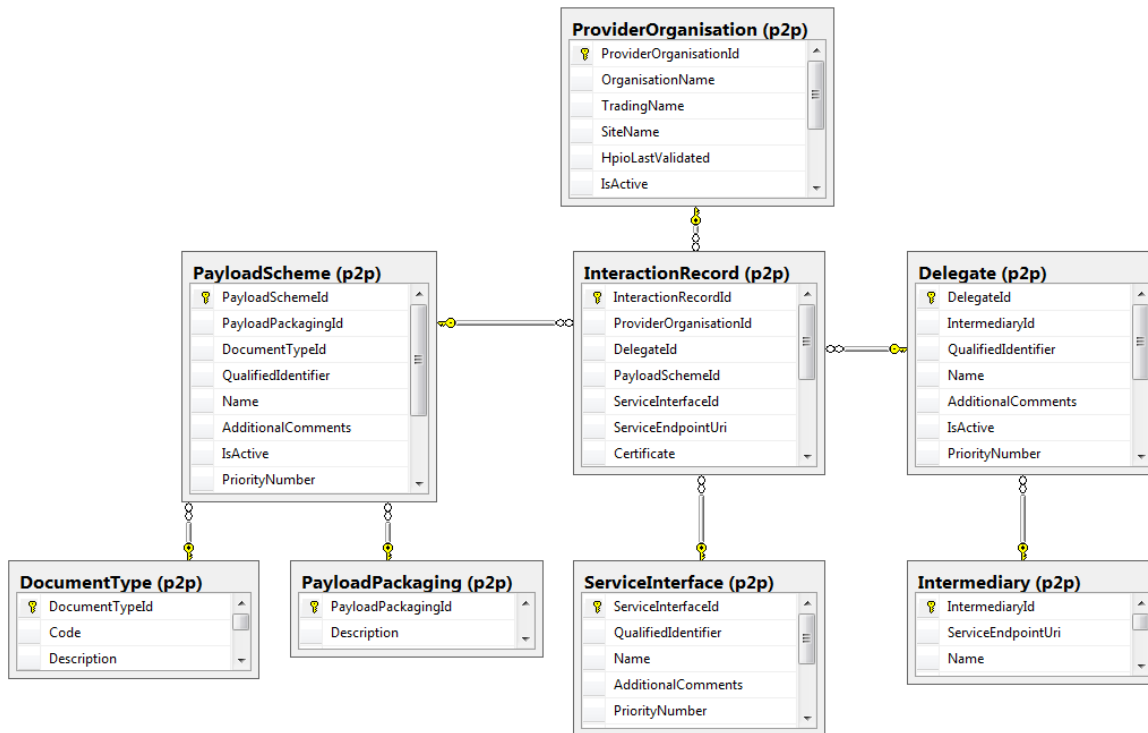
The NEPS Subscription component of the P2P Data Store has been implemented to support the NEPS Subscription Maintenance.



Schema	Table	Description
neps	Subscription	Contains a record for each subscription to NEPS held by a Health Provider Organisation within the jurisdiction. A jurisdiction may create multiple subscriptions (at most one per HPI-O) and nominate an authorised employee for each subscription. The HIPS instance will retrieve updates for each subscription using the HPI-O and authorised employee associated with that subscription. This table also stores the date and time when updates for this subscription were last received and processed by HIPS.
p2p	Accessing Organisation Network	Contains an entry for each network of organisations within a multitenant instance of P2P. This ensures that the HPI-O's and HPI-I's are obtained independently for each organisation network.
neps	Pending Change	<p>Pending changes to subscription members that HIPS has not yet applied to the LHSD database.</p> <p>When adding, updating or deleting an area of interest, the identifiers of added or removed members are stored as Pending Change records. When retrieving subscription changes, the identifiers of added, changed or removed members are also stored as Pending Change records.</p> <p>The background synchronisation service reads from this table and deletes the pending change records after it has applied the changes into the LHSD database.</p>
neps	Subscription Member Individual	A provider individual who is a member of a subscription.
neps	Subscription Member Organisation	A provider organisation that is a member of a subscription.
neps	Area Of Interest	An area of interest definition within a subscription.
neps	Area Of Interest Provider Identifier	A provider identifier that has been explicitly added to of an area of interest definition. The type of provider identifier is determined via the link to the Provider Identifier Type table. Only the NHSD URI for an individual or an organisation may be added to an area of interest.
neps	Area Of Interest Service	An organisation type or individual occupation that makes up part of an area of interest definition. Links to organisation type, service type, service unit, and individual type, specialty or specialisation reference items.
neps	Area Of Interest Geographical Area	A geographical area that makes up part of an area of interest definition. Links to suburb and state reference items.
nhsd	Provider Identifier Type	Configurable reference data table containing an entry for each type of identifier that can be assigned to a provider individual, organisation or link. Examples are: NHSD URI for Individual, NHSD URI for Organisation. Jurisdictions may add their own definitions to this table.

