Pukka-j

Nexus User Manual

Version 3.07.00

Pukka-j Limited

November 27, 2025

Contents

1 Introduction

This document provides instructions for using the functions of the **Pukka-j Nexus** software application. Nexus is a service that supports **DICOM**, **HL7**, and **HTTP** listening, allowing the configuration of sub-services that can receive data from either external or local sources. Incoming data is passed through a configurable processing pipeline before being forwarded or cached to a designated output destination.

1.1 Who Should Read This Manual?

This manual is intended for **System Administrators** working with the Pukka-j Nexus software product. It assumes a working knowledge of networking, DICOM and HL7 protocols, and system configuration tasks.

1.2 Copyright

Pukka-j's software products and related documentation contain confidential and proprietary information. Only licensees of Pukka-j are entitled to use the information contained herein. Licensees specifically granted copy and/or transfer rights may do so within the bounds of their agreement.

Unauthorized use, disclosure, reproduction, assignment, or transfer of this confidential information is strictly prohibited and will be prosecuted to the fullest extent of the law.

The software, this manual, and all related materials remain the property of Pukka-j and may not be reproduced, copied, or distributed without prior written permission from Pukka-j.

1.3 Regulatory

Pukka-j shall not be liable or obligated in any manner for bodily injury or property damage resulting from the use of this software if it is not used in strict accordance with:

- the instructions and safety precautions contained in this manual and all applicable supplements,
- all warnings on product labels, and
- the terms of the software warranty and sale agreement.

Furthermore, Pukka-j accepts no responsibility for the consequences of unauthorized changes made to the software.

1.3.1 Reporting of Incidents

Pukka-j is committed to the continuous improvement of its software and values user feedback. Users are encouraged to report any incidents or suggestions that may enhance the safety, effectiveness, or performance of the software.

1.3.2 EU Vigilance Reports

In accordance with the EU Vigilance system, all incidents – defined by applicable regulatory guidelines – must be reported to Pukka-j as soon as possible, and in any case within **48 hours** of occurrence.

1.3.3 Restrictions on the Use of the Software

Only individuals who have been trained in the use of the software by Pukka-j or by certified Pukka-j trainers are authorized to operate it. An official list of authorized operators should be maintained.

Warning: The interpretation, reporting, or storage of medical images and associated data may impact patient safety. All users must be thoroughly familiar with this manual, especially the sections outlining warnings and precautions.

Further regulatory detail is available by request:

Pukka-j Information Security Management Policy.

2 Software Prerequisites

This section outlines the software prerequisites required to enable certain functionality within the application.

2.1 Databases

The application supports various types of databases for data storage. By default, it includes support for **Java DB** (also known as **Apache Derby**), which is available as an embedded option and requires no external setup.

However, if you intend to use external databases, they must be installed and configured prior to use. The application must also be granted appropriate access privileges to allow it to automatically create and manage the required database structures.

2.1.1 MySQL

The application supports connections to **MySQL** databases. Installation instructions can be found in the official MySQL documentation. The application is compatible with both 32-bit and 64-bit environments.

Communication with the MySQL database is established via a JDBC connection string (URL) and the supplied access credentials. These credentials must be configured with sufficient privileges to allow the application to manage the database schema.

2.1.2 Microsoft SQL Server

The application also supports connections to **Microsoft SQL Server**. Like MySQL, it is compatible with both 32-bit and 64-bit environments. Installation instructions for SQL Server can be found in the official Microsoft documentation.

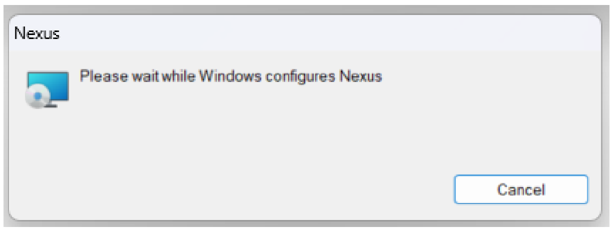
Connections to SQL Server can be made using either:

- **Standard authentication**: using a JDBC URL and SQL login credentials.
- **Integrated security**: where the operating system user running the service is granted access to the SQL Server instance.

In both cases, a suitable **JDBC driver** and a corresponding **native OS library** (DLL on Windows or SO on Linux) are required. Ensure the correct version of the driver and library is used for compatibility with your SQL Server installation.

3 Installing the Software

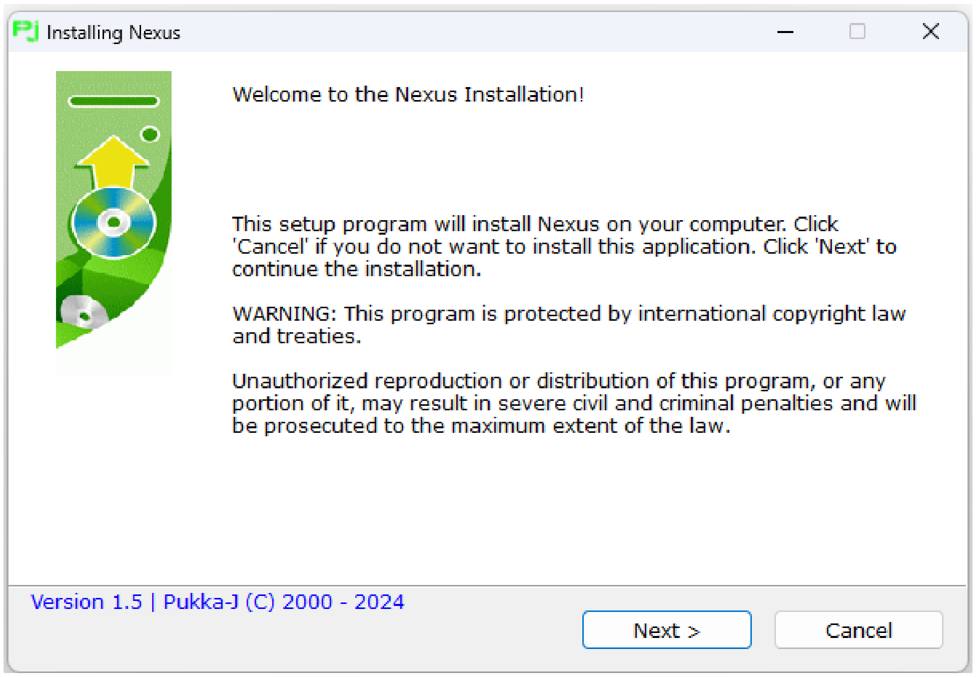
The standard installation method for Windows systems is via an `.MSI` installer, which sets up and configures the initial parameters for the Nexus service. Launching the installer will display an initial dialogue indicating that the MSI is being configured, as shown in Figure 1.


Nexus

Please wait while Windows configures Nexus

Figure 1: Initial MSI setup dialogue.

Once the MSI package has been unpacked and is ready to proceed, a welcome message is displayed, as shown in Figure 2.


Installing Nexus

Welcome to the Nexus Installation!

This setup program will install Nexus on your computer. Click 'Cancel' if you do not want to install this application. Click 'Next' to continue the installation.

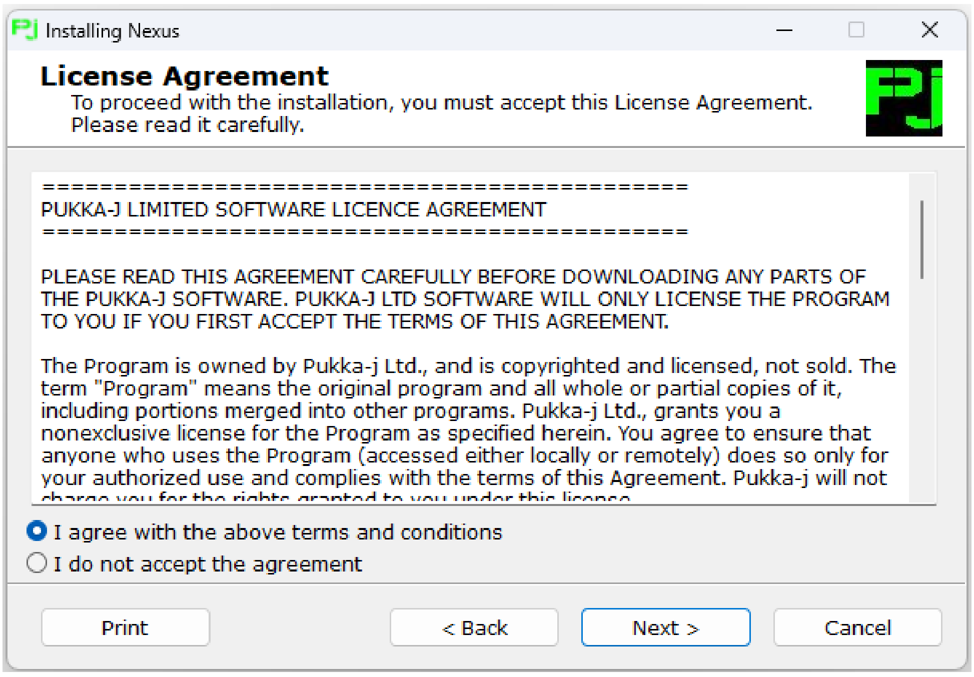
WARNING: This program is protected by international copyright law and treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties and will be prosecuted to the maximum extent of the law.

Version 1.5 | Pukka-J (C) 2000 - 2024

Figure 2: MSI welcome screen.

The welcome screen is followed by the license agreement dialogue, which must be accepted in order to proceed with the installation (Figure 3).


Installing Nexus

License Agreement

To proceed with the installation, you must accept this License Agreement.
Please read it carefully.

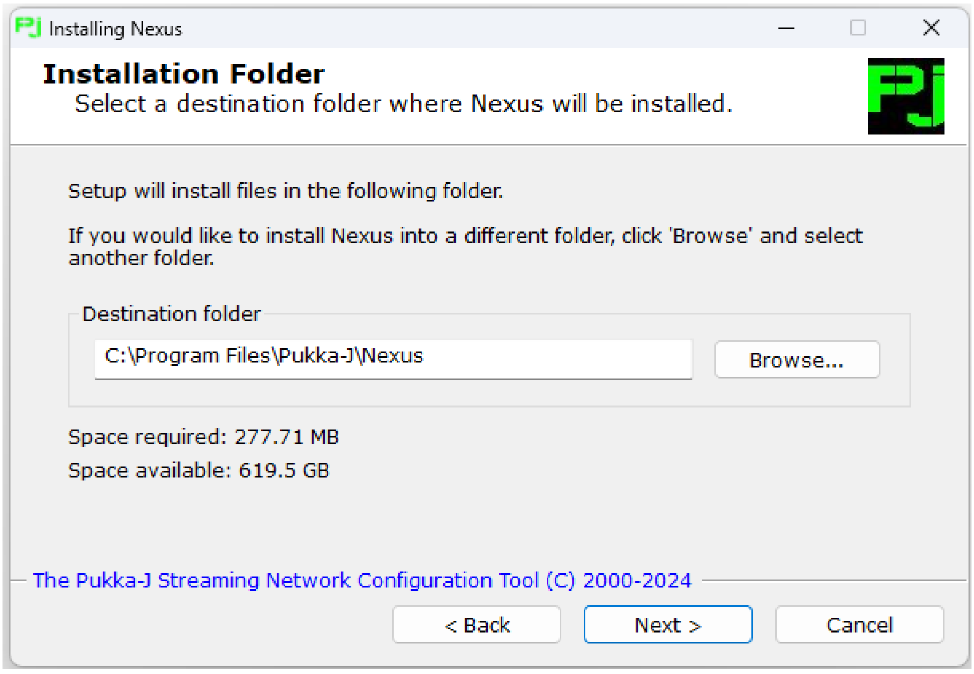
```
============================================
PUKKA-J LIMITED SOFTWARE LICENCE AGREEMENT
============================================

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE DOWNLOADING ANY PARTS OF
THE PUKKA-J SOFTWARE. PUKKA-J LTD SOFTWARE WILL ONLY LICENSE THE PROGRAM
TO YOU IF YOU FIRST ACCEPT THE TERMS OF THIS AGREEMENT.

The Program is owned by Pukka-j Ltd., and is copyrighted and licensed, not sold. The
term "Program" means the original program and all whole or partial copies of it,
including portions merged into other programs. Pukka-j Ltd., grants you a
nonexclusive license for the Program as specified herein. You agree to ensure that
anyone who uses the Program (accessed either locally or remotely) does so only for
your authorized use and complies with the terms of this Agreement. Pukka-j will not
```

[✓ I agree with the above terms and conditions]
I do not accept the agreement

Figure 3: Nexus license agreement dialogue.

Please review the license agreement carefully, select the checkbox to accept, and click **Next** to continue.

Next, the installation directory can be selected. By default, Nexus will be installed in the `Program Files` directory, as shown in Figure 4.


Installing Nexus

Installation Folder

Select a destination folder where Nexus will be installed.

Setup will install files in the following folder.

If you would like to install Nexus into a different folder, click 'Browse' and select another folder.

Destination folder: C:\Program Files\Pukka-J\Nexus

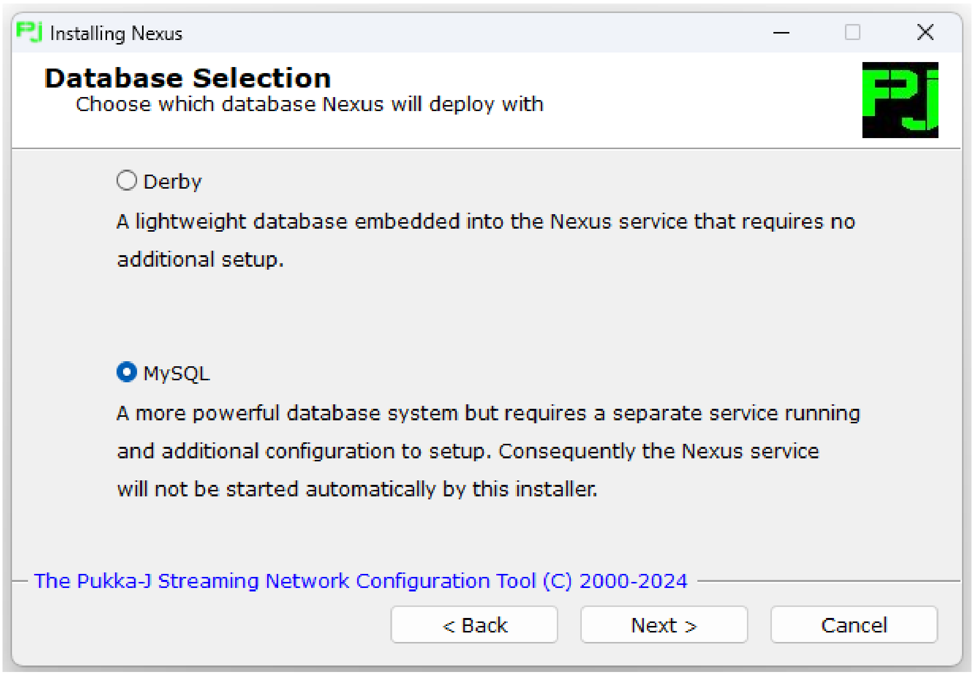
Space required: 277.71 MB

Space available: 619.5 GB

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 4: Installation location selection and disk space requirements.

Figure 5 shows the dialogue for selecting the initial system database that will be used by the Nexus service. **Derby (Java DB)** is integrated and does not require a separate installation. If you select **MySQL**, it must be installed and configured independently before installation. This selection applies only to the internal system database—additional database types can be configured later for individual nodes or services.



Database Selection
Choose which database Nexus will deploy with

Derby [✓ MySQL]

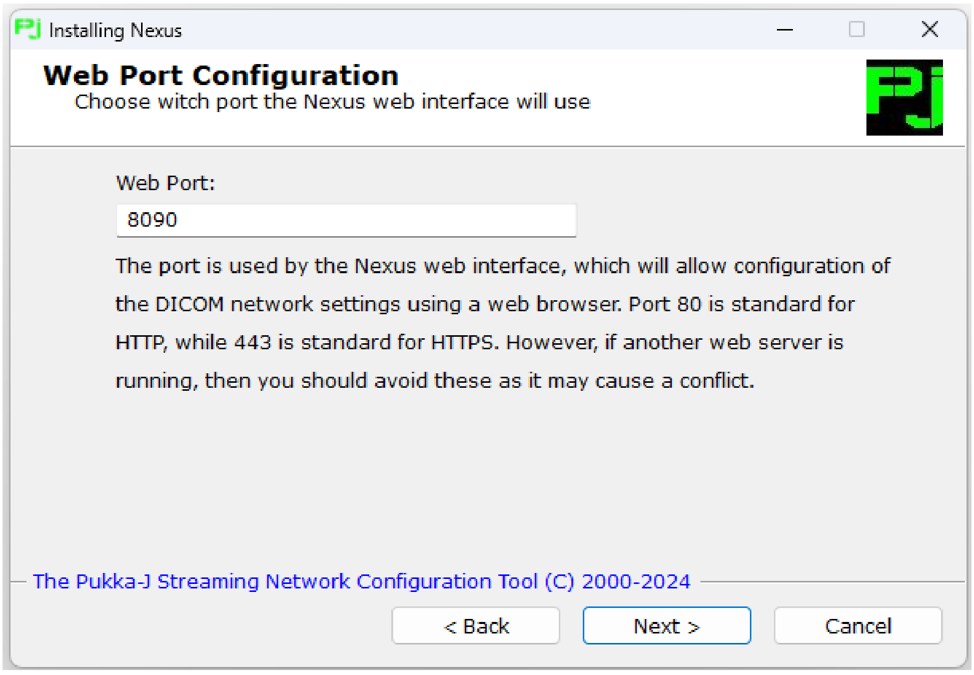
Derby: A lightweight database embedded into the Nexus service that requires no additional setup.

MySQL: A more powerful database system but requires a separate service running and additional configuration to setup. Consequently the Nexus service will not be started automatically by this installer.

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 5: System database selection dialogue.

The main HTTP(S) service port—used to access the Nexus configuration UI—is configured in the screen shown in Figure 6. This port must be available on the host machine. The standard ports are `80` for HTTP and `443` for HTTPS.



Web Port Configuration
Choose witch port the Nexus web interface will use

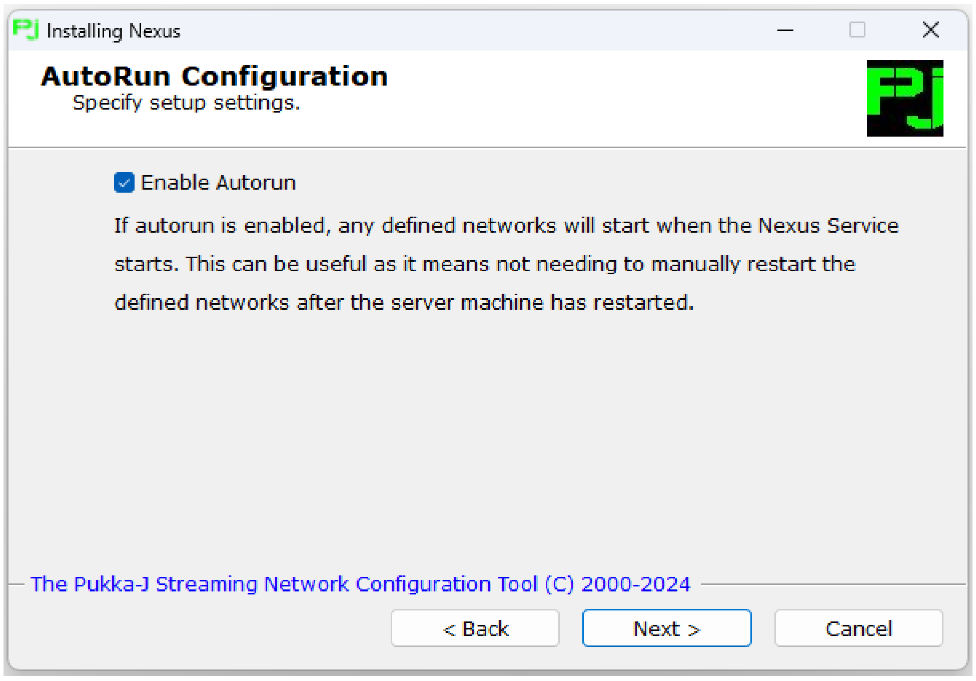
Web Port: 8090

The port is used by the Nexus web interface, which will allow configuration of the DICOM network settings using a web browser. Port 80 is standard for HTTP, while 443 is standard for HTTPS. However, if another web server is running, then you should avoid these as it may cause a conflict.

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 6: Primary web service port configuration.

Figure 7 allows you to specify whether the Nexus service should start automatically once the Windows service infrastructure is active.


Installing Nexus

AutoRun Configuration
Specify setup settings.

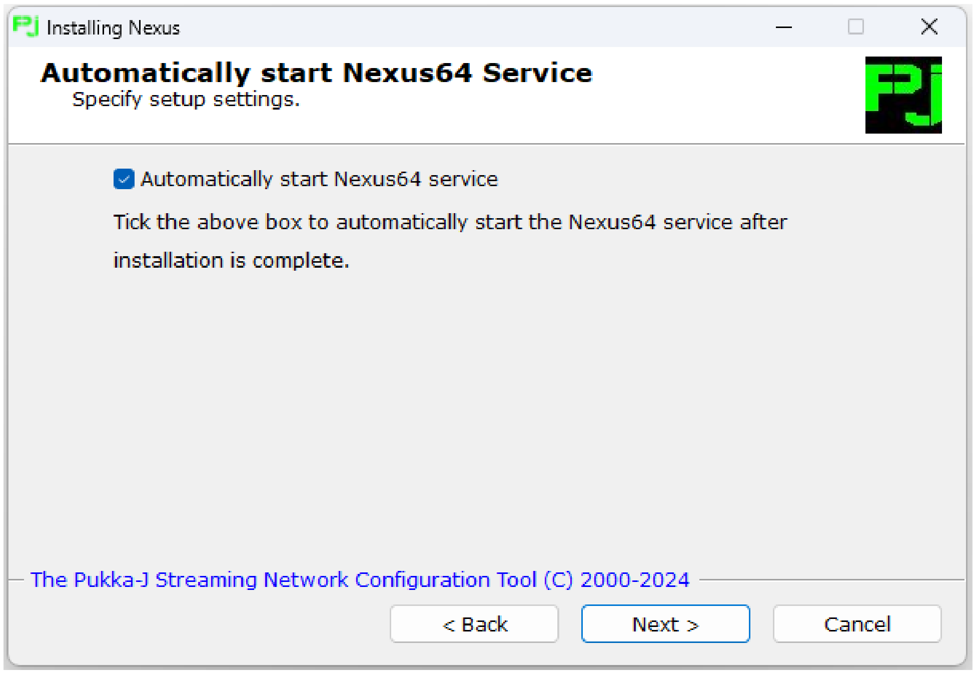
☑ Enable Autorun

If autorun is enabled, any defined networks will start when the Nexus Service starts. This can be useful as it means not needing to manually restart the defined networks after the server machine has restarted.

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 7: Nexus service autorun configuration.

You can also configure whether the underlying **Windows service** itself should start automatically after the installation is complete, as shown in Figure 8.


Installing Nexus

Automatically start Nexus64 Service
Specify setup settings.

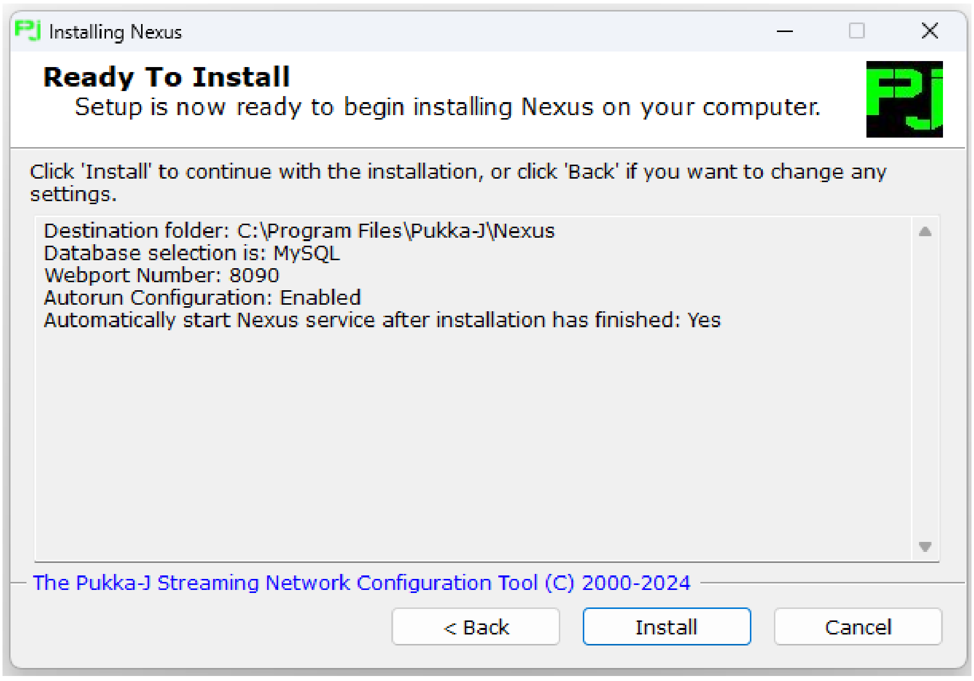
☑ Automatically start Nexus64 service

Tick the above box to automatically start the Nexus64 service after installation is complete.

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 8: Windows service start configuration.

Before installation begins, a confirmation screen is shown (Figure 9). Click **Install** to start the installation process.



Ready To Install
Setup is now ready to begin installing Nexus on your computer.

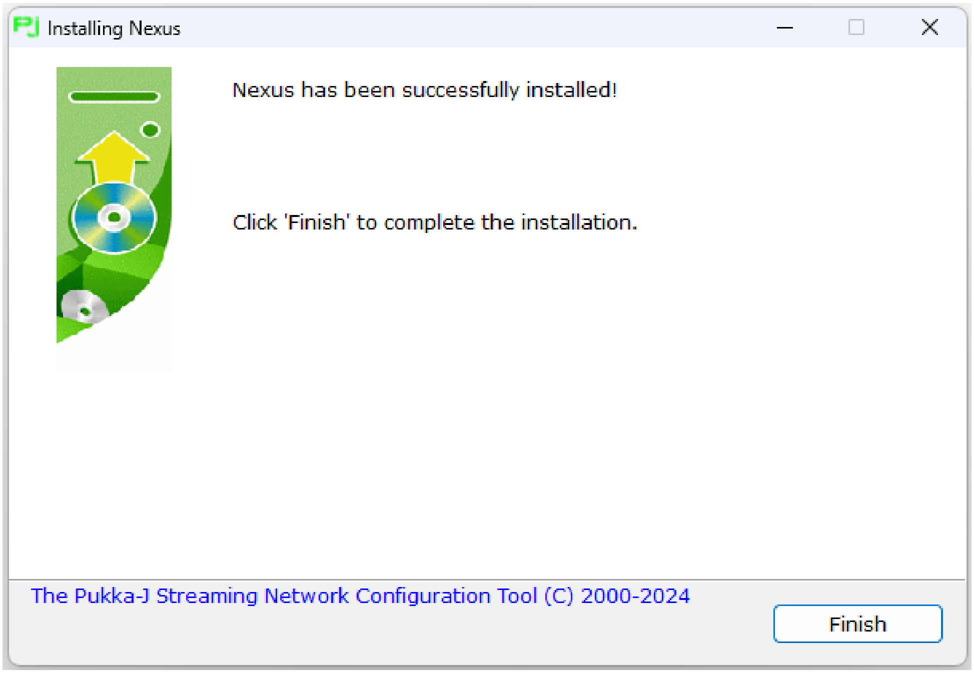
Click 'Install' to continue with the installation, or click 'Back' if you want to change any settings.

Destination folder: C:\Program Files\Pukka-J\Nexus
Database selection is: MySQL
Webport Number: 8090
Autorun Configuration: Enabled
Automatically start Nexus service after installation has finished: Yes

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 9: Installation confirmation dialogue.

Once installation is complete, a final confirmation screen is displayed, as shown in Figure 10.



Nexus has been successfully installed!

Click 'Finish' to complete the installation.

The Pukka-J Streaming Network Configuration Tool (C) 2000-2024

Figure 10: Installation completion screen.

4 Configuring the Service

While the Windows installation captures the core configuration parameters, additional settings can be customized after installation to tailor the Nexus service to your environment.

The top-level configuration file is located in the application's installation directory, typically at:

```
C:\Program Files\Pukka-j\Nexus\deployment.properties
```

4.1 Service Port

The **primary service port** defines the port on which the Nexus service starts. It is used to serve the browser-based HTTP user interface.

If `port 80` is used, it is not necessary to include the port in the URL to access the interface. The port is set using the following property:

```
service.port=80
```

If this property is not set or is commented out, port 80 is used by default.

4.2 Debug Level

The logging level can be adjusted to control the amount of information recorded in the service logs. By default, the level is set to `WARNING`, which logs warning and error messages.

To increase the verbosity, modify the property as follows:

```
service.debug.level=INFO
```

The available logging levels, in increasing verbosity, are:

```
service.debug.level=OFF
service.debug.level=SEVERE
service.debug.level=WARNING
service.debug.level=INFO
service.debug.level=CONFIG
service.debug.level=FINE
service.debug.level=FINER
service.debug.level=FINEST
service.debug.level=ALL
```

4.3 Database Settings

The primary database connection is configured using the following properties. For example, to connect to a MySQL database running locally on the default port:

```
service.db.url=jdbc:mysql://localhost:3306
```

To configure an embedded Java DB (Derby) database:

```
service.db.url=jdbc:derby:dbtest;create=true
```

Additional properties for specifying the database name and access credentials:

```
service.db.name=dbtest
service.db.user=testuser
service.db.pw=test1234
```

The configured primary database manages user accounts and UI session control for the application.

4.4 Service Settings

The following parameters control how the service behaves on startup and how it integrates with central configuration systems:

- `service.autorun=true`
  Starts the network pipelines automatically when the service is launched.
- `service.name=Nexus`
  Sets a display name for the service instance for identification.
- `service.update.name=NexusWin64`
  Defines the name of the Windows service to be restarted during remote software updates.
- `service.websocket.url=ws://localhost:8887/Desktop-8082`
  Registers the Nexus service with a central configuration system via WebSocket. The URL suffix (e.g., `Desktop-8082`) distinguishes multiple instances on the same host.
- `service.update.baseurl=http://localhost:8081`
  Specifies the location from which update files will be downloaded.
- `service.auth.http.proxy.url=http://localhost:8081`
  Enables proxying of authentication requests to a centralized authentication service.

5 Network Configuration

The network configuration is accessed by browsing to the host (and port) on which the primary service is hosted. This will display the network configuration user interface within the web browser page. The locked display only shows the operational status of the network(s) and provides a query interface onto some nodes in the network. Users may be required to enter valid user credentials to unlock the interface as show in Figure 11.

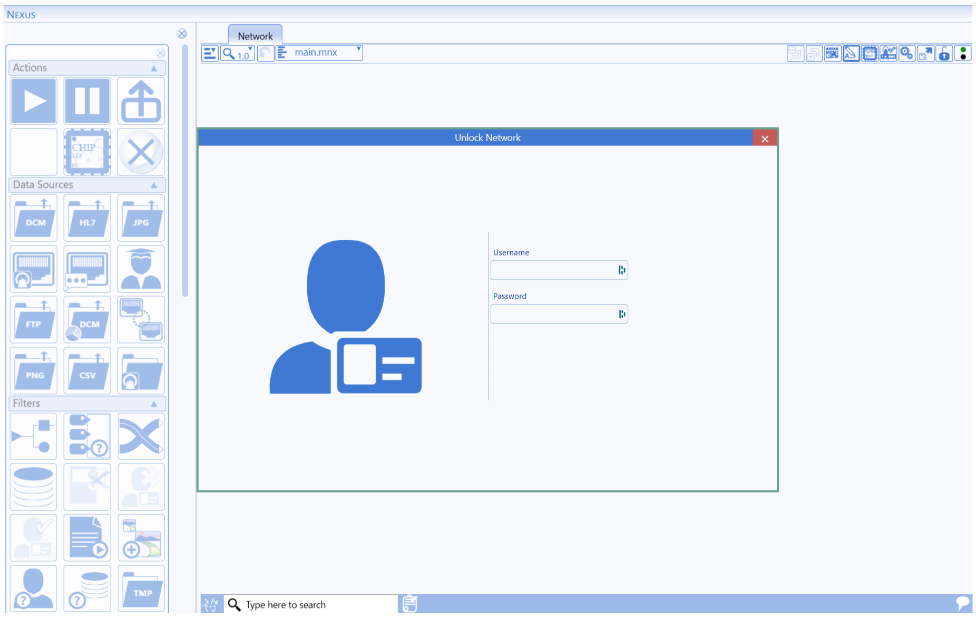


Figure 11: The figure shows the unlock dialogue that requires an admin login in order to configure the network.

The unlocked network configuration interface allows users to build a network to suit specific requirements. Some functionality may be restricted depending upon which components have been licensed. Components are added to the network by selecting the options from the left-hand sidebar. Typically, a network consists of an input node and at least one output node with any number of nodes in between depending on what the network is being designed to do.

Information on the current version and the configuration options can be found by selecting the configuration (cogs) icon at the top right of the display.

5.1 Network Actions

Network actions are used to control the status of the network and to modify its configuration settings.

5.1.1 Applying the Network Configuration

Applies the current service configuration to the server. Changes are not active until this action is performed.

The **Start** option is used to apply any configuration changes to the server. Changes made in the editor will not take effect until the configuration is applied. Once applied, any input nodes are triggered to start listening for inputs or begin processing resources, depending on their configuration.

5.1.2 Export Service

Saves the current network definition to a local file for export or backup.

The **Export** option allows the current service configuration to be saved to a local file. This does not apply the configuration to the server but provides a way to back up or share the network definition.

5.1.3 Clear Service

Clears the current service configuration from the editor. Changes are not applied until the apply action is used.

The **Clear** option removes the current network from the configuration editor. This action only affects the local editor and does not affect the server until the changes are explicitly applied.

5.1.4 Pause System

Pauses all active network pipelines.

The **Pause** option halts all running network pipelines. Input nodes will complete their current tasks before pausing. Individual pipelines can also be paused using their specific pause buttons. For example, a DICOM Listening Port will reject new incoming associations, while nodes such as a File Reader will pause after finishing their current job.

5.1.5 Chip Editor

Opens the chip node editor.

The **Chip Editor** allows users to encapsulate a group of nodes into a reusable *chip node*. This helps modularize and simplify complex network logic. Selecting this option opens the chip editor, which is described in detail in a later section.

5.2 Nexus Nodes

Functionality is added to a network by building a pipeline consisting of an **input node**, one or more **output nodes**, and optional **filter nodes** in between. While nodes share some common behaviours, each node type has configuration and operational traits unique to its role.

This section provides an overview of the main node types and their respective controls.

5.2.1 Input Node

Input nodes serve as the entry point for data into the network. They either listen on a port for inbound connections or read from a local resource.

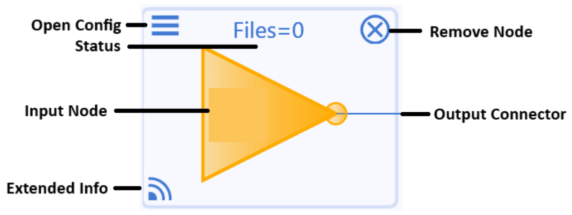


Figure 12: General view of a Nexus input node.

All configurable nodes include a **Config** button (top-left) that opens a node-specific configuration dialogue. Despite differences between nodes, many share common elements such as labelling and configuration export/import options (see Section 5.2.4).

Some nodes also include a **Remove** button, which removes the node from the current network view (changes are not saved until the network is applied).

In certain cases, nodes may include an additional button in the bottom-left corner. This indicates support for an **extended view** used to manage internal processes. For example, the Anonymisation Node includes a button to launch the Anonymisation Manager, enabling pseudonymisation management and ID data review.

When the network is running, each node reports status information. A summary is shown at the top of the node. Clicking this status label opens a detailed status panel displaying all available runtime parameters.

5.2.2 Filter Node

Filter nodes provide the primary processing functionality within a Nexus network. Their behaviour is typically data-specific (e.g., DICOM or HL7), and the available operations vary widely across node types.

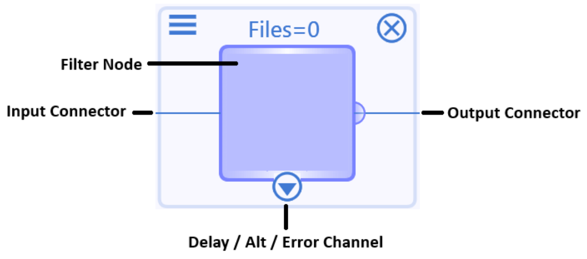


Figure 13: General view of a Nexus filter node.

Filter nodes share the same core controls and status interface as input nodes. Some filters provide a secondary output channel (shown at the bottom of the node) which can be used to route alternative data types.

For instance, a **DICOM to HL7** filter will pass DICOM data through the main output and route generated HL7 messages through the alternate channel. Refer to the documentation for each node to determine whether an alternate output is used and what it represents.

5.2.3 Output Node

Output nodes are responsible for delivering data to external systems or local destinations. A pipeline may include multiple output nodes, either due to data splitting or because of alternate channels from upstream nodes.

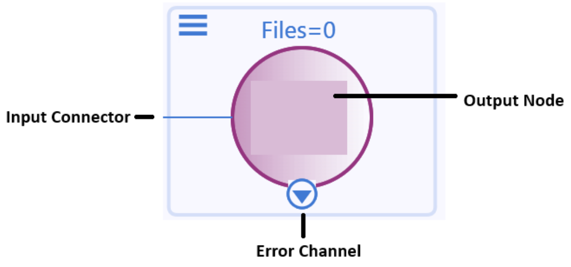


Figure 14: General view of a Nexus output node.

