Australian Government

Australian Digital Health Agency

HIPS
Initial and Clean Installation Guide (Core)

22 January 2019 v7.0

Approved for external use

Australian Digital Health Agency ABN 84 425 496 912, Level 25, 175 Liverpool Street, Sydney, NSW 2000
Telephone 1300 901 001 or email help@digitalhealth.gov.au
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 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.

Document information

Key information

Owner	Executive General Manager Innovation and Development
Contact for enquiries	Australian Digital Health Agency Help Centre Phone 1300 901 001 Email help@digitalhealth.gov.au

Product or document version history

Product or document version	Date	Release comments
1.0	February 2014	Initial release based on HIPS v4.1.
2.0	February 2015	See release note (NEHTA-2040:2015) for details of changes and bug fixes.
2.0.1 2.0.2		Unpublished updates.
2.0.3	February 2016	See release note (NEHTA-2185:2016) for details of changes and bug fixes.
6.0.0	March 2016	See release note (NEHTA-2263:2016) for details of changes and bug fixes.
6.1	November 2016	See release note (DH-2445:2016) for details of changes and bug fixes.
6.1.1	March 2018	See release note for details of changes and bug fixes.
6.1.2	May 2018	See release note for details of changes and bug fixes.
6.1.3	July 2018	See release note for details of changes and bug fixes.
6.1.4	September 2018	See release note for details of changes and bug fixes.
6.2 6.2.1	Unpublished	See release note for details of changes and bug fixes.
7.0.0	December 2018	See release note for details of changes and bug fixes.

Table of contents

1 Important Information

It is important to note that this installation document has been written as an installation guide for both a test or production environment of the HIPS product suite.

1.1 Global Unique UUID

An instance identifier may be of the UUID form or an object identifier (OID) form. The algorithm for creating a UUID ensures the uniqueness of the identifier whereas the OID form requires the software to have logic to ensure uniqueness.

When HIPS creates a unique CDA document identifier and CDA document set identifier for Discharge Summary, Pathology Report and Diagnostic Imaging Report documents, HIPS uses the OID option where achieving a unique document identifier relies on the value of the extension attribute being unique within the healthcare provider organisation.

Here is an example of document number 2143 created by the organisation DHSITESTORGD46 with HPI-O number 8003621566687292:

```
<id extension="2143" root="1.2.36.1.2001.1005.49.1.8003621566687292
.2143"
      assigningAuthorityName="HIPS DHSITESTORGD46 Doc"/>
```

According to the data types specification “The root and extension scheme means that the concatenation of root and extension shall be a globally unique identifier for the item that this value identifies”.

To ensure global uniqueness across an organisation, it is recommended that you do not configure the same HPI-O in more than one HIPS Core database. If it is necessary to do so, the numeric seed value for CDA document identifier and CDA document set identifier can be adjusted to start from a different number in each HIPS Core database using the commands:

```
DBCC CHECKIDENT ('hips.CdaDocumentNumber', RESEED, nnnnn)

DBCC CHECKIDENT ('hips.CdaSetNumber', RESEED, nnnnn)
```

Where *nnnnn* is the number to start from in each database. The available range is from 1 to 2,147,483,647 (231-1).

2 Introduction

HIPS is a communications solution to enable Patient Administration Systems and Clinical Information Systems to interact with the National My Health Record System.

The solution supplies a SOAP web service interface for integration with an Enterprise Service Bus (ESB) or other integration systems to receive HL7 V2 records from the PAS/LIS/RIS systems for patient and episode information and IHI lookups, uploading and removing pathology and diagnostic imaging reports, and to receive other CDA documents from the clinical systems for upload to the My Health Record system.

If Mirth Connect is set up and configured, then the solution can also accept HL7 V2 messages using HL7 Minimal Lower Layer Protocol (MLLP) or file drop interfaces. The solution can also be used as a broker to the digital health record system without the need of an interface to an ESB for upload and retrieval of documents from the national digital health record system.

The Australian Digital Health Agency has completed system testing, performance testing, Notice of Connection (NOC) to HI Service and My Health Record and conformance assessments of HIPS Release 7.

2.1 Purpose

The purpose of this document is to provide the technical details and steps required to install the 7 version of the HIPS product suite.

It can be used by healthcare provider organisations to install the HIPS product suite into a targeted environment.

This document describes the prerequisites and steps that are required for the HIPS product suite to be installed for the first time (clean installation).

2.2 Scope

This document covers prerequisites required for the targeted environment, the database server preparation and application server operating system preparation. The document will then go on to describe the steps required for the installation of the HIPS on the database server and application server(s).

This document does not describe any functional requirements or features of the HIPS product suite as these are covered by other documentation.

2.3 Assumptions

The following assumptions have been made during the development of this document:

- The user carrying out the installation has server administration access to the targeted database server and application server(s).
- The facility installing the HIPS product suite has appropriate software versions and server operating systems.

2.4 Definitions and Acronyms

Item	**Definition**
CAP	Conformance Assessment Process including CCA, CCD and NOC.
CCA	Conformance Compliance Assessment required before connection to production HI Service
CCD	Conformance Compliance Declaration required before connection to production My Health Record system
CDA®	HL7™ Version 3 Clinical Document Architecture
CIS	Clinical Information System
ESB	Enterprise Service Bus
IHI	Individual Healthcare Identifier
HI	Healthcare Identifiers including IHI, HPI-I and HPI-O
HL7™	Health Level Seven International
HPI-I	Healthcare Provider Identifier – Individual
HPI-O	Healthcare Provider Identifier – Organisation
LIS	Laboratory Information System
MLLP	Minimal Lower Layer Protocol used to transport HL7 messages
NASH	National Authentication Service for Health
NOC	Notice of Connection required before connection to HI Service or My Health Record system
PAS	Patient Administration System
PKI	Public Key Infrastructure
RIS	Radiology Information System
SVT	Software Vendor Test environment of the My Health Record system

3 Prerequisites

This section outlines the major prerequisites that an implementer will need to obtain before implementing HIPS in either a test or production environment.

3.1 Registration to HI Service Vendor Test Environment

For conformance testing of the integration with My Health Record, prior to a declaration of conformance and production access, you will need to arrange access to the HI Service test environment. The HI Service Registration Form which is submitted to Medicare as part of the registration process must include the details of the HIPS product, which are stored within the application configuration file. These settings are explained further in Appendix A.

The Medicare Location (Site) PKI certificate gives access to the HI Service Vendor Test Environment.

If creating the product from modified HIPS source code solution then the compiled binaries produced from the source code need to be registered in the HI Service as a new product with new product name, version and vendor details to be able to identify itself to the HI Service. This option will require a new Notice of Connection (NOC) and Conformance Compliance Accreditation (CCA) by a third-party accredited test laboratory.

If using the pre-compiled binaries or building from unmodified HIPS source code, the existing product registration for HIPS may be used, avoiding the need to complete NOC and CCA testing.

The table below states the HI Service Registration Form Fields and what application configuration setting field this should be mapped to.

Field	Description	Application Configuration Setting
Vendor ID	The registered product Vendor ID	IhiVendorId
Product Name	The registered Product Name	IhiProductName
Product Version	The registered product version number	IhiProductVersion

3.2 Registration to My Health Record SVT Environment

In order to test any My Health Record features, the organisation will need to obtain a NASH HPI-O test certificate that gives access to the My Health Record Software Vendor Testing (SVT) environment.

If creating the product from modified HIPS source code solution then the compiled binaries produced from the source code need to be registered in the My Health Record system as a new product with new product name, version and vendor details to be able to identify itself to the My Health Record system. This option will require a new Notice of Connection (NOC).

If using the pre-compiled HIPS binaries or building from unmodified HIPS source code, the existing product registration for HIPS may be used, avoiding the need to complete NOC testing, however conformance testing and CCD is still required. A unique product vendor name must be configured for CCD.

HIPS will detect that a request to interact with the My Health Record system has been made using a default value for PcehrVendorId and return an appropriate error message.

The table below states the My Health Record Service Registration Form Field and what application configuration setting field this should be mapped to. These settings are explained further in Appendix A.

Field	**Description**	**Application Configuration Setting**
Vendor	The registered Product Vendor Name	PcehrVendorId
Product Name	The registered Product Name	PcehrProductName
Product Version	The registered product version number	PcehrProductVersion

3.3 Application Server Operating Environment

To install the HIPS application server, the organisation may:

- Install on an existing workstation or notebook computer with a Windows 8 to 10 operating system. This setup is suitable for evaluation only.
- Install on a provisioned Windows Server 2008 R2 to 2016 operating system with Internet Information Services (IIS) 7.5+ and Microsoft .NET Framework 4.5.2+ installed.

  ○ **IMPORTANT:** Previous versions of HIPS only required .NET Framework 4.5+, however HIPS 7.0.0 requires .NET Framework 4.5.2 or greater.

3.4 Database Server Operating Environment

To install the HIPS Core database, the organisation may:

- Use available capacity on an existing Microsoft SQL Server 2008 R2 to 2016 database server. This setup is suitable for development and testing environments but may have insufficient capacity for the production environment.
- Install Microsoft SQL Server 2008 R2 to 2016 onto the same operating environment as the application server. This setup is suitable for development and testing environments but may have insufficient capacity for the production environment.
- Provision a separate Windows Server 2008 R2 to 2016 operating system and install Microsoft SQL Server 2008 R2 to 2016. This setup is preferable for the production environment.

3.5 Active Directory Service Account

HIPS uses Active Directory to secure its internal connections and it is recommended that an AD service account is used (one that does not expire and will not lock). This will be referred to as <HIPS_CORE_SERVICE_ACCOUNT> for the remainder of the document.

3.6 Build HIPS Products from Source Code

If you are building the HIPS products from supplied source code product, then refer to the *HIPS 7.0.0 - Build Guide* for instructions on building the binaries and associated files required for each of the HIPS Products. This must be completed before the installation of HIPS v7 can commence.

3.7 Installation Artefacts

HIPS Core installation is facilitated by a set of artefacts provided in the HIPS Core release package. These installation artefacts are assumed to be available in the filesystem of each deployment target. In the sections that follow, the installation artefact source location will be referred to as <INSTALL_SOURCE>.

Installation of HIPS Core on each Application Server is largely automated through PowerShell scripts provided as part of the installation artefacts. The minimum required version of PowerShell is **PowerShell 5.0**.

4 Database Server

4.1 Install SQL Server

If not already installed, install SQL Server on the database server. The supported SQL Server database versions for HIPS are SQL Server 2008 R2, 2012, 2014 and 2016.

For greater performance, the tempdb should be installed on a different partition to the data files. Size the tempdb to 5GB as the application will need enough processing space for performance reasons.

4.2 Create the HIPS Core Database

1. Create a new HIPS-Core database with the following settings:
   a. The initial data store rows data with PRIMARY file group requires an initial size of 1024MB, with an auto growth of 250MB and unrestricted file growth.
   b. The initial log requires an initial size of 500MB, with an auto growth of 250MB and unrestricted file growth.
2. Add the <HIPS_CORE_SERVICE_ACCOUNT> as a login to the SQL Server and assign it db_datareader and db_datawriter to the new HIPS-Core database.
3. If you will be using the Alert Monitoring DBDiskSpace alert processor, assign the <HIPS_CORE_SERVICE_ACCOUNT> the VIEW SERVER STATE permission.

IMPORTANT: In a High Availability topology that employs a SQL Server cluster, this HIPS-Core database and the SQL Server login for the <HIPS_CORE_SERVICE_ACCOUNT> must exist on all cluster nodes.

Notes:

1. HIPS also makes use of additional filegroups and database files. These are created automatically by the HIPS installation process with desired default settings such as file size and growth, but can be relocated or modified after creation if desired:
   a. Filegroups:
      i. INDEXES: Used for data files containing index data.
   b. Files:
      i. <Database Name>_index.ndf: Used by HIPS to store index data.
      ii. <Database Name>_blobs.ndf: Used by HIPS to store binary large object (BLOB) data such as queued messages.
2. For greater performance it is recommended that each type of database file (rows data, logs, indexes) should be located on different disc partitions or SAN LUNs.
3. To run the Alert Monitoring DBDiskSpace alert processor, the <HIPS_CORE_SERVICE_ACCOUNT> must have the VIEW SERVER STATE permission.

4.3 Configure and execute the HIPS Core Database Scripts

1 Locate the folder `<INSTALL_SOURCE>\HIPS-AppServer\database` and application named `HIPS.PcehrDataStore.DBUpgrade.exe`. This application will install or upgrade the HIPS core database objects and data into an existing HIPS core database.

IMPORTANT

Ideally the login used to connect to the specified SQL Server instance must be a member of the *sysadm* fixed server role. If using integrated security, this will be the domain account of the user executing the preceding command. Alternatively, modify the connection string in the preceding command to specify the user name and password for a SQL login with the appropriate membership.

In addition, ensure the default schema of the login's user in the HIPS Core database is set to [dbo]. It should not be set to [hips].

If the [SchemaVersions] table exists in the target database in a schema other than the [dbo] schema, it must be moved to the [dbo] schema prior to execution.

a Open a command prompt and change directory to the directory which contains the `HIPS.PcehrDataStore.DBUpgrade.exe` application.

b Execute the following command[1]:

```
.\HIPS.PcehrDataStore.DBUpgrade.exe upgrade-db "Data
Source=#{HIPS.Core.Database.Server};Initial
Catalog=#{HIPS.Core.Database.Name};Integrated Security=SSPI;Connect Timeout=15;"
```

Modifying the highlighted values as below:

i #{HIPS.Core.Database.Server} is the name of the SQL Server instance hosting the database.

ii #{HIPS.Core.Database.Name} is the name of the previously created database to be upgraded.

The command uses a default command timeout of 3600 seconds (1 hour) that can be adjusted via the --timeout option if desired.

If the command was successful, a green 'Success!' notice should be displayed. Any errors will be displayed in red.

2 In the same folder as the `HIPS.PcehrDataStore.DBUpgrade.exe` application there is a "`ConfigurationScripts`" subfolder containing the following script files that can be manually edited and executed if required.

a Open the "`01_HIPS_Roles.sql`" script and:

i Replace instances of '**HIPS AD Service account user**' with the domain and name of the <HIPS_CORE_SERVICE_ACCOUNT>.

ii Replace **UPDATE-AS-REQUIRED** with the names of any SQL Server users that will be members of the [Admin] role (required to create health provider organisation and facility records in a subsequent step).