### 2.3.3 Secure Message Endpoint Location

The Secure Message Endpoint Location component of the P2P Data Store has been implemented to support the LHSD Maintenance.

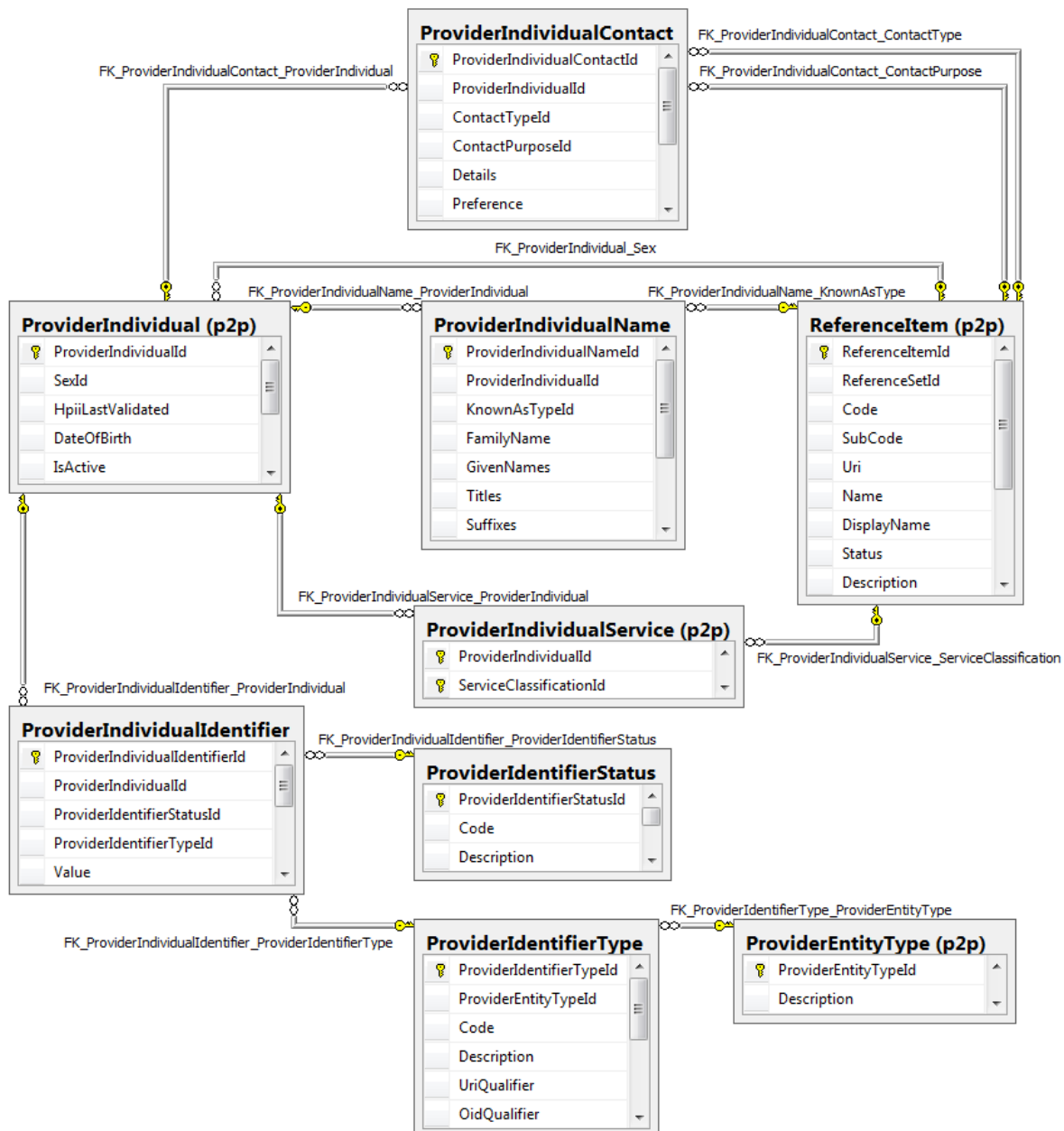


Schema	Table	Description
p2p	Provider Organisation	Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS.
p2p	Interaction Record	Contains the technical information needed to invoke a target service instance, to deliver a payload towards a target organisation. An interaction record consists of a target provider organisation, payload scheme, delegate, certificate used to encrypt the payload, service interface and endpoint URL to invoke the service. Interaction records are obtained from NEPS.
p2p	Payload Scheme (aka Service Category)	Identifies a certain type of payload and its packaging format. Payload schemes are not obtained from NEPS. This table will be pre-populated with the predefined service categories for delivery of CDA clinical documents in XDM-ZIP and MDM packaging formats. Jurisdictions can add other payload schemes if required to support additional types of payload. Each payload scheme may be associated with a type of CDA document and a payload packaging form. If the payload scheme is not associated with a document type, the payload will be treated as a raw binary payload.

Schema	Table	Description
p2p	Document Type	<p>Reference data table containing types of clinical document, for example Discharge Summary, Event Summary, Specialist Letter or eReferral. This table will be pre-populated but can be extended by the jurisdiction.</p> <p>This table contains document types supported for point-to-point sending and receiving, and is not to be confused with the table of the same name in the HIPS Core database that contains document types supported for upload to the My Health Record system.</p>