The downward-facing channel on an output node typically represents an **error channel**, used when the node fails to transmit data or encounters an error response. Error outputs can be routed to dedicated error-handling nodes in the pipeline.

5.2.4 General Configuration

Most nodes include a **Node Config** tab in their configuration dialogue, allowing the user to define a human-readable label and export configuration settings.

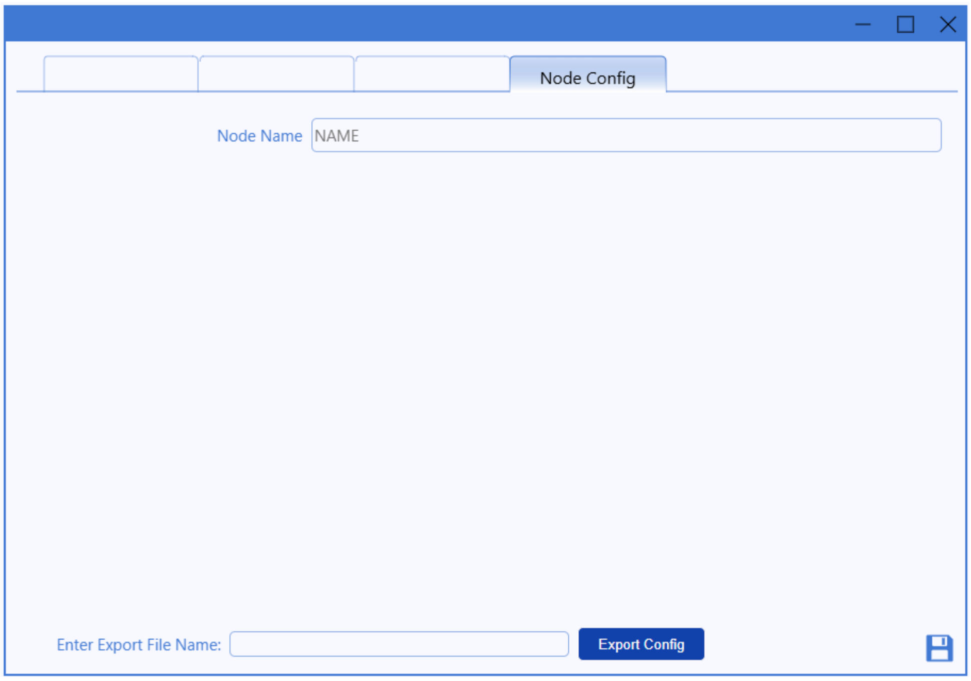


Figure 15: Common configuration tab used in most nodes.

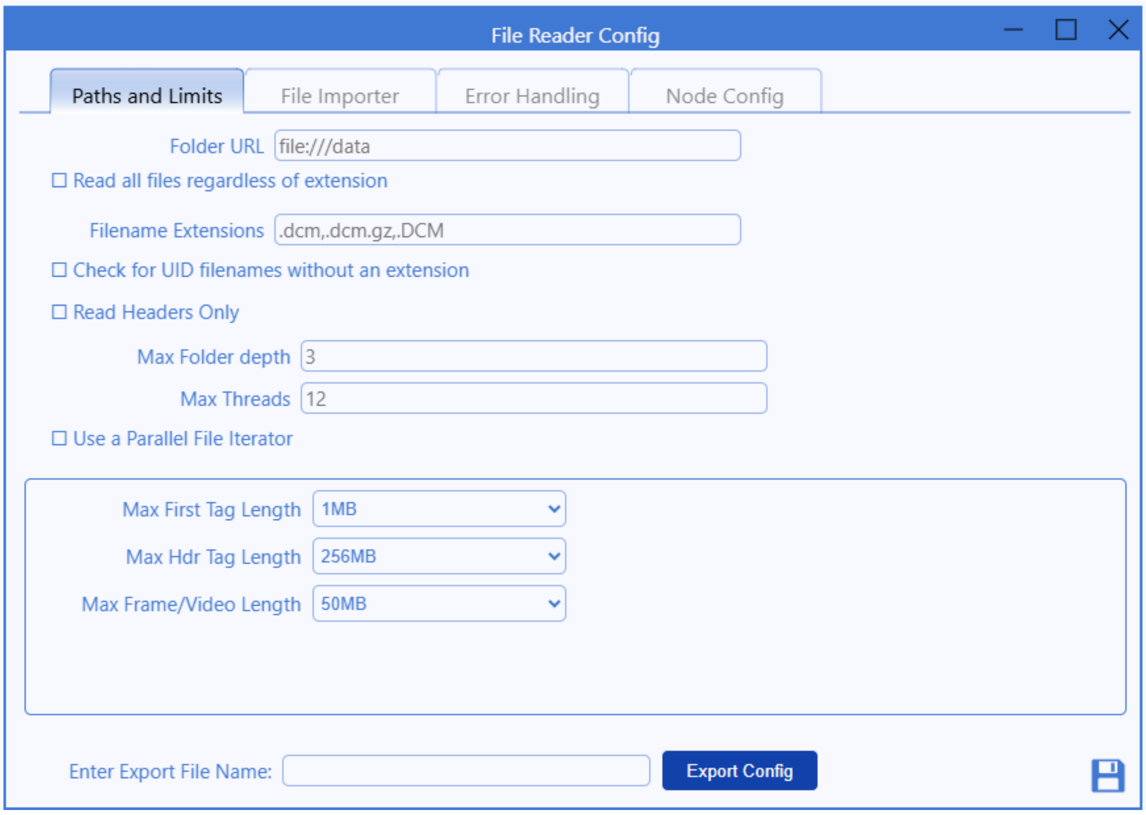
The assigned name may be used elsewhere in the Nexus interface to identify the origin of specific data. You can export the node's configuration using the **Export Config** button. To import a saved configuration, simply drag and drop the file onto the configuration dialogue.

5.3 Input Nodes

5.3.1 DICOM File Input Node

Adds a DICOM file source to the network as an input node. Reads files from disk and streams them through the pipeline.

The **DICOM File Input Node** acts as a source of DICOM files from the local or network file system to be processed by the Nexus pipeline. It is the primary mechanism for importing DICOM files into the system. The configuration panel for this node is shown in Figure 16.



Folder URL: file:///data

☐ Read all files regardless of extension

Filename Extensions: .dcm,.dcm.gz,.DCM

☐ Check for UID filenames without an extension

☐ Read Headers Only

Max Folder depth: 3

Max Threads: 12

☐ Use a Parallel File Iterator

Max First Tag Length: 1MB

Max Hdr Tag Length: 256MB

Max Frame/Video Length: 50MB

Enter Export File Name:

Figure 16: File reader configuration dialogue for reading DICOM files from a local file source.

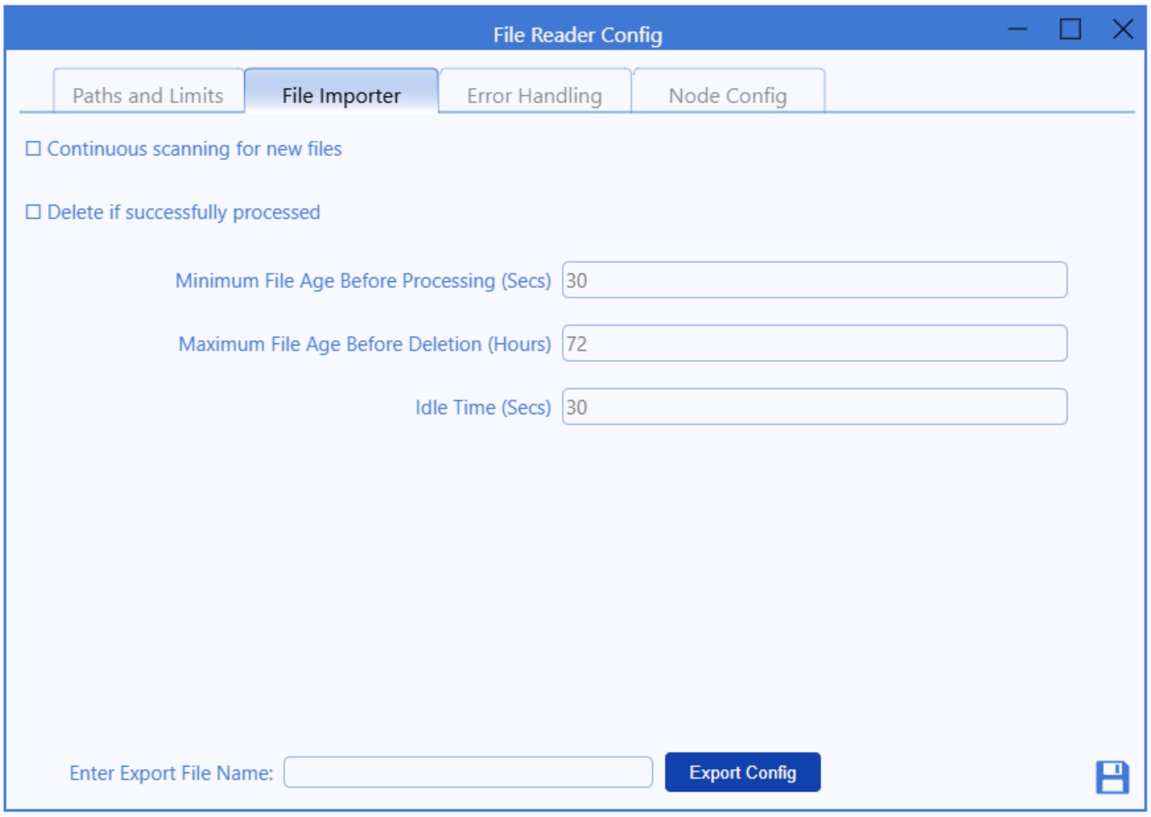
The **folder URL** specifies the local file location from which DICOM files are read. It can begin with `file://` or be a direct absolute path (e.g., `/data` or `D:\data`). Specific file extensions can be defined to filter input; by default, the system searches for `.dcm`, `.dcm.gz`, and `.DCM`. A comma-separated list of additional extensions may also be used.

Alternatively, the node can be configured to identify files that resemble DICOM UIDs by enabling the corresponding checkbox. If a ZIP archive is encountered and its extension matches one listed, the archive contents are treated as a folder and processed accordingly.

If the "**headers only**" option is enabled, the reader will extract only the DICOM headers without reading full image data. This mode is useful for building databases by passing metadata to downstream database nodes without importing the full image files.

The **maximum folder depth** setting controls how deep into the file system the node searches before dividing the workload across multiple threads. The number of threads used can also be configured.

Additionally, limits for tag and frame sizes can be specified using the relevant configuration fields, as shown in Figure 16.



☐ Continuous scanning for new files

☐ Delete if successfully processed

Minimum File Age Before Processing (Secs): 30

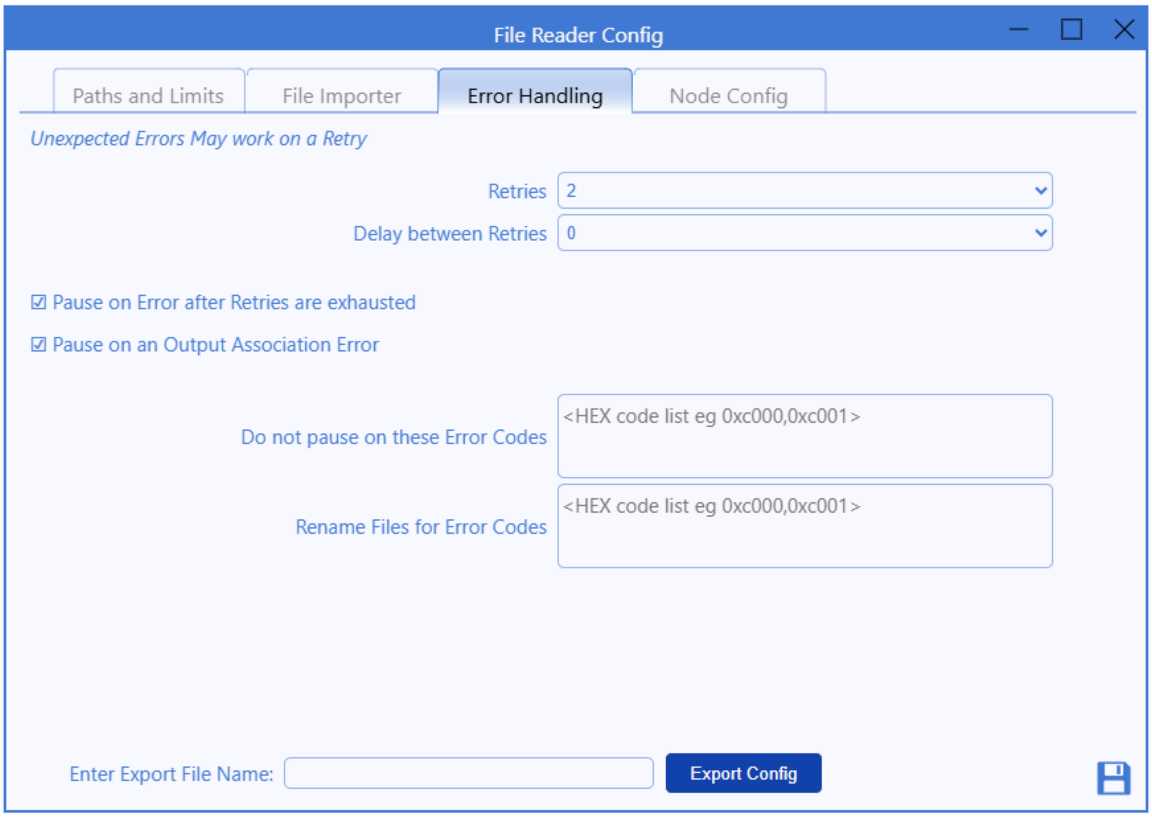
Maximum File Age Before Deletion (Hours): 72

Idle Time (Secs): 30

Figure 17: File importer options.

An alternative mode of operation for the file input node is as a continuous **importer**. In this mode, the node monitors a specified location for new files. As files appear, they are passed into the pipeline and may optionally be deleted after successful processing.

The importer can be configured using the options shown in Figure 17. The **minimum file age** setting defines how long a file must remain unmodified before it is considered safe to process—this helps avoid reading partially written files. The **maximum file age** defines the time limit for attempting to process a file before it is skipped. The **idle time** determines how long the importer waits between processing cycles.


File Reader Config

Paths and Limits | File Importer | Error Handling | Node Config

Unexpected Errors May work on a Retry

Retries: 2

Delay between Retries: 0

☑ Pause on Error after Retries are exhausted

☑ Pause on an Output Association Error

Do not pause on these Error Codes: <HEX code list eg 0xc000,0xc001>

Rename Files for Error Codes: <HEX code list eg 0xc000,0xc001>

Enter Export File Name:

Export Config

Figure 18: Error handling options available for the DICOM File Input Node.

In the event of errors during file reading or downstream processing, the DICOM File Input Node can be configured to **retry**, **pause**, or skip files depending on the error type. If the error relates to a CODEC failure, a maximum tag size violation, or an association issue, the node will not retry the same file. For association errors, the node may be configured to pause the pipeline via a checkbox option.

For all other errors, the node will retry processing the file up to a configurable maximum number of attempts. An optional delay can be set between retries. If the maximum retry limit is reached without success, the node may pause the pipeline if so configured (see Figure 18).

A list of error codes may also be defined under "**Do not pause on these Error Codes**". These codes, provided as a comma-separated list, instruct the node to skip pausing for specific error conditions.

Once configuration is complete, click the **save** icon to store the settings. Note that no configuration is applied to the live server until the **run** icon is selected from the actions section.

5.3.2 DICOM Stream Input Node

A DICOM input node that listens on a specified port for incoming DICOM data.

The **DICOM Stream Input Node** acts as a DICOM service listener, initiated on a defined port to handle inbound DICOM requests. Figure 19 shows a basic example of using a DICOM listener within a network.

The tri-bar menu icon at the top-left of the node opens the configuration dialogue. The icon at the bottom-left provides access to communication logs and statistics, if this functionality is enabled in the configuration.

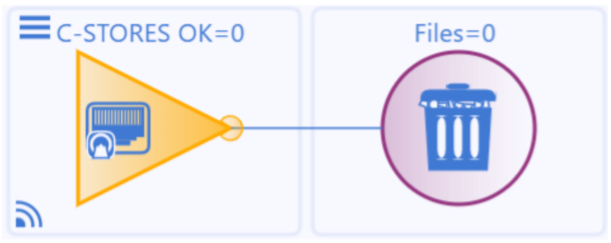
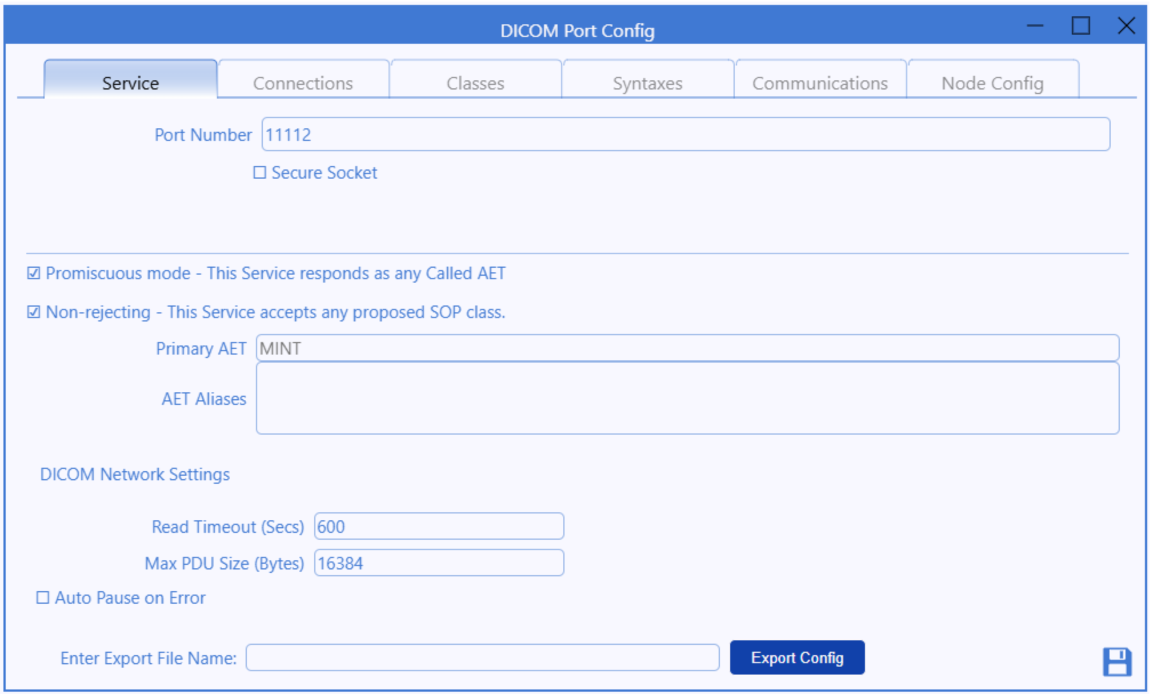


Figure 19: Example network using the DICOM Input Node.

The listener port can be configured via the configuration dialogue shown in Figure 20. Ensure the chosen port is not already in use on the server, as this would result in a binding error, preventing the node from operating correctly. Optionally, the listener may be configured to operate over SSL by enabling the *secure socket* option.



Port Number 11112
☐ Secure Socket

☑ Promiscuous mode - This Service responds as any Called AET
☑ Non-rejecting - This Service accepts any proposed SOP class.
Primary AET MINT
AET Aliases

DICOM Network Settings
Read Timeout (Secs) 600
Max PDU Size (Bytes) 16384
☐ Auto Pause on Error

Enter Export File Name:

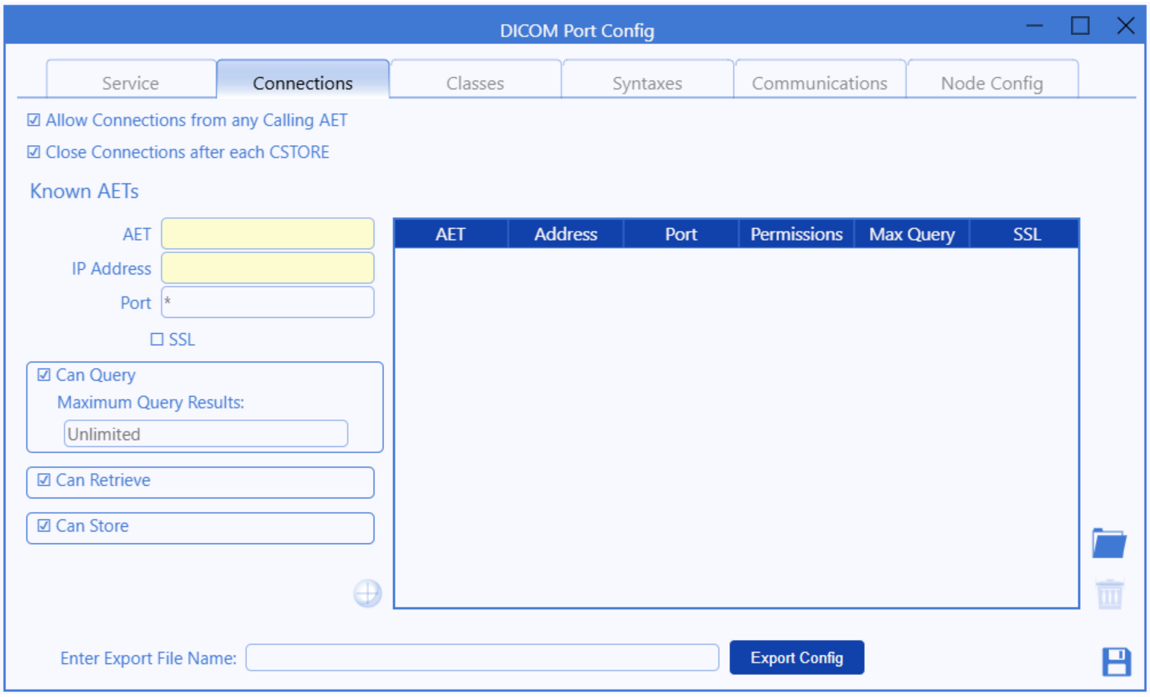
Figure 20: DICOM Input Node service configuration.

Enabling **promiscuous mode** allows the input node to accept connections from any external entity, regardless of the calling AET (Application Entity Title). By default, the input node expects incoming connections to match its primary AET or one of its configured aliases.

The **non-rejecting** option prevents the system from rejecting unknown SOP Classes or Transfer Syntaxes proposed during an association. For example, a private SOP Class not explicitly configured may still be accepted. While this increases flexibility, it may result in data being forwarded in a format that cannot be handled by downstream nodes, and further filtering may be required.

A **timeout** value can be defined to specify how long the system should wait for incoming data on a connection. The **maximum PDU size** setting determines the maximum size of data "chunks" transmitted across the connection.

The **Connections** tab enables the configuration of permitted DICOM peers. If the *Allow All Connections* option is enabled, any remote AET can communicate with the input node without being listed explicitly. Otherwise, each remote peer must be added manually with a title and IP address (or hostname). Specific permissions—such as *query*, *retrieve*, and *store*—can be granted per entity, as shown in Figure 21.



☑ Allow Connections from any Calling AET
☑ Close Connections after each CSTORE

Known AETs

AET
IP Address
Port *
☐ SSL
☑ Can Query
Maximum Query Results: Unlimited
☑ Can Retrieve
☑ Can Store

AET	Address	Port	Permissions	Max Query	SSL

Enter Export File Name:

Figure 21: DICOM Input Node connection configuration.

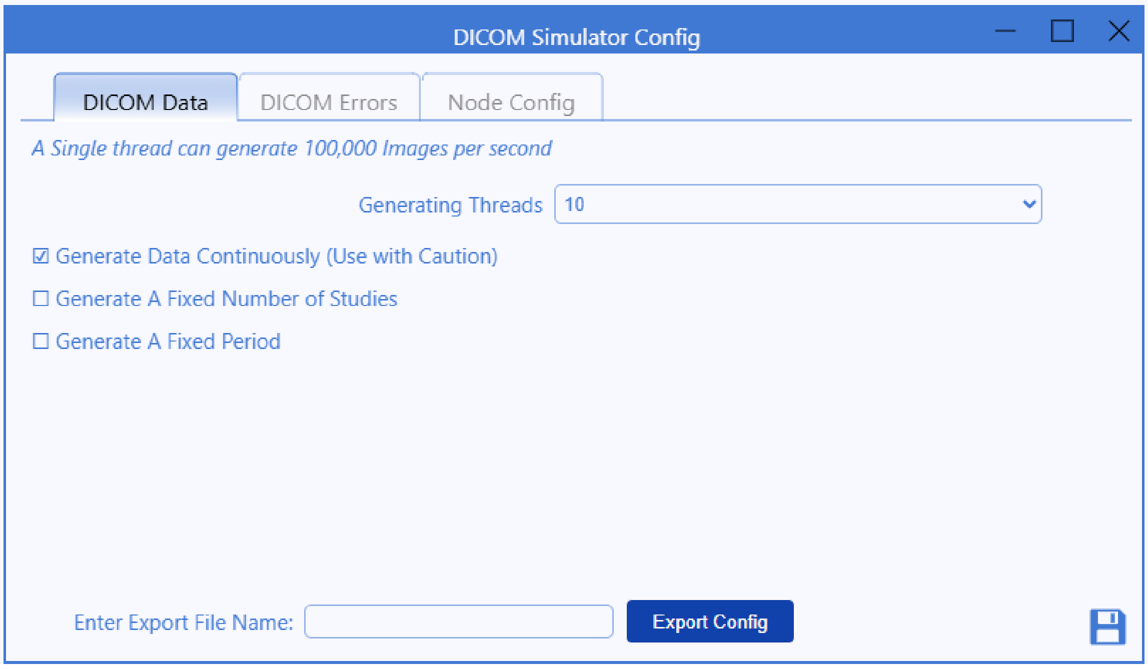
To perform a **CMOVE** operation from a DICOM Input Node, the destination AET must be present in the connections table. This allows the node to resolve the destination's IP address and port.

Additional configuration tabs allow control over the SOP Classes and Transfer Syntaxes accepted by the node. By default, only the *Implicit VR Little Endian* transfer syntax is enabled.

5.3.3 DICOM Simulator Node

A DICOM simulator node generates a DICOM input stream to simulate input from a range of modalities.

The **DICOM Simulator Node** is designed for testing and benchmarking purposes. It can be configured to create fictitious DICOM studies at a specified rate, enabling the network to be tested without requiring real inbound DICOM communications.



A Single thread can generate 100,000 Images per second

Generating Threads: 10

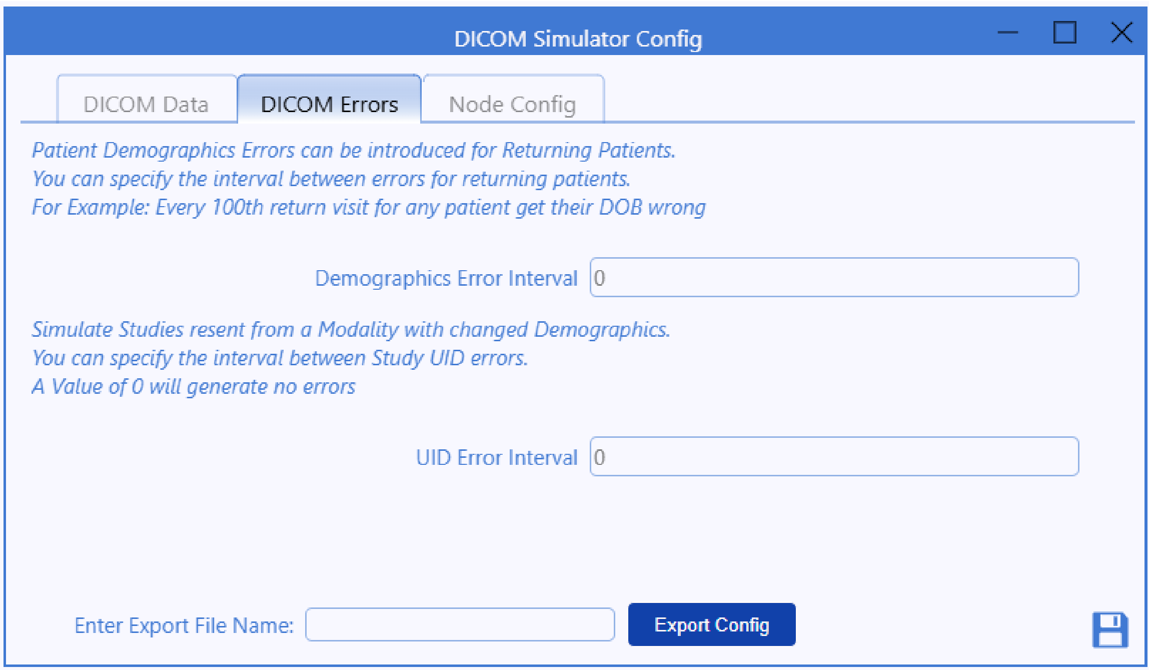
☑ Generate Data Continuously (Use with Caution)
☐ Generate A Fixed Number of Studies
☐ Generate A Fixed Period

Enter Export File Name:

Figure 22: Configuration dialogue for the DICOM Simulator.

Care should be taken when using the DICOM simulator, as it can generate large volumes of studies and images in short periods of time. Limits can be set on the number of threads used to create new entries.

If the **Generate Data** option is selected, the node will produce dummy modality data and forward it to the output node. An upper limit on the number of generated studies can be specified by enabling the option to generate a fixed number. Alternatively, the simulator can be configured to run for a defined period.


DICOM Simulator Config

DICOM Data | DICOM Errors | Node Config

Patient Demographics Errors can be introduced for Returning Patients.
You can specify the interval between errors for returning patients.
For Example: Every 100th return visit for any patient get their DOB wrong

Demographics Error Interval: 0

Simulate Studies resent from a Modality with changed Demographics.
You can specify the interval between Study UID errors.
A Value of 0 will generate no errors

UID Error Interval: 0

Enter Export File Name:

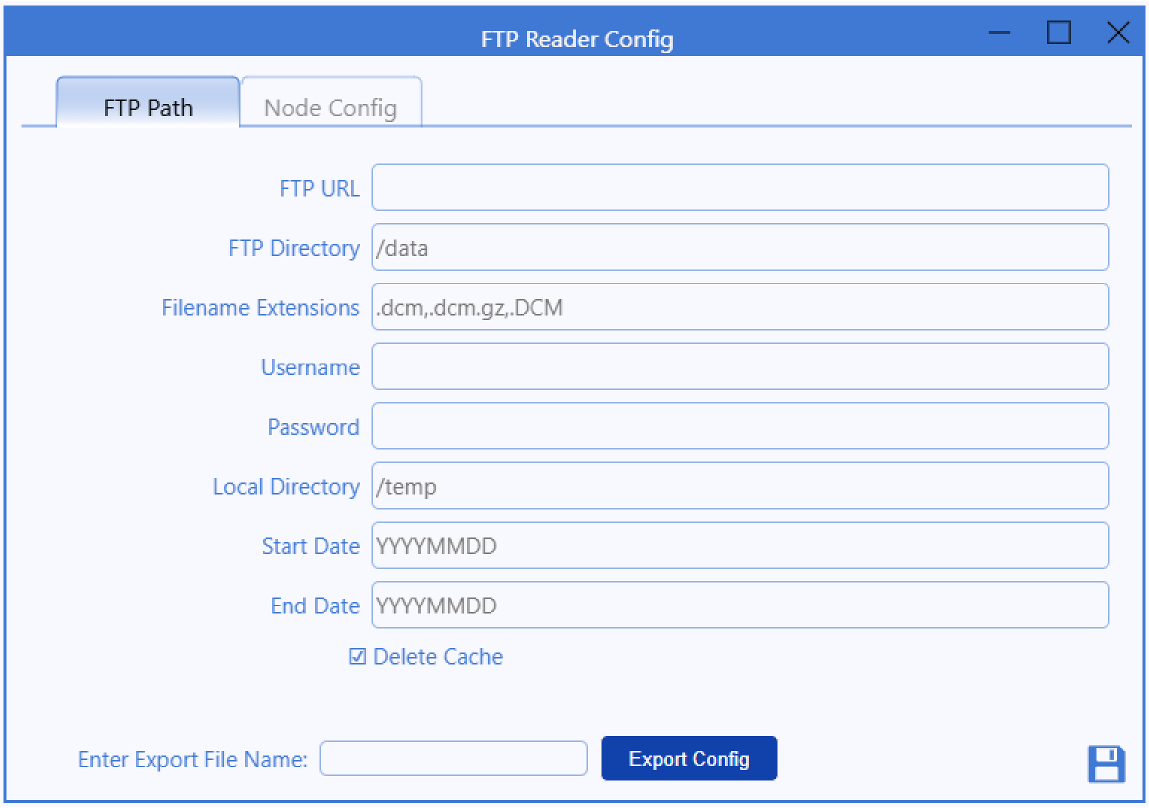
Figure 23: Configuration dialogue for error generation in the DICOM Simulator.

To support more robust testing, the simulator node can also introduce artificial **conflicts** into the generated data stream. These may include inconsistencies in patient demographic details or invalid Study UIDs. By default, the simulator is configured not to introduce any errors.

5.3.4 FTP Input Node

An FTP node reads data from an SFTP input stream and passes it to an output stream.

The **FTP Input Node** is used in scenarios where data resides on a remote storage system without DICOM access. Files are transferred securely via SFTP to a local cache, from which they are read by a file input node and sent onwards through the pipeline. Optionally, the copied files may be deleted from the local cache once processing is complete.



FTP URL:
FTP Directory: /data
Filename Extensions: .dcm,.dcm.gz,.DCM
Username:
Password:
Local Directory: /temp
Start Date: YYYYMMDD
End Date: YYYYMMDD
☑ Delete Cache
Enter Export File Name:

Figure 24: Configuration dialogue for the FTP File Reader node.

Figure 24 displays the available configuration options for the FTP file reader. The **URL** refers to the hostname or IP address of the remote SFTP server. The **FTP directory** defines the location on that server from which files will be retrieved.

File extensions can be specified to filter which files are eligible for copying. Only files matching the specified extensions will be processed. The SFTP **username** and **password** must be entered in the appropriate fields to authenticate the connection.

The **local directory** defines the temporary cache location on the host system, where files are copied before being processed and forwarded to an output node.

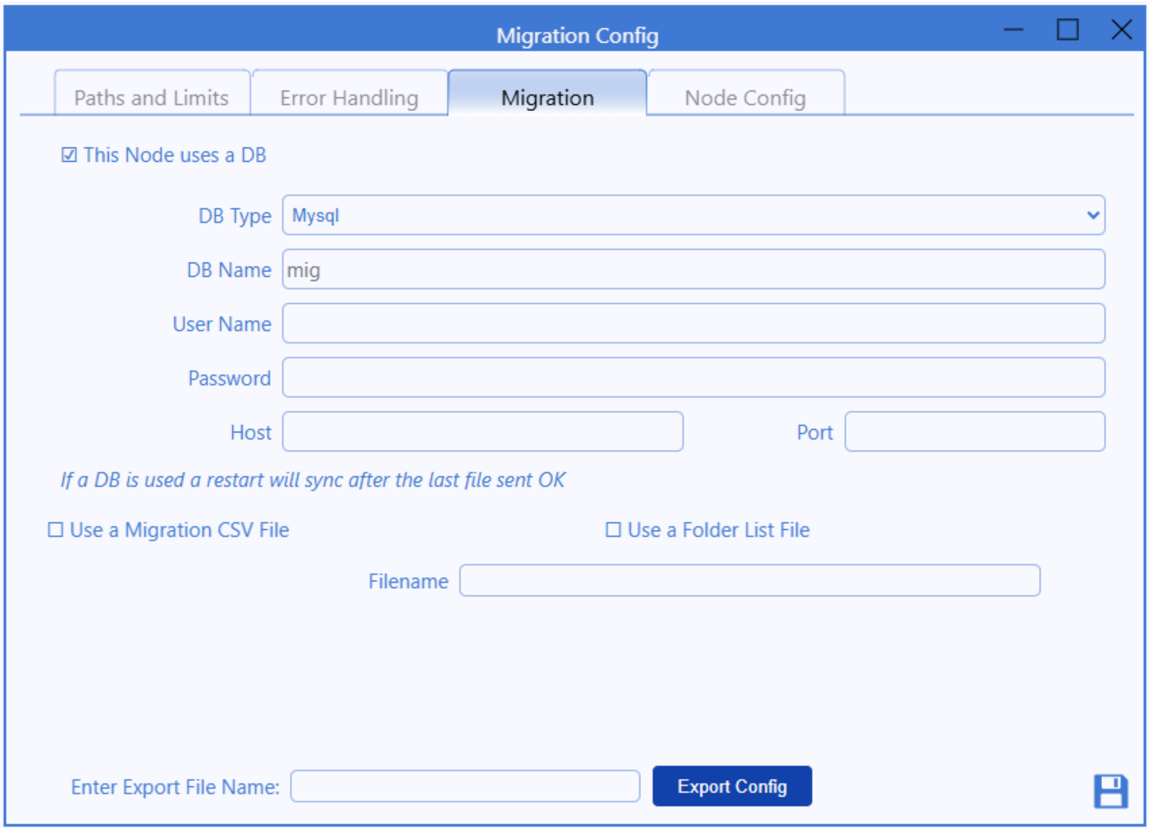
The **start date** and **end date** settings limit which files are copied, based on their file system timestamps. These dates do not correspond to DICOM study dates and should follow the DICOM date format: `YYYYMMDD`.