[1] Optionally, for help with additional command line options, execute the following:
`.\HIPS.PcehrDataStore.DBUpgrade.exe upgrade-db --help`

iii Execute the script against the target HIPS database and ensure it completes successfully.

b Open the “`02_HIPS_HealthProviderOrganisations-Facilities_Data.sql`” script and:

i Replace values as appropriate for the health provider organisation network structure and facilities (hospitals) for the target implementation. For example:

1. Replace the “OMO Name” text with the name of the person who within your organisation has the authority to make calls to the My Health Record system. This name is not visible to consumers or healthcare providers, but it will be added to the audit records in HIPS, the HI Service and the My Health Record system that are visible to the System Operator and NIO (National Infrastructure Operator) of the My Health Record system. For example, this may be the CIO, RO or OMO of your organisation.
2. Replace the “AEUID” text with the identifier or username of the person who within your organisation has the authority to make calls to the My Health Record system.
3. Replace the values of the “seed” parameters with the HPI-O and name of the seed Healthcare Provider Organisation.
4. Replace the ‘HPI-O-A’ text with the HPI-O for the Healthcare Provider Organisation.
5. Replace the ‘OU:SVT-Name A’ text with the name of the Healthcare Provider Organisation.
6. Replace the ‘Hi Cert Serial’ text with the serial number from the DHS Site PKI Certificate for the Healthcare Provider Organisation or Contracted Service Provider (CSP). If using a CSP certificate, change the value for ‘HiCsp’ from 0 to 1.
7. Replace the ‘PCEHR Cert Serial A’ text with the serial number from the NASH PKI Certificate for the Healthcare Provider Organisation or Supporting Organisation. If using a Supporting Organisation certificate, change the value for ‘PcehrCsp’ from 0 to 1.
8. Replace the ‘HPO Cert Serial’ text with the serial number from the NASH PKI Certificate for the Healthcare Provider Organisation. This certificate is required to create the digital signature when uploading documents.
9. Replace the “HIPS Hospital” description text with the local description of the facility, which can be a shortened name (e.g. RAH).
10. Replace the “HIPS Hospital” name text with the full name of the facility (e.g. Royal Adelaide Hospital).
11. Replace the address and contact details as appropriate.
12. Replace the ‘*HOSPITAL-CODE*’ text with the facility code used to identify this facility as the assigning authority of the patient MRN in ADT messages. This is generally a short code for the facility in question (e.g. Royal Adelaide Hospital is set as ‘RAH’). Replace the HospitalId in the HospitalCode record with the Hospital record within the script.

13. Replace the 'Users' hospital authorised group text with a comma-delimited list of Active Directory groups that are authorised for this facility. NOTE: There must be no spaces between values and commas.

ii Insert multiple calls to the [_CreateHealthProviderOrganisation] stored procedure as required to create multiple Healthcare Provider Organisations. NOTE: The same "seed" details can be reused if there is only a single seed Healthcare Provider Organisation.

iii Insert multiple calls to the [_CreateFacility] stored procedure if there are multiple facilities in the Healthcare Provider Organisation.

iv Execute the script against the target HIPS database and ensure it completes successfully.

c Optional. It is possible to reduce the amount of data stored when auditing interactions with the HI Service and My Health Record System by adjusting the value of the [IsEnabled] column of each record in the [hips].[SystemInteractionLogConfiguration] table. For **Non-Production** environments it may be desirable to leave the value of the [IsEnabled] column set to the default value of 1 (Enabled) to assist with diagnostic troubleshooting. For **Production** environments it may be desirable to set the value of the [IsEnabled] column to 0 (Disabled) to limit the amount of data stored for each interaction to meet the minimum conformance requirements.

d Optional. If you **ARE** planning to install and use Assisted Registration from the HIPS-UI Web product, then it is essential that the `Optional_HIPS_AssistedRegistration_Data.sql` script is executed **AFTER** the Healthcare Provider Organisation and Facility data has been created. This script will take the Healthcare Provider Organisation HPI-O records and create "Visitor" hospitals from the HPIO(s); this is required when an individual is to be registered with the My Health Record System and HIPS does not know whether or not the individual has a current episode in the hospital – hence the term "Visitor". If you **ARE NOT** planning to install and use Assisted Registration from the HIPS-UI Web product, then executing this script is not required.

i Execute the script if necessary and ensure it completes successfully.

e Optional. If the HL7(R) V2 to CDA(R) conversion functionality is to be used to upload Pathology or Diagnostic Imaging Reports to the My Health Record System, then the `Optional_PDI_HospitalCode_Data.sql` script should be executed to create the RIS/LIS System Code and assign it to each Facility.

i Modify the *RIS/LIS SYSTEM* text to be the system code that will be contained in the HL7 message MSH-4 Sending Facility and PID-3 Patient Identifier List field, i.e. 'NATA1234' or 'LSPN8234'.

ii If more than one Facility (or Laboratory) was added, then copy and paste the insert statement for each Facility created previously. You will need to query the HIPS [Hospital] table to obtain the [HospitalId] values.

iii Execute the script if necessary and ensure it completes successfully.

IMPORTANT: In a High Availability topology that employs a SQL Server cluster, the following messages added to sys.messages must exist on all cluster nodes:

50001

50002

50003

5 Application Server

The following sections describe the required steps to prepare, configure and install the HIPS Core runtime components on each Application Server.

Installation performs the following actions:

1. Ensures the Application Server meets required prerequisites, including:
   a. .NET Framework 4.5.2+ is installed.
   b. Windows operating system features are installed.
   c. NET TCP Windows Services are configured appropriately.
2. Ensures certificates required for interacting with the My Health Record system and HI Service are installed correctly:[2]
   a. Installs valid certificate files if they have been manually placed in the `<INSTALL_SOURCE>\HIPS-AppServer\certificates` folder and the filename, password and service account is specified in the configuration file located under `<INSTALL_SOURCE>\HIPS-AppServer\setup`.
   b. Sets permissions for the HPI-O, CSP and DHS Site certificates to allow the HIPS application account to full control access their private keys.
3. Deploys shared configuration settings:
   a. Copies shared configuration files (`appSettings.config` and `connectionStrings.config`) from `<INSTALL_SOURCE>\HIPS-AppServer\runtime\config` to a target location on the Application Server.
   b. Merges deployment-specific configuration data from `<INSTALL_SOURCE>\HIPS-AppServer\setup` to configuration files.
4. Deploys web components:
   a. Copies web binary and configuration files from `<INSTALL_SOURCE>\HIPS-AppServer\runtime\web` to a target location on the Application Server.
   b. Assigns required permissions to the target location on the Application Server.
   c. Merges deployment-specific configuration data from `<INSTALL_SOURCE>\HIPS-AppServer\setup` to configuration files.
   d. Creates a symbolic link to the shared configuration.
   e. Ensures required user account membership in the IIS_IUSRS local group.
   f. Creates an Internet Information Services (IIS) application pool and website using deployment-specific configuration data.
   g. Starts the IIS application pool and website.

[2] Refer to Appendix F Certificates for more information on certificate configuration.

- h Verifies the HIPS web service endpoints available via the website are accessible.

5 Deploys Queue Consumer components[3]:

- a Copies Queue Consumer binary and configuration files from `<INSTALL_SOURCE>\HIPS-AppServer\runtime\background\queue-consumer` to a target location on the Application Server.
- b Assigns required permissions to the target location on the Application Server.
- c Merges deployment-specific configuration data from `<INSTALL_SOURCE>\HIPS-AppServer\setup` to configuration files.
- d Creates a symbolic link to the shared configuration.
- e Installs the Queue Consumer instance as a Windows Service.
- f Starts the Queue Consumer Windows Service.

6 Deploys alert monitoring components:

- a Copies alert monitoring binary and configuration files from `<INSTALL_SOURCE>\HIPS-AppServer\runtime\background\alert-monitoring` to a target location on the Application Server.
- b Assigns required permissions to the target location on the Application Server.
- c Merges deployment-specific configuration data from `<INSTALL_SOURCE>\HIPS-AppServer\setup` to configuration files.
- d Creates a symbolic link to the shared configuration.
- e Installs the alert monitoring instance as a Windows Service.
- f Starts the alert monitoring Windows Service.

5.1 Configure Installation Artefacts

Perform the following steps to configure installation artefacts to reflect the target environment:

1 Open the file <INSTALL_SOURCE>\HIPS-AppServer\setup\HIPS-Core-Configuration.psd1 in a text editor such as Notepad or Notepad++.

The HIPS-Core-Configuration.psd1 file is a Windows PowerShell data file containing a PowerShell hashtable of key/value pairs representing configuration data required during installation. By default, the file contains placeholders for each value that must be modified, for example in the following line:

Path = '#{HIPS.Core.Path}\config'

Path is the key of the hashtable entry, and the highlighted text #{HIPS.Core.Path}\config is the value that will need to be reviewed and modified as appropriate.

The HIPS-Core-Configuration.psd1 file contains extensive comments describing each item that must be configured. In addition, a complete example is provided for reference in the HIPS-Core-Configuration.sample.psd1 file.

2 Review and modify the values for each item as appropriate to the target environment. These values will be used during installation and injected into configuration files as appropriate.

[3] A separate Queue Consumer instance is configured and installed for each consumer type.

3. Add required certificates to the `<INSTALL_SOURCE>\HIPS-AppServer\certificates` folder and add a reference to each certificate in the HIPS-Core-Configuration.sample.psd1 file.
4. Save and close the file.

   IMPORTANT: It may also be desirable to review and modify additional configuration settings in the following files **prior to installation**:

   a. <INSTALL_SOURCE>\HIPS-AppServer\runtime\config\appSettings.config, particularly:

      i. `HpiiValidationPeriodDays`: The number of days the organisation prefers the number of days after HPI-I validation for the HPI-I to remain to be valid for.

      ii. `AvoidProxy`: Defaults to true so that outward external connections can bypass the standard proxy, however, depending on the organisation's network and server configuration this may need to be set to false so that the outward external connections will use the proxy settings of the HIPS service account.

   b. <INSTALL_SOURCE>\HIPS-AppServer\runtime\web\web.config

   c. <INSTALL_SOURCE>\HIPS-AppServer\runtime\web\log4net.config

   d. <INSTALL_SOURCE>\HIPS-AppServer\runtime\background\queue-consumer\HIPS.AppServer.ServiceHost.QueueConsumer.exe.config

   e. <INSTALL_SOURCE>\HIPS-AppServer\runtime\background\queue-consumer\log4net.config

   f. <INSTALL_SOURCE>\HIPS-AppServer\runtime\background\alert-monitoring\HIPS.AppServer.ServiceHost.AlertMonitoring.exe.config

   g. <INSTALL_SOURCE>\HIPS-AppServer\runtime\background\alert-monitoring\log4net.config

Refer to the *Appendix* for further information regarding configuration settings.

NOTE: By default the Windows services hosting the HIPS-Core Queue Consumer and Alert Monitoring background processes are configured to detect unexpected failures that cause the service to shutdown. The service will attempt to restart automatically up to 3 times: after 1 minute, after 5 minutes following a second failure, and finally after 10 minutes following a third failure. Following 3 failed recovery attempts the service will not attempt to restart until the conditions causing the failure are investigated and the service is manually restarted. The failure count is reset every day. Each failure is logged to the System log in the Windows event log.

NOTE: Refer to Configuring the Queue Consumer in the Appendix of this document and to Configure the HIPS-Core Queue Consumer to maximise throughput in the Server Role Guidance section of the HIPS 7.0.0 - Topology and Configuration Guide for further information regarding the configuration of the HIPS-Core Queue Consumer to maximise throughput.

5.2 Execute Installation Scripts

Perform the following steps to deploy HIPS-Core runtime components to the target environment:

1. Open a Windows PowerShell console as Administrator.
2. Execute the following command to change directory to the location where the HIPS-Core installation artefacts are located:

```
cd <INSTALL_SOURCE>\HIPS-AppServer\setup
```

3 Execute the following command to install all HIPS-Core runtime components using the previously configured installation artefacts:

```
.\HIPS-Core-Install.ps1 -Interactive -ConfigurationDataFile
'HIPS-Core-Configuration.psd1' -Remove -Install -Prerequisites
-SharedConfiguration -Web –QueueConsumer -AlertMonitoring -
Certificates
```

This command will invoke the `HIPS-Core-Install.ps1` PowerShell script using the previously configured `HIPS-Core-Configuration.psd1` file as input to remove HIPS-Core (if already present), then install prerequisites, certificates, shared configuration, web, Queue Consumer and alert monitoring components.

NOTE: If it is desired to only install specific components on a certain Application Server the preceding command can be adjusted accordingly. It is important to note however that the web, queue consumer and alert monitoring components are dependent upon the shared configuration component, and their installation will fail if the shared configuration cannot be located.

4 Ensure the command completes successfully.

5.3 Application Server Self-Signed SSL Certificate

HIPS can be configured to use HTTP or HTTPS connectivity.

A risk assessment of the HIPS solution has resulted in a recommendation that all traffic between HIPS and internal applications should occur using an encrypted connection, i.e. using HTTPS rather than HTTP. However, this is not essential to the connectivity of HIPS, so if your datacentre allows HTTP communications between applications within the datacentre then this is also possible.

While it is possible to use an internal PKI certificate service, or a commercial certificate, a Self-Signed SSL certificate is also considered an acceptable solution for communication between internal applications and HIPS.