p2p	Payload Packaging	Identifies a form of packaging or wrapping to be applied to a CDA document, for example XDM-ZIP or MDM. This table will be pre-populated.
p2p	Delegate (aka Service Provider)	Identifies the organisation that is responsible for the technical operation of the target service instance. This may be the target health provider organisation but is more likely to be a messaging vendor. Delegates are obtained from NEPS, but jurisdictions can set the priority of each delegate to reflect the cost of sending messages via that delegate, to influence HIPS decision of which interaction record to use and ensure the optimal route is chosen.
p2p	Intermediary	Identifies an alternative endpoint that will accept messages and forward them on to the target organisation. If there is an intermediary defined for the delegate of the interaction record, HIPS will invoke the endpoint of the intermediary instead of directly invoking the endpoint identified in the interaction record. Jurisdictions may wish to set an intermediary and assign it to be used in place of certain delegates if those delegates will not accept messages directly from the jurisdiction.
p2p	Service Interface	<p>Identifies a web service interface and version. This table will be pre-populated with service interface values defined in ATS 5822-2010, for example the following QualifiedIdentifier represents the 2010 version of Sealed Message Delivery using the TLS Security Profile:</p> <p><a href="http://ns.electronichealth.net.au/smd/intf/SealedMessageDelivery/TLS/2010">http://ns.electronichealth.net.au/smd/intf/SealedMessageDelivery/TLS/2010</a>.</p>

### 2.3.4 Provider Individual

The Provider Individual component of the P2P Data Store is required for the LHSD Maintenance.