If the **delete cache** option is enabled, the node will automatically clear the local cache once processing of a batch is complete.

5.3.5 DICOM File Migration Node

A DICOM file migration node, based on the file input node, with additional configuration for exporting data to a new system.

The **DICOM File Migration Node** extends the functionality of the standard file input node. It is capable of tracking previously processed files using a configurable database (see Section 5.3.1) and can also be driven using a CSV file containing demographic data and file paths.


Migration Config

Paths and Limits | Error Handling | Migration | Node Config

☑ This Node uses a DB

DB Type: Mysql

DB Name: mig

User Name:

Password:

Host: Port:

If a DB is used a restart will sync after the last file sent OK

☐ Use a Migration CSV File ☐ Use a Folder List File

Filename:

Enter Export File Name:

Figure 25: Configuration tab for database and CSV file options in the DICOM File Migration Node.

Figure 25 shows the migration configuration tab, which allows you to define both a database for recording processed files and a CSV file to control the migration process. Both features must be enabled via their corresponding checkboxes.

When using a CSV file to drive migration, a specific column format is required. Each row in the CSV must contain exactly **nine columns**, corresponding to the following fields:

```
    File Path (used to locate the file, not to update metadata)
    Study UID
    Series UID
    SOP Instance UID
    Patient ID
    Patient Name
    Patient Date of Birth
    Patient Sex
    Accession Number
```

These values will be used to replace the corresponding DICOM tags in the file header.

If the option **Use a Folder List File** is selected, the system will read folder paths from a plain text file—each line representing one folder. Rather than scanning all folders from a root directory, the system will delegate folder-level processing to worker threads using this list.

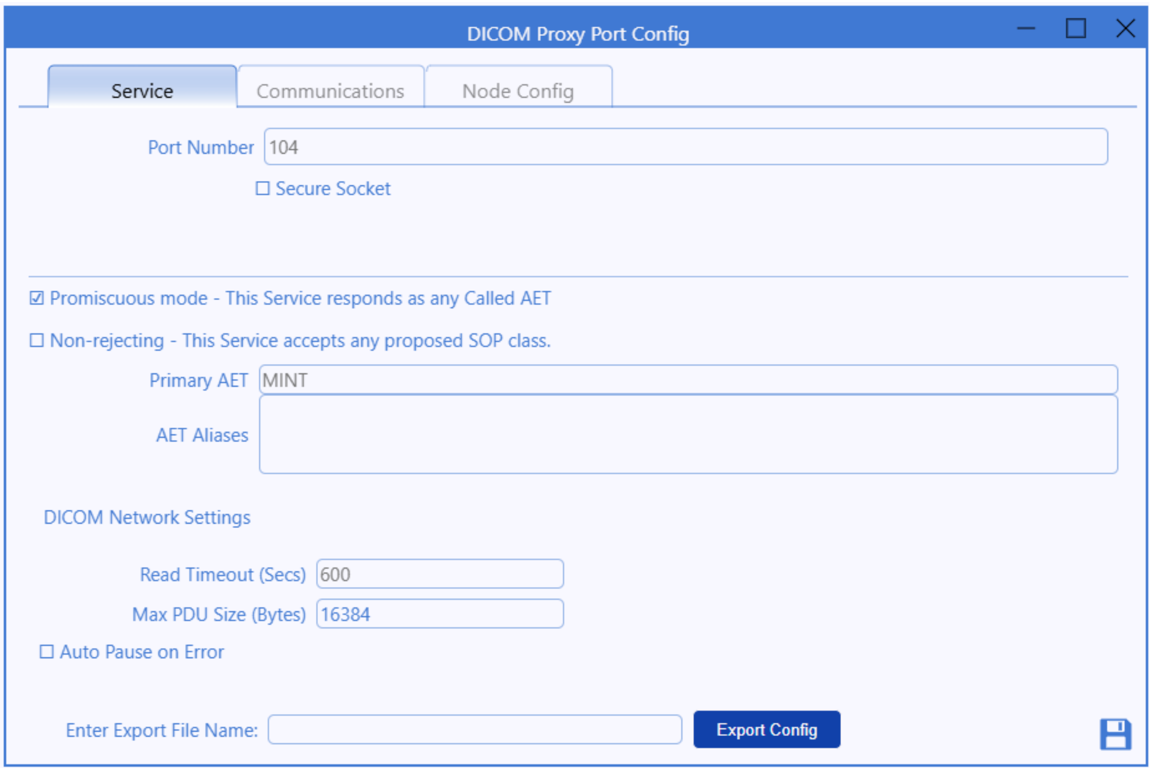
5.3.6 DICOM Proxy Input Node

A DICOM input node that redirects CFIND and CMOVE requests. Used exclusively with a Multi DICOM Output Node.

The **DICOM Proxy Input Node** is derived from the standard DICOM Input Node, but differs in that it is a **passive input node**. It does not process or handle DICOM file content directly. Instead, it redirects `CFIND` and `CMOVE` requests to a connected **Multi DICOM Output Node**.

For `CFIND` operations, the node forwards the request to all configured output nodes and aggregates the responses into a single result, which is then returned to the original requester.

For `CMOVE` operations, the request is passed to the output node for handling. The proxy node does not manage the retrieval process or receive the data being moved; instead, it assumes the target destination AET will handle the incoming data directly. The proxy node then compiles a combined `CMOVE` response from the results received from the output nodes.



Port Number 104
☐ Secure Socket

☑ Promiscuous mode - This Service responds as any Called AET
☐ Non-rejecting - This Service accepts any proposed SOP class.
Primary AET MINT
AET Aliases

DICOM Network Settings
Read Timeout (Secs) 600
Max PDU Size (Bytes) 16384
☐ Auto Pause on Error

Enter Export File Name:

Figure 26: Configuration dialogue for the DICOM CFIND and CMOVE proxy node.

The configuration for the DICOM Proxy Node is a simplified version of that used by the standard DICOM Input Node. Only the following parameters are required:

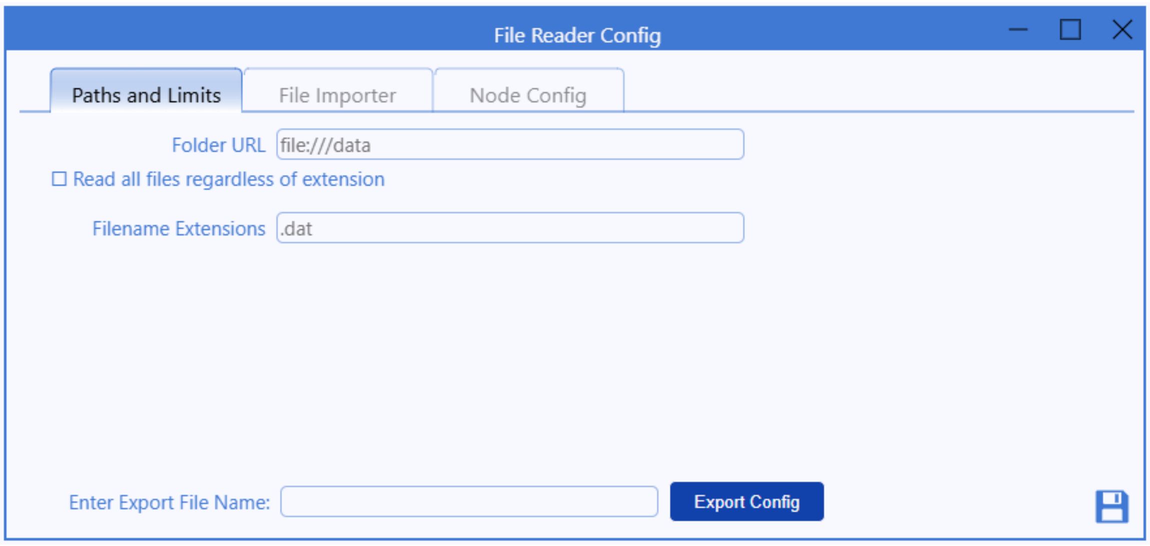
- **Port** — the network port to listen on.
- **AET** — the Application Entity Title expected by remote services.
- **Timeout** — the read timeout for inbound connections.
- **Max PDU Size** — the maximum size of transmitted data chunks.

There is no configuration for DICOM peers or SOP Class/Transfer Syntax support, as the proxy node does not handle store associations. All DICOM data transfers occur directly between the destination AET and the receivers specified by the output node.

5.3.7 HL7 File Input Node

An HL7 input node used to read HL7 messages saved as files on a file system and send them through a processing pipeline.

The **HL7 File Input Node** is designed for sending a set of previously cached HL7 messages to another system. This is particularly useful in scenarios such as forwarding stored patient registration or update messages to synchronise demographics with another system.



Folder URL: file:///data

☐ Read all files regardless of extension

Filename Extensions: .dat

Enter Export File Name:

Figure 27: HL7 file reader configuration options.

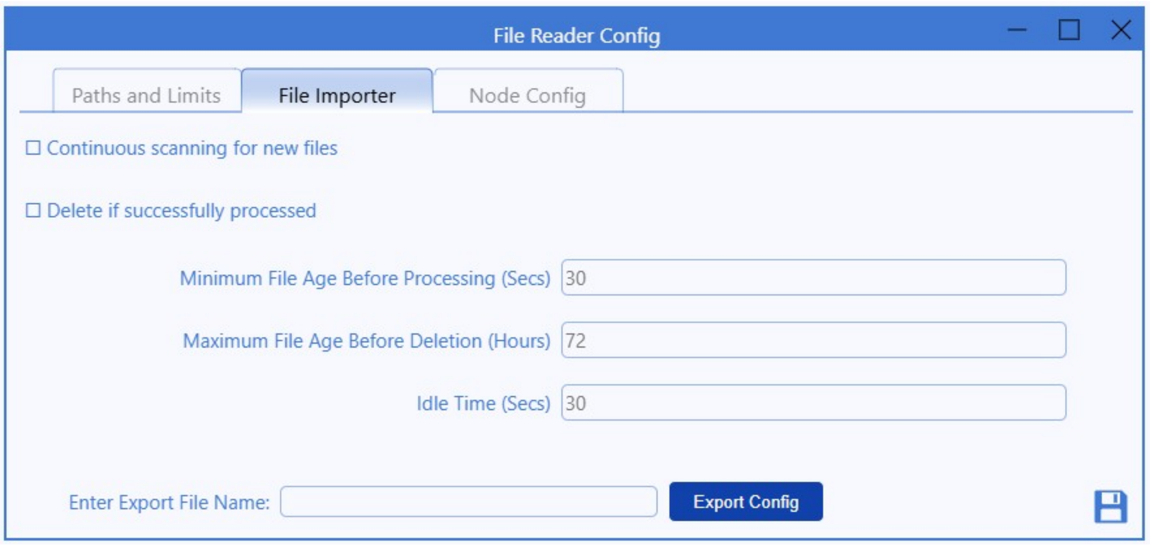
The **folder URL** defines the location of the HL7 message files. This path can be specified either with or without the `file://` prefix. For example:

```
    file://D:/data/messages
```

or

```
    D:/data/messages
```

By default, the node searches for files with a `.dat` extension. However, you may specify a custom list of extensions using a comma-separated format. The node will recursively search the directory structure for files matching the specified extensions and attempt to parse each as an HL7 message, passing them into the pipeline for further processing.



☐ Continuous scanning for new files

☐ Delete if successfully processed

Minimum File Age Before Processing (Secs): 30

Maximum File Age Before Deletion (Hours): 72

Idle Time (Secs): 30

Enter Export File Name:

Figure 28: HL7 file reader continuous mode configuration options.

As shown in Figure 28, the HL7 file reader can also be configured to operate in **continuous mode**, where it monitors the specified location for new messages and optionally deletes them after processing.

If an **ACK** (acknowledgement) is received, the corresponding message file in the source folder may be deleted automatically.

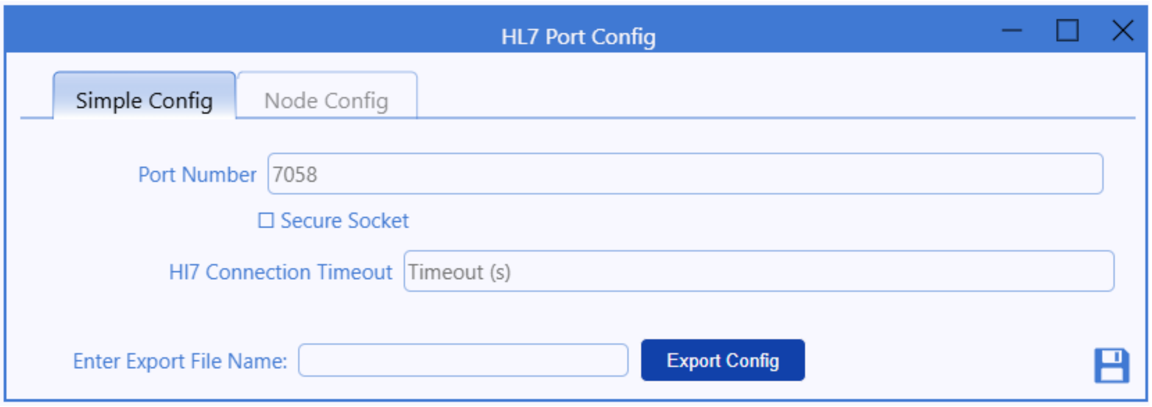
Several timing options are available for managing file handling:

- **Minimum file age** — The minimum number of seconds a file must exist before it is processed. This prevents processing of files that may still be in the process of being written. The default is 30 seconds but may be reduced safely for HL7.
- **Maximum file age before deletion** — If a file cannot be processed successfully within this period, it is removed from the processing queue. For robust handling of such cases, it is generally better to use an HL7 caching output node, which includes its own error management.
- **Idle time** — The interval (in seconds) between processing cycles, i.e., the delay between checking for newly arrived messages.

5.3.8 HL7 Stream Input Node

An HL7 input node that listens on a specified port for incoming HL7 messages and forwards them to a processing pipeline.

The **HL7 Stream Input Node** functions as a service listener, initialised on a user-defined port to handle inbound HL7 messages. The port is configurable via the node's configuration dialogue, shown in Figure 29. The selected port must be available on the host system—if it is already in use, a port binding error will occur and the node will not function correctly. Optionally, the listener can be configured to operate over SSL by enabling the *secure socket* option.


HL7 Port Config

Port Number: 7058

☐ Secure Socket

Hl7 Connection Timeout: Timeout (s)

Enter Export File Name:

Figure 29: HL7 Stream Input Node configuration options.

Incoming messages are forwarded to the output node linked to the listener. After processing, the HL7 Stream Input Node sends a reply to the originating system. The content of the reply depends on the outcome of the pipeline processing. By default, the listener accepts all HL7 message types.

A **timeout** can be configured to specify how long the system should wait to read data from a connection. By default, this timeout is set to 600 seconds. However, due to the underlying socket implementation, an initial read timeout of 20 seconds applies to verify that HL7 data is being transmitted. If no data is received during this time, the connection is closed to prevent clients from establishing idle, persistent connections, which may block other systems from connecting.

Some HL7 systems open a connection in advance and attempt to keep it permanently open. This can lead to excessive timeout warnings in the logs. Ideally, client systems should be configured to connect only when data is ready to be sent, and disconnect immediately after transmission. If this is not possible, the socket's read timeout can be adjusted by setting the following property in the deployment configuration file:

```
inet_timeout=7200
```

This property should be added to the `deployment.properties` file located in the Nexus installation directory.

5.3.9 JPG File Input Node

A JPG File Input Node reads JPG files from a specified location on disk and wraps them in a simple DICOM structure.

The **JPG File Input Node** creates a basic DICOM file using metadata derived from the image file's context. The name of the parent folder is used as the **patient name**, and a unique **patient ID** is generated for the file. All images within the same folder will be assigned the same patient ID.

By default:

- The **date of birth** is estimated as approximately 30 years before the file's last modified date.
- The **study date and time** are set to match the last modified timestamp of the file.
- The **study and series descriptions** are labelled as `JPEG Import`.
- The **image type** is set as `DERIVED\SECONDARY`.

The pixel data from the JPG file is embedded directly as the image data in the generated DICOM file.

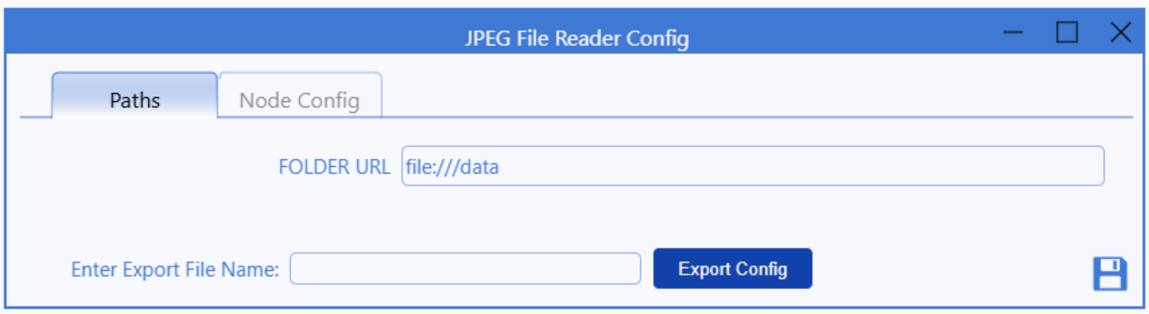


Figure 30: Configuration dialogue for defining JPG file import paths.

Figure 30 shows the configuration dialogue used to define the file paths for JPG import. Multiple import paths may be added by separating them with commas. The node will search for files with the following extensions: `.jpg`, `.jpeg`, and `.JPG`.

5.3.10 PNG File Input Node

A PNG File Input Node reads PNG files from a specified location on disk and wraps them in a simple DICOM structure.

The **PNG File Input Node** creates a basic DICOM file using metadata derived from the PNG file's context. It operates similarly to the JPG File Input Node but sets the **study** and **series descriptions** to `PNG Import`.

Like its JPG counterpart, the node uses the parent folder name as the **patient name**. All images in the same folder share a generated **patient ID**. The configuration interface is nearly identical, as shown in Figure 31. The node searches for files with the extensions `.png` and `.PNG`.

Multiple import locations can be specified using a comma-separated list of paths.

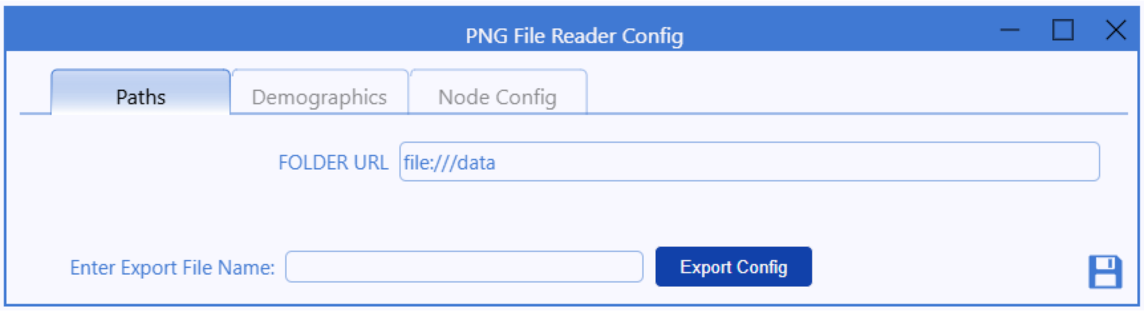
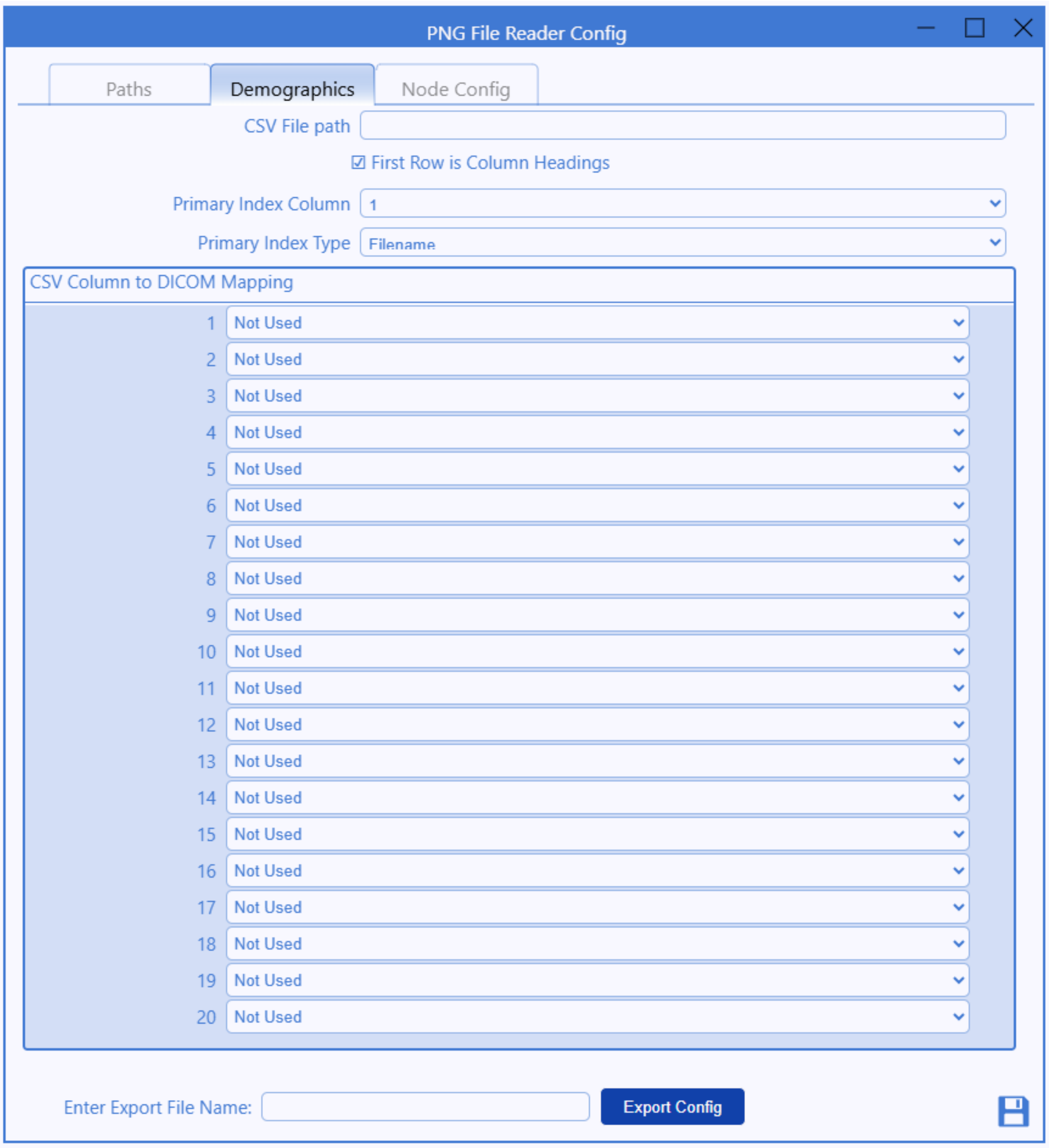


Figure 31: Configuration dialogue for defining PNG file import paths.


PNG File Reader Config

Paths | Demographics | Node Config

CSV File path:

☑ First Row is Column Headings

Primary Index Column: 1

Primary Index Type: Filename

CSV Column to DICOM Mapping

1	Not Used
2	Not Used
3	Not Used
4	Not Used
5	Not Used
6	Not Used
7	Not Used
8	Not Used
9	Not Used
10	Not Used
11	Not Used
12	Not Used
13	Not Used
14	Not Used
15	Not Used
16	Not Used
17	Not Used
18	Not Used
19	Not Used
20	Not Used

Enter Export File Name:

Export Config

Figure 32: CSV file mapping to DICOM tag configuration.

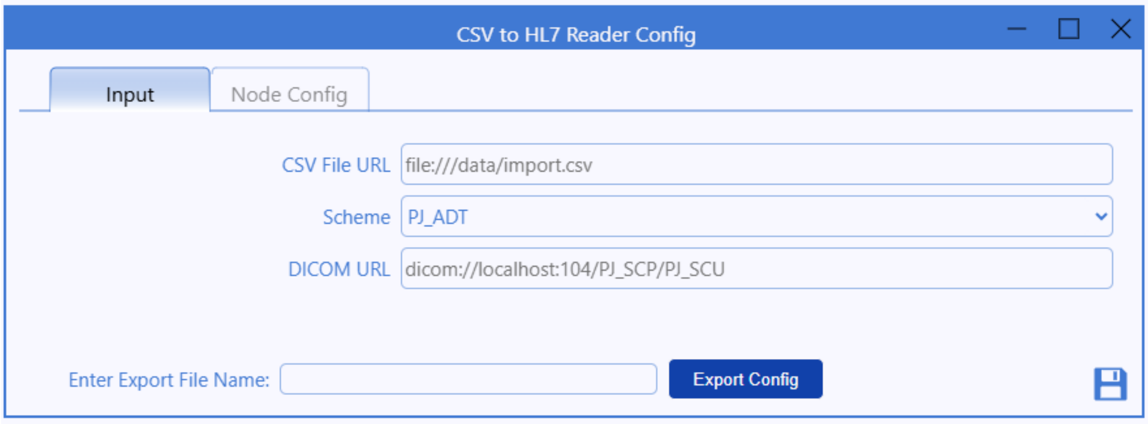
Figure 32 illustrates how a CSV info file can be used to populate DICOM tags with metadata. In the example shown, the **primary index** is the file path to the PNG image. Other fields, such as **Patient ID**, are mapped to specific CSV columns and applied to the generated DICOM wrapper.

A checkbox option is available to indicate whether the CSV file contains a header row. If enabled, the first row will be ignored during processing.

5.3.11 CSV File Input Node

A CSV to HL7 input node that converts entries in a CSV file into HL7 message files.

The **CSV File Reader Node** is used to generate HL7 messages—such as `ADT` or `ORM`—from structured entries in a CSV file. Figure 33 shows the configuration dialogue used to define the CSV file path and the expected column schema.


CSV to HL7 Reader Config

CSV File URL: file:///data/import.csv
Scheme: PJ_ADT
DICOM URL: dicom://localhost:104/PJ_SCP/PJ_SCU
Enter Export File Name:

Figure 33: Configuration dialogue for the CSV to HL7 file reader node.

The CSV input must conform to a specific schema. For **ADT messages**, the default expected mapping is shown below:

Field	**CSV Column**
Surname	8
Forename	7
Sex	10
Date of Birth	9
Patient ID 1	2
Patient ID Issuer 1	1
Patient ID 2	5
Patient ID Issuer 2	4

These fields are used to construct the `PID` segment of an ADT `A04` message.

The node can also generate **ORM messages** using a simplified schema, commonly referred to as the "ABC" format:

- Accession Number
- Date of Birth

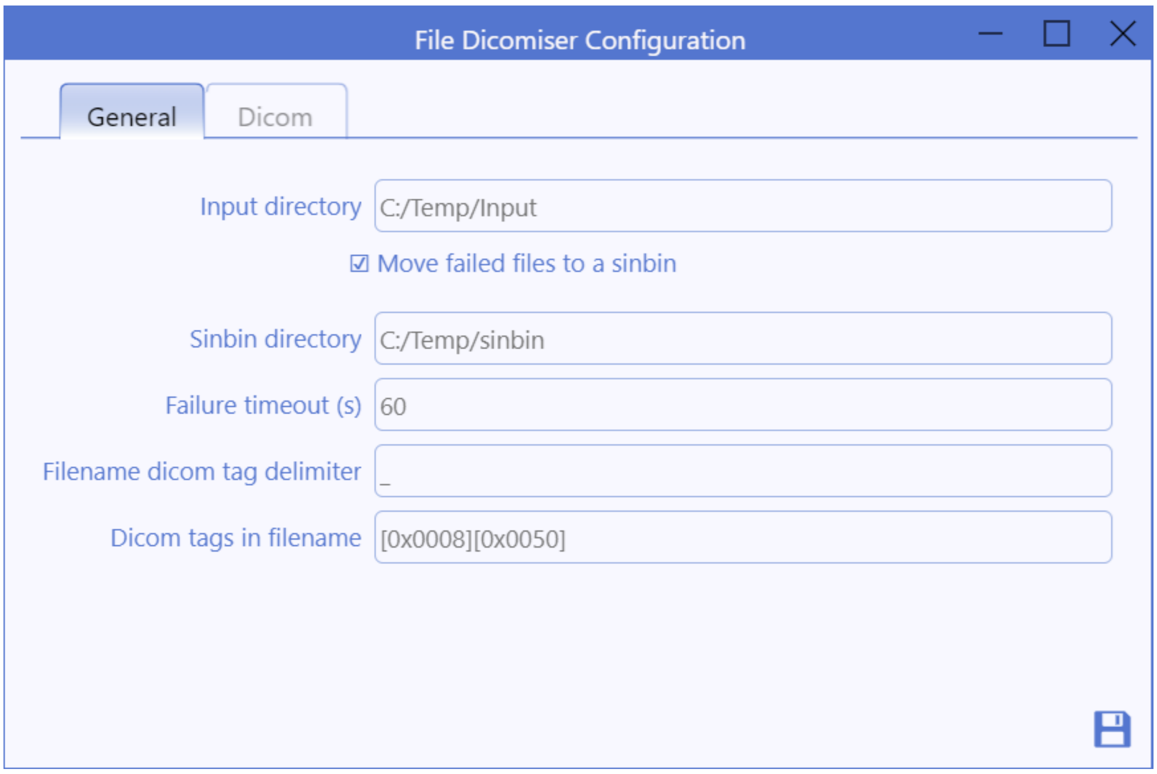
- Patient ID (Care Record Number)

When processing ORM messages, the node will perform a DICOM query to the defined URL using the values from the CSV file. The results of this query are then used to populate the full set of patient, study, and order information within the HL7 message.

5.3.12 Smart File Dicomiser

Adds an image file source to the network as an input node. Queries a DICOM source for study details based on the filename and streams them into the pipeline.

The **Smart File Dicomiser Node** is used to convert JPEG or PDF files into DICOM objects. The *smart* functionality refers to its ability to query a DICOM source for study metadata using information extracted from the filename. For example, if the filename represents an accession number, the node can query a DICOM source for matching studies and merge the results with the image. The resulting DICOM file can then be passed on through the pipeline with the appropriate study details.



Input directory	C:/Temp/Input
☑ Move failed files to a sinbin	
Sinbin directory	C:/Temp/sinbin
Failure timeout (s)	60
Filename dicom tag delimiter	_
Dicom tags in filename	[0x0008][0x0050]

Figure 34: General configuration options for the Smart File Dicomiser.

Figures 34 and 35 show the configuration options. The **Input directory** defines the folder where image files are read from. Once successfully converted and transmitted, the original image file is deleted. If a file fails to transfer, it may be retained in a specified **sin bin** directory. This must be enabled via a checkbox and a valid path provided.

File transfer failures are often transient. A configurable **timeout** value allows the system to retry before classifying the failure as final. Files are only moved to the sin bin (or deleted, if disabled) if they are older than the current time minus the timeout.

The filename is expected to contain values corresponding to DICOM tags used for the query. There is no fixed limit on the number or type of tags that may be queried, but only one matching study must be returned. These study details are merged with the image to create the output DICOM object. Multiple DICOM tags in a filename should be separated by a delimiter—by default, this is an underscore (_). This delimiter can be customised.

The **DICOM tags in filename** field should be a comma-separated list of DICOM tag identifiers (e.g., `[0x0010][0x0020]` for Patient ID). For example, to query both Patient ID and Accession Number, enter:

```
[0x0010][0x0020],[0x0008][0x0050]
```

Then for Patient ID `PJ-X12345` and Accession Number `PJ-X9876`, use the filename:

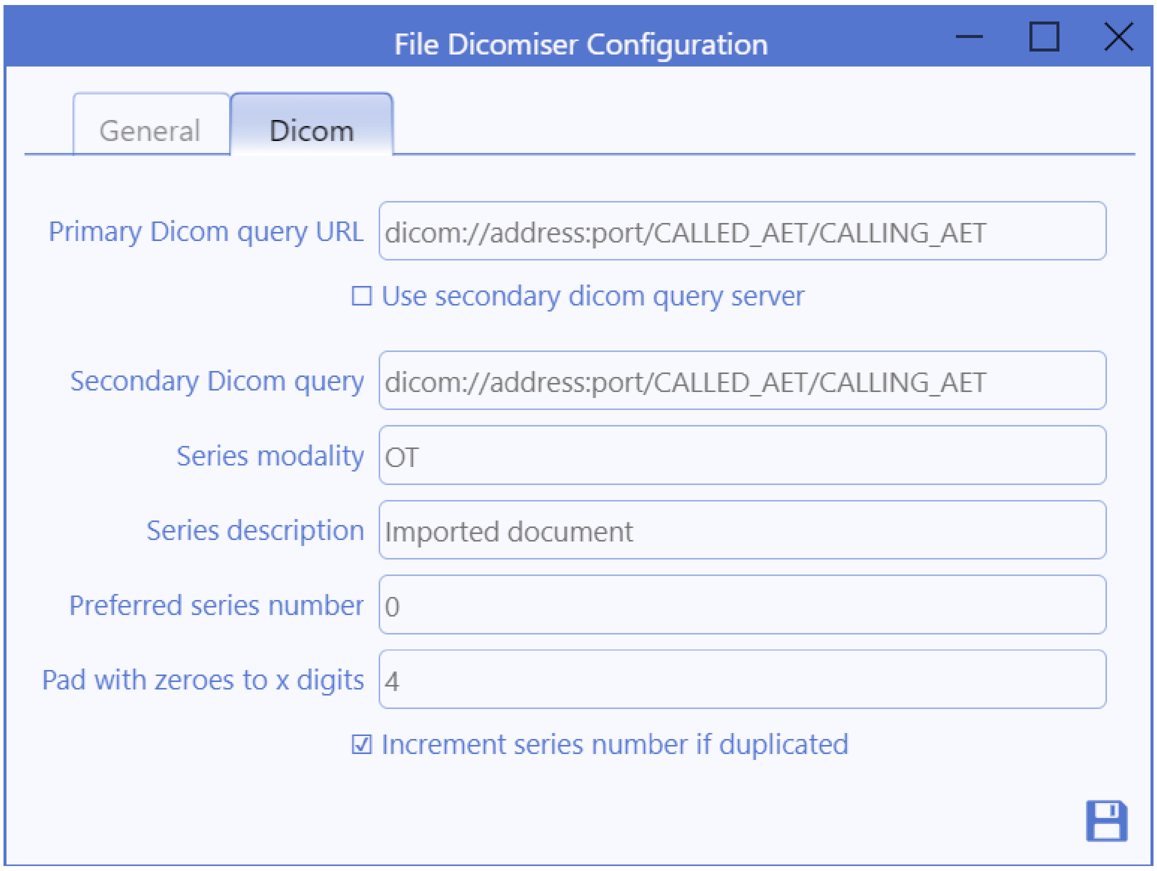
```
PJ-X12345_PJ-X9876.jpg
```

Anything appearing in the filename after the final tag is ignored. Thus, filenames such as:

```
PJ-X12345_PJ-X9876_001.jpg
PJ-X12345_PJ-X9876_002.jpg
```

are also valid.

If only one tag is defined (e.g., Patient ID), then a filename like `PJ-X12345.jpg` would only query that tag.



Primary Dicom query URL: dicom://address:port/CALLED_AET/CALLING_AET

☐ Use secondary dicom query server

Secondary Dicom query: dicom://address:port/CALLED_AET/CALLING_AET

Series modality: OT

Series description: Imported document

Preferred series number: 0

Pad with zeroes to x digits: 4

☑ Increment series number if duplicated

Figure 35: DICOM configuration options for the Smart File Dicomiser.

The DICOM configuration tab (Figure 35) allows specification of a **primary query node** and an optional **secondary query node**. The system first queries the primary source. If no matches (or multiple matches) are returned, it then queries the secondary source.

Additional parameters include:

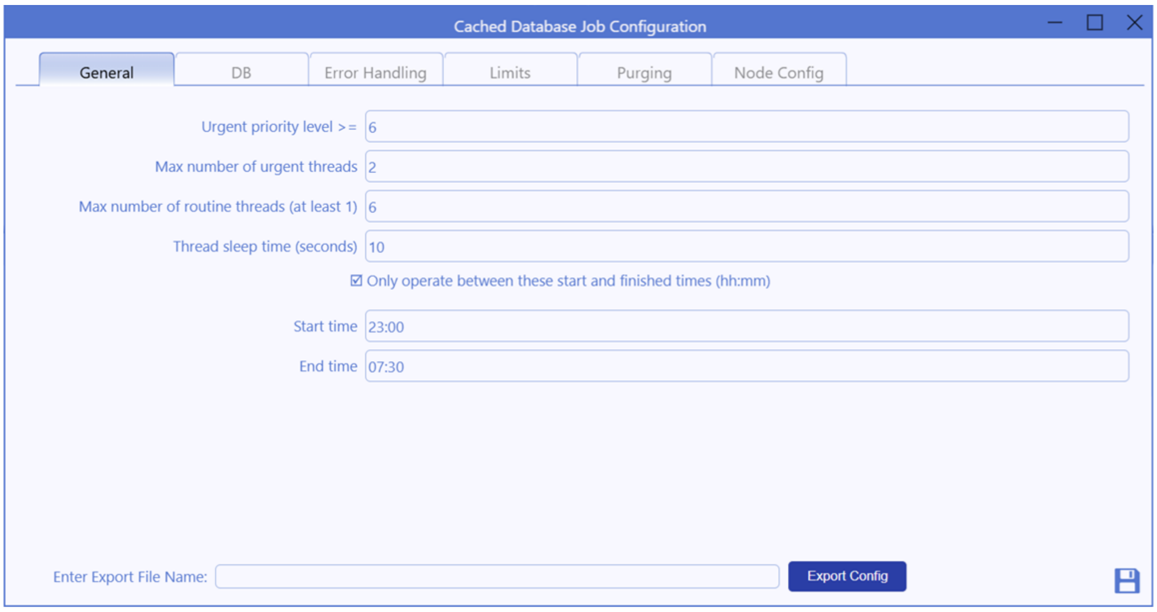
- **Preferred modality** (e.g., `OT` by default; `SC` is another common option)
- **Series description**
- **Series number** — specified as an integer
- **Series number padding** — defines leading zeroes (e.g., a padding of 3 renders `5` as `005`)

To ensure uniqueness of series numbers, enable the option **Increment series number if duplicated**. This causes the system to query both the primary and secondary nodes (if configured) to check for existing series numbers. The number is incremented until a unique value is found, which is then used in the resulting DICOM metadata before being sent downstream.