A self-signed certificate may be configured via the steps below:

1 Select the main IIS instance of the application server and double-click the “Server Certificates” icon.

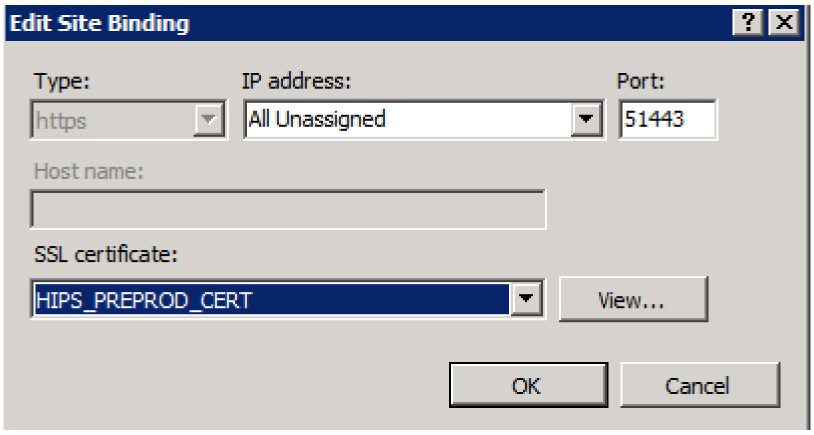
2 On the far right-click the action “Create Self-Signed Certificate…”

3 Specify a friendly name such as “HIPS_TEST_CERT” and click OK.

4 To apply the self-signed certificate - Select the website created by the HIPS-Core installation script and in the far right select the “Bindings…“ action.

5 Select the “https” row from the “Site Bindings” dialogue and click “Edit”.

6 In the “Edit Site Bindings” dialogue, select the previously created certificate from the “SSL Certificate” drop down and click OK.



Type: https
IP address: All Unassigned
Port: 51443
SSL certificate: HIPS_PREPROD_CERT

7 Close the "Site Bindings" dialogue.

5.4 Perform Optional Configuration

HIPS Core is pre-configured with commonly used defaults for much of its configuration. However, it may be necessary following the installation of HIPS Core components on the Application Server to optionally perform the following additional configuration:

1. Configuration of HIPS Core application configuration settings as described in *Appendix A Application Server Configuration Reference*
2. Configuration of HIPS UI features that are configured via the HIPS Core database as described in *Appendix B Configuring HIPS Web UI Features*
3. Configuration of HIPS Core behaviour when it receives error codes from the My Health Record System as described in *Appendix C PCEHR Error Codes*
4. Configuration of HIPS Core logging as described in *Appendix D log4net Configuration*
5. Configuration or troubleshooting of HIPS Core background processes as described in *Appendix E Background Processes*
6. Manual installation of required certificates if not previously installed as described in *Appendix F Certificates*

6 Mirth Connect Installation and Set Up

Mirth Connect is an optional component that can be set up and used to host the MLLP interfaces that connect to the HIPS SOAP web services to upload Pathology and/or Diagnostic Imaging Reports to the national digital health record system.

Mirth Connect is open-source-software that allows a HL7 message to be sent to Mirth Connect via MLLP from any ESB. The Mirth Connect software will then call the HIPS SOAP web services using the supplied HL7 message.

The configured Mirth Connect channels supplied as part of the release were built using Mirth Connect version 3.3.1 (64-bit) and can be imported on this or any later version.

Set-up requirements and the Mirth Connect tool can be downloaded from the Mirth website:

http://www.mirthcorp.com/community/wiki/display/mirth/Getting+Started+Guide

If Mirth Connect is to be implemented to receive the HIPS Acknowledgement, this is the final acknowledgement that HIPS receives from the My Health Record Service when a pathology or diagnostic imaging report is uploaded, then an extra setting is required to be updated in the application configuration file. This is described in *Appendix A Application Server Configuration Reference*.

6.1 Channel Installation

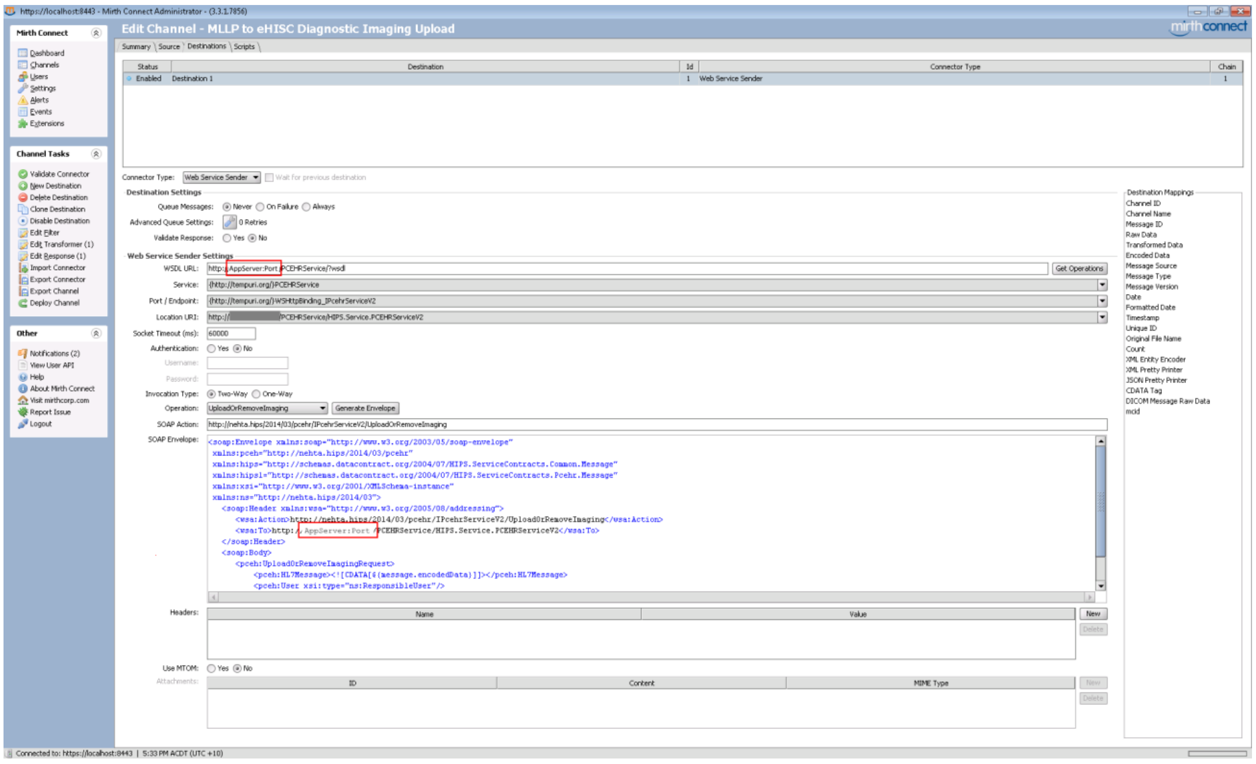
To install the channels required for the Pathology and Diagnostic Imaging Report uploads:

1. Launch the Mirth Connect Administration tool.
2. Open the **Channels** tab.
3. Under Channel tasks click on **Import Channel.**
4. Open the `HIPS-Mirth\channels\` folder in the Core Software Package and select the `HIPS Acknowledgement to MLLP.xml`. Click **Open.**
5. Repeat step 4 for each of the following channels:
   a. `MLLP to HIPS Diagnostic Imaging Upload.xml`
   b. `MLLP to HIPS Notify PAS Event.xml`
   c. `MLLP to HIPS Pathology Upload.xml`
6. If you want to use the file drop functionality, that is where a HL7 message can be dropped into a folder to be consumed by Mirth then import the following channels:
   a. `Filedrop ADT to MLLP.xml`
   b. `Filedrop Imaging to MLLP.xml`
   c. `Filedrop Pathology to MLLP.xml`
   d. `Filedrop Upload-Ack from MLLP.xml`
7. To configure the MLLP Channels to point to HIPS.
8. Open the “MLLP to HIPS Diagnostic Imaging Upload” Channel.

a Select the **Source** tab.

i Update the **Source** settings to listen on the Port for your Mirth configuration.

b Select the **Destinations** tab.



Status	Destination	Id	Connector Type	Chain
Enabled	Destination 1	1	Web Service Sender	1

Connector Type: Web Service Sender

Destination Settings

Queue Messages: [✓ Never] On Failure Always

Advanced Queue Settings: 0 Retries

Validate Response: Yes [✓ No]

Web Service Sender Settings

WSDL URL: http://AppServer:Port/PCEHRService/?wsdl

Service: {http://tempuri.org/}PCEHRService

Port / Endpoint: {http://tempuri.org/}WSHttpBinding_IPcehrServiceV2

Location URI: http://[redacted]/PCEHRService/HIPS.Service.PCEHRServiceV2

Socket Timeout (ms): 60000

Authentication: Yes [✓ No]

Invocation Type: [✓ Two-Way] One-Way

Operation: UploadOrRemoveImaging

SOAP Action: http://nehta.hips/2014/03/pcehr/IPcehrServiceV2/UploadOrRemoveImaging