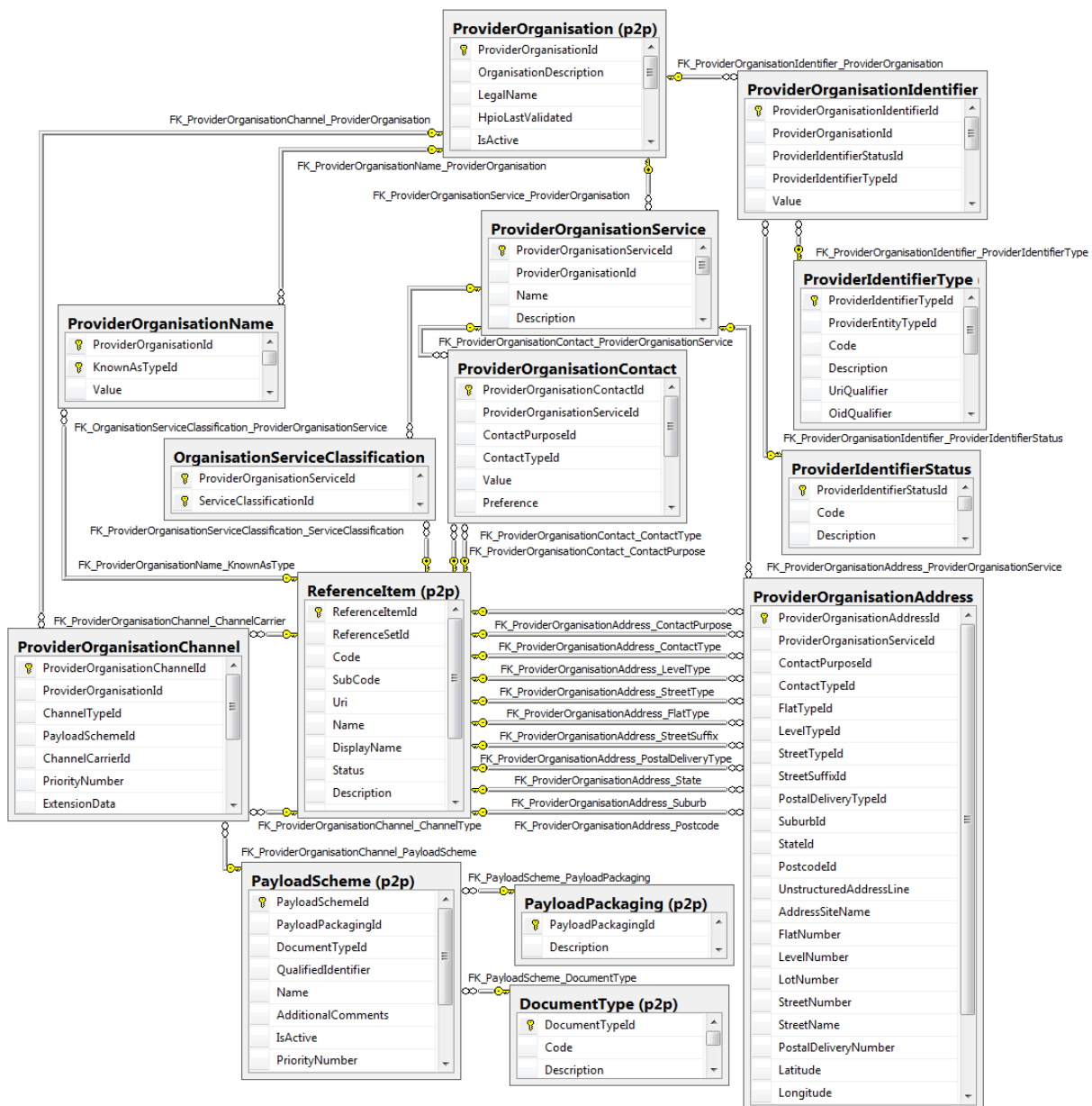


Schema	Table	Description
p2p	Provider Individual	Contains a record for each Healthcare Provider Individual whose information has been obtained from the NHSD via NEPS. <ul style="list-style-type: none"> <li>• Date of birth</li> <li>• Timestamp when HPI-I was last validated</li> <li>• Whether the individual is part of the active set, covered by a subscription to NEPS.</li> </ul>
p2p	Provider Individual Contact	Stores a set of electronic contact details for the individual, for example their mobile phone number and/or email address.

Schema	Table	Description
p2p	Provider Individual Name	Stores the set of names that the individual is known by.
p2p	Provider Individual Service	Stores the set of provider types, specialties and/or specialisations that the individual belongs to.
p2p	Reference Item	Reference data table containing the NEPS-provided codes including: <ul style="list-style-type: none"> <li>• Contact type</li> <li>• Contact purpose</li> <li>• Sex</li> <li>• Name usage</li> <li>• Individual service classification</li> </ul>
p2p	Provider Individual Identifier	Contains external identifiers of the individual provider. The NEPS ID, NHSD ID and HPI-I are stored here. Jurisdictions may add locally assigned identifiers of the healthcare individual.
p2p	Provider Identifier Status	Reference data table containing the various status that an HPI-I, HPI-O or other identifier may have, for example, Active, Deactivated, Retired, or Resolved.
p2p	Provider Identifier Type	Reference data table containing an entry for each type of identifier that can be assigned to a provider individual, organisation or link. Examples are: NEPS ID, NHSD ID, HPI-I, HPI-O. Jurisdictions may add their own definitions to this table.
p2p	Provider Entity Type	Reference data table containing an entry for each of the three logical entities (Individual, Organisation and Link).