5.3.13 Database Job Forwarder

Reads DICOM files from forwarding jobs in a database.

The Database Job Forwarder reads from a database to collect DICOM files and then send them along the pipeline. Jobs are created by the *Forwarding Node* (see section 5.5.14) and saved to a configurable database. In the *General* configuration (see figure 36), the user is able to set the level that is considered *urgent* and can adjust how many concurrent workers will forward urgent jobs or non-urgent jobs (the default is four of each). In the unlikely event a user sets zero concurrent urgent workers, urgent jobs are processed by the routine workers. A user is not able to set zero routine workers. An advanced user may which to adjust the sleeptime between finishing the current batch of DICOM studies to forward and collecting the next 200 jobs. Users may want to only forward DICOM files during off-peak times and there is the option to do so in the configuration; simply tick the box and then set the start and finish times of the forwarding service (in hh:mm or hh:mm:ss format).

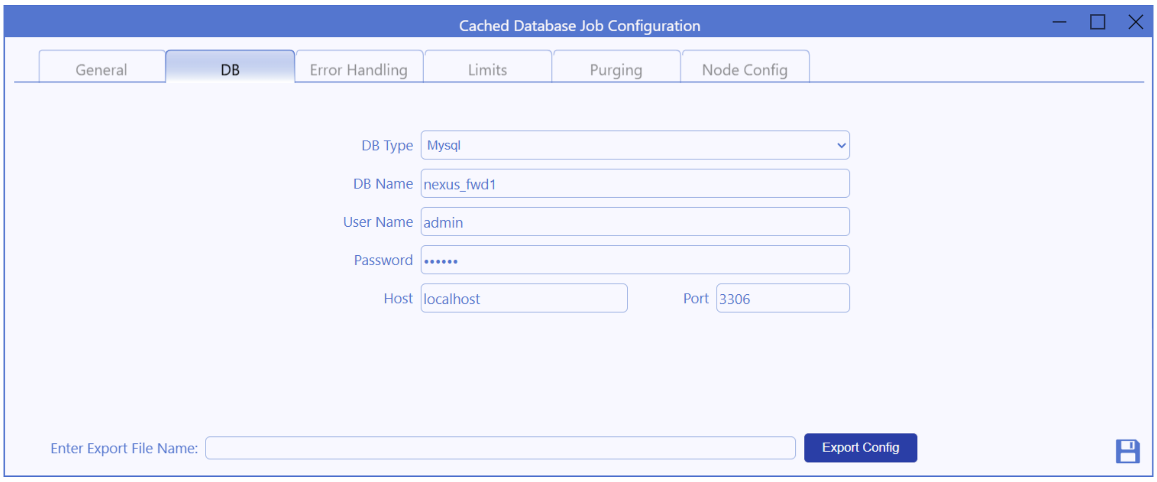


Urgent priority level >= 6
Max number of urgent threads 2
Max number of routine threads (at least 1) 6
Thread sleep time (seconds) 10
☑ Only operate between these start and finished times (hh:mm)
Start time 23:00
End time 07:30
Enter Export File Name:

Figure 36: Database Job Forwarder general configuration.

Database Job Forwarders have exclusive access to read their own database to prevent race conditions for different pipelines reading the same data. The database configuration is shown in figure 37, where users can set the database type, the database name, username and password for access, host, and IP port. If a user configures multiple Database Job Forwarder nodes to read the same database, the Nexus network will fail to start correctly.

Most users do not want to indefinately store the data that is being forwarded and the Database Job Forwarder provides configurable puring options (figure 38). The user may choose the purge the files from the file system, the database entries for the forwarding job, or both, by ticking the relevant boxes. A further option allows the user to choose whether or not to delete only items where forwarding was successful, or all items. The user must set the retention time of the data for

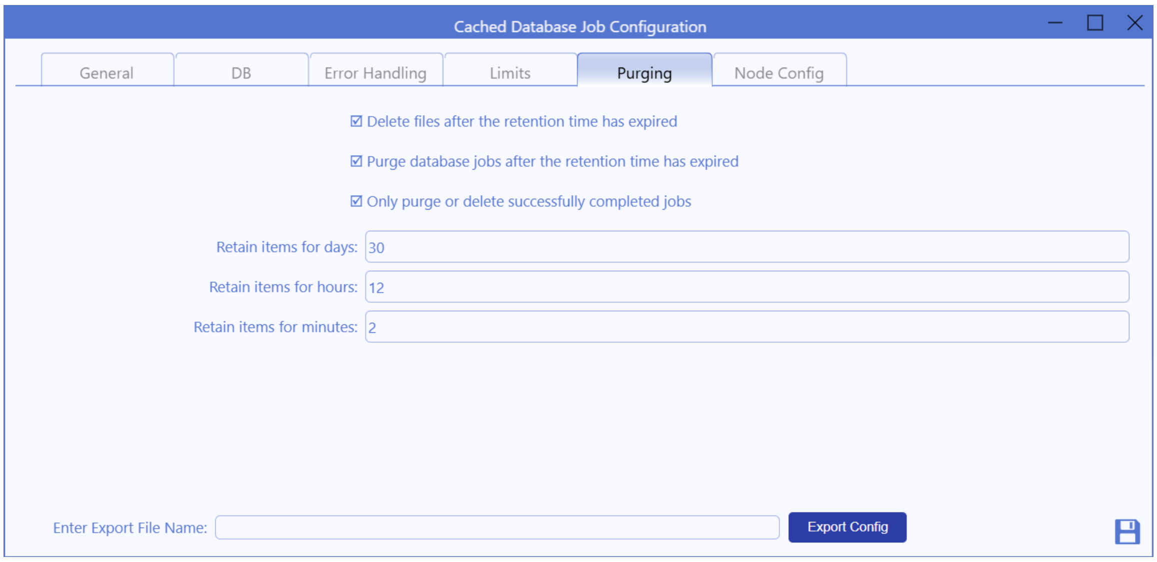


DB Type: Mysql
DB Name: nexus_fwd1
User Name: admin
Password: ••••••
Host: localhost
Port: 3306
Enter Export File Name:

Figure 37: Database Job Forwarder database configuration.

the purging service to activate/ The user is able to enter any numerical values into the day, hour and minute fields, which are added together to create the time the data is retained. Updating the retention time in the configuration will not affect the retention times of jobs already created.

Purge jobs are added to the running *Cleaner* service (see section 6.9). The cleaner service must therefore have access to the databases defined in the Database Job Forwarder configuration (which is the normal configuration).



☑ Delete files after the retention time has expired
☑ Purge database jobs after the retention time has expired
☑ Only purge or delete successfully completed jobs
Retain items for days: 30
Retain items for hours: 12
Retain items for minutes: 2
Enter Export File Name:

Figure 38: Database Job Forwarder purging configuration.

5.4 Filter Nodes

Filter nodes enable users to configure specific operations on the data stream as it passes through the network. For example, DICOM files can have their headers modified dynamically as data is streamed into, through, or out of the network.

Similar to input nodes, filter nodes feature a **configuration icon** located at the top left of the node within the pipeline view. Some filter nodes also include a **view icon** in the bottom left corner, which provides access to extended views or tools relevant to that node's functionality.

Certain filter nodes are capable of routing generated output via a **time-dependent channel**, which is visually represented as a secondary pipe extending from the bottom of the node. This functionality is activated by selecting the **clock icon**.

5.4.1 Stream Splitter Node

A stream splitter node allows the stream to be duplicated or routed along multiple paths in the streaming pipeline.

The **DICOM Stream Splitter Node** enables the data stream to be split into multiple paths, either duplicating or redirecting the data to different parts of the network. Figure 39 shows the default appearance of the splitter when first added to the network. Additional pipes can be created by clicking the **plus** icon in the bottom-left corner of the node. The default splitter type is `DICOM`, but it should be changed to `HL7` in the configuration if it is to process HL7 messages.

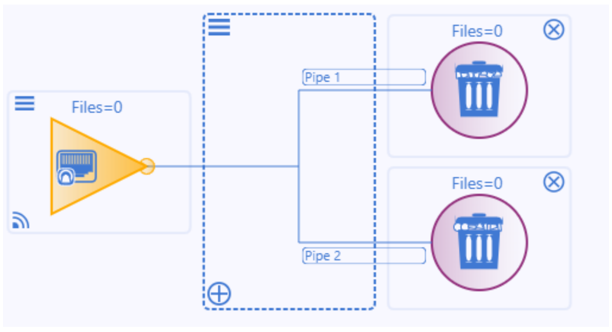
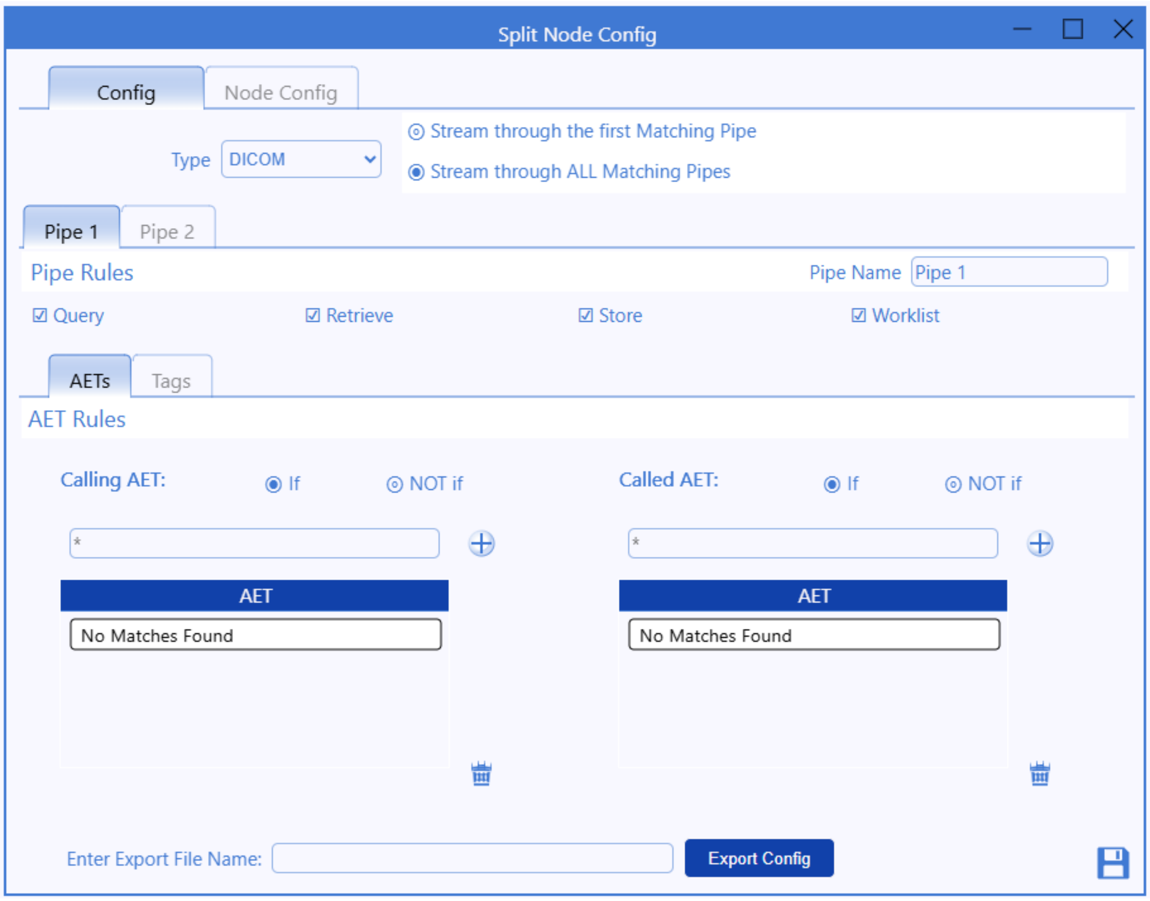


Figure 39: DICOM Stream Splitter pipeline controls (AETs).

DICOM stream splitting can be based on either Application Entity Titles (AETs) or specific DICOM tag values in the header. The mode is selected by using the appropriate radio button, as shown in Figure 40. Output pipelines can be named by editing the **Pipe Name** field.



Type: DICOM

Stream through the first Matching Pipe [✓ Stream through ALL Matching Pipes]

Pipe Name: Pipe 1

☑ Query ☑ Retrieve ☑ Store ☑ Worklist

Calling AET: [✓ If] NOT if

Called AET: [✓ If] NOT if

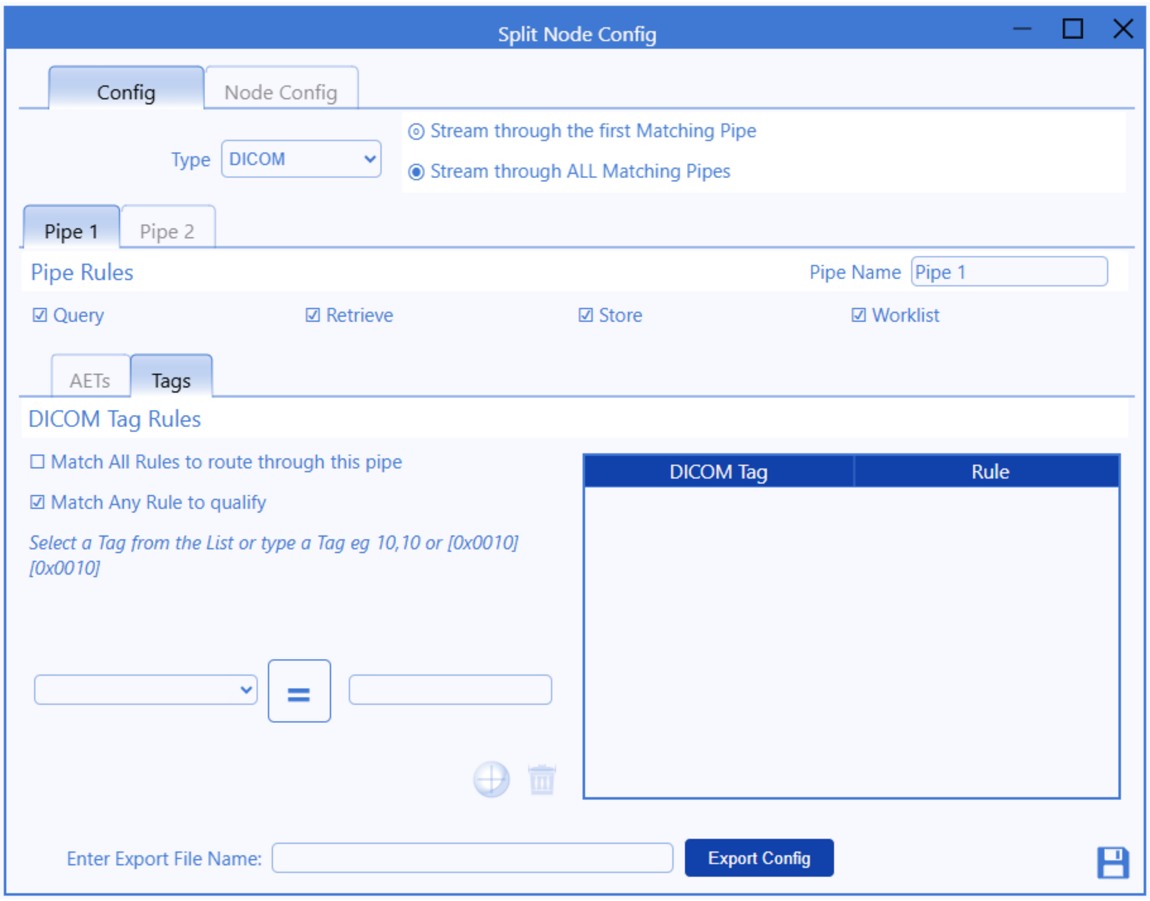
AET
No Matches Found

AET
No Matches Found

Enter Export File Name:

Figure 40: DICOM Stream Splitter configuration dialogue (AET-based routing).

Routing rules can be defined using the **AET Rules** section, enabling positive or negative matches based on both calling and called AETs.


Split Node Config

Config | Node Config

Type: DICOM

○ Stream through the first Matching Pipe

● Stream through ALL Matching Pipes

Pipe 1 | Pipe 2

Pipe Rules

Pipe Name: Pipe 1

☑ Query ☑ Retrieve ☑ Store ☑ Worklist

AETs | Tags

DICOM Tag Rules

☐ Match All Rules to route through this pipe

☑ Match Any Rule to qualify

Select a Tag from the List or type a Tag eg 10,10 or [0x0010] [0x0010]

DICOM Tag	Rule

Enter Export File Name:

Export Config

Figure 41: DICOM Stream Splitter configuration dialogue (tag-based routing).

Tag-based routing is configured by choosing a tag, selecting a match type from the dropdown, and entering the target value. The comparison operator can be cycled by clicking the icon between the dropdown and the input field. Supported match types include: `Equals`, `Not Equals`, `Wildcard Match`, `Not Wildcard Match`, `Greater Than`, `Greater Than or Equal`, `Less Than`, `Less Than or Equal`, `Between`, and `Not Between`.

The splitter can also route data based on DICOM operations, such as sending queries down one path and images down another. This is controlled via the **Query**/**Retrieve**/**Store**/**Worklist** checkboxes.

To support HL7 stream splitting, change the splitter **Type** to `HL7`. This switches the configuration interface to the HL7 routing view, shown in Figure 42. Pipes are labelled in the same way as for DICOM.

In HL7 mode, mapping rules may follow one of two logic types: `Match All Rules` (logical AND) or `Match Any Rule` (logical OR). Each rule references a location in the HL7 message by defining:

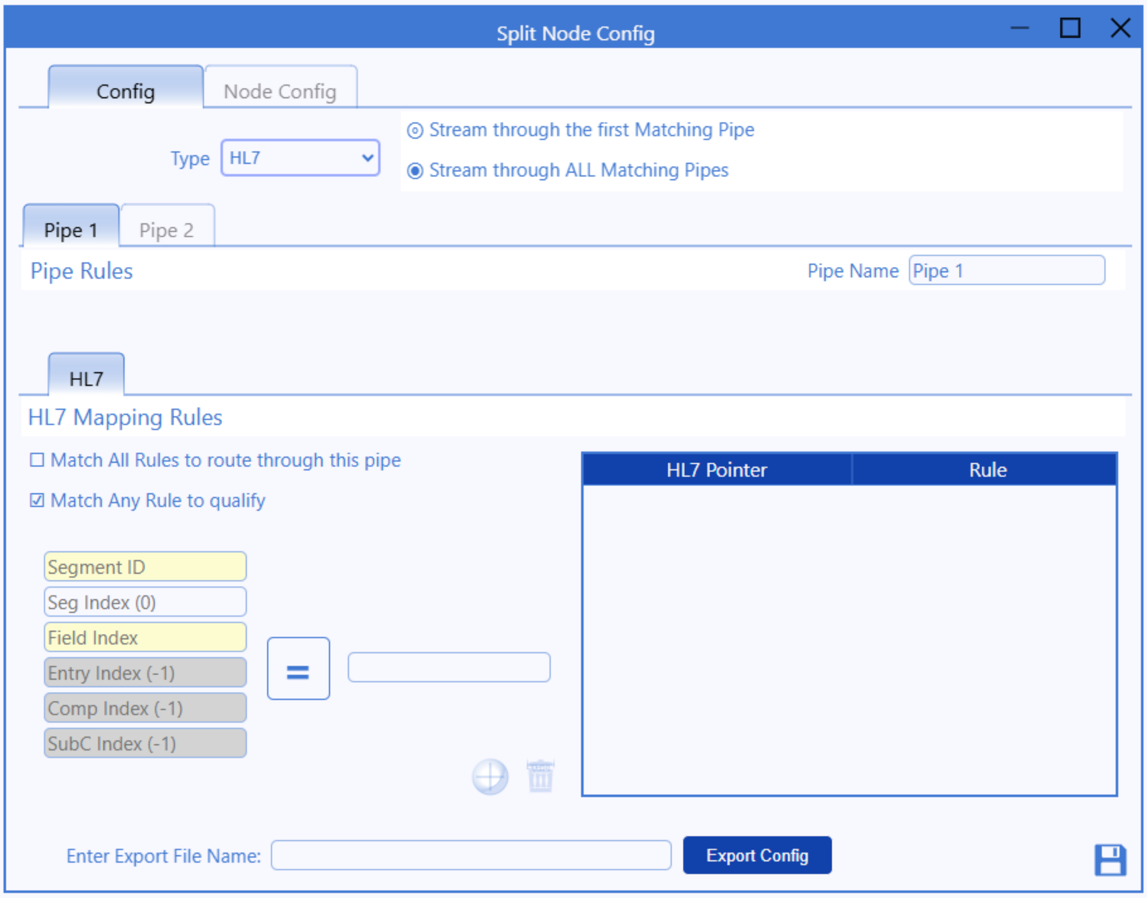
- **Segment ID** – e.g., `PID`
- **Segment Index** – index of the matching segment (zero-based)

- **Field Index** – index of the field within the segment (field 0 = segment ID)
- **Entry Index**, **Component Index**, **Sub-component Index** – used for nested field structures (all zero-based)

For example, to reference NHS number `6666666666` in the message shown in Listing 1, the rule would use: Segment ID = `PID`, Segment Index = `0`, Field Index = `3`, Entry Index = `1`, Component Index = `0`. To route based on the "NHS" identifier, use Component Index = `3`.

Listing 1: Example HL7 report message.

```
MSH|^~\&|WEBCLIENT|PACS|RIS.REPORTS|RIS|20141202140633.0||ORU^R01|
    EBRGyk.36763-EC2E6X.1758|P^I|2.4
PID||QP26^^^^PID|QP26^^^^PID~6666666666^^^NHS^NHSN||Bloggs^Joe
    ^^^||19810121|M|||,^^^^^||||||||||
...
```


Split Node Config

Config | Node Config

Type: HL7

( ) Stream through the first Matching Pipe
(●) Stream through ALL Matching Pipes

Pipe 1 | Pipe 2

Pipe Rules — Pipe Name: Pipe 1

HL7

HL7 Mapping Rules

☐ Match All Rules to route through this pipe
☑ Match Any Rule to qualify

Segment ID
Seg Index (0)
Field Index
Entry Index (-1)
Comp Index (-1)
SubC Index (-1)

HL7 Pointer	Rule

Enter Export File Name: — Export Config

Figure 42: HL7 Stream Splitter node configuration dialogue.

To prioritise rule evaluation, you can reorder output pipes by right-clicking on a pipe and choosing the appropriate menu option. Figure 43 shows the interface for moving an output pipe upwards. This affects the rule-matching precedence during routing.

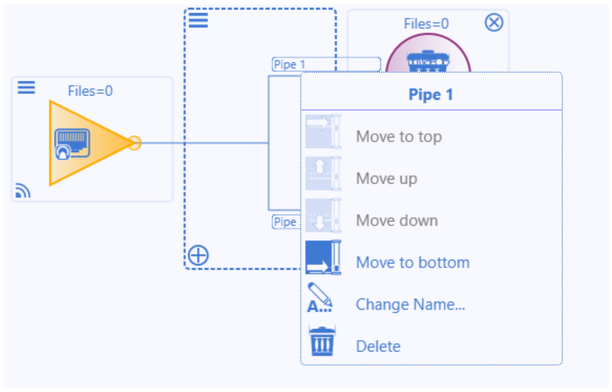


Figure 43: Reordering output pipes at the Stream Splitter.

5.4.2 DICOM Tag Mapping Node

A tag mapping node allows the manipulation of the DICOM metadata stream as it passes through the node.

The **Tag Mapping Node** enables on-the-fly modification of DICOM tags as data is streamed through the network. There are five modes of tag manipulation, each accessed via a dedicated tab in the configuration dialogue, as shown in Figure 44.

Set Tag

Allows a specified tag to be assigned either a fixed value or the value from another tag. The assignment can be made conditional on the calling or called AET, and on the current value of the tag. AET values may include wildcards (`*`). Tags can also be encoded as byte arrays using the `bytes:` prefix, which ensures proper binary encoding of structured values.

For example, to encode the frame increment pointer sequence:

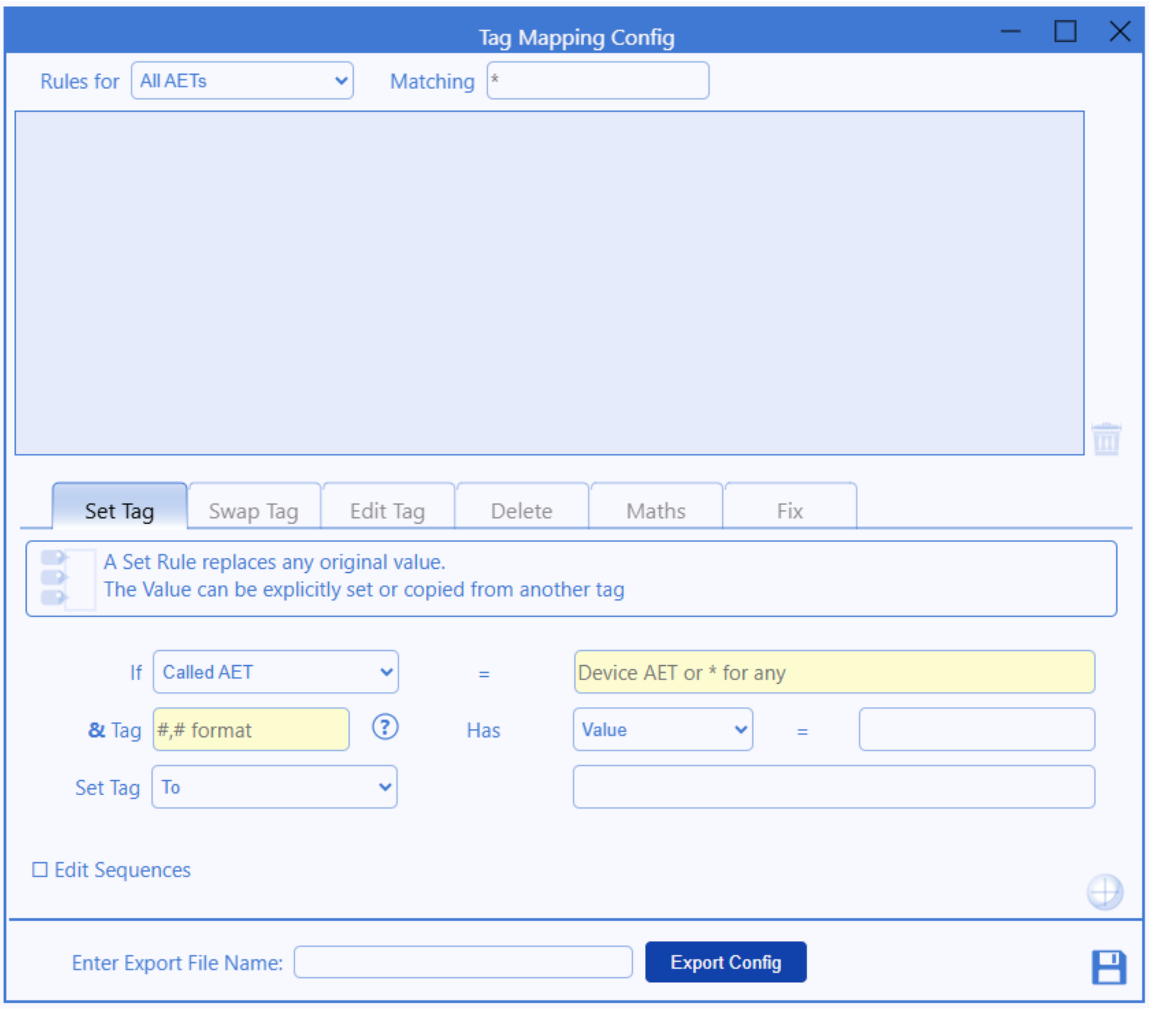
```
[0x0054][0x0010][0x0054][0x0020]
```

you would write:

```
bytes:0x0054/0x0010/0x0054/0x0020
```

which produces the little-endian byte array:
`[0x54, 0x0, 0x10, 0x0, 0x54, 0x0, 0x20, 0x0].`

If the **Update Sequences** option is enabled, matching tags within sequences will also be modified in addition to the top-level tag.


Tag Mapping Config

Rules for: All AETs
Matching: *

A Set Rule replaces any original value.
The Value can be explicitly set or copied from another tag

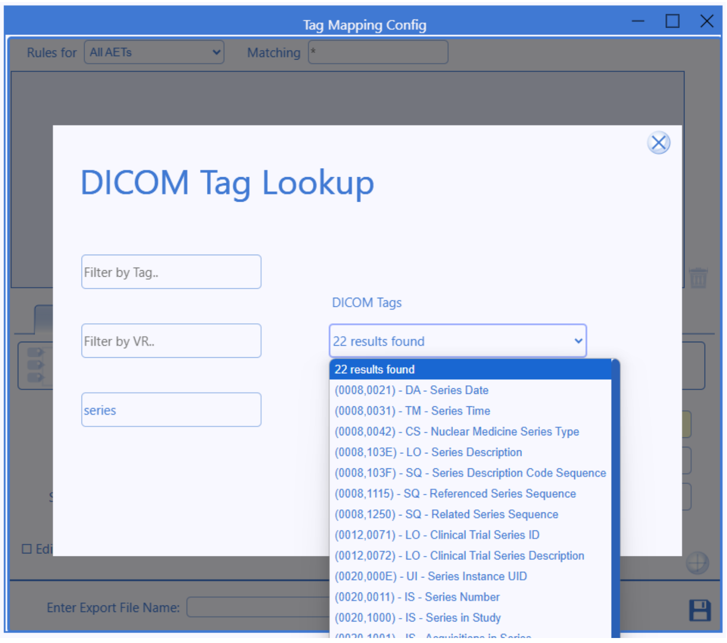
If: Called AET = Device AET or * for any
& Tag: #,# format Has: Value =
Set Tag: To

☐ Edit Sequences

Enter Export File Name:

Figure 44: The Tag Mapping `Set` configuration dialogue.

Tags can be selected using the DICOM tag lookup tool (Figure 45) to the right of the tag input field. This lookup allows tags to be searched by tag ID, value representation (VR), or name. For example, searching for "series" may return multiple matches. Select the required tag and click `Apply`.


Tag Mapping Config
Rules for
All AETs
Matching
DICOM Tag Lookup
DICOM Tags
Enter Export File Name:
Edi

Filter by Tag..

Filter by VR..

series

22 results found

22 results found
(0008,0021) - DA - Series Date
(0008,0031) - TM - Series Time
(0008,0042) - CS - Nuclear Medicine Series Type
(0008,103E) - LO - Series Description
(0008,103F) - SQ - Series Description Code Sequence
(0008,1115) - SQ - Referenced Series Sequence
(0008,1250) - SQ - Related Series Sequence
(0012,0071) - LO - Clinical Trial Series ID
(0012,0072) - LO - Clinical Trial Series Description
(0020,000E) - UI - Series Instance UID
(0020,0011) - IS - Series Number
(0020,1000) - IS - Series in Study

Figure 45: DICOM tag selection dialogue.

Swap Tag

Exchanges the values of two tags. Like the `Set Tag` mode, this can be made conditional on AET or the existing value of one of the tags.

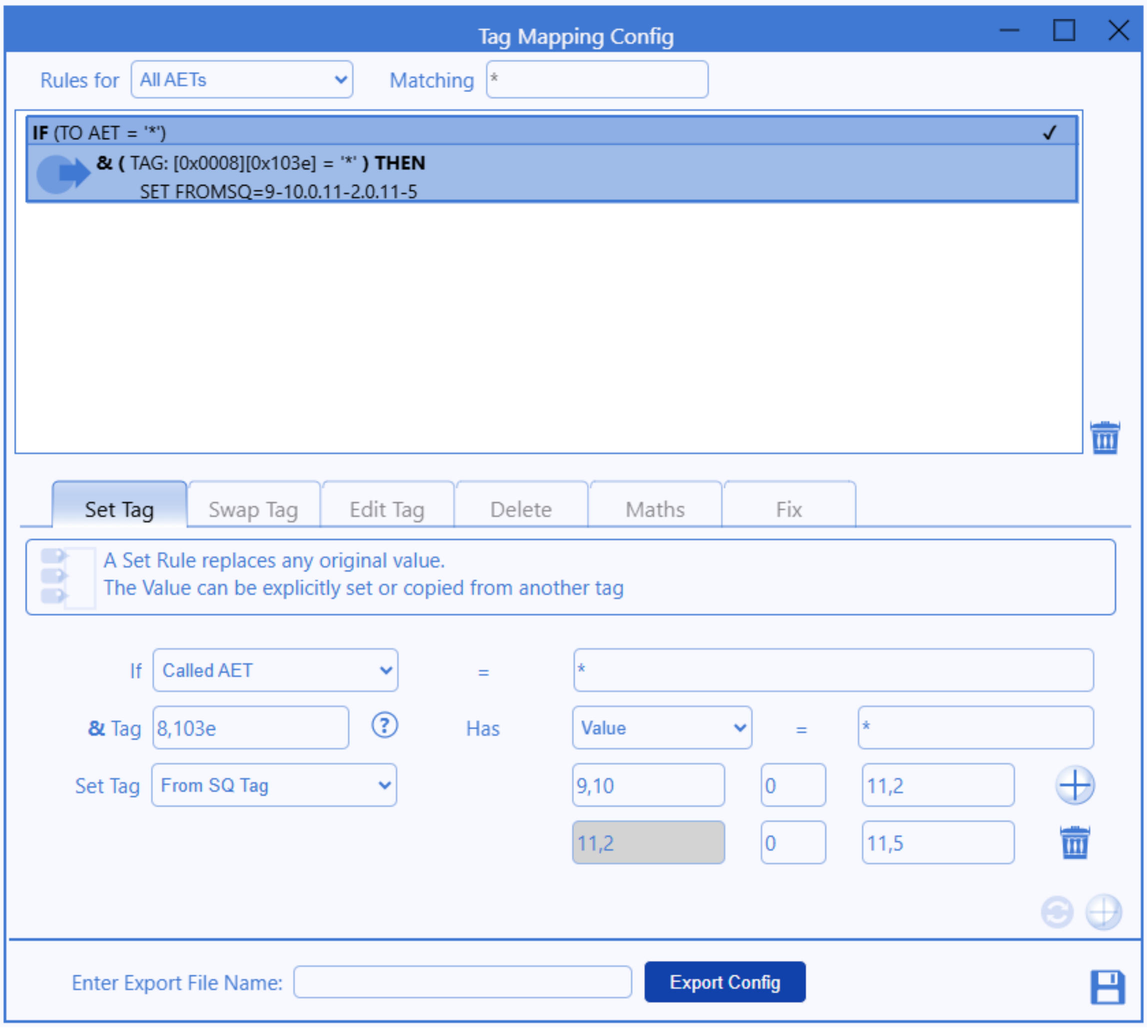
Edit Tag

Provides tools to modify tag values by prefixing, suffixing, or replacing parts of the value. For prefix and suffix edits, the adjacent toggle button allows switching between `Add` and `Remove`. Rules can be conditioned on AET and optionally the tag value using wildcards.

To remove a prefix or suffix, it must be present in the tag value. For replacing values, specify the search pattern (with optional wildcards) in the `Has Value` field and the replacement value in the corresponding `Replace` field.

Figure 46 illustrates an example where the Series Description tag `[0x0008][0x103e]` is populated using the value of tag `[0x0011][0x0005]` within the first item of a private sub-sequence `[0x0011][0x0002]`, itself nested inside `[0x0009][0x0010]`:

```
[0x0008][0x103e] =
[0x0009][0x0010]
---------------->[0x0011][0x0002]
---------------->---------------->[0x0011][0x0005]
```


Tag Mapping Config

Rules for: All AETs

Matching: *

IF (TO AET = '*')
& (TAG: [0x0008][0x103e] = '*') THEN
SET FROMSQ=9-10.0.11-2.0.11-5

A Set Rule replaces any original value.
The Value can be explicitly set or copied from another tag

If: Called AET = *

& Tag: 8,103e Has Value = *

Set Tag: From SQ Tag 9,10 0 11,2

11,2 0 11,5

Enter Export File Name:

Figure 46: Setting a top-level tag using a value from a nested sequence.