SOAP Envelope:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:pceh="http://nehta.hips/2014/03/pcehr"
 xmlns:hips="http://schemas.datacontract.org/2004/07/HIPS.ServiceContracts.Common.Message"
 xmlns:hipsl="http://schemas.datacontract.org/2004/07/HIPS.ServiceContracts.Pcehr.Message"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ns="http://nehta.hips/2014/03">
   <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://nehta.hips/2014/03/pcehr/IPcehrServiceV2/UploadOrRemoveImaging</wsa:Action>
      <wsa:To>http://AppServer:Port/PCEHRService/HIPS.Service.PCEHRServiceV2</wsa:To>
   </soap:Header>
   <soap:Body>
      <pceh:UploadOrRemoveImagingRequest>
         <pceh:HL7Message><![CDATA[${message.encodedData}]]></pceh:HL7Message>
         <pceh:User xsi:type="ns:ResponsibleUser"/>
```

Use MTOM: Yes [✓ No]

Connected to: https://localhost:8443 | 5:33 PM ACDT (UTC +10)

i Update the **WSDL URL** to:

http://[AppServerName]:[Port]/HIPS.Service.PathologyImagingService.svc?wsdl

where [AppServerName] is the HIPS application server name and [Port] is the port number the HIPS web services are running on.

ii Under the Channel Tasks menu, click Save Changes.

9 Click on the **Channels** menu item under the **Mirth Connect** menu.

10 Repeat steps 8 and 9 for the “MLLP to HIPS Pathology Upload” Channel.

11 Open the “MLLP to HIPS Notify PAS Event” Channel.

a Select the **Source** tab.

i Update the **Source** settings to listen on the Port for your Mirth configuration.

b Select the **Destinations** tab.

i Update the **WSDL URL** to:

http://[AppServerName]:[Port]/HIPS.Service.DatabaseLoaderService.svc?wsdl

where [AppServerName] is the HIPS application server name and [Port] is the port number the HIPS web services are running on.

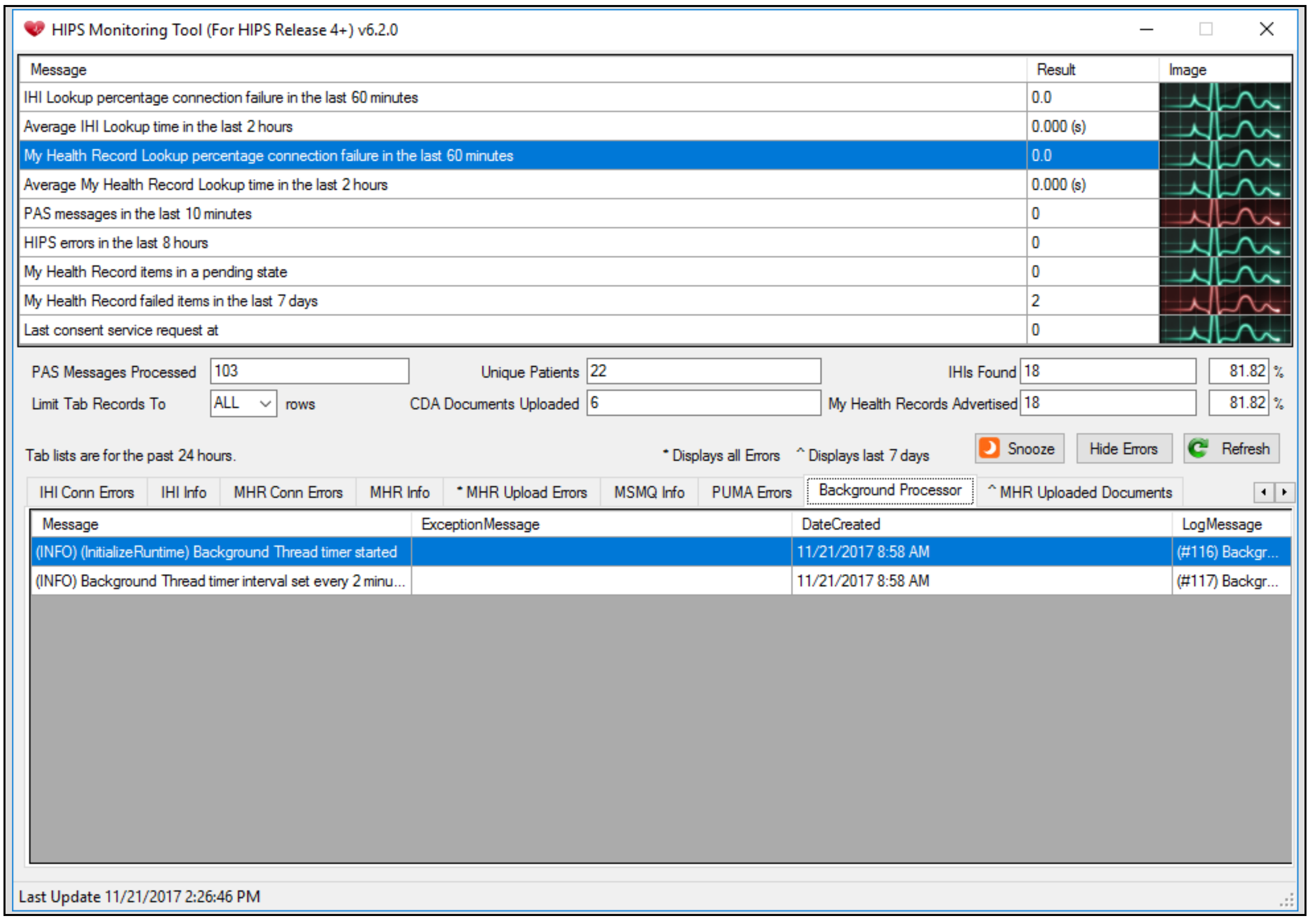
ii Under the Channel Tasks menu, click Save Changes.

12 Copy the accept-ack-message.jar to the MIRTH_HOME\custom-lib directory, then restart Mirth.

7 HIPS Monitoring Tool Set Up

The HIPS Monitoring Tool is an optional component that can be set up and used by the application support team to allow them to monitor and proactively take any actions required when an alert is raised through the interface.

The tool is a simple windows application and can be configured to display only certain alerts. The tool has been designed to run on Windows 8+ with Microsoft .NET Framework 4.5 installed.



Message	Result	Image
IHI Lookup percentage connection failure in the last 60 minutes	0.0	
Average IHI Lookup time in the last 2 hours	0.000 (s)	
My Health Record Lookup percentage connection failure in the last 60 minutes	0.0	
Average My Health Record Lookup time in the last 2 hours	0.000 (s)	
PAS messages in the last 10 minutes	0	
HIPS errors in the last 8 hours	0	
My Health Record items in a pending state	0	
My Health Record failed items in the last 7 days	2	
Last consent service request at	0	

PAS Messages Processed	103	Unique Patients	22	IHIs Found	18	81.82 %
Limit Tab Records To	ALL rows	CDA Documents Uploaded	6	My Health Records Advertised	18	81.82 %

Tab lists are for the past 24 hours. * Displays all Errors ^ Displays last 7 days

Message	ExceptionMessage	DateCreated	LogMessage
(INFO) (InitializeRuntime) Background Thread timer started		11/21/2017 8:58 AM	(#116) Backgr...
(INFO) Background Thread timer interval set every 2 minu...		11/21/2017 8:58 AM	(#117) Backgr...

Last Update 11/21/2017 2:26:46 PM

7.1 Configuration

Section	Option Name	Suggested Value	Description
Connection String	Data Source	*Name of SQL server hosting HIPS Core database*	HIPS Monitoring Tool will attempt to connect to this SQL server.
	Initial Catalog	HIPS_CORE_TEST	HIPS Monitoring Tool will connect to the database with this name.

Section	Option Name	Suggested Value	Description
	Integrated Security	SSPI	If set to SSPI as recommended, then the Monitoring Tool will use the current Windows account credentials for authentication to the database. Otherwise, and this is not recommended, set to false and provide a User ID and Password. This will only work if SQL Server Authentication is enabled on the server.
Application Settings	HipsErrorsTab	True	Flag to turn the *HIPS Error* Tab display on or off.
	HipsInfoTab	False	Flag to turn the *HIPS Info* Tab display on or off.
	IhiConnectionsErrorsTab	True	Flag to turn the *IHI Conn Errors* Tab display on or off.
	IhiInfoTab	True	Flag to turn the *IHI Info* Tab display on or off.
	PcehrErrorsTab	True	Flag to turn the *MHR Conn Errors* Tab display on or off.
	PcehrInfoTab	False	Flag to turn the *MHR Info* Tab display on or off.
	BackgroundProcessorTab	False	Flag to turn the *Background Processor* Tab display on or off.
	MsmqInfoTab	True	Flag to turn the *MSMQ Info* Tab display on or off.
	PumaErrorsTab	False	Flag to turn the *PUMA Errors* Tab display on or off.
	PcehrUploadErrorsTab	False	Flag to turn the *MHR Upload Errors* Tab display on or off.
	MergeTab	False	Flag to turn the *Merge Info* Tab display on or off.
	PcehrUploadDocumentTab	True	Flag to turn the *MHR Upload Documents* Tab display on or off.
	AlertTimerIntervalMinutes	1	The Alert interval in minutes.

THIS IS THE COMPLETION OF THE INSTALLATION OF HIPS

Appendix A Application Server Configuration Reference

The application configuration files should be pre-configured for each release. This is a detailed explanation of the contents for implementers.

Most application-wide configuration options are held in one of the following files:

- `config\appSettings.config`
- `config\connectionStrings.config`
- `web\web.config`
- `web\log4net.config`
- `background\queue-consumer\HIPS.AppServer.ServiceHost.QueueConsumer.exe.config`
- `background\queue-consumer\log4net.config`
- `background\alert-monitoring\HIPS.AppServer.ServiceHost.AlertMonitoring.exe.config`
- `background\alert-monitoring\log4net.config`

Section	**Option Name**	**Suggested Value**	**Description**
Connection String	Data Source	*Name of SQL server*	HIPS will attempt to connect to this SQL server.
	Initial Catalog	HIPS_Core_Test	HIPS will use the database with this name.

Section	Option Name	Suggested Value	Description
	Integrated Security	SSPI	If set to SSPI as recommended, then HIPS will use the IIS application pool's Windows account credentials for authentication to the database. Otherwise, and this is not recommended, set to false and provide a User ID and Password. This will only work if SQL Server Authentication is enabled on the server.
CDA Service	baseAddress	net.tcp://servername:**50000**/	The CDA service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/CdaService/	The CDA service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/CdaService/	The CDA service will be accessible using HTTPS protocol at this URL.
Consent Service	baseAddress	net.tcp://servername:**50000**/	The Consent service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/ConsentService/	The Consent service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/ConsentService/	The Consent service will be accessible using HTTPS protocol at this URL.
Database Loader Service	baseAddress	net.tcp://servername:**50000**/	The Database Loader service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/DatabaseLoaderService/	The Database Loader service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/DatabaseLoaderService/	The Database Loader service will be accessible using HTTPS protocol at this URL.
Document Service	baseAddress	net.tcp://servername:**50000**/	The Document service will be accessible using net.tcp protocol at this URL.

Section	Option Name	Suggested Value	Description
	baseAddress	http://servername:**50500**/DocumentService/	The Document service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/DocumentService/	The Document service will be accessible using HTTPS protocol at this URL.
IHI Service	baseAddress	net.tcp://servername:**50000**/	The IHI services will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/IHIService/	The IHI services will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/IHIService/	The IHI services will be accessible using HTTPS protocol at this URL.
PCEHR Service	baseAddress	net.tcp://servername:**50000**/	The PCEHR services will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/PCEHRService/	The PCEHR services will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/PCEHRService/	The PCEHR services will be accessible using HTTPS protocol at this URL.
Pathology Imaging Service	baseAddress	net.tcp://servername:**50000**/	The Pathology Imaging Service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/PathologyImagingService/	The Pathology Imaging Service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/PathologyImagingService/	The Pathology Imaging Service will be accessible using HTTPS protocol at this URL.
Reference Service	baseAddress	net.tcp://servername:**50000**/	The Reference service will be accessible using net.tcp protocol at this URL.

Section	Option Name	Suggested Value	Description
	baseAddress	http://servername:**50500**/ReferenceService/	The Reference service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/ReferenceService/	The Reference service will be accessible using HTTPS protocol at this URL.
Assisted Registration Service	baseAddress	net.tcp://servername:**50000**/	The Reference service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/AssistedRegistrationService/	The Reference service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/AssistedRegistrationService/	The Reference service will be accessible using HTTPS protocol at this URL.
HPI-I Service	baseAddress	net.tcp://servername:**50000**/	The Reference service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/HpiiService/	The Reference service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/HpiiService/	The Reference service will be accessible using HTTPS protocol at this URL.
Hi Reference Service	baseAddress	net.tcp://servername:**50000**/	The Reference service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/HiReferenceService/	The Reference service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/HiReferenceService/	The Reference service will be accessible using HTTPS protocol at this URL.
Patient Service	baseAddress	net.tcp://servername:**50000**/	The Reference service will be accessible using net.tcp protocol at this URL.

Section	Option Name	Suggested Value	Description
	baseAddress	http://servername:**50500**/PatientService/	The Reference service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/PatientService/	The Reference service will be accessible using HTTPS protocol at this URL.
Ack Service	baseAddress	net.tcp://servername:**50000**/	The Ack Service will be accessible using net.tcp protocol at this URL.
	baseAddress	http://servername:**50500**/AckService/	The Ack service will be accessible using HTTP protocol at this URL.
	baseAddress	https://servername:**50443**/AckService/	The Ack service will be accessible using HTTPS protocol at this URL.
Ack Service Client Endpoint	address	http://localhost:663/services.AckService	The endpoint address for the Ack Service. This can either be set to be Mirth Connect HIPS Acknowledgement to MLLP Channel or if Mirth Connect is not implemented the HIPS Acknowledgement Service end point which will require the FileLocation appsetting (below) to be set.
App Settings	IhiProductName	<Product Name of the new product based on the HIPS source code>	Product Name registered with DHS.
	IhiProductVersion	<Product Version of the new product based on the HIPS source code>	Product Version registered with DHS.
	IhiVendorId	<Vendor Name of the organisation that has created the new product based on the HIPS source code>	Vendor ID registered with DHS.
	IhiVendorQualifier	http://ns.electronichealth.net.au/id/hi/vendorid/1.0	This URL qualifies the vendor ID in the header of each request to the HI Service. This is a fixed value defined by DHS technical specifications.
	IhiUserQualifierProviderIndividual	http://ns.electronichealth.net.au/id/hi/vendorid/1.0	

Section	Option Name	Suggested Value	Description
	IhiUserQualifierHiUser	http://ns.*healthdomain*.*stateorterritory*.gov.au/id/{0}/userid/1.0	This URL qualifies the domain in which the interactive user's login is identified to DHS, with {0} replaced by the supplied domain,
	IhiUserQualifierAuthorisedEmployee	http://ns.*healthdomain*.*stateorterritory*.gov.au/id/ae/userid/1.0	This URL qualifies the user identifier supplied for the authorised employee for non-interactive processing.
	IhiHpioQualifier	http://ns.electronichealth.net.au/id/hi/hpio/1.0	This URL qualifies an HPI-O in the header of a request to HI Service, but is only used for Contracted Service Providers (CSP) and not if the health provider organisation itself makes the call. This is a fixed value defined by DHS technical specifications.
	IhiValidationPeriodDays	1	HIPS will revalidate an IHI before returning it to the calling system or using it in a call to the My Health Record system, if it was not obtained or last validated within the number of days specified here. For organisations that do not connect HIPS to the HI Service, it is necessary to set this to a large number (e.g. 999) to prevent HIPS attempting revalidation.
	HiServiceUrl	Vendor Environment: https://www5.medicareaustralia.gov.au/cert/soap/services/ Production Environment: https://www3.medicareaustralia.gov.au/pcert/soap/services/	HIPS will connect to the HI Service at this URL, which is that of the HI Service vendor or production environment.
	PcehrProductName	<Product Name of the new product based on the HIPS source code>	Product Name registered with My Health Record SVT or Production.
	PcehrProductVersion	<Product Version of the new product based on the HIPS source code>	Product Version registered with My Health Record SVT or Production.
	PcehrVendorId	<Vendor Name of the organisation that has created the new product based on the HIPS source code>	Vendor ID registered with My Health Record SVT or Production.

Section	Option Name	Suggested Value	Description
	PcehrRole	CIS	HIPS is designed to communicate with My Health Record B2B Gateway as part of a Clinical Information System (CIS).
	CheckDoesPcehrExist	true	Whether HIPS should check the ‘digital health record’ advertised status of each patient immediately upon obtaining his/her IHI.
	CdaUserIdQualifierFormat	http://ns.*healthdomain*.*stateterritory*.gov.au/id/cda/userid/1.0/{0}	This URL qualifies the user identifier in the CDA signature file for CDA packaging. Replace healthdomain and stateterritory. The {0} will be replaced with the author’s identifier from the CDA document.
	IhiCleanupProcessMinutes	60	Interval to wake up background thread and process service error IHI calls.
	LookupRefreshSeconds	600	This is the waiting time between cached data refreshes in seconds.
	PatientMasterOverride	False	Allows client apps to provide temporary patient demographic overrides within HIPS.
	AvoidProxy	True	Avoid standard Proxies.
	DoesPCEHRExistUrl	https://b2b.ehealthvendortest.health.gov.au/doesPCEHRExist	HIPS will connect to this URL to check the ‘digital health record’ advertised status of an IHI.
	UploadDocumentUrl	https://b2b.ehealthvendortest.health.gov.au/uploadDocument	HIPS will connect to this URL to upload or supersede a document to the My Health Record system.
	GetDocumentUrl	https://b2b.ehealthvendortest.health.gov.au/getDocument	HIPS will connect to this URL to download a document from the My Health Record system.
	GetViewUrl	https://b2b.ehealthvendortest.health.gov.au/getView	HIPS will connect to this URL to request a view from the My Health Record system.
	RemoveDocumentUrl	https://b2b.ehealthvendortest.health.gov.au/removeDocument	HIPS will connect to this URL to remove a document from the My Health Record system.

Section	Option Name	Suggested Value	Description
	ListDocumentUrl	https://b2b.ehealthvendortest.health.gov.au/getDocumentList	HIPS will connect to this URL to list documents available to download from the My Health Record system.
	GainPCEHRAccessUrl	https://b2b.ehealthvendortest.health.gov.au/gainPCEHRAccess	HISP will connect to this URL to gain access to a digital health record.
	RegisterPcehrUrl	https://b2b.ehealthvendortest.health.gov.au/registerPCEHR	HIPS will connect to this URL to register a digital health record.
	DoesPCEHRExistTimeoutSeconds	120	Connection Timeout (in seconds) for DoesPCEHRExist Service. Defaults to 60 if not included in the configuration file.
	DocumentProductionTimeoutSeconds	300	Connection Timeout (in seconds) for Document Production Services : UploadDocument, RemoveDocument. Defaults to 300 if not included in the configuration file.
	DocumentConsumptionTimeoutSeconds	120	Connection Timeout (in seconds) for Document Consumption Services : GetDocument, GetDocumentList, GetChangeHistoryView, GainPCEHRAccess. Defaults to 120 if not included in the configuration file.
	IhiSearchTimeoutSeconds	120	Connection Timeout (in seconds) for IHI Search. Defaults to 60 if not included in the configuration file.
	HpiiSearchTimeoutSeconds	120	Connection Timeout (in seconds) for HPI-I Search. Defaults to 60 if not included in the configuration file.

Section	Option Name	Suggested Value	Description
	BypassHIService	false	This configuration setting should be kept as “false” except for: **1. Pathology and diagnostic imaging service providers** who wish to pass validated IHI and HPI-I numbers in the ORU messages and disallow HIPS from validating them with the HI Service. **2. Custom Assisted Registration installations** at sites where the HI Service is not to be used for Assisted Registration then this is set to true. This can only be set to true IF the Assisted Registration web service is used outside of the HIPS-UI. Note setting this true will mean that HIPS will need to pass conformance testing again to use Assisted Registration outside of the HIPS-UI solution of Assisted Registration.
	RegisteredDateOfBirthEnabled	False	**ONLY used if you expect patient to provide a different date of birth to what they have recorded at Medicare** This setting instructs HIPS to store the Medicare registered Date of Birth separately to the patients current Date of Birth and use to use the registered Date of Birth if the current Date of Birth does not return a valid IHI for the patient.
	PathologyReportUploadLocation	C:\Uploads\Pathology\	The folder location where the HIPS service will retrieve Pathology reports when a Reference Pointer is used rather than Embedded reports in the HL7 message.
	ImagingReportUploadLocation	C:\Uploads\Imaging\	The folder location where the HIPS service will retrieve Imaging reports when a Reference Pointer is used rather than Embedded reports in the HL7 message.
	DiagnosticImagingReportDocumentFormatCode	1.2.36.1.2001.1006.1.222.4	The Document Format Code to use for 3A level Pathology Reports. This value indicates if the HPI-I Enforced or HPI-I Relaxed template is to be used to upload the Diagnostic Imaging Reports to the My Health Record system. By default, it is set to HPI-I Enforced. Refer to the Diagnostic Imaging Report Template Package Library for applicable values.

Section	Option Name	Suggested Value	Description
	PathologyReportDocumentFormatCode	1.2.36.1.2001.1006.1.220.4	The Document Format Code to use for the Diagnostic Imaging Reports 3A. This value indicates if the HPI-I Enforced or HPI-I Relaxed template is to be used to upload the pathology reports to the My Health Record system. By default, it is set to HPI-I Enforced. Refer to the Pathology Report Template Package Library for applicable values.
	HpiiValidationPeriodDays	1	The number of days any stored HPI-I's are valid for.
	FileLocation	C:\mirth\upload-ack	The location where application Acknowledgements will be saved, if Mirth or another ESB provided endpoint is not invoked.
	UseHL7MessageDateTime	True	Flag to indicate if the check for earlier pending HL7 messages for pathology or diagnostic imaging uploads should use the HL7 Message Date Time or not. If false will use the DateCreated in the HL7MessageLog table.
	NashProviderOid	1.2.36.174030967.1.10.1.1	The NASH policy OID used in the NASH certificate validation.
	AllowCreationDummyMRN	True	Flag to indicate whether a Dummy MRN should be created for the Patient in the specified Hospital when HIPS cannot find a Patient in the Hospital.
	QueueLimit	1000	The limit for pending operations allowed on the queue. HIPS will return a SOAP fault to the caller of the UploadOrRemovePathology and UploadOrRemoveImaging services when there are more than this number of operations on the queue. This avoids a large backlog of messages being processed too quickly and overloading the database.
	PathologyThrottlingMilliseconds	350	The period of time to wait before processing inbound pathology HL7 messages. This delay limits the rate at which messages are placed on the queue and avoids overloading the database.

Section	Option Name	Suggested Value	Description
	ImagingThrottlingMilliseconds	350	The period of time to wait before processing inbound imaging HL7 messages. This delay limits the rate at which messages are placed on the queue and avoids overloading the database.
	LogStackTrace	false	Whether to write the stack trace for each HIPS exception into the SystemErrorLog table to enable debugging. This would not be recommended for a production environment.
	Database.CommandTimeout	30	Database Command Timeout (in seconds).
	Mrn.Padding	9	Configuration of zero padding for local identifiers. Value must be between 0 to 40.
	Ihi.AllowMHRAccessForDuplicates	false	Configuration to allow MHR access for duplicates patient.
	IhiSearchCriteriaReuseIntervalMinutes	1440	IHI Search Criteria Reuse Interval (minutes) **NOTE**: This should be greater than (IhiValidationPeriodDays setting * 60 * 24) to be used.
	PcehrExistsReuseIntervalMinutes	1440	Specifies (in minutes) how long the previous result of the does PCEHR exist enquiry should be reused before being re-queried.
	DBLockMaximumRetry	5	A positive integer representing the maximum number of retry operations when a DBLockException is caught.
	DBLockRetryDelay	500	A positive integer representing the number of milliseconds HIPS waits before retrying an operation when a DBLockException is caught.
	PathologyImaging.CheckForPendingMessages	true	A Boolean value that determines if incoming HL7 messages are rejected if there are any messages for the same patient in a Pending state.

Appendix B Configuring HIPS Web UI Features

B.1 Enabling Patient Registration

To enable the patient registration feature in HIPS UI, set the config value for “EnableRegisterUI“ to *true*. The configuration is managed in the [hips].[HospitalCode] table in the HIPS Core database.

NOTE: Enabling patient registration for a facility will also enable accessing the My Health Record for duplicate patients in that facility.

NOTE: A new code system named “EnableRegisterUI” has been added to the [hips].[CodeSystem] table in the HIPS Core database.

Each hospital has its own configuration for each code system available in the [hips].[CodeSystem] table. Hence, you need to get the [HospitalId] from the [hips].[Hospital] table and [CodeSystemId] for “EnableRegisterUI” from the [hips].[CodeSystem] table.

The [hips].[HospitalCode] table has the following columns:

Column	**Type**	**Description**
HospitalId	int	The Hospital identifier from Hospital table.
CodeSystemId	int	The Code System Identifier CodeSystem table.
Code	varchar(256)	The value for the specific hospital and code system.
DateCreated	datetime	The date and time when the record was created.
UserCreated	varchar(256)	The domain and login of the user that created this record.
DateModified	datetime	The date and time when the record was last modified.
UserModified	varchar(256)	The domain and login of the user that last modified this record.

Query to get [CodeSystemId] for “EnableRegisterUI”:

```
SELECT [CodeSystemId] FROM [hips].[CodeSystem] WHERE [Code] = 'EnableRegisterUI'
```

An example insert statement is:

```
INSERT INTO [hips].[HospitalCode]
([HospitalId],[CodeSystemId],[Code],[DateCreated],[UserCreated],[DateModified],[UserModified
])
VALUES (1,119,'true',GETDATE(),'HIPS',GETDATE(),'HIPS')
```

B.2 Enabling the “Show All Patients” option

To enable the show all patients feature in HIPS UI set the config value for “ShowAllPatientsUI “ to *true*. The configuration is managed in the [hips].[HospitalCode] table in the HIPS Core database.

NOTE: A new code system named “ShowAllPatientsUI” has been added to the [hips].[CodeSystem] table in the HIPS Core database.

Each hospital has its own configuration for each code system available in [hips].[CodeSystem] table. Hence, you need to get the [HospitalId] from the [hips].[Hospital] table and [CodeSystemId] for “ShowAllPatientsUI” from the [hips].[CodeSystem] table.

The [hips].[HospitalCode] table has the following columns:

Column	Type	Description
HospitalId	int	The Hospital identifier from Hospital table.
CodeSystemId	int	The Code System Identifier CodeSystem table.
Code	varchar(256)	The value for the specific hospital and code system.
DateCreated	datetime	The date and time when the record was created.
UserCreated	varchar(256)	The domain and login of the user that created this record.
DateModified	datetime	The date and time when the record was last modified.
UserModified	varchar(256)	The domain and login of the user that last modified this record.

Query to get [CodeSystemId] for “ShowAllPatientsUI”:

```
SELECT [CodeSystemId] FROM [hips].[CodeSystem] WHERE [Code] = 'ShowAllPatientsUI'
```

An example insert statement is:

```
INSERT INTO [hips].[HospitalCode]
([HospitalId],[CodeSystemId],[Code],[DateCreated],[UserCreated],[DateModified],[UserModified
])
VALUES (1,120,'true',GETDATE(),'HIPS',GETDATE(),'HIPS')
```

B.3 Configuring the “All Facilities” option

The steps to create the “All” facility are the same as the creation of other facilities except the hospital’s code must be configured as “ALL” (case sensitive).

Refer to step 2b in section Configure and execute the HIPS Core Database Scripts for an example script for creating facilities. Only the [_CreateFacility] stored procedure is required to be executed in this case.

Appendix C PCEHR Error Codes

PCHER error codes are now stored in a new table in the HIPS Core DB – [hips].[PcehrErrorCode].

The [hips].[PcehrErrorCode] table has the following columns:

Column	Type	Description
PcehrErrorCodeId	int	This the Primary Key and is auto incremented.
Code	varchar(100)	The PCEHR Error Code.
Description	varchar(100)	A description for the PCEHR Error Code.
ErrorCodeType	varchar(100)	The type of PCEHR error code. Types supported by HIPS are: • PCEHR_COMPLETED_WITH_WARNINGS_CODES • PCEHR_PERMANENT_FAILURE_CODES • PCEHR_SUCCESS_MESSAGES • PCEHR_TRANSIENT_FAILURE_CODES Any other type will default to an Unknown fault type.
DateCreated	datetime	The date and time when the record was created.
UserCreated	varchar(256)	The domain and login of the user that created this record.
DateModified	datetime	The date and time when the record was last modified.
UserModified	varchar(256)	The domain and login of the user that last modified this record.

C.1 Adding new PCEHR Error Codes

PCEHR error codes can now be managed by HIPS administrators using SQL Server Management Studio (SSMS).

An example insert statement is:

```
INSERT INTO [hips].[PcehrErrorCode]

([Code], [Description], [ErrorCodeType], [DateCreated], [UserCreated], [DateModified],
[UserModified])

VALUES ('PCEHR_ERROR_3007', 'PCEHR ERROR 3007',
'PCEHR_COMPLETED_WITH_WARNINGS_CODES', GETDATE(),'HIPS', GETDATE(), 'HIPS');
```

C.2 Preconfigured PCEHR Error Codes

The following table represents the PCEHR error codes that are inserted during initial installation/upgrade of HIPS 7.0.0.

Code	ErrorCodeType	Description
PCEHR_ERROR_3007	PCEHR_COMPLETED_WITH_WARNINGS_CODES	PCEHR ERROR 3007
PCEHR_ERROR_3002	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3002
PCEHR_ERROR_2501	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 2501
PCEHR_ERROR_3001	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3001
PCEHR_ERROR_3002	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3002
PCEHR_ERROR_3003	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3003
PCEHR_ERROR_3004	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3004
PCEHR_ERROR_3005	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3005
PCEHR_ERROR_3006	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3006
PCEHR_ERROR_3008	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3008
PCEHR_ERROR_3501	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3501
PCEHR_ERROR_3502	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3502
PCEHR_ERROR_3503	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 3503
PCEHR_ERROR_0001	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0001
PCEHR_ERROR_0002	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0002
PCEHR_ERROR_0003	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0003
PCEHR_ERROR_0004	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0004
PCEHR_ERROR_0006	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0006
PCEHR_ERROR_0007	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0007
PCEHR_ERROR_0008	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0008
PCEHR_ERROR_0009	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0009
PCEHR_ERROR_0010	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0010
PCEHR_ERROR_0501	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0501
PCEHR_ERROR_0502	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0502
PCEHR_ERROR_0503	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0503
PCEHR_ERROR_0504	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0504
PCEHR_ERROR_0505	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0505
PCEHR_ERROR_0506	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0506
PCEHR_ERROR_0509	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0509
PCEHR_ERROR_0510	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0510
PCEHR_ERROR_0511	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0511
PCEHR_ERROR_0512	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0512

Code	**ErrorCodeType**	**Description**
PCEHR_ERROR_0513	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0513
PCEHR_ERROR_0514	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0514
PCEHR_ERROR_0519	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0519
PCEHR_ERROR_0520	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0520
PCEHR_ERROR_0521	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0521
PCEHR_ERROR_0522	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0522
PCEHR_ERROR_0523	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0523
PCEHR_ERROR_0524	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0524
PCEHR_ERROR_0525	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0525
PCEHR_ERROR_0526	PCEHR_PERMANENT_FAILURE_CODES	PCEHR ERROR 0526
MEDVIEW_REPOSITORY_003	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 003
MEDVIEW_REPOSITORY_004	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 004
MEDVIEW_REPOSITORY_006	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 006
MEDVIEW_REPOSITORY_007	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 007
MEDVIEW_REPOSITORY_008	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 008
MEDVIEW_REPOSITORY_010	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 010
MEDVIEW_REPOSITORY_011	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 011
MEDVIEW_REPOSITORY_012	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 012
MEDVIEW_REPOSITORY_014	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 014
MEDVIEW_REPOSITORY_015	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 015
MEDVIEW_REPOSITORY_016	PCEHR_PERMANENT_FAILURE_CODES	MEDVIEW REPOSITORY 016
PCEHR_SUCCESS	PCEHR_SUCCESS_MESSAGES	PCEHR SUCCESS
urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success	PCEHR_SUCCESS_MESSAGES	urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success
PCEHR_ERROR_0005	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0005
PCEHR_ERROR_0011	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0011
PCEHR_ERROR_0012	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0012
PCEHR_ERROR_0013	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0013
PCEHR_ERROR_0014	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0014
PCEHR_ERROR_0507	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0507
PCEHR_ERROR_0515	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0515
PCEHR_ERROR_0516	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0516
PCEHR_ERROR_0517	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0517

Code	ErrorCodeType	Description
PCEHR_ERROR_0518	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0518
PCEHR_ERROR_0527	PCEHR_TRANSIENT_FAILURE_CODES	PCEHR ERROR 0527

Appendix D log4net Configuration

HIPS Core uses log4net which offers flexible logging configuration options for implementers. The default settings are configured in the log4net.config file which can be found in the HIPS installation folder alongside the web.config.

D.1 Configuring Appenders

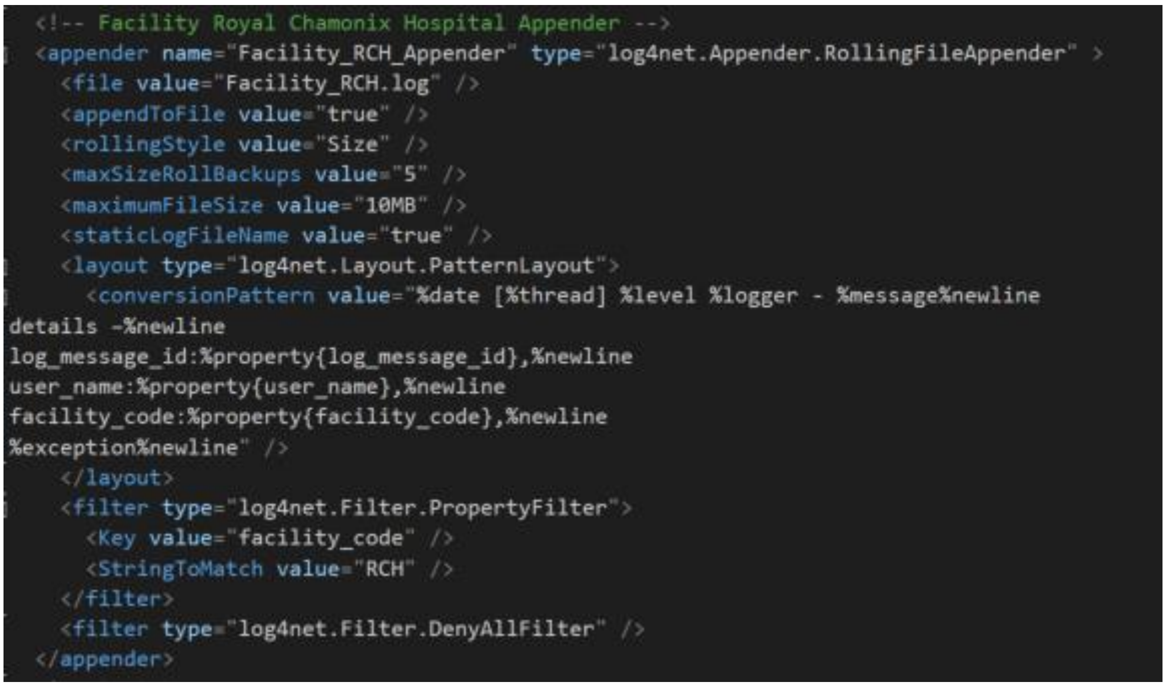
log4net supports many types of appenders for logging to different formats. The HIPS Core configuration file is pre-configured to use the following types of appenders:

- ADO.NET – Used to log to the hips.SystemErrorLog table
- RollingFileAppender – Used to log to the HIPS_Core.log file (located in the same folder as the log4net.config by default)

For more configuration examples refer to the log4net config documentation.

D.2 Filtering log messages based on custom properties

It is often useful to apply custom property filters to appenders. The following RollingFileAppender example contains a filter element which has been configured to only log messages where the custom property facility_code is 'RCH' and outputs any logged messages to a Facility_RCH.log file:



```
  <!-- Facility Royal Chamonix Hospital Appender -->
  <appender name="Facility_RCH_Appender" type="log4net.Appender.RollingFileAppender" >
    <file value="Facility_RCH.log" />
    <appendToFile value="true" />
    <rollingStyle value="Size" />
    <maxSizeRollBackups value="5" />
    <maximumFileSize value="10MB" />
    <staticLogFileName value="true" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%thread] %level %logger - %message%newline
details -%newline
log_message_id:%property{log_message_id},%newline
user_name:%property{user_name},%newline
facility_code:%property{facility_code},%newline
%exception%newline" />
    </layout>
    <filter type="log4net.Filter.PropertyFilter">
      <Key value="facility_code" />
      <StringToMatch value="RCH" />
    </filter>
    <filter type="log4net.Filter.DenyAllFilter" />
  </appender>
```

This is particularly useful for organisations with multiple facilities, as it provides an easy way to filter logs relating to each facility.

D.3 Configuring additional loggers for different HIPS components

The HIPS log4net configuration file contains some useful and currently disabled examples that will log specific messages from the specified code components within HIPS to individual files. These include:

- HIPS.AppServer.ServiceHost
- HIPS.CommonBusinessLogic

When enabled, these loggers will only log events specific to the namespace into the configured destination (currently files). This provides a more detailed view of events in particular areas of HIPS. This can be further refined by specifying more detail in the namespace, down to the class name. The following common business logic namespaces may be useful to log to individual files:

- HIPS.CommonBusinessLogic.Ihi
- HIPS.CommonBusinessLogic.HL7
- HIPS.CommonBusinessLogic.Consent
- HIPS.CommonBusinessLogic.Hi
- HIPS.CommonBusinessLogic.Hpii
- HIPS.CommonBusinessLogic.Pcehr

Note: The logger name must match the namespace exactly if using this method.

D.4 Configuring the layout pattern

The default layout pattern used by HIPS Core to write to files specifies the date, facility, thread, level, logger, message, exception, and all custom properties. If needed, the layout conversion patterns can be modified to show only relevant details for any appenders.

Default log4net properties can be specified in the layout conversion pattern by adding a '%' in front of the property name. For a full list of log4net properties available refer to the log4net Pattern Layouts documentation.

D.5 HIPS Core custom properties

The following list of custom properties are written by HIPS Core (when available) and can be used in the logger layout conversion patterns:

- user_name
- log_message_id
- soap_message_id
- hpio_number
- ihi_number
- facility_mrn
- facility_code
- source_system_episode_id

- filler_order_numbers
- hips_hospital_id
- hips_patient_master_id
- hips_patient_id
- hips_episode_id
- hips_hl7_message_log_id
- hl7_message_control_id
- hl7_message_content
- hl7_message_sending_application
- hl7_message_sending_facility
- hl7_message_date_time
- cda_source_system_set_id
- cda_source_system_document_id
- fault_identifier
- hpii_number
- local_provider_code

Note: properties prefixed with hips_ represent an SQL identifier from the corresponding HIPS table.

IMPORTANT: By default, the ihi_number property is not included in log messages as it is considered personally identifiable information (PII). Careful consideration should be applied to including this property in log messages and securing log destinations appropriately.

These custom properties can be added to layout conversion patterns with the following syntax:

```
property{<propertyname>}
```

For example, the following can be used to display the username property:

```
user_name:%property{user_name}
```

D.6 Filtering Log Levels

HIPS Core is currently configured to logs messages with the INFO, ERROR and FATAL log level. Log levels can be filtered for each logger. The default is ALL.

Appendix E Background Processes

E.1 Queue Consumer

NOTE: The Queue Consumer is configured and installed by the installation scripts in section *5 Application Server*. However, the following sections provide a more detailed reference if further information on configuring or managing the Queue Consumer is required.

HIPS employs an intermediate queuing component to decouple the acceptance of messages for processing from the actual processing of those messages. It currently uses the intermediate queue in two scenarios:

- For requests from HIPS service consumers to interact with the My Health Record system for the purposes of **uploading** or **removing** a clinical document or pathology or diagnostic imaging report.
- For **acknowledgements** to HIPS service consumers of the completion of an interaction with the My Health Record system for the purposes of uploading a pathology or diagnostic imaging report.

An executable component, the HIPS Queue Consumer, must be configured and deployed on one or more HIPS application servers to consume and process queued operations. Multiple instances of the Queue Consumer can be deployed on the same application server or across several application servers. Each instance of the Queue Consumer is dedicated to processing messages from a specific HIPS queue.

The deployment artefacts for the Queue Consumer are located in the `<INSTALL_SOURCE>\HIPS-AppServer\runtime\background\queue-consumer` folder in the HIPS install media.

E.1.1 Configuring the Queue Consumer

IMPORTANT: Changes to the Queue Consumer configuration settings require that the Queue Consumer executable be restarted before the changes will take effect. Refer to the following *Running the Queue Consumer* section for further information on starting and stopping the Queue Consumer executable.

Prior to installing or starting the Queue Consumer it must be configured via its configuration file, named *HIPS.AppServer.ServiceHost.QueueConsumer.exe.config*. The critical sections to be reviewed and updated in the configuration file are:

- `connectionStrings`: References the shared `connectionStrings.config` file to obtain a valid connection string for connecting to the HIPS-Core database. No change should be required.
- `appSettings`: References the shared `appSettings.config` file for common application settings. Contains Queue Consumer specific settings as described below.
- `system.serviceModel/client`: If HL7 acknowledgements will be sent, configure the value of the `address` attribute for the "AckServiceEndPoint" client endpoint. This is configured by default during installation.

In addition to the configuration settings described above, the Queue Consumer may specify the following specific settings in the `appSettings` section:

Name	Description	Minimum	Maximum	Default
QueueConsumer.BatchDegreeOfParallelism	The degree of parallelism for processing messages in a batch. Set to an empty string for maximum parallelism.	1	50	*null*
QueueConsumer.BatchSize	A positive integer value representing the maximum number of queued messages to be acquired in each batch.	1	100	30
QueueConsumer.MaximumRetries	A positive integer value representing the maximum number of times a message lease can be marked for retry before the message is marked as failed.	100	5000	5000
QueueConsumer.MaximumRetryDelay	A positive integer value representing the maximum number of seconds a message lease can be extended when the message has been marked for retry.	300	86400	3600
QueueConsumer.MaximumRetrying	A positive integer value representing the maximum number of message leases that can be retrying at any given time.	10	1000	100
QueueConsumer.PollingInterval	A positive integer value representing the number of seconds between each time the consumer will poll for available messages. It is suggested that this value be adjusted based on the primary workload of the site. For sites whose primary workload is pathology or diagnostic imaging uploads, it is suggested to set this value lower, for example to 5. For other sites it is suggested that this value be left as its default or set higher, for example to 60.	5	300	10
QueueConsumer.RetryDelay	A positive integer value representing the number of seconds a message lease is extended when the message lease has been marked for retry.	5	600	300

Reasonable initial default values for each of these settings are provided in the configuration file. It is vitally important to understand the effect that changes to any of these settings will have on the Queue Consumer and how it will process messages and handle retries.

Instance-specific values may be configured to override the default value by adding another setting with a key prefixed with the name of the instance. In the following example, the default value of “10” for the *QueueConsumer.PollingInterval* setting is overridden for the “ABC.DEF” instance with a value of “20”:

```
<appSettings>
  <!-- ... -->
  <!-- A positive integer value representing the number of seconds
between each time the consumer will poll for available messages. -->
  <add key="QueueConsumer.PollingInterval" value="10" />
  <!-- Instance-specific settings. Prefix key name with instance name. --
>
  <add key="ABC.DEF:QueueConsumer.PollingInterval" value="20" />
  <!-- ... -->
</appSettings>
```

The Queue Consumer also employs log4net for diagnostic logging. A default log4net configuration file named *log4net.config* is included with the Queue Consumer artefacts, and is configured to log to the following destinations:

- `AdoNetAppender_SystemErrorLogTable` writes log messages at "information" level or above (such as errors) to the *SystemErrorLog* table in the HIPS Core database.
- `RollingFileAppender_Log` writes log messages at "Debug" level or above (essentially all messages) to separate files for each configured Queue Consumer instance.
    - A separate file named *HIPS.AppServer.ServiceHost.QueueConsumer_{qc:instanceName}.log* is created in the same directory as the Queue Consumer executable for each Queue Consumer instance, where *{qc:instanceName}* is the name of the Queue Consumer instance that wrote the log message.
    - Additionally, the appender is configured to roll its log files when they reach 20MB, up to five times. This means that by default, for each Queue Consumer instance, at any time there may be 5x20MB log files, with the five most recent files per instance preserved.
- `ConsoleAppender` writes log messages at "Debug" level or above (essentially all messages) to the console when the Queue Consumer is run from the console (described below).

The conversion pattern for both the `RollingFileAppender_Log` and the `ConsoleAppender` is configured to write the following Queue Consumer specific information to the output, which may be useful for diagnostics purposes:

- `%thread`: The managed thread ID, which can be useful for tracking the processing performed in multi-threaded applications such as the Queue Consumer.
- `%property{qc:instanceName}`: The name of the Queue Consumer instance that wrote the log message.
- `%property{qc:consumerType}`: The type of the Queue Consumer that wrote the log message (see below).
- `%property{qc:MessageIdentifier}`: The unique identifier of the queued operation that was being processed by the Queue Consumer instance.

Consult the log4net documentation for additional information on configuring the log4net configuration file.

E.1.2 Running the Queue Consumer

The Queue Consumer may be run in one of two ways:

1. From the command line, for example in a Command Prompt or Windows PowerShell window.
2. Installed as a Windows Service and started like any other service from the Services console provided by the Windows operating system.

Running the Queue Consumer from the command line is only recommended for development or testing purposes, where it is desirable to observe the output from the Queue Consumer in real-time. It is not suitable for Production use.

Regardless of how the Queue Consumer is run, you must specify the following mandatory arguments:

- `instance`: Uniquely identifies the Queue Consumer instance when it is installed as a Windows Service, in the HIPS Core database and in diagnostic logs. The value specified for the instance argument must be 35 characters or less in length.
- `consumerType`: Specifies the type of processing performed by a Queue Consumer instance, must be one of the following values:
    - **MHROperations** creates a Queue Consumer instance capable of processing queued document upload or remove requests against the My Health Record system
    - **HL7Acknowledgements** creates a Queue Consumer instance capable of processing queued HL7 V2 acknowledgement messages
    - **HL7Reports** creates a Queue Consumer instance capable of pre-processing pathology or diagnostic imaging reports submitted as HL7 V2 messages, which includes conversion to CDA prior to upload to the My Health Record system

IMPORTANT: To function correctly, there must be at least one instance of the Queue Consumer for each consumer type active at all times.

To start the Queue Consumer from the command line (for development or testing purposes only):

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Queue Consumer executable is deployed.
3. Execute the following from the command line:

    ```
    .\HIPS.AppServer.ServiceHost.QueueConsumer.exe -
    instance:MHROperations.1 -consumerType:MHROperations
    ```

    Replace the highlighted values as appropriate.
4. Ensure the Queue Consumer starts and logs output to the console window.
5. The Queue Consumer will continue to run as long as the console window is open.
6. To stop the Queue Consumer, press CTRL+C (the “CONTROL” key and the “C” key at the same time).

To install the Queue Consumer as a Windows Service:

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Queue Consumer executable is deployed.
3. Execute the following from the command line:

    ```
    .\HIPS.AppServer.ServiceHost.QueueConsumer.exe install -
    instance:MHROperations.1 -consumerType:MHROperations
    ```

    Replace the highlighted values as appropriate.
4. Ensure the output in the console window reflects that the Queue Consumer has been installed.
5. Open the Services console (Start → Services).
6. Confirm that the Queue Consumer instance is listed in the list of services, with a name of “HIPS-Core Message Queue Consumer (<*consumerTypeDisplay*>) (Instance: <*instanceName*>)” where <*consumerTypeDisplay*> is a display name corresponding to the value specified for the `consumerType` argument, and <*instanceName*> is the value specified for the `instanceName` argument.

Once the Queue Consumer has been installed as a Windows Service it can be started and stopped from the Services console like any other service:

1. In the Services console, locate the Queue Consumer instance to be started.
2. Right-click the instance, and select “Start”.
3. Confirm the instance starts without error. If it does not start, refer to the troubleshooting section below.

By default, the Queue Consumer is also configured to start automatically when the host application server starts, and to shutdown automatically when the host application server is shut down or restarts.

When installing the Queue Consumer as a Windows Service, it is important to review and configure settings related to the Windows Service, such as:

- Startup. By default, the Queue Consumer is installed to startup automatically with the host. This can be changed as appropriate, for example for the service to start automatically but delayed, or to only be started manually.
- Log on. By default, the Queue Consumer is installed to log on as the Local System account. The HIPS installation scripts override this to use the HIPS Core service account specified in the installation artefacts. It is particularly important to ensure that the user account used has appropriate permissions, for example to access the network and to connect to the HIPS Core database. If using a non-domain account such as Local System, the connection string used to connect to the HIPS Core database must specify a user name and password for a SQL Server login.
- Recovery. By default, the Queue Consumer is installed to take no action when it fails unexpectedly (which should be rare). Alternative recovery options can be specified to attempt to restart the service or run an external program upon failure.

These settings can be configured manually through the Services console. Alternatively, many of these settings may also be configured from the command line when installing the Queue Consumer instance, refer to TopShelf command line for further information.

To uninstall a Queue Consumer instance that has been installed as a Windows Service:

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Queue Consumer executable is deployed.
3. Execute the following from the command line:

   ```
   .\HIPS.AppServer.ServiceHost.QueueConsumer.exe uninstall -
   instance:MHROperations.1 -consumerType:MHROperations
   ```

   Replace the highlighted values as appropriate.
4. Ensure the output in the console window reflects that the Queue Consumer has been uninstalled.

E.1.3 Troubleshooting

This section provides guidance on troubleshooting issues with the Queue Consumer.

As a first step in troubleshooting, it is desirable to review the output logged by the Queue Consumer to the *HIPS.AppServer.ServiceHost.QueueConsumer.log* files. This may provide valuable diagnostic information to assist with further troubleshooting. Additional information may also be located in the Application and System logs in the Windows Event Log.

E.1.3.1 Issue: Queue Consumer fails to start

If the Queue Consumer fails to start, it will output information that can be used to diagnose the reason for the failure. If it is run from the command line, it will output this information in the console window. If it is run as a Windows Service, this information will be written to the Application and System logs in the Windows Event Log.

The most common issues that can cause the Queue Consumer to fail on startup are:

Issue	Corrective Action
Incorrect arguments specified on command line	If the value of the `consumerType` argument is not specified correctly, the Queue Consumer will fail to start. Specify one of the values provided previously. When the Queue Consumer is installed as a Windows Service, you can locate the arguments provided to the Queue Consumer executable in the *Path to executable* field for the corresponding service entry in the Services console.
Account does not have correct permissions	As indicated previously, the account used by the Queue Consumer requires a number of permissions. If the error information indicates that the Queue Consumer cannot connect to the HIPS Core database, review the value of the connection string configured in the `connectionStrings` section of the configuration file to ensure it is correct. If the connection string is configured to use Windows integrated security, ensure the Windows user account being used to run the Queue Consumer has appropriate network and database access permissions. Alternatively, if the connection string is configured to use SQL Server authentication, ensure the SQL Server login has appropriate database permissions.

E.1.3.2 Issue: Queue Consumer cannot be removed

Generally, the Queue Consumer should be installed and removed using the PowerShell scripts provided as part of the installation artefacts. Occasionally upon removal, one or more Queue Consumer instances may remain listed in the Services console in a “Disabled” state. Attempts to re-install the Queue Consumer instance with the same instance name will fail as the operating system has marked the previously installed service for deletion but has not yet completed the action.

In this case, it is likely that the Queue Consumer instance has been successfully removed, but an open Windows handle is preventing the operating system from completely removing it from its list of installed services. Commonly this is caused by the Services console retaining a handle, and closing and reopening the Services console will result in releasing the handle and the removal of the service being completed. Occasionally other operating system processes may be holding this open handle, and the only way for it to be released is for the Application Server to be restarted.

It is important to understand that this issue is not specific to the Queue Consumer, but is a Windows operating system issue. For further information, refer to the following links:

- https://github.com/Topshelf/Topshelf/issues/35
- https://stackoverflow.com/questions/225275/how-to-force-uninstallation-of-windows-service

E.2 Alert Monitoring

NOTE: The Alert Monitoring service is configured and installed by the installation scripts in section *5 Application Server*. However, the following sections provide a more detailed reference if further information on configuring or managing the Alert Monitoring service is required.

Prior to the release of HIPS 7.0, implementers were required to manually monitor HIPS for potential issues. The addition of the Alert Monitoring service introduces proactive reporting of alert conditions, allowing implementers to quickly identify and resolve potential issues before they cause problems.

An executable component, the HIPS Alert Monitoring service, should be configured and deployed on each HIPS application servers to process alert items that are configured in the HIPS Core database.

The deployment artefacts for the Alert Monitoring service are in the `<INSTALL_SOURCE>\HIPS-AppServer\runtime\background\alert-monitoring` folder in the HIPS install media.

E.2.1 Configuring the Alert Monitoring Service

IMPORTANT: Changes to the Alert Monitoring configuration settings require the Alert Monitoring executable be restarted before the changes will take effect. Refer to the *Running the Alert Monitoring* section for further information on starting and stopping the Alert Monitoring executable.

Prior to installing or starting the Alert Monitoring service it must be configured via its configuration file, named *HIPS.AppServer.ServiceHost.AlertMonitoring.exe.config*. The critical sections to be reviewed and updated in the configuration file are:

- `connectionStrings`: References the shared `connectionStrings.config` file to obtain a valid connection string for connecting to the HIPS-Core database. No change should be required.

In addition to the configuration settings described above, the Alert Monitoring service may specify the following specific settings in the `appSettings` section:

Name	Description	Default
AlertConsumer.ShutdownTimeout	The number of seconds Windows will wait before shutting down the service.	10

The Alert Monitoring service also employs log4net for diagnostic logging. A default log4net configuration file named *log4net.config* is included with the Alert Monitoring artefacts, and is configured to log to the following destinations:

- `AdoNetAppender_SystemErrorLogTable` writes log messages at “information” level or above (such as errors) to the *SystemErrorLog* table in the HIPS Core database.
- `RollingFileAppender_Log` writes log messages at “ALL” level to a HIPS_AlertMonitoring.
  - Additionally, the appender is configured to roll its log files when they reach 20MB, up to five times. This means that by default, for each Alert Monitoring instance, at any time there may be 5x20MB log files, with the five most recent files per instance preserved.
- `ConsoleAppender` writes log messages at “ALL” level to the console when the Alert Monitoring executable is run from the console (described below).

The conversion pattern for both the `RollingFileAppender_Log` and the `ConsoleAppender` is configured to write the following Alert Monitoring specific information to the output, which may be useful for diagnostics purposes:

- `%property{alert_item_name}`: The name of the current executing alert item that wrote the log message.
- `%property{alert_item_id}`: The identifier of the current executing alert item that wrote the log message.
- `%property{host}`: The server DNS host name.

Consult the log4net documentation for additional information on configuring the log4net configuration file.

E.2.2 Running the Alert Monitoring Service

The Alert Monitoring service may be run in one of two ways:

1. From the command line, for example in a Command Prompt or Windows PowerShell window.
2. Installed as a Windows Service and started like any other service from the Services console provided by the Windows operating system.

Running the Alert Monitoring service from the command line is only recommended for development or testing purposes, where it is desirable to observe the output from the Alert Monitoring service in real-time. It is not suitable for Production use.

To start the Alert Monitoring service from the command line (for development or testing purposes only):

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Alert Monitoring executable is deployed.
3. Execute the following from the command line:
   ```
   .\HIPS.AppServer.ServiceHost.AlertMonitoring.exe
   ```
4. Ensure the Alert Monitoring starts and logs output to the console window.
5. The Alert Monitoring will continue to run as long as the console window is open.
6. To stop the Alert Monitoring, press CTRL+C (the “CONTROL” key and the “C” key at the same time).

To install the Alert Monitoring as a Windows Service:

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Alert Monitoring executable is deployed.
3. Execute the following from the command line:
   ```
   .\HIPS.AppServer.ServiceHost.AlertMonitoring.exe install -
   instance:Alerts.1
   ```
   Replace the highlighted value as appropriate.
4. Ensure the output in the console window reflects that the Alert Monitoring has been installed.
5. Open the Services console (Start → Services).
6. Confirm that the Queue Consumer instance is listed in the list of services, with a name of “HIPS-Core Alert Monitoring (Instance: *<instanceName>*)” where *<instanceName>* is the value specified for the `instanceName` argument.

Once the Alert Monitoring has been installed as a Windows Service it can be started and stopped from the Services console like any other service:

1. In the Services console, locate the Alert Monitoring instance to be started.
2. Right-click the instance and select “Start”.
3. Confirm the instance starts without error. If it does not start, refer to the troubleshooting section below.

By default, the Alert Monitoring is also configured to start automatically when the host application server starts, and to shutdown automatically when the host application server is shut down or restarts.

When installing the Alert Monitoring as a Windows Service, it is important to review and configure settings related to the Windows Service, such as:

- Startup. By default, the Alert Monitoring is installed to startup automatically with the host. This can be changed as appropriate, for example for the service to start automatically but delayed, or to only be started manually.
- Log on. By default, the Alert Monitoring is installed to log on as the Local System account. The HIPS installation scripts override this to use the HIPS Core service account specified in the installation artefacts. It is particularly important to ensure that the user account used has appropriate permissions, for example to access the network and to connect to the HIPS Core database. If using a non-domain account such as Local System, the connection string used to connect to the HIPS Core database must specify a user name and password for a SQL Server login.
- Recovery. By default, the Alert Monitoring is installed to take no action when it fails unexpectedly (which should be rare). Alternative recovery options can be specified to attempt to restart the service or run an external program upon failure.

These settings can be configured manually through the Services console. Alternatively, many of these settings may also be configured from the command line when installing the Alert Monitoring instance, refer to TopShelf command line for further information.

To uninstall an Alert Monitoring instance that has been installed as a Windows Service:

1. Open a new Command Prompt or Windows PowerShell window.
2. Change directory to the directory where the Alert Monitoring executable is deployed.
3. Execute the following from the command line:

   ```
   .\HIPS.AppServer.ServiceHost.AlertMonitoring.exe uninstall -
   instance:Alerts.1
   ```

   Replace the highlighted value as appropriate.
4. Ensure the output in the console window reflects that the Alert Monitoring has been uninstalled.

E.2.3 Configuring Alert Items

E.2.3.1 Alert Item Table

Alert items are managed in the hips.AlertItem table which has the following columns:

Column	Description
Alert Item Id	The alert item identifier.

Column	Description
Name	The alert item name. The following alert item names can be used to define the type of processor the alert item should be executed by: • Certificate Processor: **HICertExpiry**, **MyHRCertExpiry**, **HPOCertExpiry** • Application Server Disk Space Processor: **AppDiskSpace** • Database Server Disk Space Processor: **DBDiskSpace** • Store Procedure Processor: Any name (this is the default processor).
StoredProcName	(Optional) The name of the stored procedure that the alert item should execute.
StoredProcParameters	(Optional) The stored procedure parameters that the processor should provide to the stored procedure referenced in the StoredProcName column. This should be provided as a simple JSON array. For example: '{ "FirstParameter": 123, “SecondParameter”, “xyz” }'
Threshold	A Threshold must always be provided and is used by all processors. The Threshold can be provided in a human friendly format. The following types of threshold are supported by each processor: • Certificate Processor: **Timespan**. • Application Server Disk Space Processor: **Information size**. • Database Server Disk Space Processor: **Information size**. • Store Procedure Processor: **Information size, Timespan, Percentage or Count**. Refer to the Alert Item Formats section below for valid examples.
Period	(Optional) The Period can be provided in a human friendly format and will be used by the Stored Procedure Processor only. The period will be provided as a parameter to the specified stored procedure if present. Refer to the Alert Item Formats section below for valid timespan examples.
Frequency	The Frequency can be provided in a human friendly format and is used by all processors. The Frequency determines how often the alert item should be executed. Refer to the Alert Item Formats section below for valid timespan examples.
LevelCode	Foreign Key to the Code column of the LogLevel table. Defines the log4net log level code assigned to this AlertItem.

Column	Description
Message	The message that should be alerted if the Threshold is exceeded. The message can include tokens which will be replaced by the processors. For example: "HI Service Certificate for Health Provider Organisation {name} expires on {value} in less than threshold {threshold}." The following tokens are generic and can be used in any processor: • {Threshold} • {Period} • {Frequency} • {LevelCode} Each processor has the following additional tokens: • Certificate Processor: ○ {name} – the name of the certificate ○ {value} – the expiry date of the certificate • Application Server Disk Space Processor: ○ {drive} – the name of the drive ○ {host} – the name of the host server ○ {value} – the amount of gigabytes remaining • Database Server Disk Space Processor: ○ {drive} – the name of the drive ○ {host} – the name of the host server ○ {value} – the amount of megabytes remaining • Store Procedure Processor: ○ {count} – the number of rows returned ○ {value} – the value of the first row
Active	Determines whether the alert item should be processed.

E.2.3.2 Predefined Alert Items

The following predefined alert items have been configured in the HIPS Core database table hips.AlertItems. These alert items can be marked inactive if not required, but they should not be deleted.

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
1	HICertExpiry			14 days	NULL	1 day	FATAL	HI Service Certificate for Health Provider Organisation {name} expires on {value} in less than threshold {threshold}.	1
2	MyHRCertExpiry			14 days	NULL	1 day	FATAL	My Health Record certificate for Health Provider Organisation {name} expires on {value} in less than threshold {threshold}.	0
3	HPOCertExpiry			14 days	NULL	1 day	FATAL	NASH PKI Certificate for Healthcare Provider Organisation {name} expires on {value} in less than threshold {threshold}.	1
4	AppDiskSpace			5 GB	NULL	1 hour	FATAL	Disk space {value} remaining on drive {drive} on HIPS application server {host} is less than threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
5	DBDiskSpace	dbo.GetDBDiskSpace		5 GB	NULL	1 hour	FATAL	Disk space {value} remaining on drive {drive} on HIPS database server {host} is less than threshold {threshold}.	1
6	MinInboundPASRequests	Hips. HL7MessageInbound MinCount	{ "SendingApplication" :"[PAS SendingApplication]" }	3	30 minutes	15 minutes	ALERT	Number of messages {value} from {SendingApplication} in the past {period} is lower than threshold {threshold}.	1
7	MaxInboundPASRequests	Hips. HL7MessageInbound MaxCount	{ "SendingApplication" :"[PAS SendingApplication]" }	200	5 minutes	15 minutes	SEVERE	Number of messages {value} from {SendingApplication} in the past {period} exceeds threshold {threshold}.	1
8	MinInboundLISRequests	Hips. HL7MessageInbound MinCount	{ "SendingApplication" :"[LIS SendingApplication]" }	3	30 minutes	15 minutes	ALERT	Number of messages {value} from {SendingApplication} in the past {period} is lower than threshold {threshold}.	1
9	MaxInboundLISRequests	Hips. HL7MessageInbound MaxCount	{ "SendingApplication" :"[LIS SendingApplication]" }	1000	5 minutes	15 minutes	SEVERE	Number of messages {value} from {SendingApplication} in the past {period} exceeds threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
10	MinInboundRISRequests	Hips. HL7MessageInboundMinCount	{ "SendingApplication" :"[RIS SendingApplication]" }	3	30 minutes	15 minutes	ALERT	Number of messages {value} from {SendingApplication} in the past {period} is lower than threshold {threshold}.	1
11	MaxInboundRISRequests	Hips. HL7MessageInboundMaxCount	{ "SendingApplication" :"[RIS SendingApplication]" }	100	5 minutes	15 minutes	SEVERE	Number of messages {value} from {SendingApplication} in the past {period} exceeds threshold {threshold}.	1
12	PendingUploadQueue	Hips.MessageQueuePendingCount		100	NULL	15 minutes	SEVERE	Number of pending items on upload queue {value} exceeds threshold {threshold}.	1
13	FailedUploads	Hips. MessageQueueFailedCount		10%	15 minutes	5 minutes	SEVERE	Proportion of failed uploads {value} in the past {period} exceeds threshold {threshold}.	1
14	IHIAlerts	Hips.IhiLookupAlertCount		10%	15 minutes	5 minutes	CRITICAL	Proportion of IHI Alerts {value} in the past {period} exceeds threshold {threshold}.	1
15	MHRQueue	Hips.MessageQueuePendingDuration		5 minutes	NULL	20 minutes	SEVERE	MHR Queue processing time {value} exceeds threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
16	DBErrors	Hips.SystemErrorLogDBErrorCount		5	10 minutes	10 minutes	CRITICAL	Number of database errors {value} in HIPS Core database in the past {period} exceeds threshold {threshold}.	1
17	HIServiceUnavailable	Hips. IhiLookupAlertUnavailableMinutes		5 minutes	NULL	10 minutes	CRITICAL	No successful connections to the HI Service for {value} exceeding threshold {threshold}.	1
18	MyHRServiceUnavailable	Hips. PcehrAuditUnavailableMinutes		5 minutes	NULL	10 minutes	CRITICAL	No successful connections to the My Health Record system for {value} exceeding threshold {threshold}.	1
19	MinMyHRUploadRequests	hips.SystemInteractionLogMinCount	{ "SystemInteractionID": 35 }	1	1 hour	1 hour	CRITICAL	Number of upload requests {value} in the past {period} is lower than threshold {threshold}.	1
20	IHIConsumerSearch	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 10 }	3 seconds	10 minutes	10 minutes	SEVERE	An IHI consumer search in the past {period} took {value} seconds exceeding threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
21	HPIIBatchSearch	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 1 }	10 seconds	10 minutes	10 minutes	SEVERE	An HPI-I Batch Search submit or retrieve in the past {period} took {value} seconds exceeding threshold {threshold}.	1
22	HPIIIndividualSearch	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 2 }	5 seconds	10 minutes	10 minutes	SEVERE	An HPI-I Individual Search in the past {period} took {value} seconds exceeding threshold {threshold}.	1
23	MyHRDoesPcehrExist	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 20 }	3 seconds	10 minutes	10 minutes	SEVERE	A check for an advertised My Health Record in the past {period} took {value} seconds exceeding threshold {threshold}.	1
24	MyHRUploadDocument	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 35 }	20 seconds	10 minutes	10 minutes	SEVERE	The upload of a document to the My Health Record in the past {period} took {value} seconds exceeding threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
25	MyHRRemoveDocument	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 32 }	5 seconds	10 minutes	10 minutes	SEVERE	The removal of a document from the My Health Record in the past {period} took {value} seconds exceeding threshold {threshold}.	1
26	MyHRRetrieveDocument	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 33 }	20 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a document from the My Health Record in the past {period} took {value} seconds exceeding threshold {threshold}.	1
27	MyHRGetChangeHistoryView	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 24 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Change History View in the past {period} took {value} seconds exceeding threshold {threshold}.	1
28	MyHRGetViewPD	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 26 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Prescription and Dispense View in the past {period} took {value} seconds exceeding threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
29	MyHRGetViewDIR	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 27 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Diagnostic Imaging Report View in the past {period} took {value} seconds exceeding threshold {threshold}.	1
30	MyHRGetViewHRO	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 28 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Health Record Overview in the past {period} took {value} seconds exceeding threshold {threshold}.	1
31	MyHRGetViewMO	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 29 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Medicare Overview in the past {period} took {value} seconds exceeding threshold {threshold}.	1
32	MyHRGetViewPR	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 30 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Pathology Report View in the past {period} took {value} seconds exceeding threshold {threshold}.	1

AlertItemId	Name	StoredProcName	StoredProcParameters	Threshold	Period	Frequency	LevelCode	Message	Active
33	MyHRGetDocumentList	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 25 }	10 seconds	10 minutes	10 minutes	SEVERE	The retrieval of a Document List in the past {period} took {value} seconds exceeding threshold {threshold}.	1
34	MyHRGainAccessWithCode	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 21 }	5 seconds	10 minutes	10 minutes	SEVERE	An attempt to gain access with code in the past {period} took {value} seconds exceeding threshold {threshold}.	1
35	MyHRGainAccessWithoutCode	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 22 }	5 seconds	10 minutes	10 minutes	SEVERE	An attempt to gain access without code in the past {period} took {value} seconds exceeding threshold {threshold}.	1
36	MyHRGainEmergencyAccess	Hips.SystemInteractionLogMaxElapsed	{ "SystemInteractionID": 23 }	5 seconds	10 minutes	10 minutes	SEVERE	An attempt to gain emergency access in the past {period} took {value} seconds exceeding threshold {threshold}.	1

E.2.3.3 Customising Alert Items

Implementers can modify existing alert items if they want to adjust alerting metrics. The Threshold, Period, Frequency and Message are reasonably safe fields to update. It is recommended that implementers save a backup of the table before modification in case the update does not work as expected.

Implementers are also able to create their own alert items by adding new rows to the hips.AlertItem table. The following alert item names should be used to define the type of processor the alert item should use:

Processor Type	Valid Names
Certificate Processor	HICertExpiry MyHRCertExpiry HPOCertExpiry
Application Server Disk Space Processor	AppDiskSpace
Database Server Disk Space Processor	DBDiskSpace
Store Procedure Processor (default processor)	Any name can be used.

A custom stored procedure can be added to the HIPS database and referenced by name in an Alert Item StoredProcName field. The stored procedure should handle the logic for whether an alert should be generated and return a single value with a column alias if the threshold is exceeded, for example:

The following returns a count of records from the hips.SystemInteractionLog that have a DateCreated within the @Period where the SystemInteractionId = @SystemInteractionId, if @Count is less than @Threshold.