## 2.3.5 Provider Organisation

The Provider Organisation component of the P2P Data Store is required for the LHSD Maintenance.



Schema	Table	Description
p2p	Provider Organisation	<p>Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS. In this table are stored the:</p> <ul style="list-style-type: none"> <li>• Legal Name</li> <li>• Organisation Description</li> <li>• Timestamp when HPI-O was last validated</li> <li>• Whether the organisation is part of the active set, covered by a subscription to NEPS.</li> </ul>

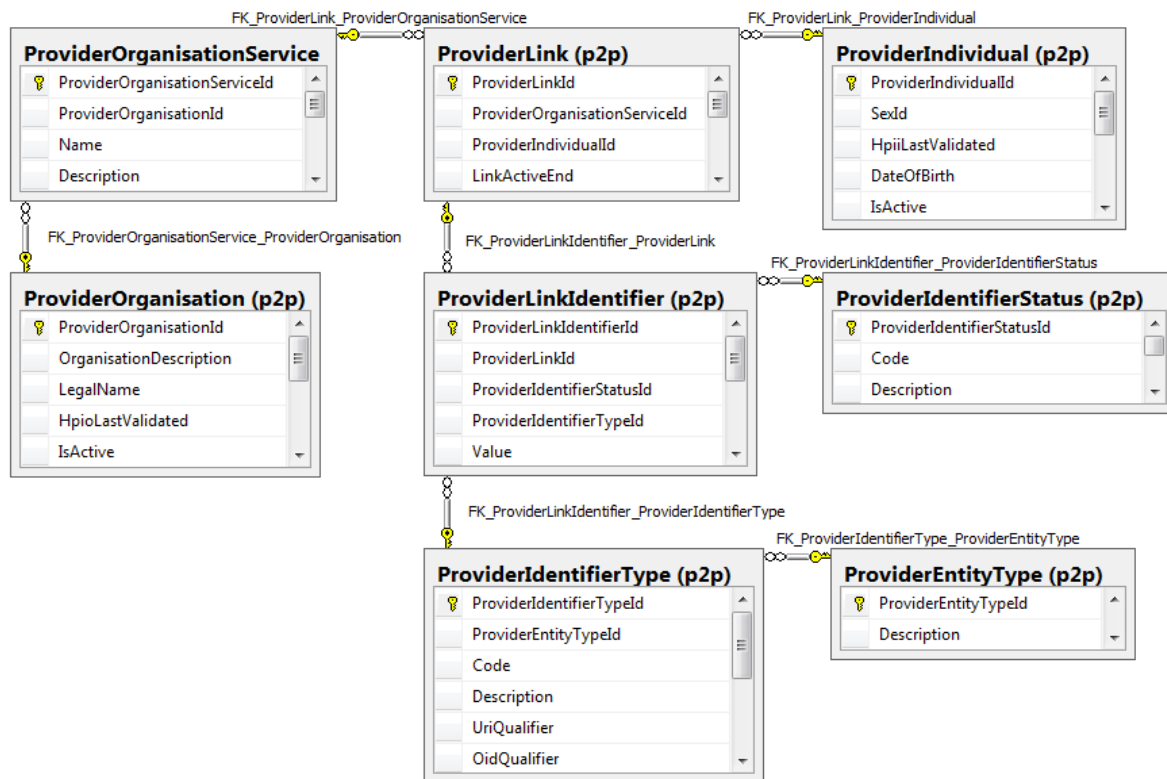
Schema	Table	Description
p2p	Provider Organisation Identifier	Contains external identifiers of the organisation. The NEPS ID, NHSD ID and HPI-O are stored here. Jurisdictions may add locally assigned identifiers of the healthcare organisation.
p2p	Provider Identifier Status	Reference data table containing the various status that an HPI-I, HPI-O or other identifier may have, for example, Active, Deactivated, Retired, or Resolved.
p2p	Provider Identifier Type	Reference data table containing an entry for each type of identifier that can be assigned to a provider individual, organisation or link. Examples are: NEPS Individual ID, NEPS Organisation ID, HPI-I, HPI-O. Jurisdictions may add their own definitions to this table.
p2p	Provider Entity Type	Reference data table containing an entry for each of the three logical entities (Individual, Organisation and Link).
p2p	Provider Organisation Service	Contains a record for each service location operated by the organisation.
p2p	Organisation Service Classification	Stores the service classifications (organisation types, services and service units) that apply to the organisation service location.
p2p	Provider Organisation Name	Stores the set of names associated with the organisation. Each name is qualified by a name usage or 'Known As' code.
p2p	Reference Item	Reference data table containing the NEPS-provided codes for: <ul style="list-style-type: none"> <li>• service classification</li> <li>• name usage</li> <li>• contact purpose (usage)</li> <li>• contact type (medium)</li> <li>• channel type</li> <li>• channel carrier</li> <li>• various address components</li> </ul>
p2p	Provider Organisation Contact	Stores a set of electronic contact details for an organisation service location, for example their phone number, fax number and email address.
p2p	Provider Organisation Address	Stores a set of addresses for the organisation service location.
p2p	Provider Organisation Channel	Stores a set of channel preferences for the organisation. Each channel preference indicates a channel through which the organisation wants to receive a certain type of clinical documents. There can be multiple acceptable channels for the same or different payload schemes, ordered by preference.