Delete Tag

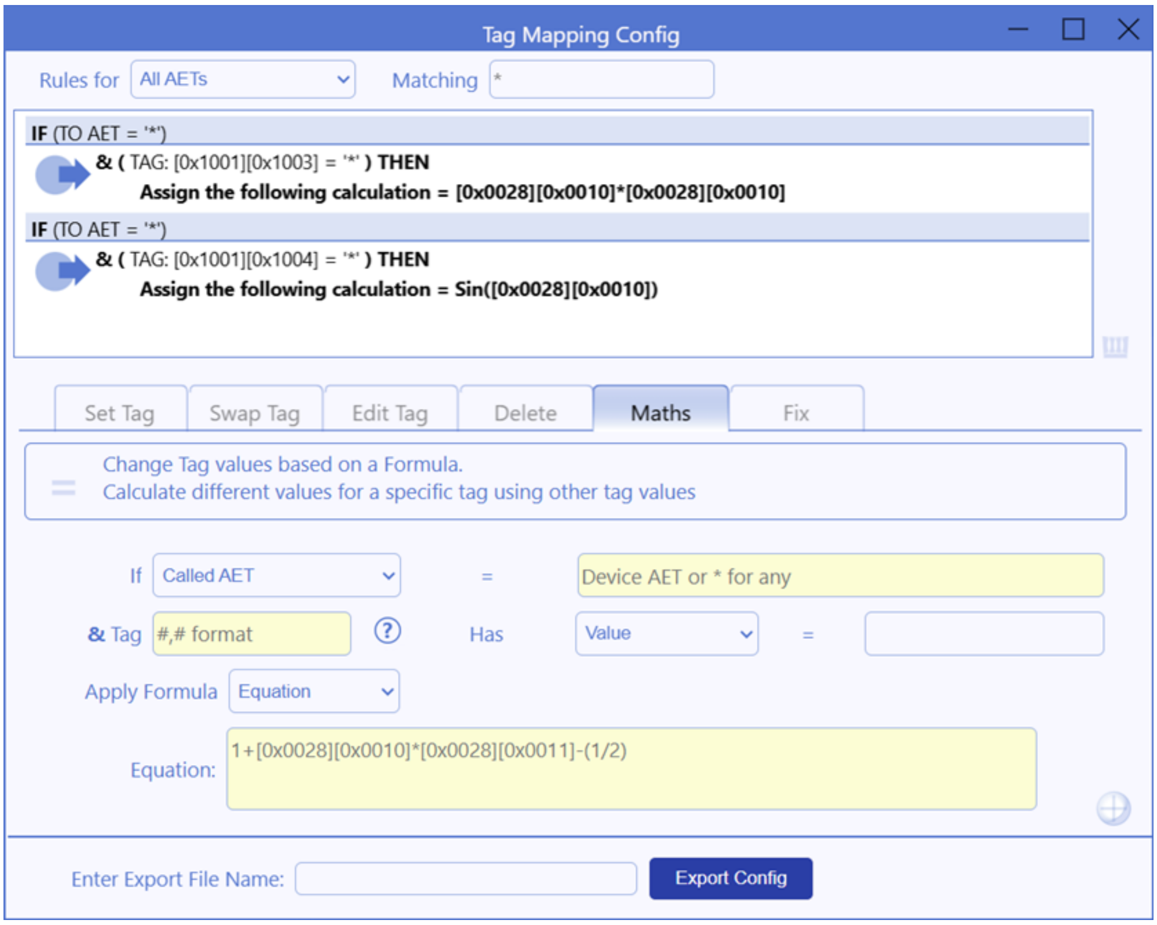
Enables removal of a tag or group of tags from the dataset. This operation can also be conditioned on the value of another tag or the AET.

Maths Tag

Applies numerical operations to tag values. The user can define their own equation or use a pre-defined expression. The `Apply Formula` drop-down menu has options for `Equation` and `Bed Height`.

The `Bed Height` option alters the y-component of tag `(0x0020, 0x0032)`—Patient Image Position—which may need to be adjusted to reconcile differences between modalities and display workstations.

The `Equation` option allows uses to enter their own equation, the result of which is assigned to the selected tag. Users may enter integer or floating point numbers, Dicom tags, standard mathematical operators, supported mathematical functions or constants. Figure 47 shows two configured equations assigned to tags `[0x1001][0x1003]` and `[0x1001][0x1003]`, the area at the bottom includes a hint for entering equations.


Tag Mapping Config

Rules for: All AETs
Matching: *

IF (TO AET = '*')
& (TAG: [0x1001][0x1003] = '*') THEN
Assign the following calculation = [0x0028][0x0010]*[0x0028][0x0010]
IF (TO AET = '*')
& (TAG: [0x1001][0x1004] = '*') THEN
Assign the following calculation = Sin([0x0028][0x0010])

Change Tag values based on a Formula.
Calculate different values for a specific tag using other tag values

If: Called AET = Device AET or * for any
& Tag: #,# format Has: Value =
Apply Formula: Equation
Equation: 1+[0x0028][0x0010]*[0x0028][0x0011]-(1/2)

Enter Export File Name:

Figure 47: Setting a top-level tag using a value from a nested sequence.

Supported Operators: standard supported operations are + (add), - (subtract), * (multiply), / (divide), ^ (exponent) and () (round brackets).

Supported Functions: $\sin x$, $\cos x$, $\tan x$, $\text{round}(x)$, $\ln x$ and $\log x$ (base 10). Trigonometric functions will interpret the input unit as radians. The `round` function rounds the input number to the nearest integer ($1.1 \rightarrow 1$ and $1.5 \rightarrow 2$).

Supported constants: For the users' convenience, two common constants have been made available, π and e (Euler's number). To access these, type `pi` or `(e)` in your equation. It is important to note the Euler's number *must* be surrounded by brackets to differentiate it from the use of an 'e' in scientific notation; `1e3` would return 1000 and very different to `1*(e)^3`, which returns approximately 20.

Dicom tags: Groups and tags *must* be entered in hexadecimal format surrounded by square brackets. For example, `[0x0028][0x0010]` would be the format required to extract the value of *Pixel Rows* in a calculation. The value in a tag must be either numerical or an operator, any other values would simply cause the calculation to fail.

The Dicom standard allows sequences to have repeating elements, so to extract a value from a sequence item, the user is required to enter the sequence tag, followed by a stop, the array index of the sequence (the first item has index zero and this is expected to be the index most commonly used), a further stop and then the next tag (and further indexes and tags if required). For instance, to get the value of the first occurrence of *X Ray Tube Current* within the *Exposure Dose Sequence*, use the following format:

```
[0x0040][0x030e].0.[0x0018][0x1151]
```

Or to get the value of the second occurrence of *X Ray Tube Current* within the same sequence, use:

```
[0x0040][0x030e].1.[0x0018][0x1151]
```

Some Dicom tags have a multiplicity greater than one and contain multiple values. The evaluator can treat these as an array much like when dealing with sequences. For example, *Pixel Spacing* has two values; to extract the row spacing use `[0x0028][0x0030].0` and to extract the column spacing use `[0x0028][0x0030].1`.

Notes on use: Calculations are performed in standard mathematical (BODMAS) order. Spaces are ignored and users may add them to neaten expressions if they wish. Users must be wary that 'inferred multiplication' is not completely supported (see table 1 for details). If there is any doubt as to whether the inferred multiplication would work on a given equation, be risk averse and assume it won't.

Expression	Evaluation
`(1 + 2)(1 + 3)`	$(1+2)\times(1+3)$
`2(1 + 2)`	Unsupported
`2Sin(pi)`	$2 \times \sin \pi$

Table 1: Inferred Multiplication Support

Fix Tag

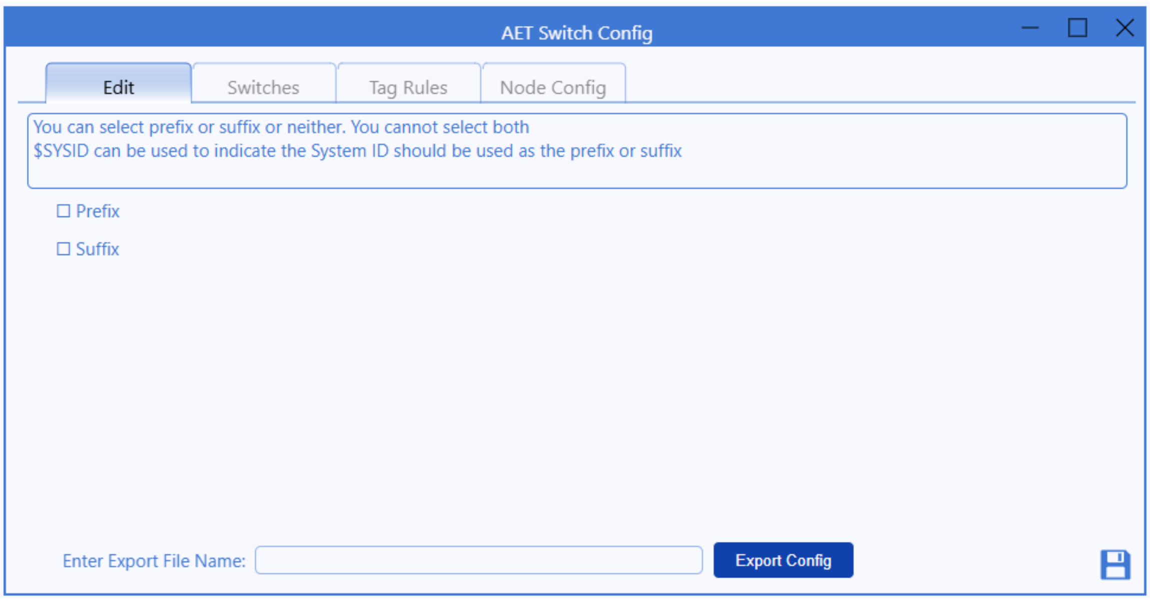
Checks for tags with odd-length values and automatically applies padding to meet the DICOM standard.

When any edit is applied, **group length tags** are automatically removed, as their contents are no longer accurate and they are not required by the standard.

5.4.3 DICOM AET Switcher Node

An AET switcher node allows manipulation of the association stream, cross-connecting an input to an output using different identifiers.

The **AET Switcher Node**, illustrated in Figure 48, enables modification of the Application Entity Titles (AETs) used during communication between a DICOM input and output source. AETs can be prefixed or suffixed with either a fixed string or a dynamic system property. For example, using `$SYSID` applies the value of the `system.id` property (as defined in the Nexus *deployment* properties file) as a prefix or suffix to the AET.



You can select prefix or suffix or neither. You cannot select both
$SYSID can be used to indicate the System ID should be used as the prefix or suffix

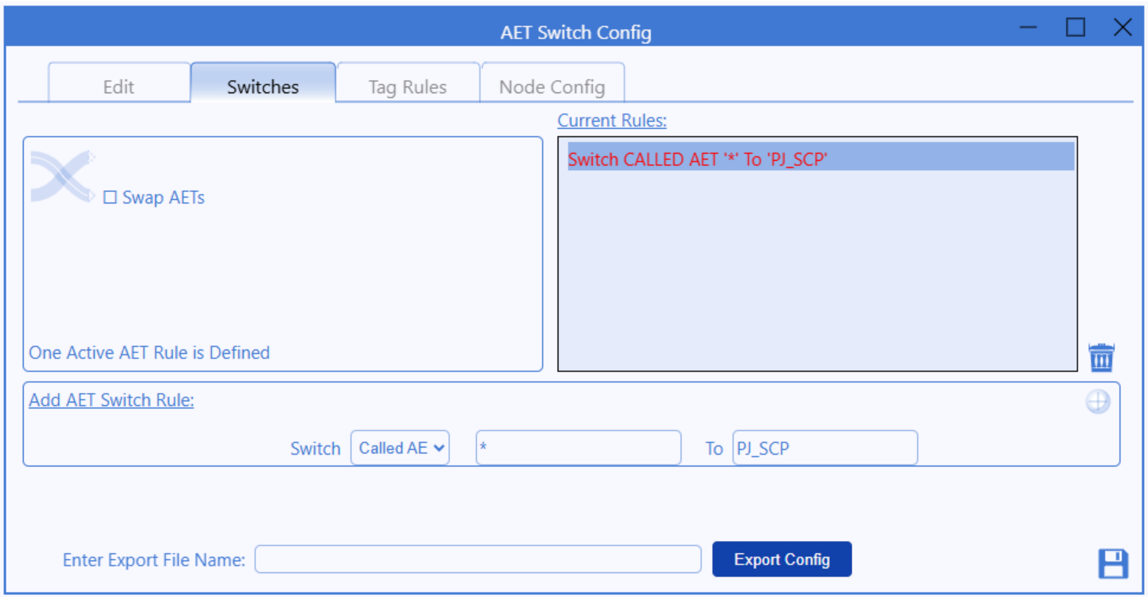
☐ Prefix

☐ Suffix

Enter Export File Name:

Figure 48: Configuring the AET Switcher Node with prefix/suffix values.

The **Switches** tab provides configuration options for replacing AETs with fixed values based on incoming values. As shown in Figure 49, all incoming `called AETs` (represented by `*`) can be replaced with a defined string, such as `PJ_SCP`. There is also an option to **swap the called and calling AETs** within the association using the provided tick box.



☐ Swap AETs

One Active AET Rule is Defined

Current Rules:

Switch CALLED AET '*' To 'PJ_SCP'

Add AET Switch Rule:

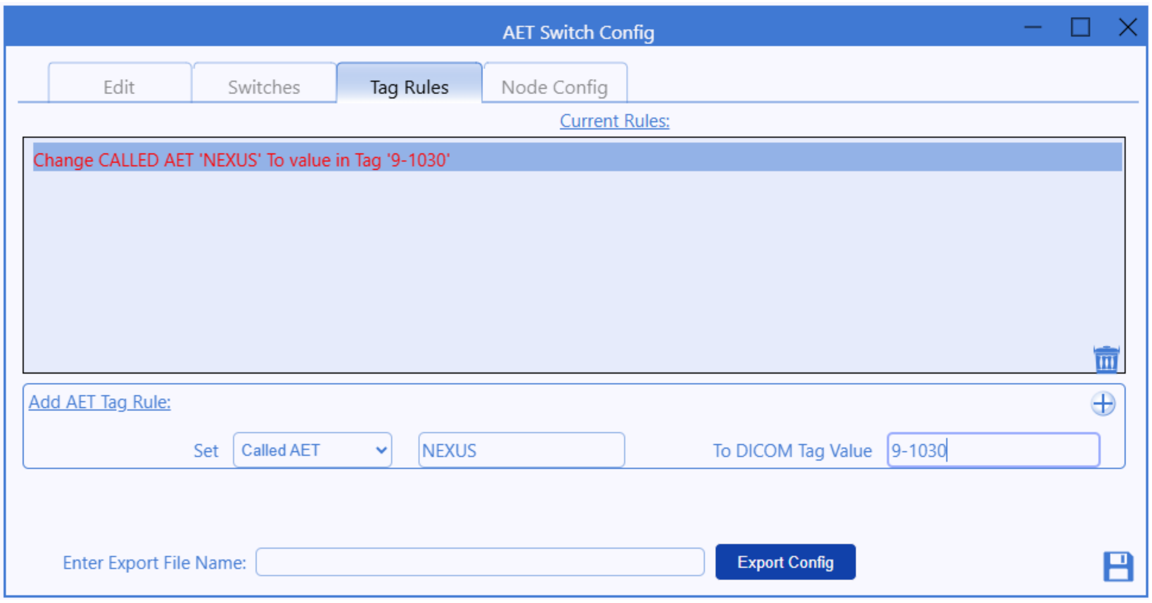
Switch: Called AE | * | To: PJ_SCP

Enter Export File Name:

Figure 49: Switching AETs to fixed values based on incoming AETs.

The **Tag Rules** tab allows the use of values stored in the DICOM header to set AETs dynamically. This is especially useful in workflows where original AETs were written into DICOM headers before storage. When the file is read back in, these values can be extracted and used to restore the original AETs. For example, if the original `called AET` was written to a private tag such as `[0x0009][0x1030]`, this tag's value can be assigned as the active AET when the current value is `NEXUS`, as shown in Figure 50.

The Tag Mapping Node can then be used to remove this temporary tag after restoration, ensuring a clean dataset.



Change CALLED AET 'NEXUS' To value in Tag '9-1030'

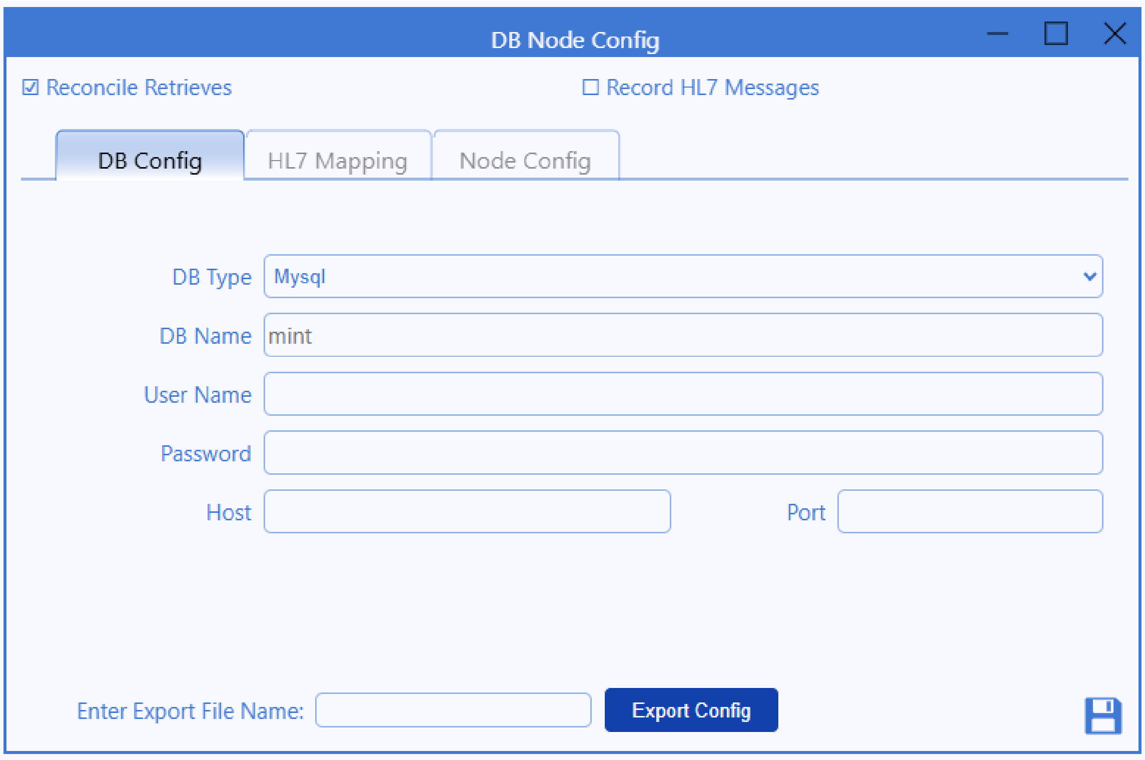
Set: Called AET — NEXUS — To DICOM Tag Value: 9-1030

Figure 50: Using DICOM tag values to dynamically assign AETs.

5.4.4 Database Node

A database node records data from the stream passing through it. Other nodes may also maintain their own internal database connections.

The **Database Node** enables a defined recording database to capture information from the data stream as it flows through the network. While this node provides general database recording functionality, some other nodes may manage their own database connections internally as part of their specific configuration.


DB Node Config

☑ Reconcile Retrieves ☐ Record HL7 Messages

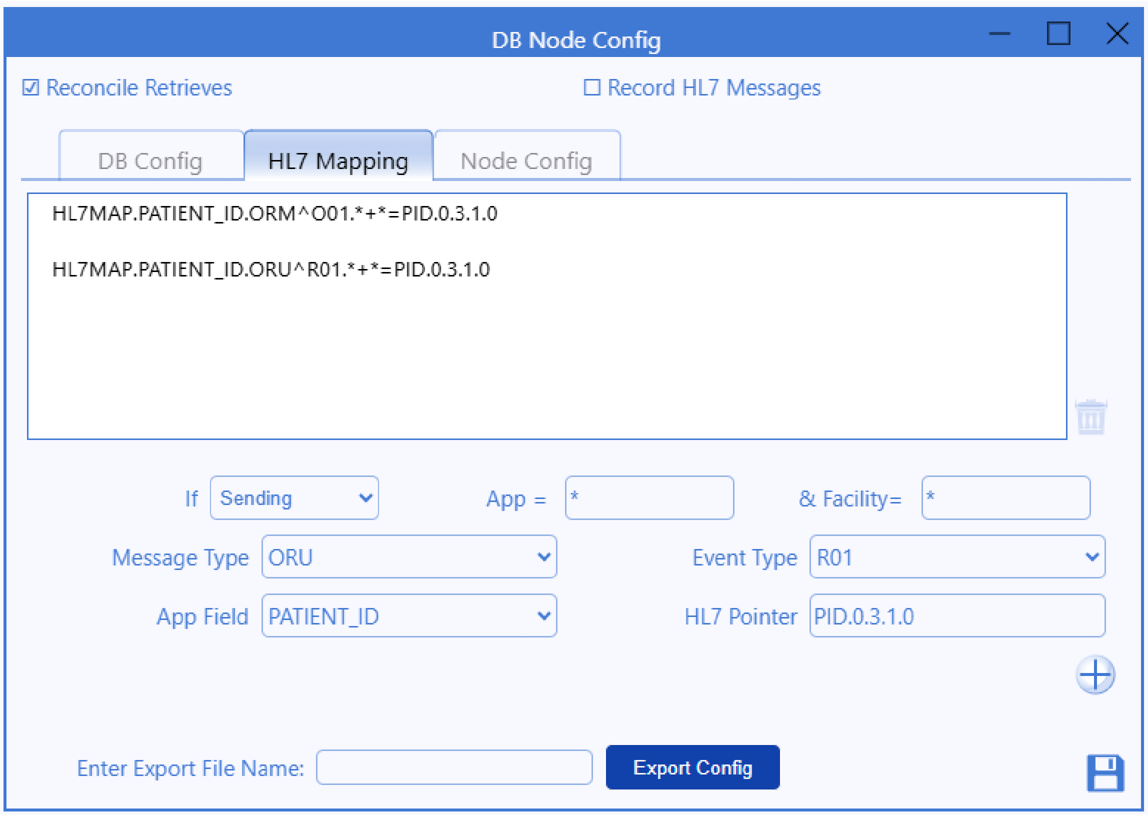
DB Type	Mysql
DB Name	mint
User Name	
Password	
Host	
Port	

Figure 51: Database node connection configuration.

Three types of database implementations are supported: **MySQL**, **Microsoft SQL Server**, and **Java DB** (Apache Derby). The desired implementation is selected via the `DB Type` dropdown menu.

The `Database Name` field defines the name of the database to be used. If the database does not already exist, it will be created automatically when the network is activated—assuming the user has the necessary permissions.

Enabling the `Reconcile Retrieves` option allows the server to update the DICOM header with primary values retrieved from the database prior to sending the data in a query-retrieve response.


DB Node Config

☑ Reconcile Retrieves ☐ Record HL7 Messages

DB Config | HL7 Mapping | Node Config

```
HL7MAP.PATIENT_ID.ORM^O01.*+*=PID.0.3.1.0

HL7MAP.PATIENT_ID.ORU^R01.*+*=PID.0.3.1.0
```

If	Sending	App =	*	& Facility=	*
Message Type	ORU			Event Type	R01
App Field	PATIENT_ID			HL7 Pointer	PID.0.3.1.0

Enter Export File Name:

Figure 52: Database node HL7 mapping configuration.

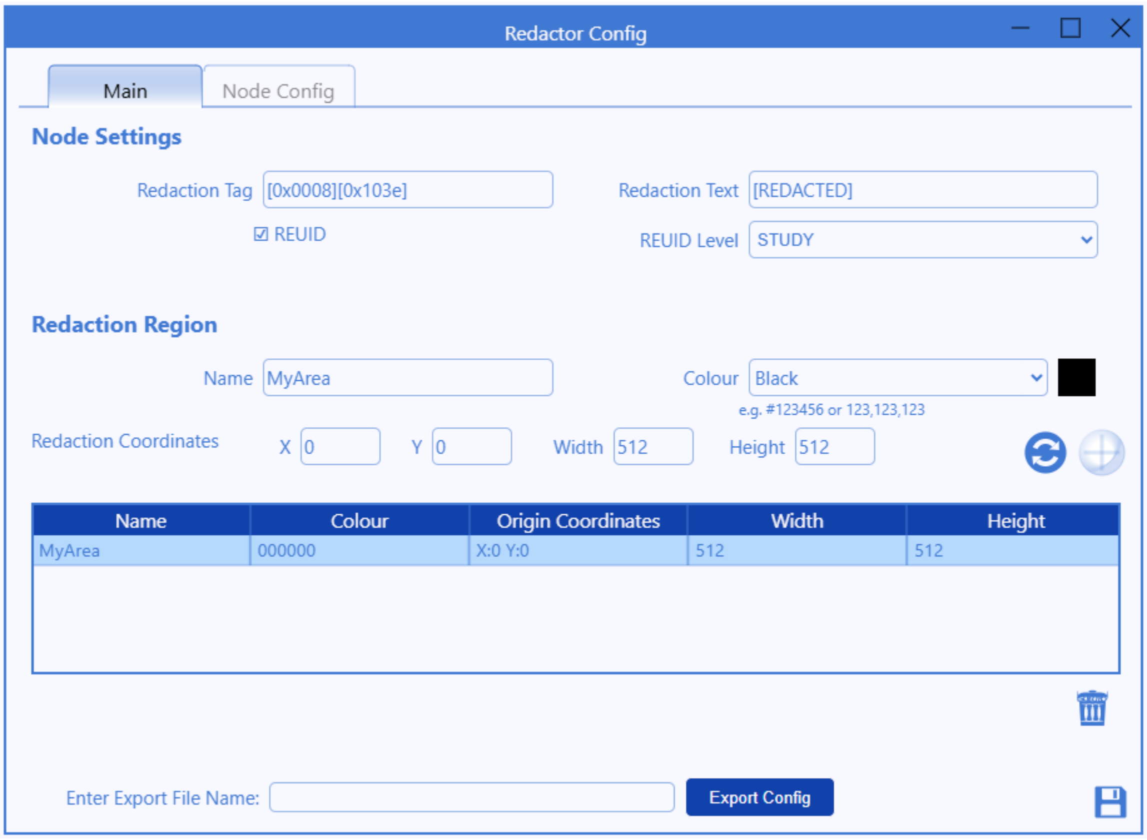
Figure 52 shows how HL7 message fields can be mapped to database fields when processing specific message types. Using Listing 1 as an example, the configuration in this figure maps the patient ID to the second entry in `PID` Field 3, Component 0. In the example message, this corresponds to the value `6666666666`.

Ticking the `Record HL7 Messages` option instructs the system to store incoming HL7 messages in the `Messages` table of the configured database.

5.4.5 Redactor Node

The DICOM Redactor allows pixel data to be redacted for specific regions based on tag values.

The **DICOM Redactor Node** is used to redact visible patient-identifiable information from image data. A common use case is removing the demographic banner in ultrasound (US) images, either for anonymisation purposes or due to incorrect patient data. The redactor also supports removing overlays or banners from DEXA secondary capture reports.



Node Settings

Redaction Tag: [0x0008][0x103e]

Redaction Text: [REDACTED]

☑ REUID

REUID Level: STUDY

Redaction Region

Name: MyArea

Colour: Black

e.g. #123456 or 123,123,123

Redaction Coordinates: X 0 Y 0 Width 512 Height 512

Name	Colour	Origin Coordinates	Width	Height
MyArea	000000	X:0 Y:0	512	512

Enter Export File Name:

Figure 53: DICOM redaction region and tag edit rules configuration.

As shown in Figure 53, various options are available in the redactor configuration. Entering a valid DICOM tag in the `Redaction Tag` field allows you to append the text specified in the `Redaction Text` field to that tag's value. This is useful when it is necessary to insert a placeholder or indication that redaction has occurred.

You can also configure the node to generate new UIDs for the dataset. By enabling the `REUID` tick box, the system will replace UIDs at the selected level: `Study`, `Series`, or `Image`. Selecting a level causes the node to generate new UIDs at that level and all child levels beneath it. This is important when preserving referential integrity after editing patient data or removing content.

The node supports defining custom redaction regions. These regions are created by specifying: - the colour of the redaction box, - the `X, Y` coordinate (top-left origin is `0,0`), - and the width and height of the redaction area.

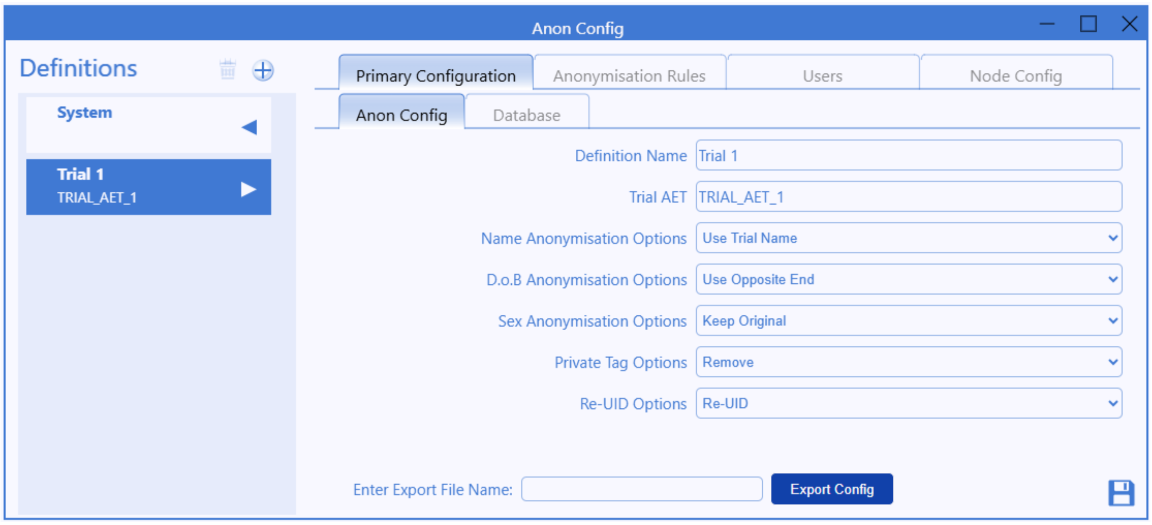
The colour can be selected from predefined options or entered manually. To define a custom colour, use either hexadecimal format (`#RRGGBB`) or decimal RGB values (`RRR,GGG,BBB`). For example, `#FF0000` or `255,0,0` would specify red.

Coloured redaction regions are only visible on colour images. For greyscale images, the region will default to black, regardless of the colour value entered.

5.4.6 Anonymiser Node

Anonymises or edits specified tags in the DICOM header as data passes through the node.

The **DICOM Anonymiser Node** modifies header metadata within DICOM files to remove or replace patient-identifiable information. Specific anonymisation rules can be applied based on the called AET, referred to as the *Trial AET*. If no specific rule is matched, the system's default anonymisation configuration is used.



Definition Name: Trial 1
Trial AET: TRIAL_AET_1
Name Anonymisation Options: Use Trial Name
D.o.B Anonymisation Options: Use Opposite End
Sex Anonymisation Options: Keep Original
Private Tag Options: Remove
Re-UID Options: Re-UID
Enter Export File Name:

Figure 54: Anonymisation trial AET definition and top-level configuration.

A new trial can be created by selecting the (+) icon next to the "Definitions" header. Each trial includes a name, the database configuration for storing anonymisation records, and a defined *Trial AET*—the destination AET used when sending data to invoke the rules.

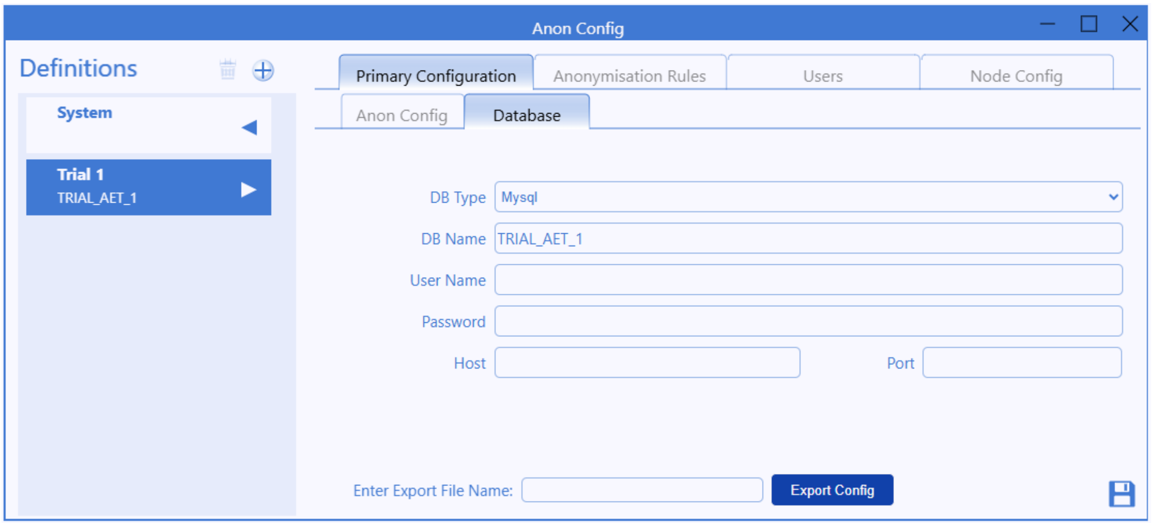
Patient name anonymisation defaults to replacing the name with the Trial AET. Other options include: `Use Pre-set`, `Use Blank`, or `Keep Original`. The pre-set option requires pre-defined anonymised names to be configured for the trial.

Date of birth anonymisation defaults to `Use Opposite End`: if the original date is in the first half of the month, it becomes the 28th; if in the second half, it becomes the 1st. This retains an approximate age while maintaining privacy. Other options include using invalid dates (e.g., 0th or 32nd), removing the date, or keeping the original.

Sex can either be preserved (default) or replaced with "U" (unknown).

Private tags are removed by default but may be retained. Specific tag-level anonymisation is described later.

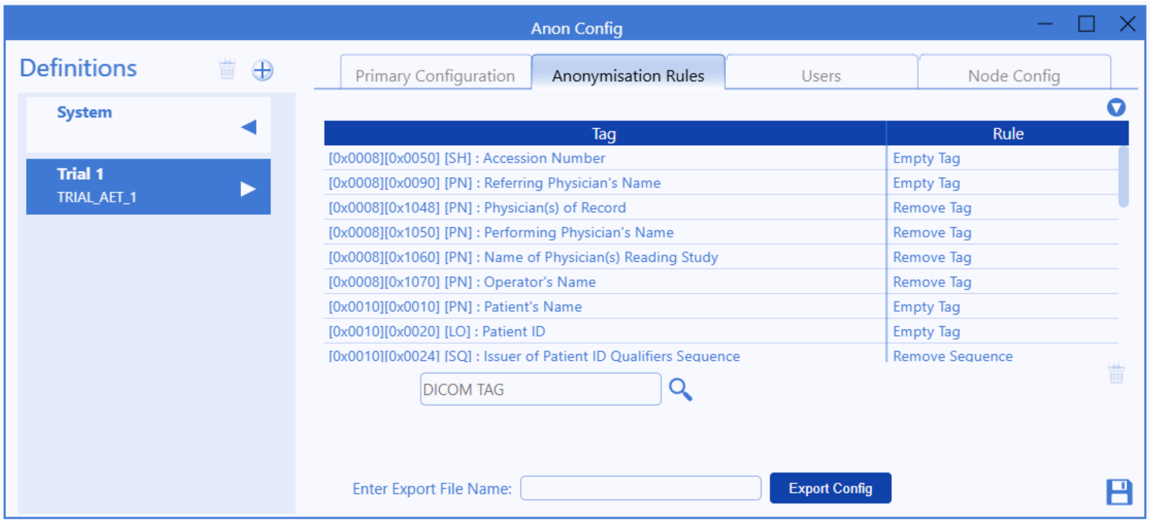
To protect identifiers, DICOM UIDs can be replaced. The `Re-UID` option creates and stores replacement UIDs, allowing consistency between related DICOM objects (e.g., maintaining links between CT images and RTPLANs). These replacement UIDs are reused for consistency within each trial.



DB Type	Mysql
DB Name	TRIAL_AET_1
User Name	
Password	
Host	
Port	
Enter Export File Name:	

Figure 55: Anonymisation trial AET database configuration.

Figure 55 shows the configuration for the trial's database. This database stores both anonymisation and re-UID mappings. Typically, the trial AET is used as the database name.



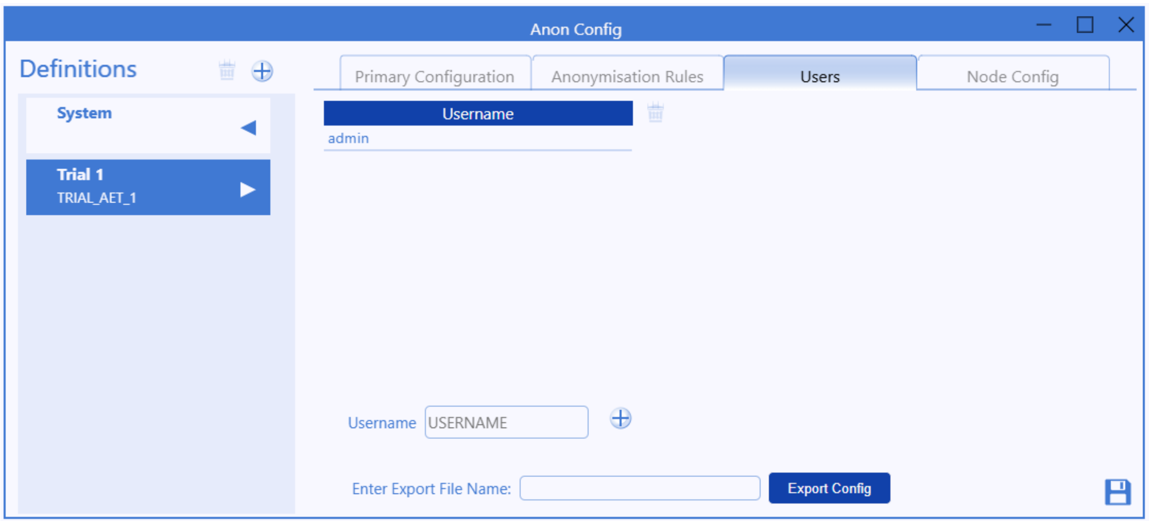
Tag	Rule
[0x0008][0x0050] [SH] : Accession Number	Empty Tag
[0x0008][0x0090] [PN] : Referring Physician's Name	Empty Tag
[0x0008][0x1048] [PN] : Physician(s) of Record	Remove Tag
[0x0008][0x1050] [PN] : Performing Physician's Name	Remove Tag
[0x0008][0x1060] [PN] : Name of Physician(s) Reading Study	Remove Tag
[0x0008][0x1070] [PN] : Operator's Name	Remove Tag
[0x0010][0x0010] [PN] : Patient's Name	Empty Tag
[0x0010][0x0020] [LO] : Patient ID	Empty Tag
[0x0010][0x0024] [SQ] : Issuer of Patient ID Qualifiers Sequence	Remove Sequence

Figure 56: Anonymisation options for individual tags.