```
@SystemInteractionId INT,
@Period INT, -- as seconds
@Threshold INT -- number to compare with @Count


 DECLARE @Count INT = (SELECT COUNT(sil.[SystemInteractionLogId])
                       FROM [hips].[SystemInteractionLog] sil
                       WHERE sil.DateCreated > DATEADD(SECOND,-@Period, GETDATE())
                         AND sil.SystemInteractionId = @SystemInteractionId);

IF @Count < @Threshold
       SELECT @Count AS Result
```

Notes:

- An alias is required for data selected by the procedure otherwise the processor will ignore it.
- If a value is returned to the processor, an alert will be generated.
- When a processor sends a Period or Threshold as a timespan to a stored procedure it is always provided as seconds.

E.2.3.4 Alert Item Formats

Threshold, Period and Frequency can all be provided in readable formats. The following formats are supported:

Type	Supported Formats
Information size	KB | MB | GB | TB For example:

	"1000 MB", "5 GB" "1.5 TB"
Timespan	minute[s], | second[s] | hour[s] | day[s] For example: "20 seconds", "30 minutes", "5 days"
Percentage	Number that includes '%' (must be within range 0-100%). For example: "50%"
Count	Standard number For example: "5"

E.2.4 Troubleshooting

This section provides guidance on troubleshooting issues with the Alert Monitoring.

As a first step in troubleshooting, it is desirable to review the output logged by the Alert Monitoring service to the *HIPS_AlertMonitoring.log* files. This may provide valuable diagnostic information to assist with further troubleshooting. Additional information may also be located in the Application and System logs in the Windows Event Log.

E.2.4.1 Issue: Alert Monitoring service fails to start

If the Alert Monitoring service fails to start, it will output information that can be used to diagnose the reason for the failure. If it is run from the command line, it will output this information in the console window. If it is run as a Windows Service, this information will be written to the Application and System logs in the Windows Event Log.

The most common issues that can cause the Alert Monitoring to fail on startup are:

Issue	Corrective Action
Account does not have correct permissions	As indicated previously, the account used by the Alert Monitoring requires a number of permissions. If the error information indicates that the Alert Monitoring cannot connect to the HIPS Core database, review the value of the connection string configured in the connectionStrings section of the configuration file to ensure it is correct. If the connection string is configured to use Windows integrated security, ensure the Windows user account being used to run the Alert Monitoring has appropriate network and database access permissions. Alternatively, if the connection string is configured to use SQL Server authentication, ensure the SQL Server login has appropriate database permissions. To run the Alert Monitoring DBDiskSpace alert processor, the <HIPS_CORE_SERVICE_ACCOUNT> must have the VIEW SERVER STATE permission.

E.2.4.2 Issue: Alert Monitoring service cannot be removed

Generally, the Alert Monitoring service should be installed and removed using the PowerShell scripts provided as part of the installation artefacts. Occasionally upon removal, one or more Alert Monitoring instances may remain listed in the Services console in a “Disabled” state. Attempts to re-install the Alert Monitoring instance with the same instance name will fail as the operating system has marked the previously installed service for deletion but has not yet completed the action.

In this case, it is likely that the Alert Monitoring instance has been successfully removed, but an open Windows handle is preventing the operating system from completely removing it from its list of installed services. Commonly this is caused by the Services console retaining a handle, and closing and reopening the Services console will result in releasing the handle and the removal of the service being completed. Occasionally other operating system processes may be holding this open handle, and the only way for it to be released is for the Application Server to be restarted.

It is important to understand that this issue is not specific to the Alert Monitoring service, but is a Windows operating system issue. For further information, refer to the following links:

- https://github.com/Topshelf/Topshelf/issues/35
- https://stackoverflow.com/questions/225275/how-to-force-uninstallation-of-windows-service

Appendix F Certificates

F.1 Installation

The installation script imports the following certificates into the Local Computer:

- HPI-O, CSP and DHS Site certificates into the Personal store.
- The Medicare Australia Root Certification Authority into the Trusted Root Certification Authorities store.
- The Medicare Australia Organisation Certification Authority into the Intermediate Certification Authorities store.

F.2 Nash PKI Certificate

The NASH PKI Certificate for Healthcare Provider Organisations is supplied by DHS for connection to the My Health Record B2B Services. This certificate is used for HPI-O and CSP organisations.

HIPS validates the NASH certificates by performing a policy OID check, chain trust check and a revocation check. Some extra configuration is required for these validations.

For the policy OID check, the for web.config file templates included in the installation package, but if you are upgrading from a previous version then the extra setting will need to be included. See the NashProviderOid application setting in Appendix A for the setting value.

For the revocation check, HIPS connects back to the certificate authority to download the Certificate Revocation List (CRL) so that it can check the status of the certificate. The following URL for a test certificate is:

http://matest.certificates-australia.com.au/TestMAOCA/latest.crl

For a production certificate, it is a different URL that can be found in the CRL Distribution Point property of the certificate.

For sites that need this connection to go through a proxy server, there is further information in the below linked Microsoft blog post about how to specify the proxy settings for CRL checks.

https://blogs.msdn.microsoft.com/jpsanders/2011/02/21/certificate-revocation-list-crl-check-and-winhttp-proxy-settings/

F.3 DHS Site Certificate

The DHS Site PKI Certificate (or DHS HI Network Organisation PKI Certificate) that is supplied by DHS for connection to the HI Service B2B Services.

If your integration strategy is to supply pre-validated IHI numbers to HIPS and disable the built-in IHI search / validation functionality, then this certificate may not be required. If you are using the DatabaseLoaderService then this certificate is essential.