Schema	Table	Description
p2p	Payload Scheme	Identifies a certain type of payload and its packaging format. Payload schemes are not obtained from NEPS. This table will be pre-populated with the predefined service categories for delivery of CDA clinical documents in XDM-ZIP and MDM packaging formats. Jurisdictions can add other payload schemes if required to support additional types of payload. Each payload scheme may be associated with a type of CDA document and a payload packaging form. If the payload scheme is not associated with a document type, the payload will be treated as a raw binary payload.
p2p	Payload Packaging	Identifies a form of packaging or wrapping to be applied to a CDA document, for example XDM-ZIP or MDM. This table will be pre-populated.
p2p	Document Type	Reference data table containing types of clinical document that are supported for point-to-point sending and receiving, for example Discharge Summary, Event Summary, Specialist Letter or eReferral. This table will be pre-populated but can be extended by the jurisdiction.
p2p	Channel Carrier	Reference data table containing NEPS-provided codes that identify a messaging vendor that will accept secured email, such as Argus, HealthLink or Medical Objects.
p2p	Channel Type	Reference data table containing NEPS-provided codes that identify a type of channel, such as Post, Fax, Secured Email or SMD.

### 2.3.6 Provider Link

The Provider Link component of the P2P Data Store is required for LHSD Maintenance. The provider link represents the relationship between an individual provider and the organisation services that employ the individual.