Figure 56 displays the tag-level configuration tab. Tags can be anonymised, blanked, or replaced with static text. Rules apply recursively across sequences. For example, entering tag `8,80` and pressing return will auto-format the tag. Options differ for sequence and non-sequence tags. To add a rule, click the (+) icon. Existing rules can be edited or deleted using the appropriate table controls.

Anonymisation rule sets can be exported or imported using the download and upload buttons in the top-right corner of the tab.

The final configuration tab defines authorised users of the anonymisation record. A user is added by entering their username and clicking the (+) icon. Only authorised users may view or edit



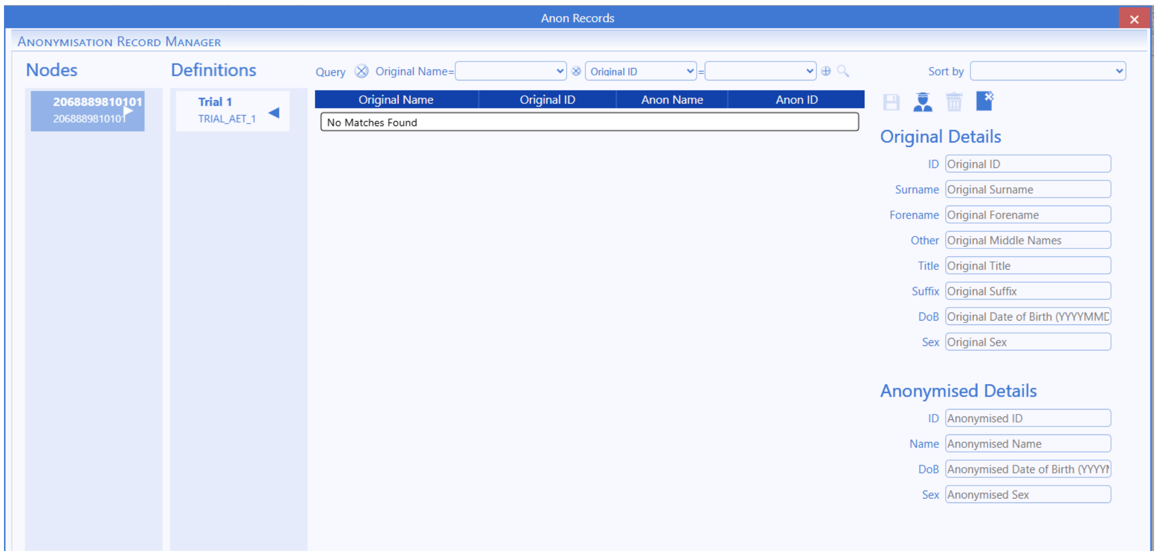
Username
admin

Figure 57: User configuration for a trial AET.

anonymisation mappings.

Once all configuration steps are complete, select the `Save` icon. Changes are not applied until the server is running and the node is active.

Managing Anonymisation Records



Original Name	Original ID	Anon Name	Anon ID
No Matches Found			

Figure 58: The anonymisation record manager user interface.

When a network is started, the Anonymisation Record Manager can be accessed by selecting the icon at the bottom left of the anonymiser node. As shown in Figure 58, users can:

- Predefine anonymisation records,
- Search, modify or delete existing records,

- View associated trials and nodes.

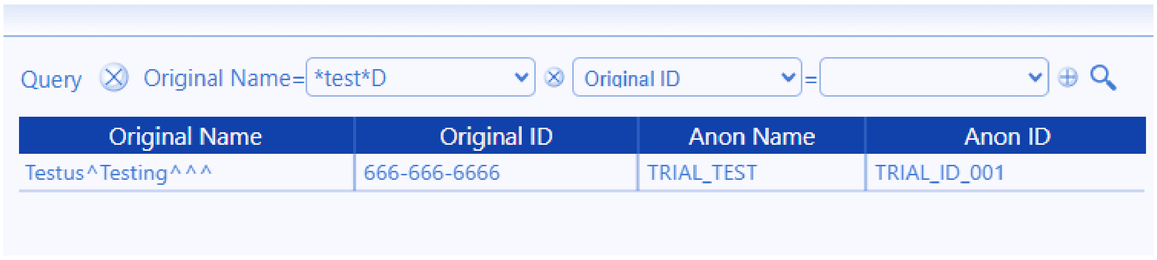
Only trials assigned to the current user will be visible.

A CSV file can be dropped onto the interface to populate records. The file should be in one of the following formats:

```
Original DICOM Name (e.g. Surname^Forename)
Original ID
Original Date of Birth
Original Sex
Anonymised Name
Anonymised ID
Anonymised Date of Birth
Anonymised Sex
```

or:

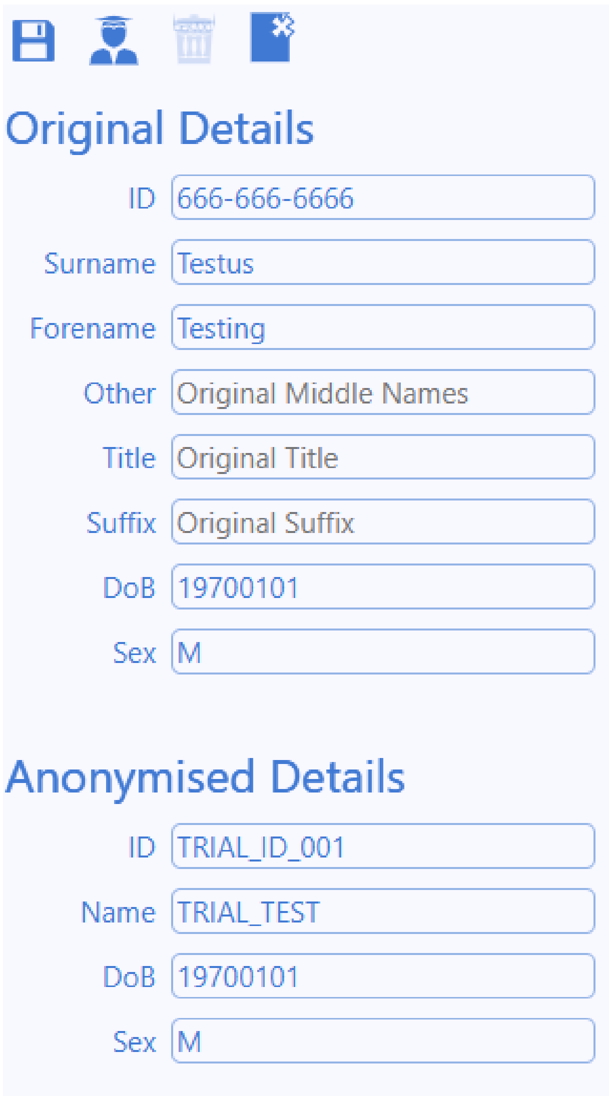
```
Original Surname
Original Forename
Original Middlename
Original Title
Original Suffix
Original ID
Original Date of Birth
Original Sex
Anonymised Name
Anonymised ID
Anonymised Date of Birth
Anonymised Sex
```


Query Original Name= *test*D Original ID =

Original Name	Original ID	Anon Name	Anon ID
Testus^Testing^^^	666-666-6666	TRIAL_TEST	TRIAL_ID_001

Figure 59: Anonymisation record manager query and results.

Figure 59 shows the query interface. A wildcard (*) can be used in search values to find partial matches. Results appear in a table format.



Original Details

ID	666-666-6666
Surname	Testus
Forename	Testing
Other	Original Middle Names
Title	Original Title
Suffix	Original Suffix
DoB	19700101
Sex	M

Anonymised Details

ID	TRIAL_ID_001
Name	TRIAL_TEST
DoB	19700101
Sex	M

Figure 60: Anonymisation record manager editor panel.

Selecting a row opens the full record in the editor (Figure 60). Records can be edited and saved, or cleared using the clear icon. Note: Records are only applied when data is sent using the defined Trial AET (e.g. `TRIAL_AET_1`).

Entries can be deleted using the bin icon, or all records cleared using the delete-all icon. The delete-all operation is irreversible and requires user confirmation.

5.4.7 De-Anonymiser Node

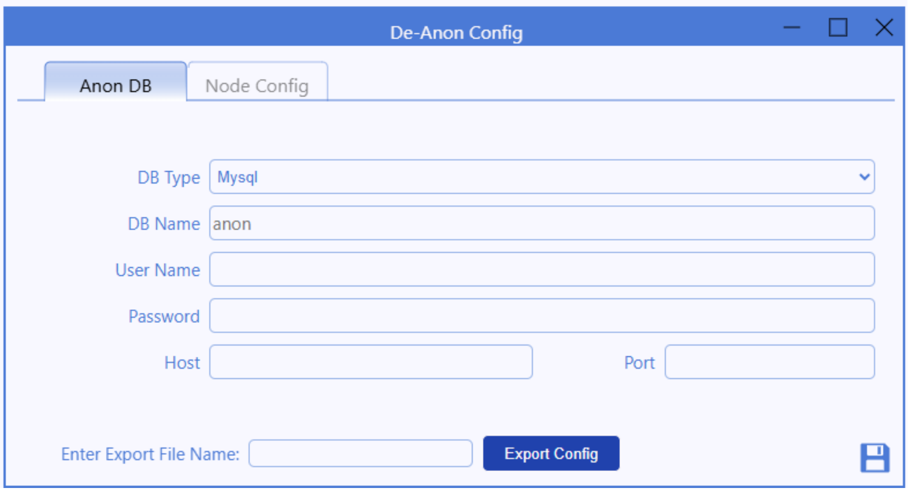
The DICOM De-Anonymiser restores the original primary demographics and UIDs to a study previously anonymised by the Nexus DICOM Anonymiser Node.

The **De-Anonymiser Node** is used in conjunction with the **Anonymiser Node** to re-identify DICOM objects that were previously anonymised within a Nexus network. This includes restoring original patient demographics and UID values by referencing the anonymisation record stored in a dedicated trial database.

In order for de-anonymisation to function correctly:

- The same trial information must be configured in both the Anonymiser and De-Anonymiser nodes.
- Data must be received via the same AET (Application Entity Title) used during anonymisation.

When these conditions are met, the De-Anonymiser Node queries the appropriate database using the incoming anonymised values and replaces them with the original values stored in the anonymisation mapping.



DB Type	Mysql
DB Name	anon
User Name	
Password	
Host	
Port	

Figure 61: De-Anonymiser Node configuration.

Figure 61 illustrates the configuration options for the De-Anonymiser Node. This configuration is a simplified counterpart to the one used by the Anonymiser Node, and it is essential that settings are consistent between the two for de-anonymisation to succeed.

If **study-level data** must also be reconciled following de-anonymisation, the De-Anonymiser Node should be connected upstream of a **Reconciliation Node** in the network. This allows restored demographic data to be harmonised with other systems.

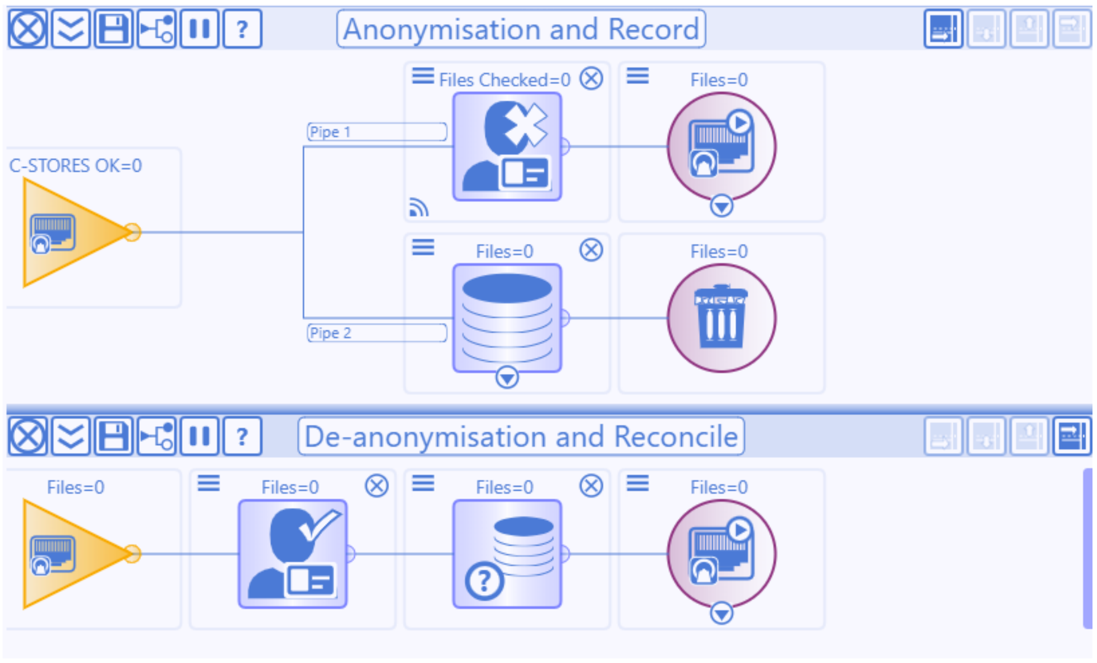


Figure 62: Example network using De-Anonymiser and Reconciliation Nodes.

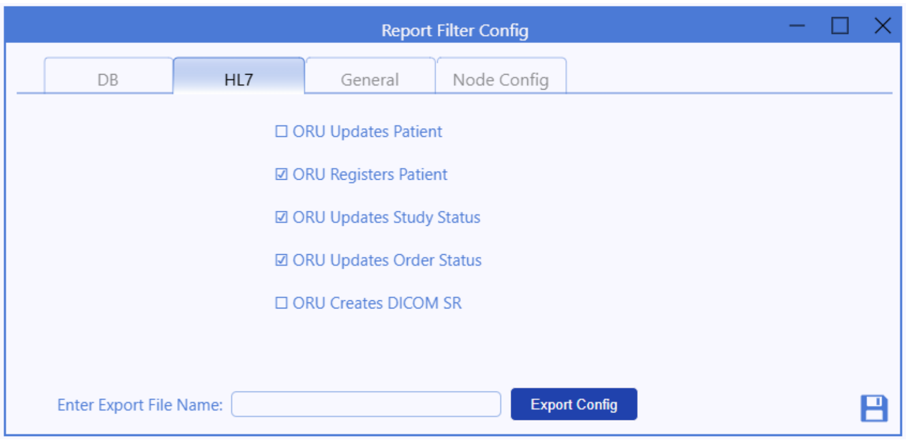
5.4.8 Report Filter Node

A node used to handle HL7 and DICOM reports, and to convert between them.

The **Report Filter Node** is primarily used for integration with *DICOM Explorer*, enabling the conversion of reports between formats including database entries, HL7 ORU messages, and DICOM Structured Reports (SR).

The node includes a standard database configuration, typically pointing to the DICOM Explorer database. Figure 63 illustrates the configuration options available on the *HL7* tab. One of the key features is the ability to generate a DICOM SR from an incoming HL7 ORU message. The generated SR is routed through the **time-dependent channel** of the node, represented by the downward pipe in the network diagram.

In addition to format conversion, the *General* tab provides an option to store the report in the configured database.


Report Filter Config

DB | HL7 | General | Node Config

☐ ORU Updates Patient

☑ ORU Registers Patient

☑ ORU Updates Study Status

☑ ORU Updates Order Status

☐ ORU Creates DICOM SR

Enter Export File Name:

Figure 63: HL7 configuration options for the Report Filter Node.

5.4.9 Decompression Node

A node that decompresses pixel data for images using supported transfer syntaxes.

The **Decompression Node** is responsible for decompressing compressed DICOM pixel data as it passes through the network. Decompression will occur if the system has access to the appropriate codec libraries for the transfer syntax used.

Some compression formats can be decompressed using standard codecs available within the Java Runtime Environment (JRE). However, certain formats may require additional native or third-party libraries to be installed in the system's `lib` directory.

Optionally, the node allows modification of the **Called AET** (Application Entity Title) during decompression. This can be useful when specific workflows or routing behaviours are required based on the decompressed status of the image.

5.4.10 Data Consistency Node

The Data Consistency / Validation Node is used to validate patient demographics, as well as study and series-level DICOM attributes.

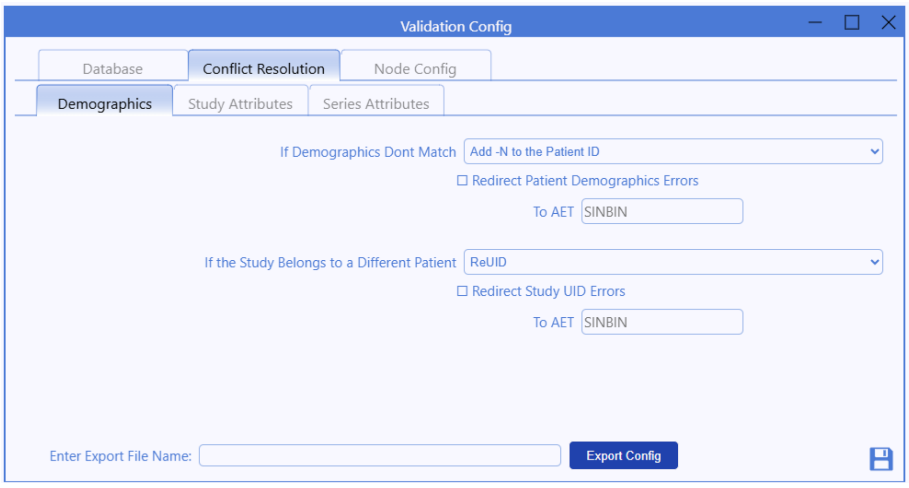
The **Data Consistency Node** validates DICOM objects by comparing header information against records stored in a database. The node uses a standard database configuration page to establish database connectivity. The configured database is queried for existing patient and study data to determine how to resolve any inconsistencies.

Figure 64 illustrates how the node handles conflicts when patient names or demographic details in the DICOM header do not match the corresponding database records for the patient ID. Options include:

- Appending an incremental suffix to the patient ID for each demographic error,
- Rejecting the data outright,
- Replacing the conflicting values in the DICOM header with the database values.

Additionally, the node can reroute errored data to a designated Application Entity Title (AET) by modifying the destination AET when a mismatch is encountered.

The second configuration option addresses cases where a `StudyInstanceUID` already exists in the database but is associated with a different patient. In this case, the system can be instructed to either generate a new study UID or reject the data. The Called AET may also be changed to redirect such studies to a specialised handler node.



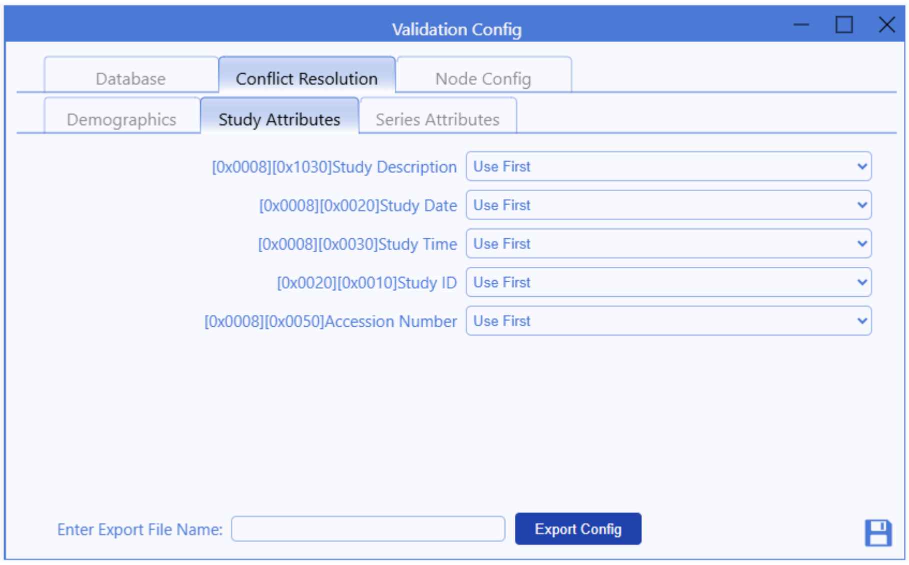
If Demographics Dont Match: Add -N to the Patient ID
☐ Redirect Patient Demographics Errors
To AET: SINBIN
If the Study Belongs to a Different Patient: ReUID
☐ Redirect Study UID Errors
To AET: SINBIN
Enter Export File Name:

Figure 64: Demographics conflict resolution configuration options.

Conflicts at the **study-level** and **series-level** can also be managed for specific DICOM tags. Figures 65 and 66 demonstrate the configuration pages where handling rules for tag-level discrepancies are defined.

If conflicting values are detected for a given tag across multiple DICOM objects belonging to the same study or series, the system can be set to:

- Use the first received value,
- Use the last received value, or
- Ignore the discrepancy entirely.


Validation Config

Database | Conflict Resolution | Node Config

Demographics | Study Attributes | Series Attributes

[0x0008][0x1030]Study Description: Use First

[0x0008][0x0020]Study Date: Use First

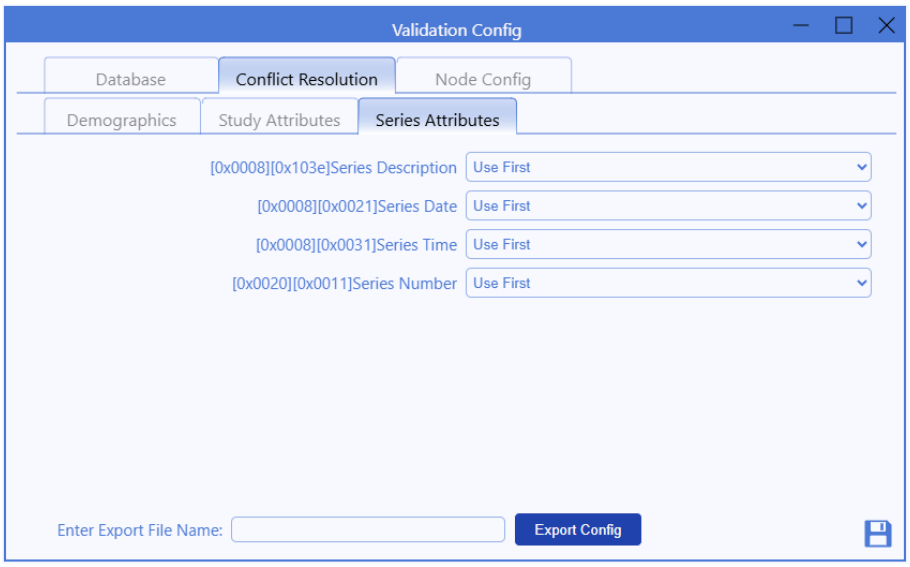
[0x0008][0x0030]Study Time: Use First

[0x0020][0x0010]Study ID: Use First

[0x0008][0x0050]Accession Number: Use First

Enter Export File Name:

Figure 65: Study-level DICOM attribute conflict resolution configuration.


Validation Config

Database | Conflict Resolution | Node Config

Demographics | Study Attributes | Series Attributes

[0x0008][0x103e]Series Description: Use First

[0x0008][0x0021]Series Date: Use First

[0x0008][0x0031]Series Time: Use First

[0x0020][0x0011]Series Number: Use First

Enter Export File Name:

Figure 66: Series-level DICOM attribute conflict resolution configuration.

5.4.11 Data Reconciler Node

The Data Reconciler Node retrieves authoritative metadata from a database using the SOP Instance UID and updates the DICOM header accordingly.

The **Data Reconciler Node** is used to correct or complete metadata in a DICOM object by referencing an authoritative external database source.

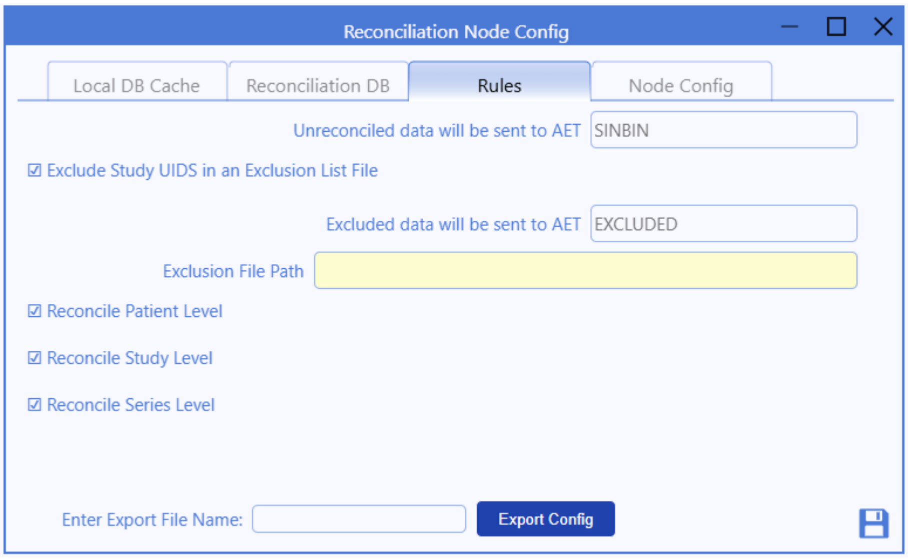
This node uses two database connections:

- A **local database**, used for initial cache checks.

- An **external authoritative database**, queried when information is not available locally.

When added to a network, the node inspects each DICOM object's `SOPInstanceUID` and attempts to retrieve associated metadata from the authoritative database. If the record is not already cached and caching is enabled, the node retrieves and stores the complete Study/Series/Instance information for all objects within the same study. This caching minimises redundant lookups for subsequent objects.

Once a match is found, the node updates the DICOM header. The specific fields and tags affected will depend on the schema of the external database in use.


Reconciliation Node Config

Unreconciled data will be sent to AET: SINBIN

☑ Exclude Study UIDS in an Exclusion List File

Excluded data will be sent to AET: EXCLUDED

Exclusion File Path:

☑ Reconcile Patient Level

☑ Reconcile Study Level

☑ Reconcile Series Level

Enter Export File Name:

Figure 67: Reconciliation exception handling configuration.

Figure 67 illustrates the exception handling options available within the node. If a SOP Instance UID fails to resolve during reconciliation, the node can modify the **Called AET** to route the affected object along a specific pathway in the network. By default, unresolved records are assigned the AET `SINBIN`, with variations such as `SINBIN_NOMATCH` for unmatched records and `SINBIN_MULTI` for conflicts where multiple matches are found.

The prefix of the Called AET can be customised, though care must be taken to ensure that the final AET remains within the DICOM-standard 16-character limit, especially when suffixes are appended.

An exclusion list can also be defined by specifying a file containing Study Instance UIDs. If a reconciled record matches an entry in this list, the node redirects the object using a separate AET (default: `EXCLUDED`). This behaviour is configured on the third tab of the node's settings interface.

When the **Pukka-J schema** is selected on the reconciliation tab, the node enables extended matching logic: if a `SOPInstanceUID` fails to reconcile, the node can fall back to matching using parent identifiers such as `SeriesInstanceUID`, `StudyInstanceUID`, or `PatientID`.

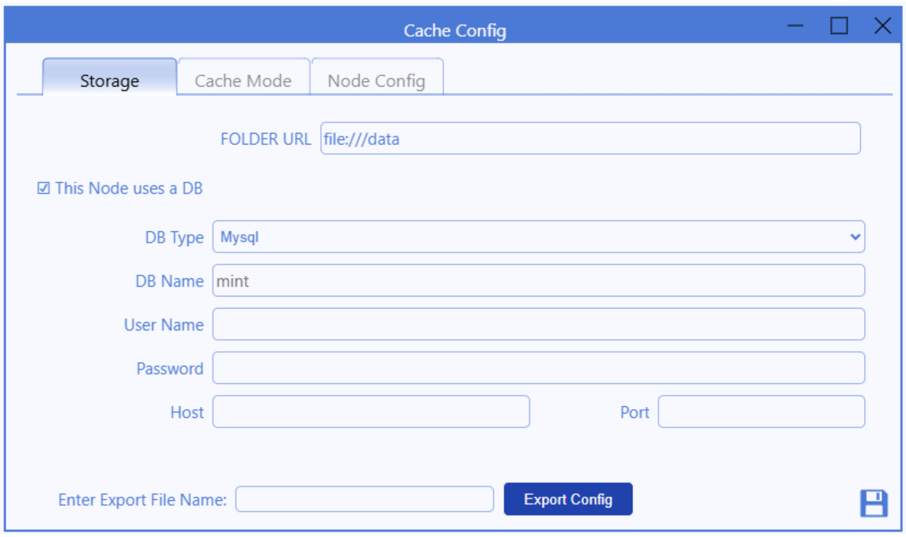
This advanced reconciliation is especially useful in workflows involving the **De-Anonymiser Node**, where metadata needs to be re-attached to original studies following processing of anonymised datasets.

5.4.12 Temporary Cache Node

The Temporary Cache Node enables the short-term storage of data for use in time-dependent workflows or error handling pathways.

The **Temporary Cache Node** stores data passing through the network, enabling flexible routing and deferred processing. It is typically employed in two scenarios:

- As part of an **error channel** from an output node, capturing failed transactions for later inspection or rerouting.
- In **time-dependent workflows**, where data needs to be held and forwarded at a later stage in the pipeline.



FOLDER URL file:///data

☑ This Node uses a DB

DB Type Mysql

DB Name mint

User Name

Password

Host

Port

Enter Export File Name:

Figure 68: Temporary cache node configuration interface.

The **Folder URL** field defines the local file system path where cached data is stored. This path may optionally begin with `file://`, followed by an absolute directory reference (e.g., `/data` on Unix-based systems or `d:\data` on Windows systems).

If the cache requires integration with a database—for example, to index or log cached entries—this can be configured using the same method as described for the Database Node (see Section 5.4.4).

The node supports two operational modes:

- **Active Mode**: Automatically attempts to forward cached data to the output channel. On successful transmission, the cached entry is deleted.

- **Passive Mode**: Does not automatically forward data. Cached entries are retained and must be retrieved manually or by an external trigger.

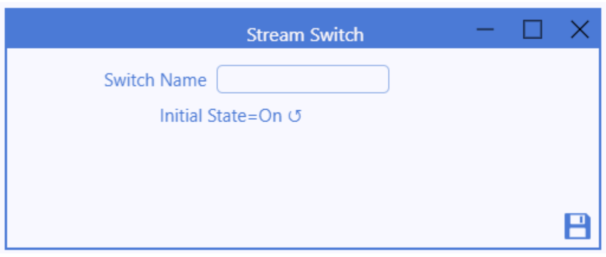
This dual-mode configuration provides both automated and manual flexibility for managing transient data within dynamic processing pipelines.

5.4.13 Switch Node

The Switch Node provides control over whether a particular data pathway within the network is active.

The **Switch Node** functions as a controllable gateway within the dataflow network. It is used to dynamically enable or disable specific routes, allowing greater flexibility in how and when data is processed or directed.

This can be particularly useful in scenarios where certain pathways are only required during specific workflows or under particular conditions.


Stream Switch

Switch Name

Initial State=On ↺

Figure 69: Switch Node configuration dialogue.

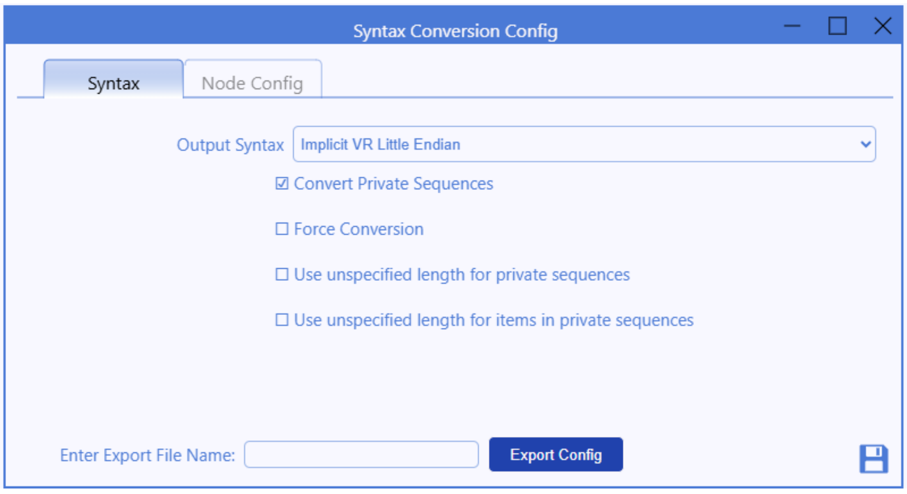
The node can be assigned a unique **name**, which is used to identify it for remote activation or automation scripts. The **initial state** of the switch (on or off) can be toggled via the arrow icon in the configuration interface.

Once active, the switch permits the passage of data through its connected output path; when deactivated, it halts the flow until re-enabled.

5.4.14 Syntax Convertor Node

The Syntax Convertor Node transforms DICOM headers between implicit and explicit transfer syntaxes.

The **Syntax Convertor Node** is used to alter the transfer syntax of DICOM headers, typically when converting from *Explicit Little Endian* to *Implicit Little Endian*. Caution should be taken when performing this conversion, particularly with private tags—especially sequences—as their explicit value representation (VR) encoding is lost. Since private tags are not part of the public DICOM dictionary, other systems may struggle to interpret them correctly after conversion.



Output Syntax: Implicit VR Little Endian

☑ Convert Private Sequences

☐ Force Conversion

☐ Use unspecified length for private sequences

☐ Use unspecified length for items in private sequences

Enter Export File Name:

Figure 70: Syntax Convertor Node configuration options.

Figure 70 illustrates the available configuration settings. By default, the option to convert private sequences is enabled. This setting determines whether private sequences within the DICOM header should be converted along with the rest of the data.

The final two options specify how lengths are encoded in private sequences—either using a delimiter (with a length of -1) or by assigning a specific length.

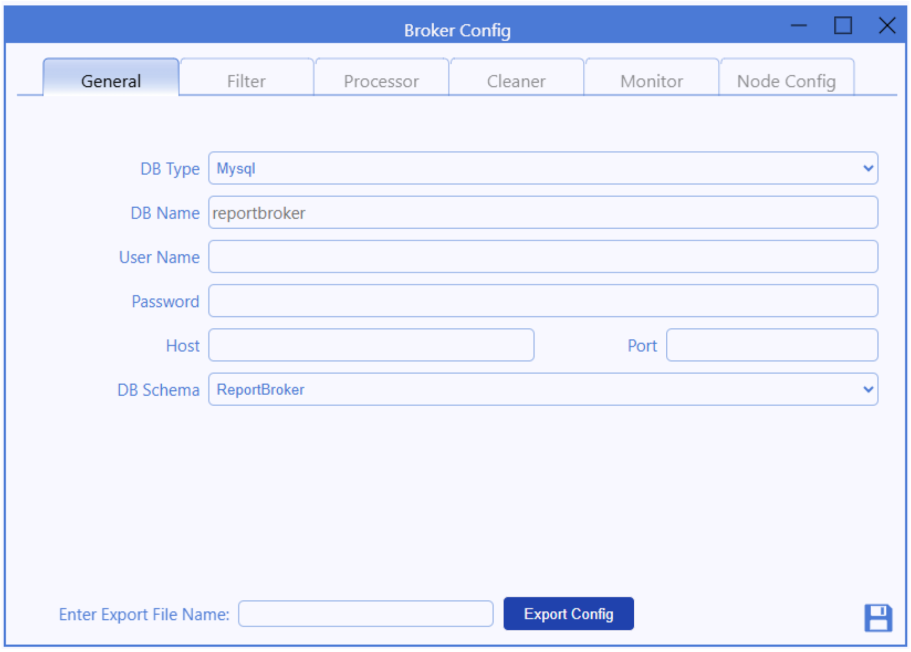
The **Force Conversion** option compels the system to reserialise the header in the chosen output syntax, even if the original header claims to already use that syntax. This is particularly useful in scenarios where inconsistencies exist—such as headers labelled as implicit but containing elements encoded using explicit syntax, often within sequences.

5.4.15 Report Broker Node

BROKER

A broker node enables asynchronous communication with external systems to determine handling strategies for incoming data.

The report broker node manages incoming data asynchronously. It is typically used to receive and temporarily cache data using a file output node, after which it queries external systems to determine how to process or route the cached data. Each broker may run multiple sub-processes, each performing specific tasks.

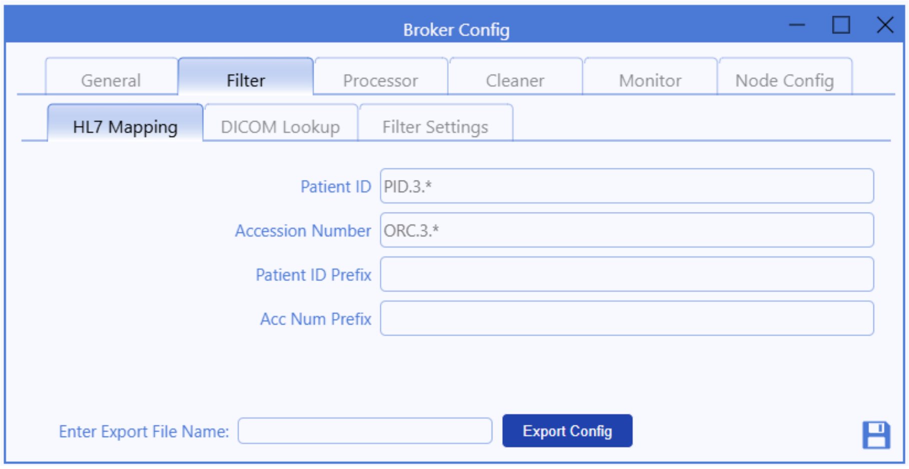


DB Type: Mysql
DB Name: reportbroker
User Name:
Password:
Host:
Port:
DB Schema: ReportBroker
Enter Export File Name:

Figure 71: General broker configuration dialogue.