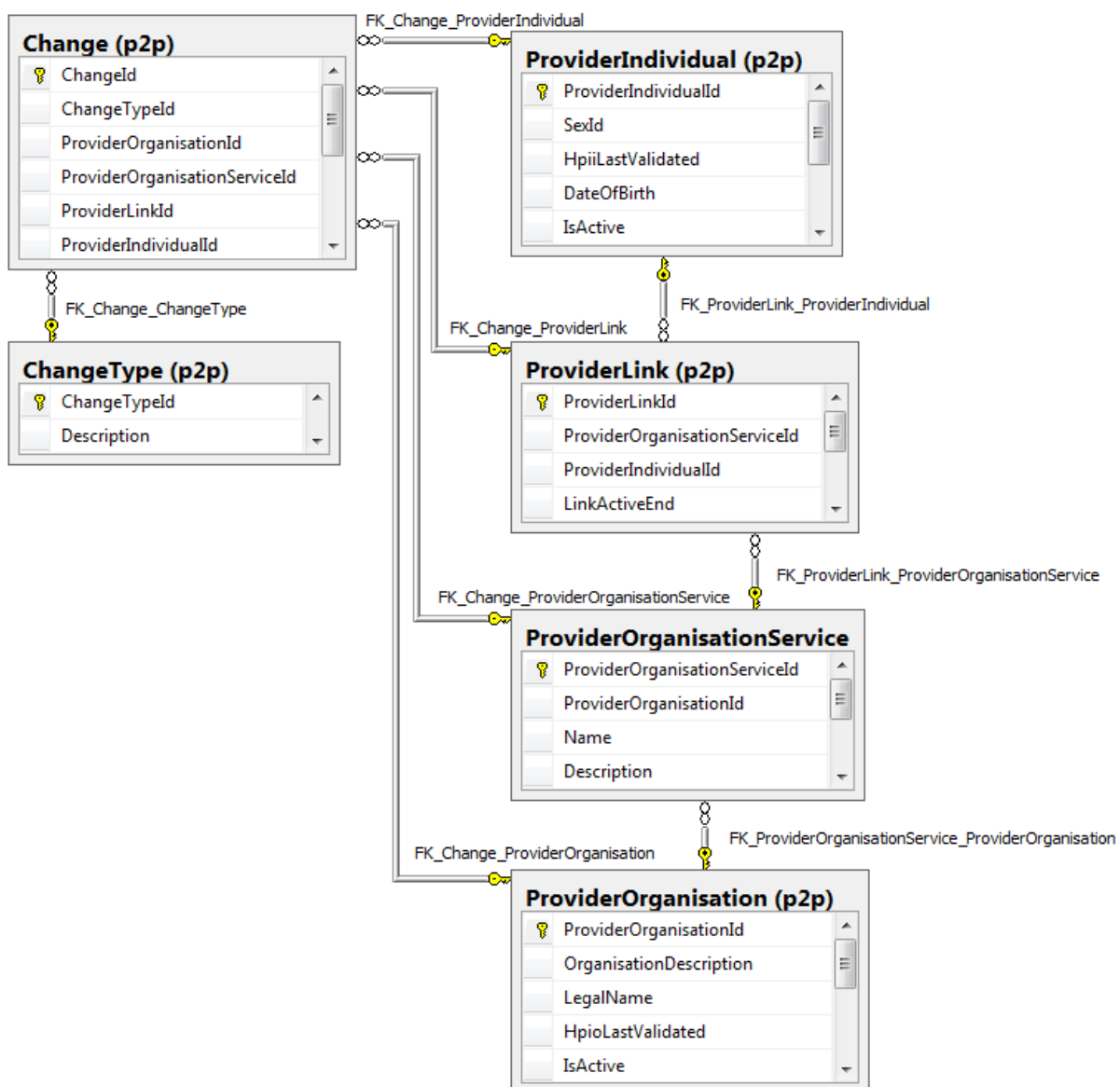


Schema	Table	Description
p2p	Provider Link	Contains a record for each association between a Healthcare Provider Individual and a service location of a Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS. When the link becomes inactive, the date on which the individual left the organisation is stored in this table.
p2p	Provider Individual	Contains a record for each Healthcare Provider Individual whose information has been obtained from the NHSD via NEPS.
p2p	Provider Organisation	Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS.
p2p	Provider Link Identifier	Contains external identifiers of the link. Jurisdictions may add locally assigned identifiers of the link. The Medicare Provider Number is an example of an identifier that is assigned for each organisation service location where the individual works and would logically be stored as a Provider Link Identifier.
p2p	Provider Identifier Status	Reference data table containing the various status that an identifier may have, for example, Active, Deactivated, Retired, or Resolved.

Schema	Table	Description
p2p	Provider Identifier Type	Reference data table containing an entry for each type of identifier that can be assigned to a provider individual, organisation or link. Examples are: NEPS ID, NHSD ID, HPI-I, HPI-O. Jurisdictions may add their own definitions to this table.
p2p	Provider Entity Type	Reference data table containing an entry for each of the three types of logical entity (Individual, Organisation and Link).

### 2.3.7 Change Tracking

The Change Tracking component of the P2P Data Store is required for LHSD Change Retrieval. Changes are tracked at the level of the individual, link, organisation service and organisation.



Schema	Table	Description
p2p	Change	<p>Contains a record for each tracked change to a logical entity whose information has been obtained from the NHSD via NEPS. This table stores the following information:</p> <ol style="list-style-type: none"><li>1. Sequentially allocated change number</li><li>2. Date and time when the change was recorded</li></ol> <p>Each change also references a change type and one logical entity, which may be an Organisation, Organisation Service, Individual or Link.</p>
p2p	Change Type	Reference data table containing the various types of change that are tracked (Creation, Modification and Deactivation).
p2p	Provider Organisation	Contains a record for each Healthcare Provider Organisation whose information has been obtained from the NHSD via NEPS.
p2p	Provider Organisation Service	Contains a record for each location at which an organisation provides healthcare services.
p2p	Provider Link	Contains a record for each association between a healthcare provider individual and a healthcare provider organisation service location.
p2p	Provider Individual	Contains a record for each Healthcare Provider Individual whose information has been obtained from the NHSD via NEPS.