As shown in Figure 71, the general settings configure the broker's database, which stores information about cached items pending processing. A schema can be selected to define the broker's overall role, with `Report Broker` as the default.

The **Filter** is the first stage in asynchronous handling. When an HL7 message arrives, key information is extracted and recorded in the database. The HL7 Mapping tab, shown in Figure 72, allows the user to define where the accession number and patient ID are extracted from within the HL7 message.



Patient ID: PID.3.*
Accession Number: ORC.3.*
Patient ID Prefix:
Acc Num Prefix:
Enter Export File Name:

Figure 72: HL7 mapping configuration for the broker filter.

Multiple patient ID entries can be mapped using wildcard notation or comma-separated lists. For example:

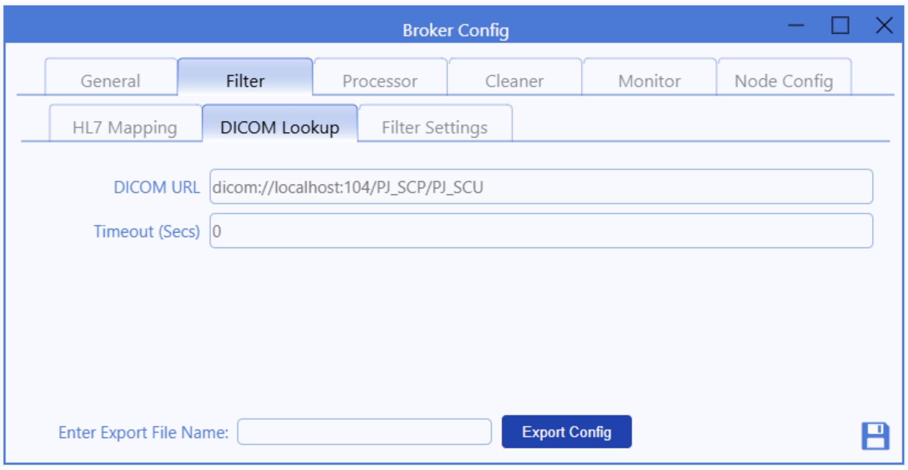
```
PID.0.3.*.0
```

This maps all first components of entries in field 3 of the first PID segment. Accession numbers are mapped from a single location, defaulting to:

```
ORC.0.3.0.0
```

The broker uses these values to query an external DICOM archive, as defined by the DICOM lookup configuration (Figure 73). The URL format is:

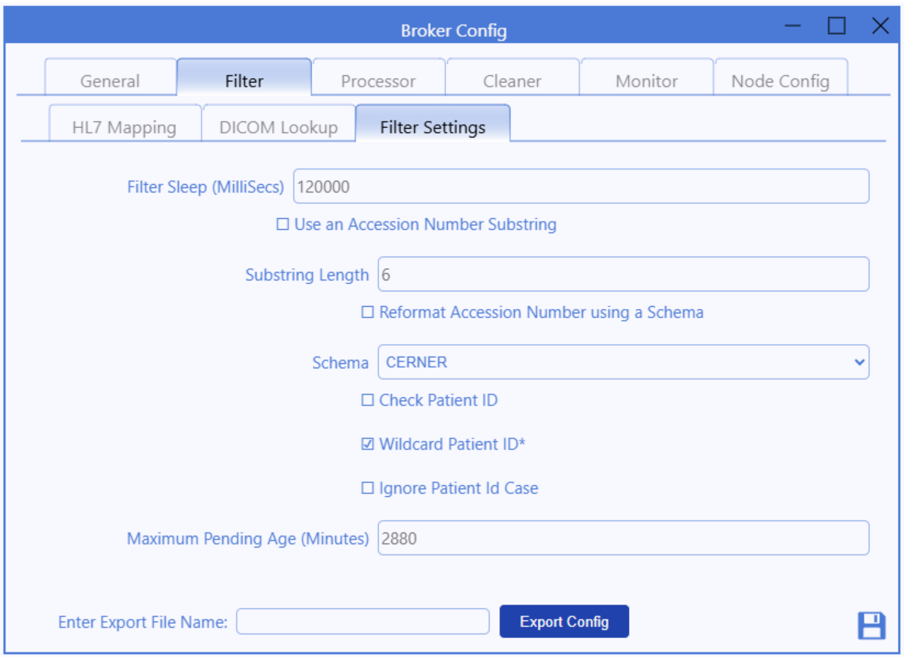
```
dicom://{ipaddress}:{port}/{called_aet}/{calling_aet}
```



DICOM URL: dicom://localhost:104/PJ_SCP/PJ_SCU
Timeout (Secs): 0
Enter Export File Name:

Figure 73: DICOM lookup configuration.

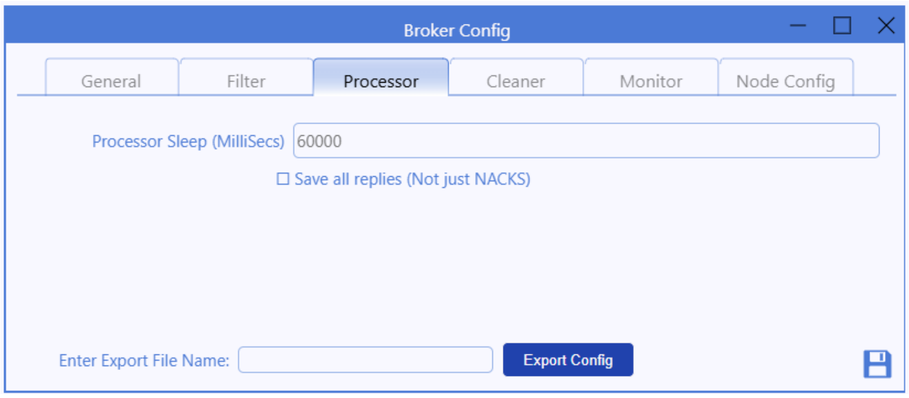
If a match is found, the filter updates the database entry status to `ACCEPTED`. The filter settings in Figure 74 define reformatting rules, sleep duration between filter runs, and fallback mechanisms such as substring accession matching or reformatting for specific vendors.



Filter Sleep (MilliSecs): 120000
☐ Use an Accession Number Substring
Substring Length: 6
☐ Reformat Accession Number using a Schema
Schema: CERNER
☐ Check Patient ID
☑ Wildcard Patient ID*
☐ Ignore Patient Id Case
Maximum Pending Age (Minutes): 2880
Enter Export File Name:

Figure 74: Additional filter configuration settings.

After a successful match, the **Processor** component takes over. It periodically polls for jobs, retrieves cached data, updates it using the DICOM query results, and forwards it. Processed entries may optionally be stored or discarded, depending on configuration (Figure 75).



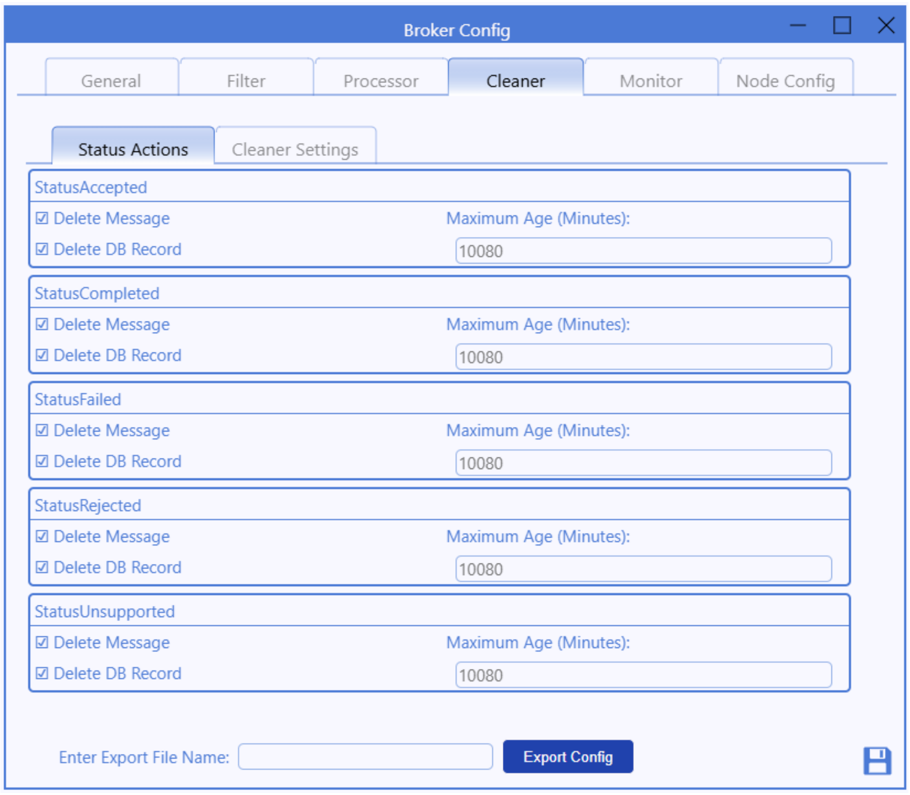
Processor Sleep (MilliSecs) 60000

☐ Save all replies (Not just NACKS)

Enter Export File Name:

Figure 75: Processor configuration dialogue.

To manage ageing entries, the **Cleaner** process periodically checks database records and removes old entries based on status-specific rules, as configured in Figures 76 and 77. By default, the cleaner runs daily at 11pm.



StatusAccepted
☑ Delete Message — Maximum Age (Minutes): 10080
☑ Delete DB Record

StatusCompleted
☑ Delete Message — Maximum Age (Minutes): 10080
☑ Delete DB Record

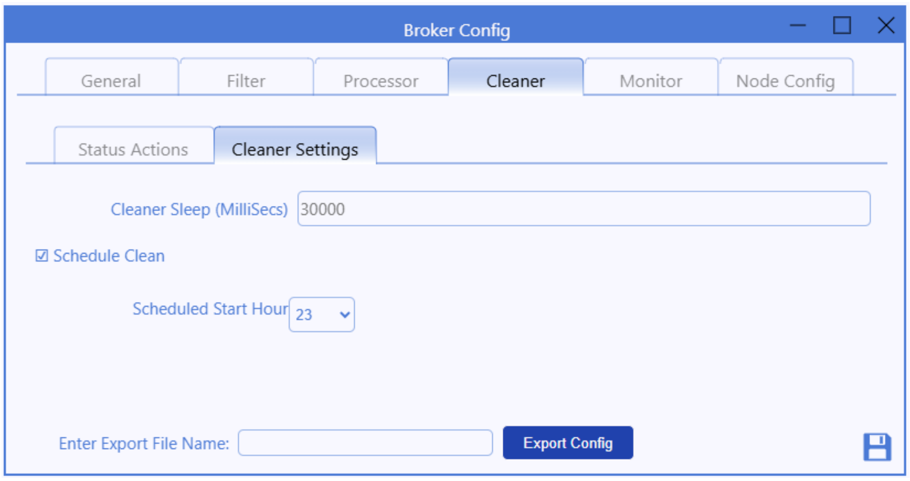
StatusFailed
☑ Delete Message — Maximum Age (Minutes): 10080
☑ Delete DB Record

StatusRejected
☑ Delete Message — Maximum Age (Minutes): 10080
☑ Delete DB Record

StatusUnsupported
☑ Delete Message — Maximum Age (Minutes): 10080
☑ Delete DB Record

Enter Export File Name:

Figure 76: Cleaner configuration for status-specific expiry.



Cleaner Sleep (MilliSecs) 30000

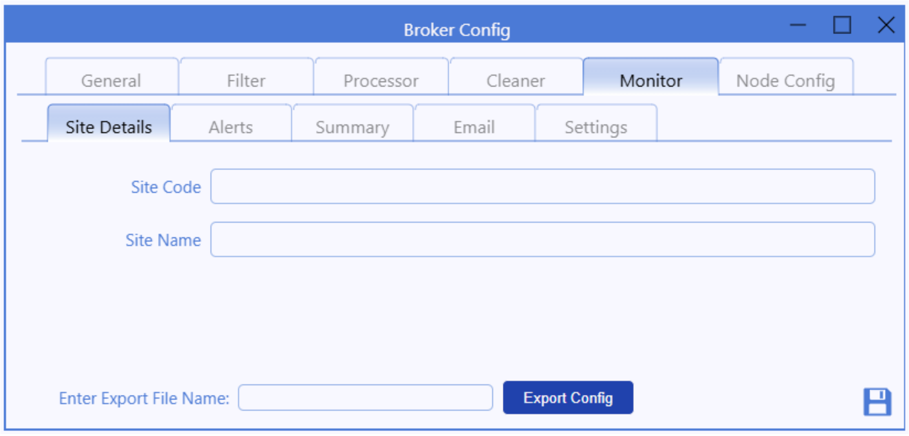
☑ Schedule Clean

Scheduled Start Hour 23

Enter Export File Name:

Figure 77: Cleaner scheduling settings.

The **Monitor** process tracks the operation of the broker's components. Site information (Figure 78) and alert mechanisms (Figures 79 and 80) can be configured to email notifications when processes fail or summaries are due. SMTP server details are set in Figure 81.

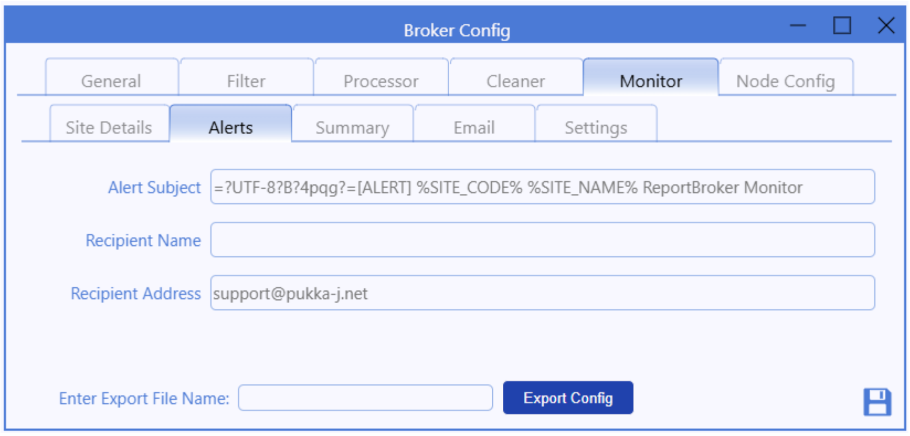


Site Code

Site Name

Enter Export File Name:

Figure 78: Site identification for broker monitoring.



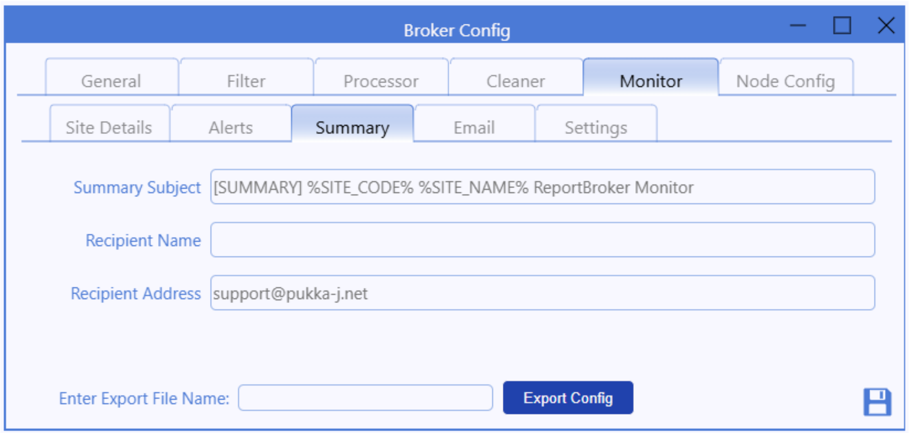
Alert Subject =?UTF-8?B?4pqg?=[ALERT] %SITE_CODE% %SITE_NAME% ReportBroker Monitor

Recipient Name

Recipient Address support@pukka-j.net

Enter Export File Name:

Figure 79: Alert email configuration.



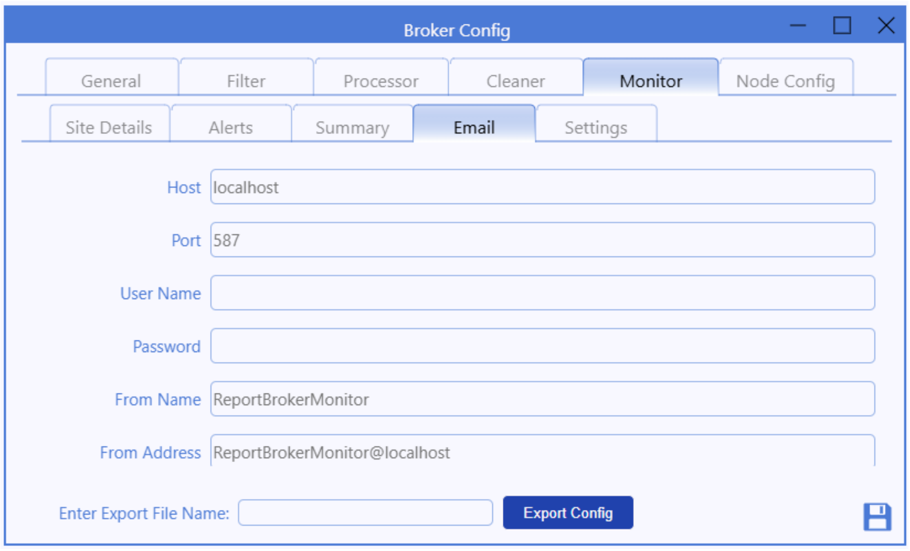
Summary Subject: [SUMMARY] %SITE_CODE% %SITE_NAME% ReportBroker Monitor

Recipient Name:

Recipient Address: support@pukka-j.net

Enter Export File Name:

Figure 80: Summary email scheduling for monitoring.



Host: localhost

Port: 587

User Name:

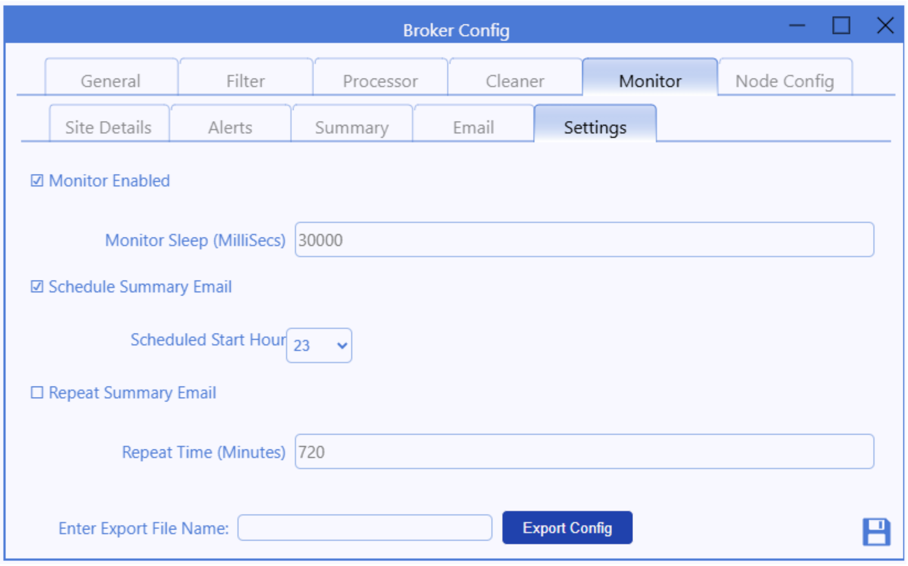
Password:

From Name: ReportBrokerMonitor

From Address: ReportBrokerMonitor@localhost

Enter Export File Name:

Figure 81: SMTP configuration for email delivery.



☑ Monitor Enabled

Monitor Sleep (MilliSecs): 30000

☑ Schedule Summary Email

Scheduled Start Hour: 23

☐ Repeat Summary Email

Repeat Time (Minutes): 720

Enter Export File Name:

Figure 82: Broker monitor scheduling settings.

When the network is active, the status of all broker sub-processes can be viewed using the feed icon located at the bottom-left corner of the broker node in the network.

5.4.16 DICOM to HL7 Node

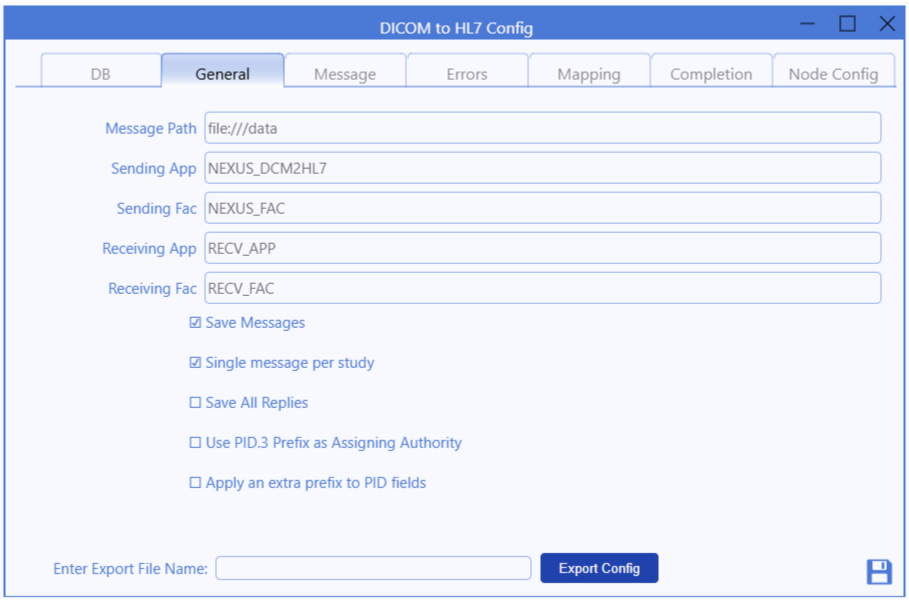
A node that generates an HL7 message based on a DICOM input.

The DICOM to HL7 Node enables the generation of HL7 messages from incoming DICOM objects. Demographic information in the DICOM header is used to populate the PID segment of the HL7 message. Additional message content can be configured using fixed values or extracted from other DICOM tags. One HL7 message is generated per study for general message creation.

The node maintains a database of processed studies and generated messages. The database is configured via the "DB" tab using a standard connection interface. Processed studies are recorded in the "StudyLevel" table, while message content and responses are stored in the "Messages" table. The Study UID is used as the unique identifier for processed studies.

Users may specify the database type, name, and login credentials. The host and port must be provided for externally hosted databases.

General message settings are configured in the "General" tab (Figure 83). These include the file path for storing messages, the sending and receiving application/facility identifiers, whether to store all replies or only failures, and an option to apply a prefix as the assigning authority in PID.2 and PID.3.

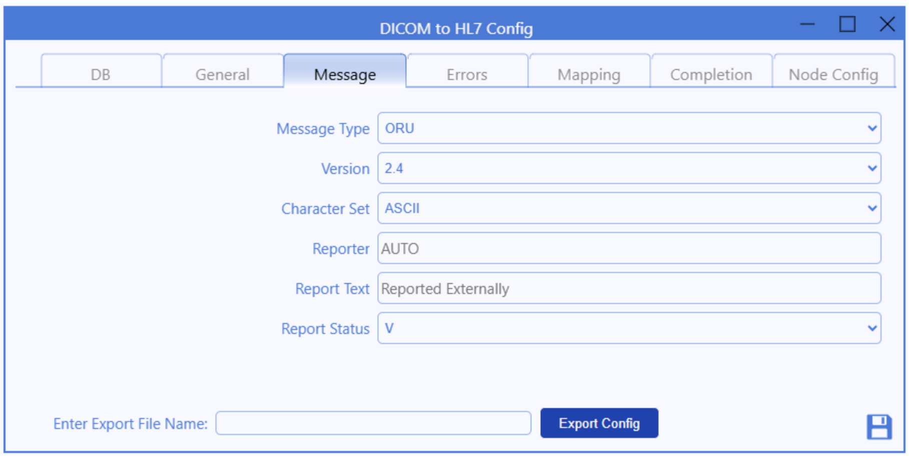


Message Path	file:///data
Sending App	NEXUS_DCM2HL7
Sending Fac	NEXUS_FAC
Receiving App	RECV_APP
Receiving Fac	RECV_FAC

☑ Save Messages
☑ Single message per study
☐ Save All Replies
☐ Use PID.3 Prefix as Assigning Authority
☐ Apply an extra prefix to PID fields

Figure 83: General node and HL7 message configuration.

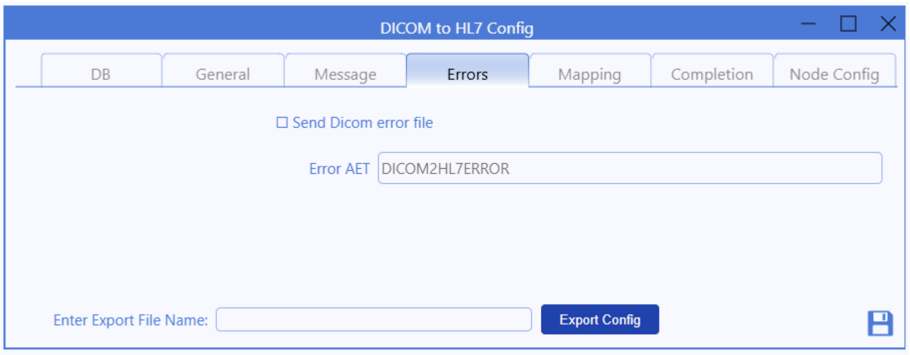
Message-specific settings are found on the "Message" tab (Figure 84). For ORU messages, the user can set the report status (OBR.25 and OBX.11), the principal result interpreter (OBR.32), and the report text (OBX.5). If the report text contains line breaks ('\n'), multiple OBX segments are used.



Message Type: ORU
Version: 2.4
Character Set: ASCII
Reporter: AUTO
Report Text: Reported Externally
Report Status: V
Enter Export File Name:

Figure 84: Message content and header configuration.

Error handling is configured in the "Error" tab (Figure 85). In the event of an error, a basic DICOM file is generated and passed through the output. The calling AET used in such cases can be set here, allowing for redirection through notification nodes if required. See the appendix for a worked network example.



☐ Send Dicom error file
Error AET: DICOM2HL7ERROR
Enter Export File Name:

Figure 85: DICOM error output configuration.

DICOM-to-HL7 field mappings are configured on the "Mapping" tab (Figure 86). Segment types are required to be the segment ID at the start of the segment being referenced. "Field" is also mandatory. All unspecified components default to 0. Only positive integers are valid.

The "DICOM" field includes a dropdown with common tags, or users can enter tags manually in short format (e.g., 10,20). Pressing the plus button validates and adds the mapping to the table. Mappings can be removed by selecting the row and clicking the trash icon.

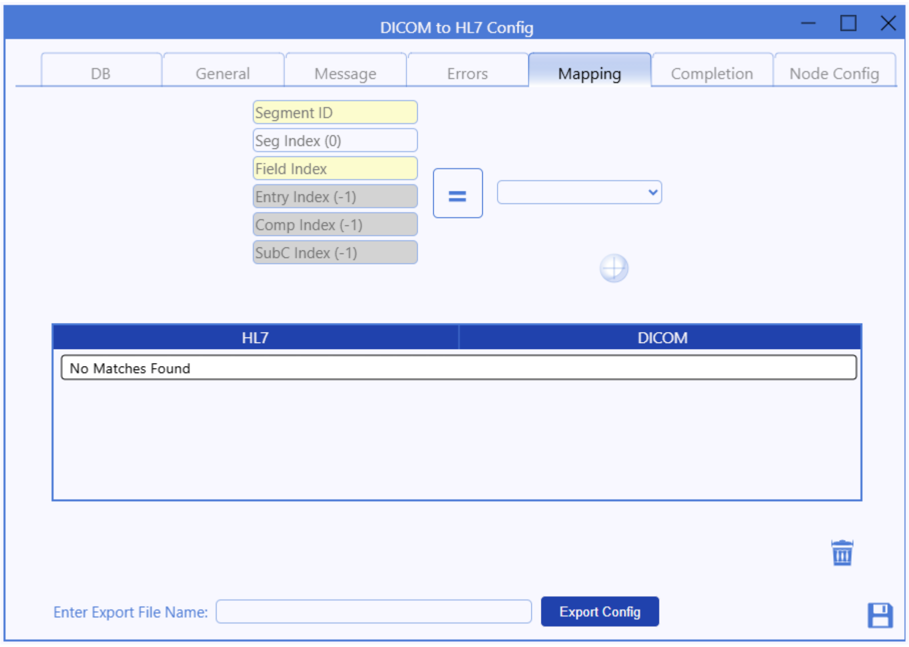


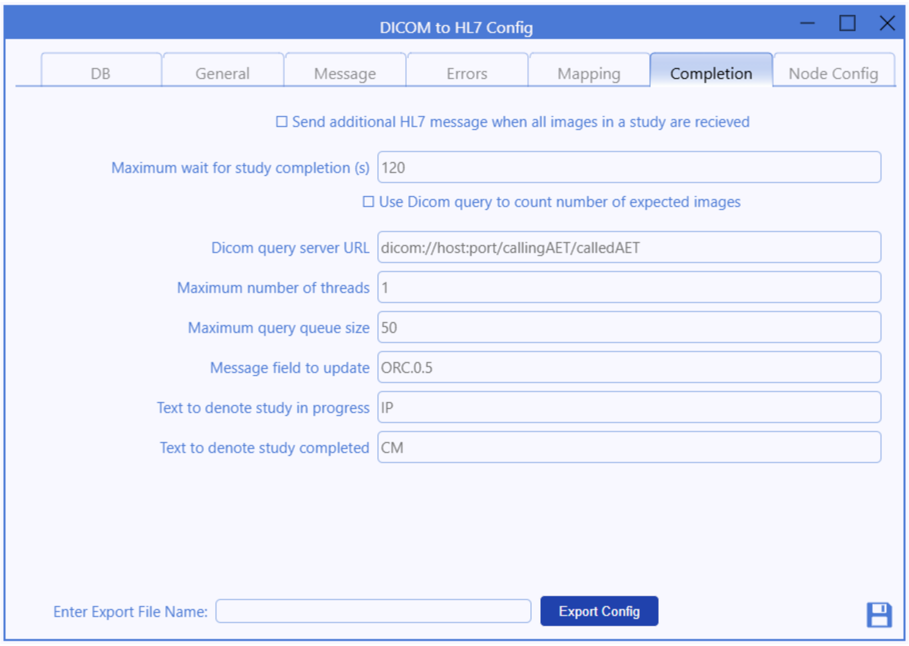
Figure 86: DICOM-to-HL7 mapping configuration.

Completion settings, shown in Figure 87, define when a follow-up HL7 message is sent after all images in a study are received. If the receiver is a Pukka-j Nexus service, the service will mark the study as complete in the “OrderManagement” database table.

To enable this, tick the checkbox for “Send additional HL7 message when all images in a study are received”. Then, specify the HL7 field to be updated (default: ORC.0.5) and the strings used for “in progress” (IP) and “complete” (CM) states.

By default, if no new images are received within 120 seconds after the last image, the study is considered complete. This timeout is configurable. Alternatively, you may use a DICOM query to check the expected number of images—tick “Use DICOM query to count number of expected images” and supply the DICOM server URL.

For high-throughput systems, you can increase the thread count and queue length (default: 50) to handle more concurrent study queries.


DICOM to HL7 Config

☐ Send additional HL7 message when all images in a study are recieved

Maximum wait for study completion (s)	120
	☐ Use Dicom query to count number of expected images
Dicom query server URL	dicom://host:port/callingAET/calledAET
Maximum number of threads	1
Maximum query queue size	50
Message field to update	ORC.0.5
Text to denote study in progress	IP
Text to denote study completed	CM

Figure 87: HL7 study completion message configuration.

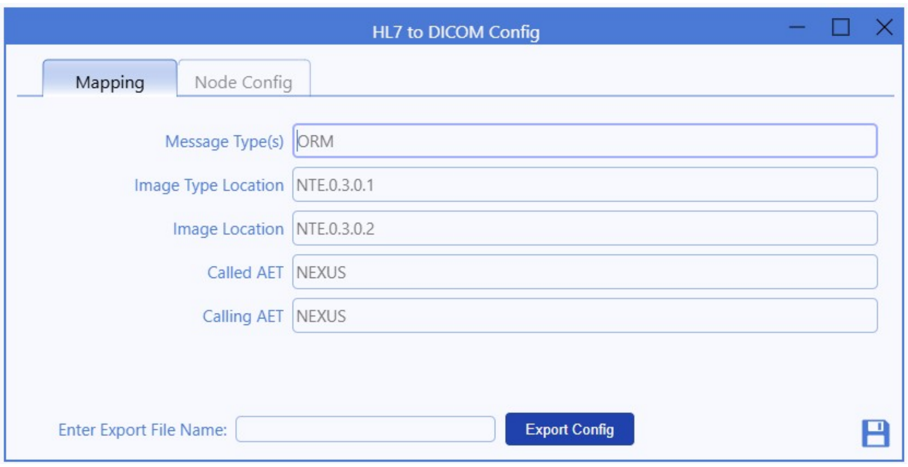
5.4.17 HL7 to DICOM Node

A node for converting Base64-encoded data within an HL7 message into a DICOM file.

The HL7 to DICOM Node extracts Base64-encoded bytes from a designated HL7 field and generates a corresponding DICOM file, which is output via the secondary output channel of the node.

If the data is a JPEG image, a DICOM Secondary Capture (SC) object is created with the JPEG as pixel data. If the data is a PDF file, an Encapsulated PDF DICOM object is created. Patient and study-level metadata are derived from standard HL7 segments—PID, ORC, and OBR—and are used to populate the DICOM header accordingly.

If a DICOM Study UID is present in the ZDS.1 field, it will be used in the output file. This ensures the resulting document is stored alongside related images on the destination system.


HL7 to DICOM Config

Message Type(s): ORM
Image Type Location: NTE.0.3.0.1
Image Location: NTE.0.3.0.2
Called AET: NEXUS
Calling AET: NEXUS
Enter Export File Name:

Figure 88: HL7 to DICOM tag mapping configuration.

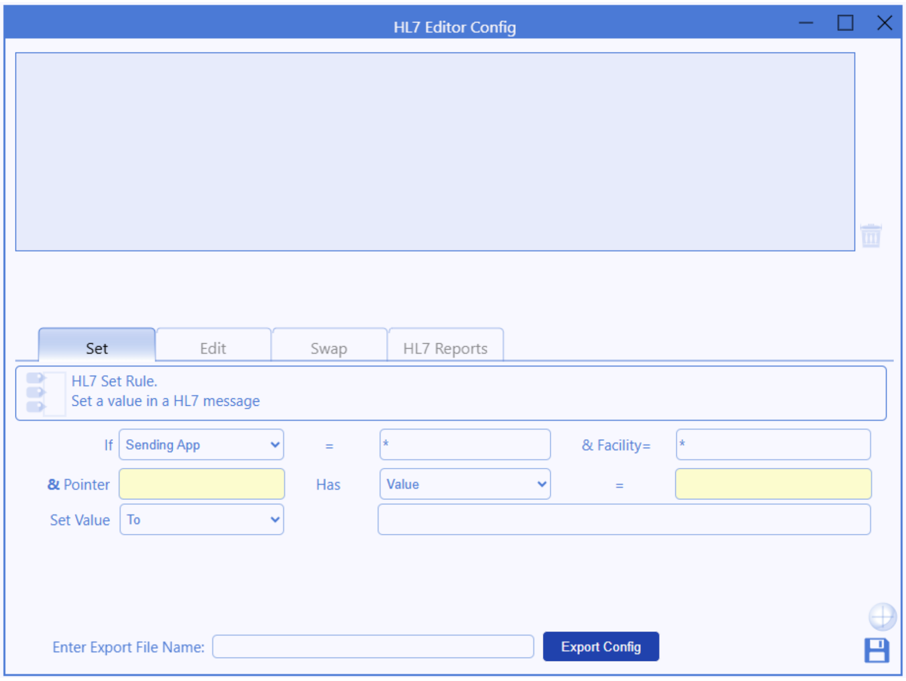
Figure 88 illustrates the configuration options available for this node. The supported HL7 message types can be defined here, with “ORM” being the default. Message types may be listed using any separator (e.g., ‘ORM,OMG‘ or ‘ORM OMG‘).

The node reads two HL7 field locations: one for the encoded image bytes and another for identifying the data type (e.g., ‘JPG‘, ‘PDF‘). By default, it reads the third field of the first NTE segment, interpreting the first component as the Base64 image data and the second as the data type.

5.4.18 HL7 Integration Engine Node

An HL7 Integration Node (HL7 Editor) used for operations involving the manipulation of HL7 message content.

The HL7 Editor is designed to apply specific modifications to an HL7 message as it traverses a processing pipeline. It supports four types of message manipulation—**Set**, **Edit**, **Swap**, and **HL7 Reports**—each accessible via a separate tab within the configuration dialogue, as shown in Figure 89.


HL7 Editor Config

Set | Edit | Swap | HL7 Reports

HL7 Set Rule.
Set a value in a HL7 message

If: Sending App = * & Facility= *

& Pointer: Has Value =

Set Value: To

Enter Export File Name:

Export Config

Figure 89: HL7 Editor – Set Rule configuration tab.

The **Set** operation assigns a specified HL7 pointer a value which may be:

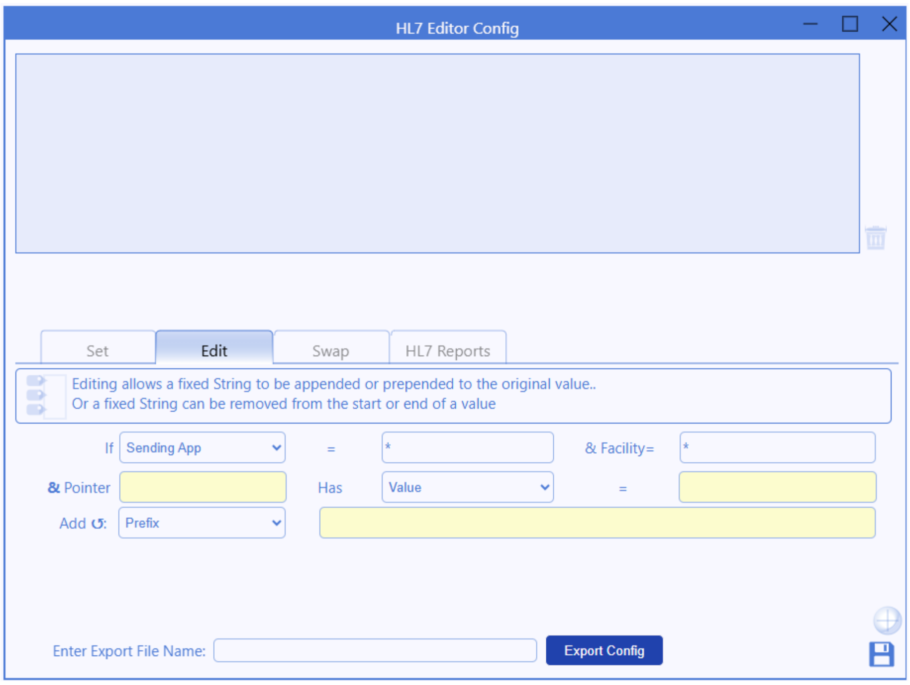
- A fixed value,
- The value of another HL7 pointer, or
- The first non-empty value from a list of HL7 pointers.

This assignment can be made conditional upon the sending or receiving application/facility or the current value of the pointer. Wildcards ('*') are permitted for matching AETs and values.

The **Edit** operation allows modification of an HL7 pointer's value through:

- Prefixing or suffixing the value,
- Replacing all or part of the original value using a specified pattern.

The toggle button next to the prefix drop-down switches between 'Add' and 'Remove', applicable for prefix or suffix modifications. Replace mode supports wildcard substitution using '*' to match and replace segments of the pointer value. Configuration for the Edit tab is shown in Figure 90.



Editing allows a fixed String to be appended or prepended to the original value..
Or a fixed String can be removed from the start or end of a value

If Sending App = * & Facility= *
& Pointer Has Value =
Add: Prefix

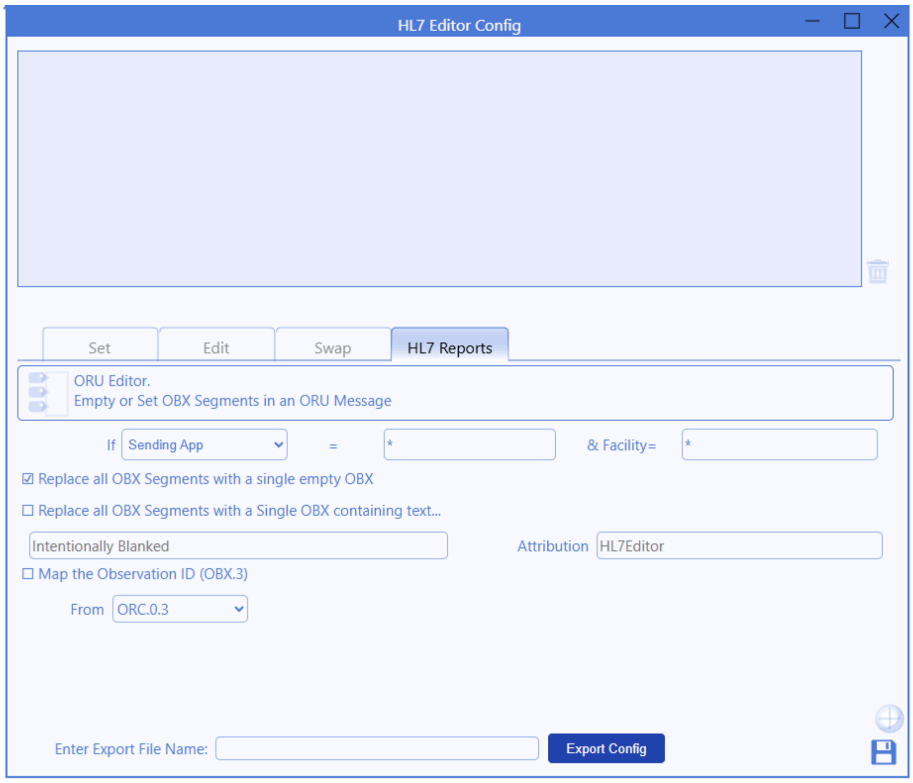
Enter Export File Name:
Export Config

Figure 90: Edit Rule configuration tab.

The **Swap** operation exchanges the values between two HL7 pointers. Like other options, this can be conditionally applied based on application, facility, or the existing value of the tag.

The **HL7 Reports** tab, shown in Figure 91, allows configuration of rules that modify OBX segments in the HL7 message. Rules can replace all OBX segments with a single, empty-content segment or with a segment containing a replacement value. These rules may be selectively applied based on the sending or receiving application/facility.

An optional mapping allows the Observation ID (OBX Field 3) to be sourced from a specific field, with the default set to field 3 of the first ORC segment, i.e., ‘ORC.0.3‘. The attribution field indicates the origin of the report content—typically the reporter’s name. For automated edits, this may be set to a default such as ‘AUTO-EDIT‘.


HL7 Editor Config

ORU Editor.
Empty or Set OBX Segments in an ORU Message

If Sending App = * & Facility= *

☑ Replace all OBX Segments with a single empty OBX

☐ Replace all OBX Segments with a Single OBX containing text...

Intentionally Blanked   Attribution HL7Editor

☐ Map the Observation ID (OBX.3)

From ORC.0.3

Enter Export File Name:

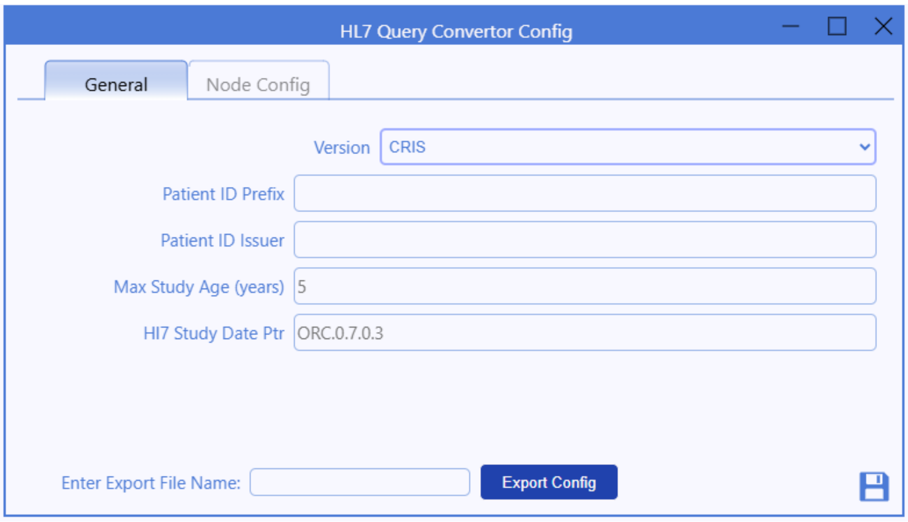
Figure 91: HL7 Reports configuration tab.

5.4.19 HL7 Query Convertor Node

An HL7 Query Convertor Node used for converting HL7 RSP message content.

The HL7 Query Convertor Node is designed to transform a stream of HL7 RSP messages—typically received in response to a QBP query sent to a RIS—into a series of standard report and order messages. This is particularly useful when external systems do not support HL7 RSP messages but require the information they contain in a different format.

Figure 92 illustrates the configuration interface, where the RIS manufacturer is selected. This selection determines how the incoming RSP messages are parsed and interpreted.


HL7 Query Convertor Config

General | Node Config

Version: CRIS

Patient ID Prefix:

Patient ID Issuer:

Max Study Age (years): 5

Hl7 Study Date Ptr: ORC.0.7.0.3

Enter Export File Name:

Figure 92: Configuration for converting HL7 RSP messages into a stream of ORUs and ORMs.

The **Patient ID Prefix** field allows a prefix to be added to the patient ID in the generated output. Additionally, the **Patient ID Issuer** field enables the user to specify a preferred issuer. As RSP PID segments may contain multiple patient IDs from different issuers, this field ensures that the ID for the selected issuer is placed first in the list in the output message.

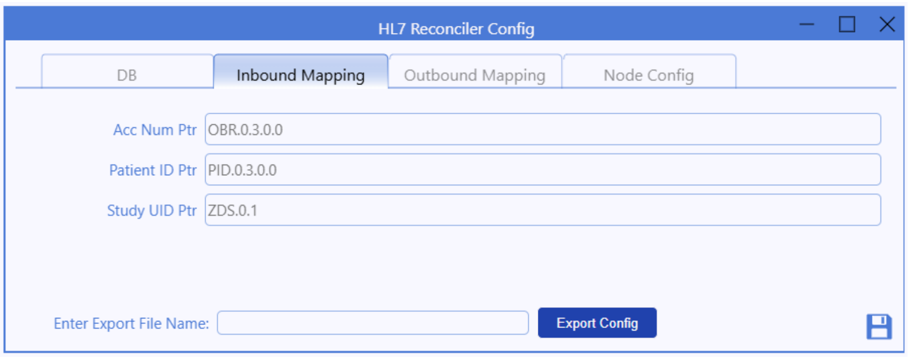
RSP messages can contain extensive historical data, some of which may not be relevant. To filter out outdated records, the **Maximum Study Age** field defines the upper age limit for results to be retained. The **Study Date Pointer** field specifies the HL7 location from which the study date is extracted. Studies and reports older than the configured maximum age will be excluded from the output stream.

5.4.20 HL7 Reconciler Node

An HL7 Reconciler Node used to augment HL7 message content based on order or report details retrieved from a database.

When generating an order from a DICOM file, the information included in the HL7 message created by the DICOM to HL7 Node is inherently limited to the contents of the original DICOM header. The HL7 Reconciler Node addresses this limitation by performing a lookup against a database to retrieve additional information for the corresponding study or accession number. This allows extra fields to be populated in the HL7 message to enrich the order or report context.

Figure 93 shows the configuration interface where the node is instructed how to extract primary identifiers from the incoming HL7 message. These identifiers are then used to query the database for more comprehensive information.


HL7 Reconciler Config

DB | Inbound Mapping | Outbound Mapping | Node Config

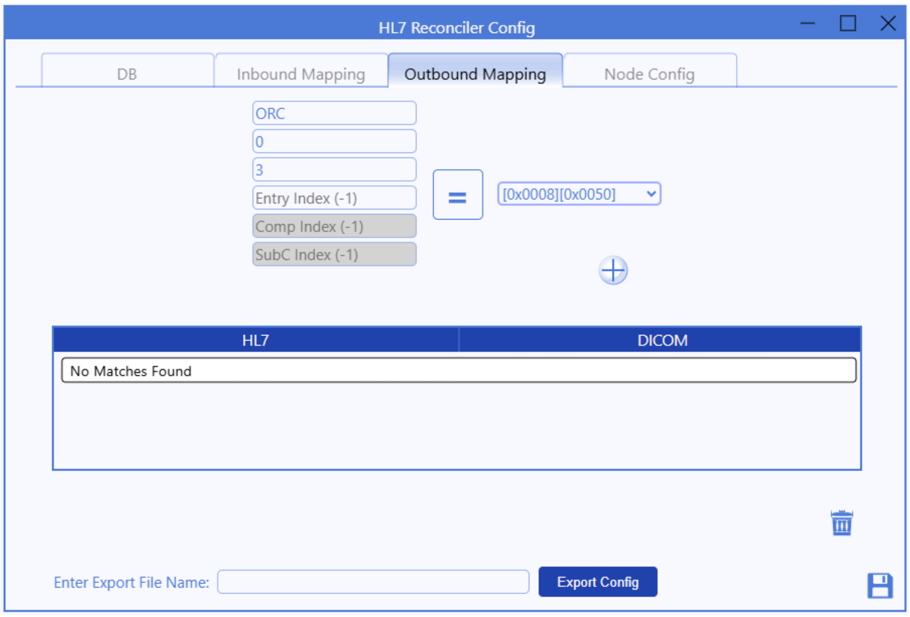
Acc Num Ptr: OBR.0.3.0.0
Patient ID Ptr: PID.0.3.0.0
Study UID Ptr: ZDS.0.1

Enter Export File Name:

Figure 93: Configuration for extracting primary IDs from HL7 messages in the HL7 Reconciler Node.

Where available, Study UID is used preferentially to locate the relevant record in the `OrderTable` of the configured database. The node uses a standard database configuration panel to define the database type, host, and access credentials. If the Study UID is not present in the HL7 message, the combination of Patient ID and Accession Number will instead be used for the lookup.

Once the relevant data is retrieved from the database, Figure 94 illustrates how the node can be configured to map database fields to specific HL7 pointer locations, thereby injecting the additional information directly into the HL7 message.


HL7 Reconciler Config

DB | Inbound Mapping | Outbound Mapping | Node Config

ORC
0
3
Entry Index (-1)
Comp Index (-1)
SubC Index (-1)

= [0x0008][0x0050]

HL7	DICOM
No Matches Found	

Enter Export File Name:

Figure 94: Mapping configuration for inserting retrieved data into HL7 message fields.

The destination location in the HL7 message is defined using HL7 segment and field pointers (e.g., `ORC.0.3`). The corresponding source field from the database is selected from a dropdown menu on the right. In the illustrated example, the Accession Number is being mapped into the `ORC.3` field of the HL7 message.

This mapping capability allows the HL7 Reconciler Node to enhance message completeness and ensure that downstream systems receive more accurate and detailed data than is available from the DICOM file alone.

5.4.21 Alert Node

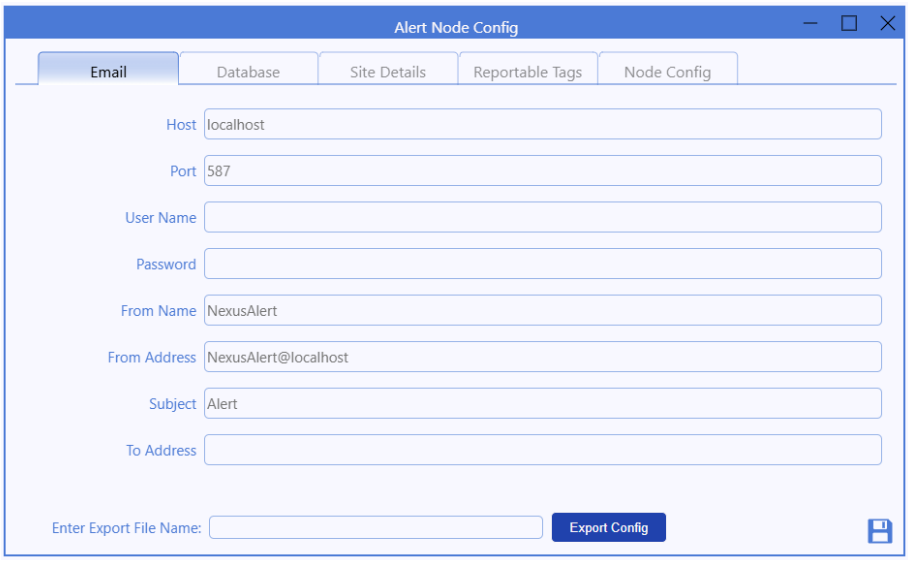
A node that sends a notification per study when data passes through it.

The Alert Node (Notification Node) is used to send an email message to a specified address whenever data passes through it. When combined with a splitter, this allows users to be notified when data matching a configured DICOM specification is detected. This functionality is particularly useful for identifying and alerting users to potential errors or inconsistencies in received data.

The node communicates notifications via email. If multiple recipients need to be notified, the destination address can be configured to use an email relay, which then distributes the message to the appropriate users.

SMTP configuration is required to enable the node to send emails. These settings should be provided according to the requirements of the SMTP service in use. In many cases, the destination and sending name fields may need to be left blank, although this will depend on the configuration of the mail server.

The subject line of the email can be customised to reflect the nature of the notification. The configuration for the email parameters is illustrated in Figure 95.



Host	localhost
Port	587
User Name	
Password	
From Name	NexusAlert
From Address	NexusAlert@localhost
Subject	Alert
To Address	

Figure 95: Email configuration for the Alert Node.

The Alert Node also uses a database to track the emails that are pending or have been sent. The database keeps a record of the studies (by UID) that have already triggered a notification, preventing duplicate messages. Database configuration is handled using the standard database setup interface.

In environments with multiple Alert Nodes, either across different sites or within the same network, it may be useful to include a site name or site ID in the notification. This can be configured via the Site Details tab. By using placeholders `%SITE_CODE%` and `%SITE_NAME%` in the subject line, these values will automatically be substituted when the message is sent.

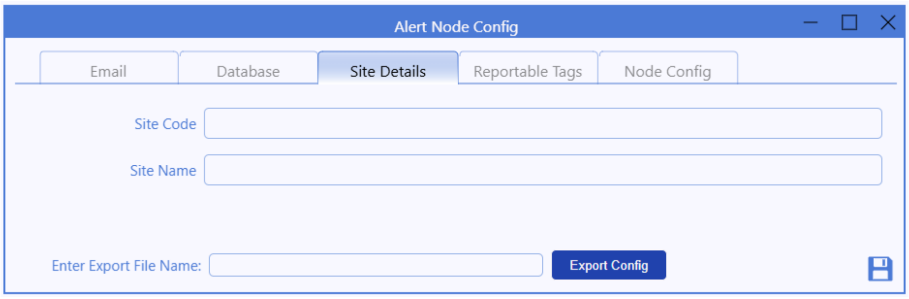
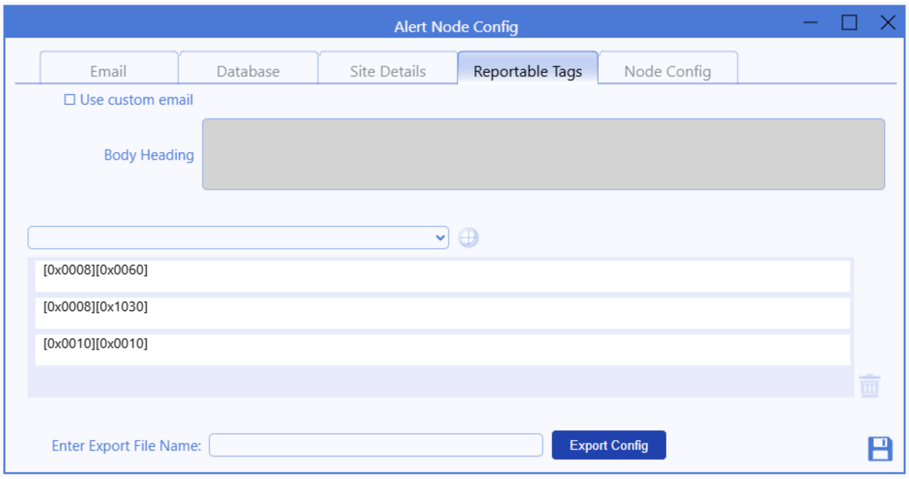


Figure 96: Site code configuration for the Alert Node.

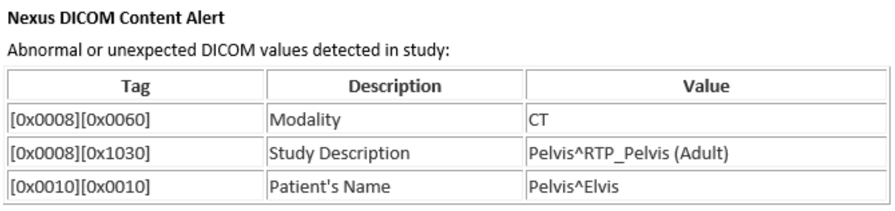
The final configuration tab enables the inclusion of specific DICOM tag content in the body of the notification email. Tags can be added to the list, and their corresponding values will be included in the generated message. Users may also define a fully customised email template by enabling the "Use custom email" option and populating the "Body Heading" field. Custom email templates may include standard HTML formatting for advanced presentation.



[0x0008][0x0060]
[0x0008][0x1030]
[0x0010][0x0010]

Figure 97: DICOM tag configuration for email notifications.

For example, including the modality, study description, and patient name tags could produce an email such as the one shown in Figure 98.



Nexus DICOM Content Alert

Abnormal or unexpected DICOM values detected in study:

Tag	Description	Value
[0x0008][0x0060]	Modality	CT
[0x0008][0x1030]	Study Description	Pelvis^RTP_Pelvis (Adult)
[0x0010][0x0010]	Patient's Name	Pelvis^Elvis

Figure 98: Email output with included DICOM tag content.

5.4.22 Image Transform Node

Used to rotate or flip the pixel data as it passes through the pipeline.

The Image Transform Node is used to rotate or flip the pixel data within DICOM files. There are typically eight operations available (including "No Operation"), as listed below:

ID	Transform
0	No flip, no rotation
1	Rotate 90°
2	Rotate 180°
3	Rotate 270°
4	Horizontal flip about the central Y-axis
5	Rotate 90° and flip
6	Rotate 180° and flip
7	Rotate 270° and flip

The transform operation is applied by first rotating the image, followed by an optional horizontal flip.

Figure 99 shows the configuration dialogue for the Image Transform Node. As the pixel data is modified, it is recommended that a new SOP Instance UID is assigned to the image to reflect the change. The node includes an option to automatically assign new UIDs to all files processed through it.

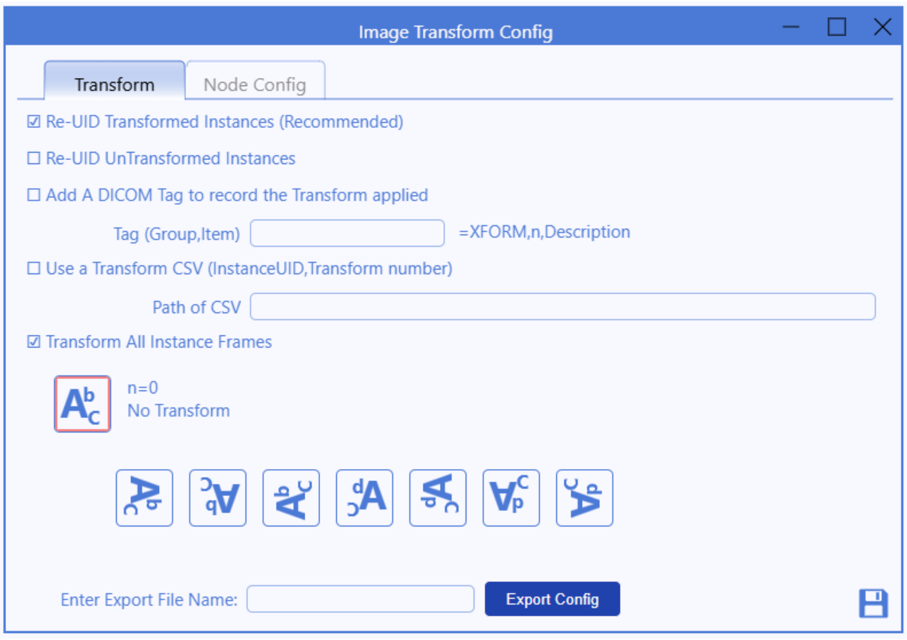


Image Transform Config

Transform | Node Config

☑ Re-UID Transformed Instances (Recommended)

☐ Re-UID UnTransformed Instances

☐ Add A DICOM Tag to record the Transform applied

Tag (Group,Item) =XFORM,n,Description

☐ Use a Transform CSV (InstanceUID,Transform number)

Path of CSV

☑ Transform All Instance Frames

n=0
No Transform

Enter Export File Name: Export Config

Figure 99: Image Transform configuration dialogue.

To document the transformation in the DICOM header, users can specify a DICOM tag where the transformation details will be recorded as a text string, for example: `XFORM,1,Rotate 90`.

There are two modes of operation for the transform node:

1. **CSV-based selective transformation** — The node references a CSV file containing SOP Instance UIDs and associated transformation types. As each image passes through, its UID is checked against the list. If a match is found, the corresponding transformation is applied; otherwise, the image is passed through unchanged.

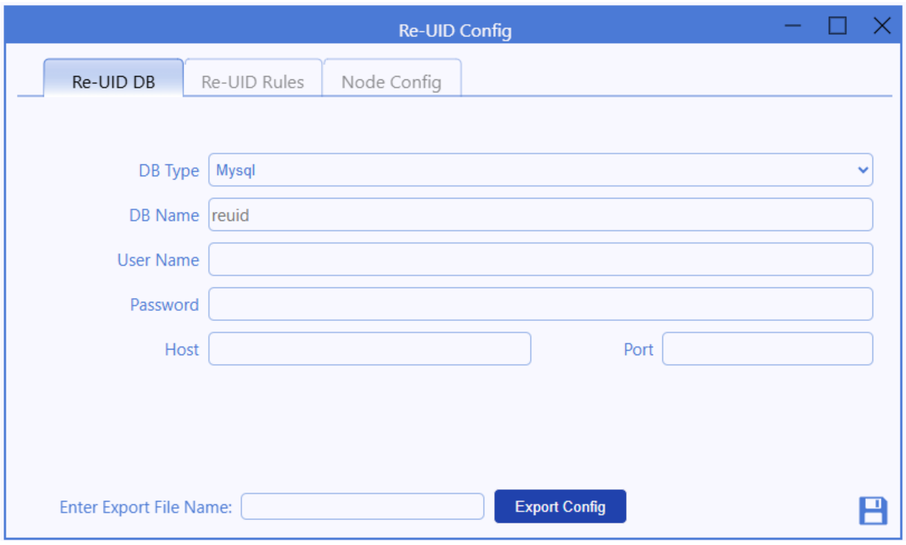
2. **Uniform transformation** — The node applies the same transformation to all files. This mode is useful for correcting known misalignments, such as rotating all images in a series by 180°.

5.4.23 DICOM Re-UID Node

The DICOM Re-UID Node is used to consistently replace UIDs in DICOM files across a series or study.

The DICOM Re-UID Node is used to replace any Universal Unique Identifiers (UIDs) in a DICOM file. By leveraging a database, the node retains a reference to previously modified UIDs, enabling consistent re-identification while preserving the structural relationships between study, series, and instance UIDs.

Figure 100 shows the standard database configuration for the Re-UID Node. The database maintains a record of mappings in a table named `reUidTable`, where new UIDs are associated with the original UIDs and the corresponding patient IDs.



DB Type	Mysql
DB Name	reuid
User Name	
Password	
Host	
Port	

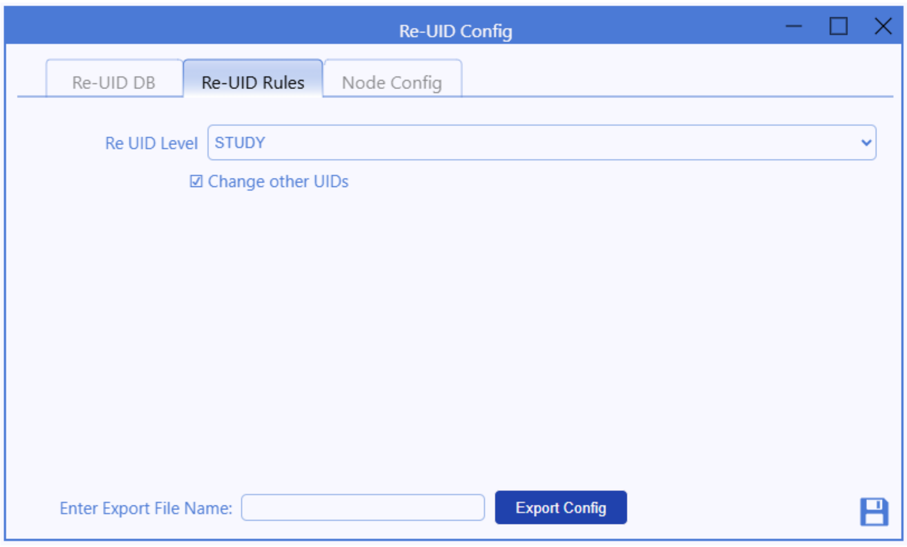
Figure 100: DICOM Re-UID Node database configuration.

By default, the node modifies all primary and reference UIDs within the DICOM file. Certain fields, such as SOP Class UIDs, are not modified. The re-UID configuration dialogue, shown in Figure 101, allows customisation of UID replacement behaviour.

The primary UID replacement scope is selected using the drop-down menu:

- **STUDY** (default): changes Study, Series, and Instance UIDs. - **SERIES**: changes only Series and Instance UIDs. - **IMAGE**: changes only the Instance UID.

In addition to the primary UIDs, there may be other UIDs present (e.g., Frame of Reference UID). These can be included in the re-identification process using the "Change other UIDs" tick box. This is enabled by default. If unticked, only the UIDs selected via the primary level drop-down will be modified.



Re UID Level: STUDY

☑ Change other UIDs

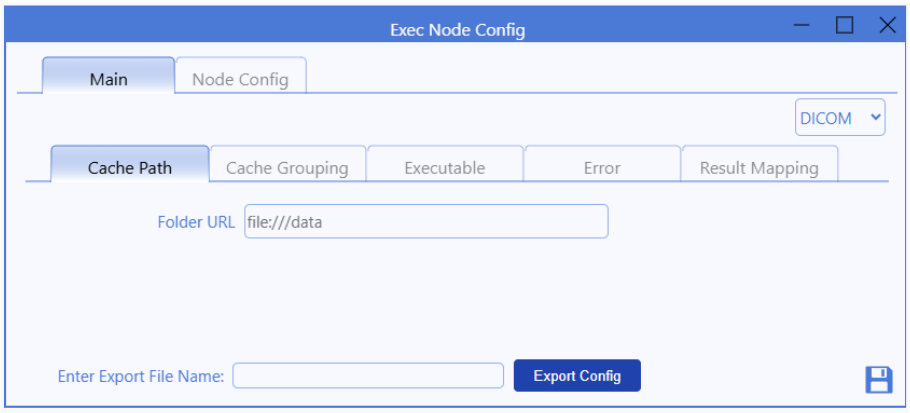
Enter Export File Name:

Figure 101: DICOM Re-UID rules configuration.

5.4.24 Exec Node

A node that invokes an external process or script when data passes through it. Results may be returned and forwarded downstream.

The Exec Node enables the invocation of external processes on the host operating system via a script or executable, providing flexible extension capabilities to the Nexus environment.



Folder URL file:///data

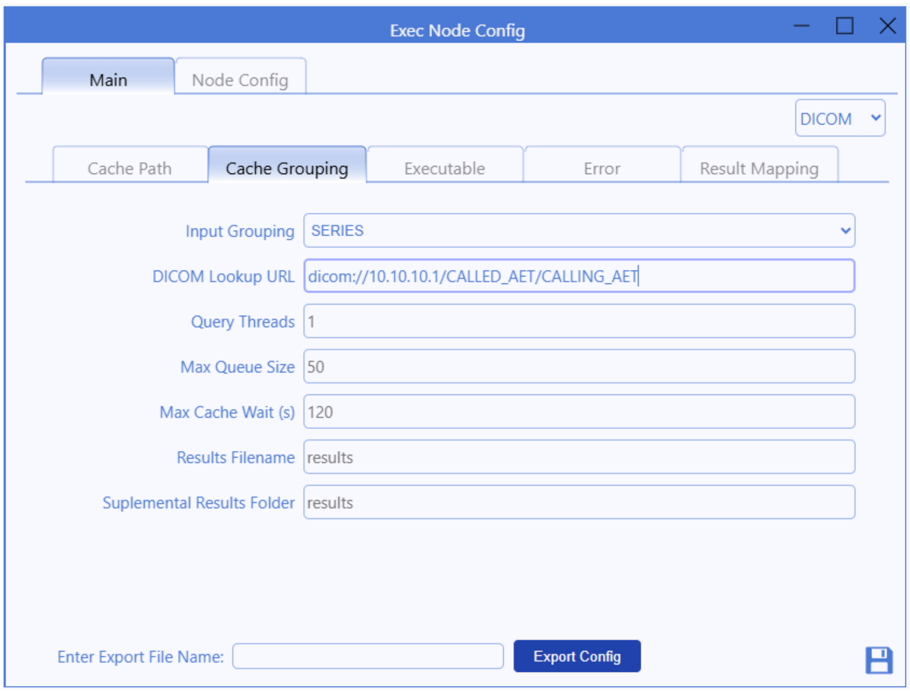
Enter Export File Name:

Figure 102: DICOM file output configuration for caching inbound data for external processing.

When a DICOM file is processed by the Exec Node, it is written to disk using an embedded DICOM File Writer Node, as configured in Figure 102. The path to the file is passed to the external process specified in the “Executable” tab (see Figure 104). If the external process requires all images for a given study or series before execution, grouping can be configured (Figure 103).

In this grouped mode, a DICOM C-FIND query is used to determine the expected number of images. A monitor tracks the number of incoming files. Once the expected number is received—or if a maximum wait timeout elapses—the process is triggered. Key configuration fields include:

- **Max Cache Wait (s)**: Maximum time to wait before processing if not all expected images arrive. - **Query Threads**: Number of concurrent threads for the C-FIND query. - **Max Queue Size**: Maximum number of studies queued for monitoring. - **Results Filename**: Name of the file expected to be produced by the external script. - **Supplemental Results Folder**: Directory for any secondary files (e.g., heatmaps, DICOM SCs) to be forwarded downstream.

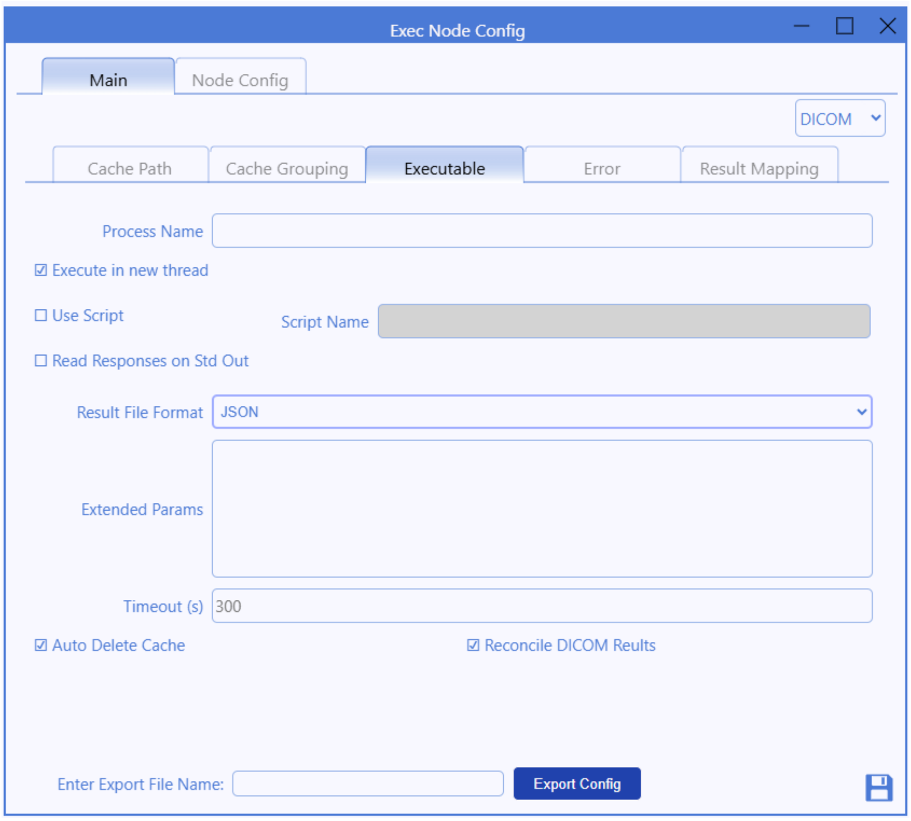


Input Grouping: SERIES
DICOM Lookup URL: dicom://10.10.10.1/CALLED_AET/CALLING_AET
Query Threads: 1
Max Queue Size: 50
Max Cache Wait (s): 120
Results Filename: results
Suplemental Results Folder: results
Enter Export File Name:

Figure 103: Cache grouping configuration.

The external command may be a script interpreter (e.g., `cmd.exe`, `powershell.exe`) or a direct call to a compiled executable. Output may be read from standard output, or from a generated response file (text or JSON). If a response file is used, it should share the same filename as the input DICOM but with a `.txt` or `.json` extension. Additional parameters can be passed using the “Extended Params” input—each on a new line. Resulting DICOM files placed in a `results` subdirectory will be picked up and forwarded.

Selecting “Delete Cache” will remove all cached files post-execution. If “Reconcile Results” is ticked, DICOM headers in output files will be updated with original patient and study information.


Exec Node Config

Main | Node Config

DICOM

Cache Path | Cache Grouping | Executable | Error | Result Mapping

Process Name:

☑ Execute in new thread

☐ Use Script — Script Name:

☐ Read Responses on Std Out

Result File Format: JSON

Extended Params:

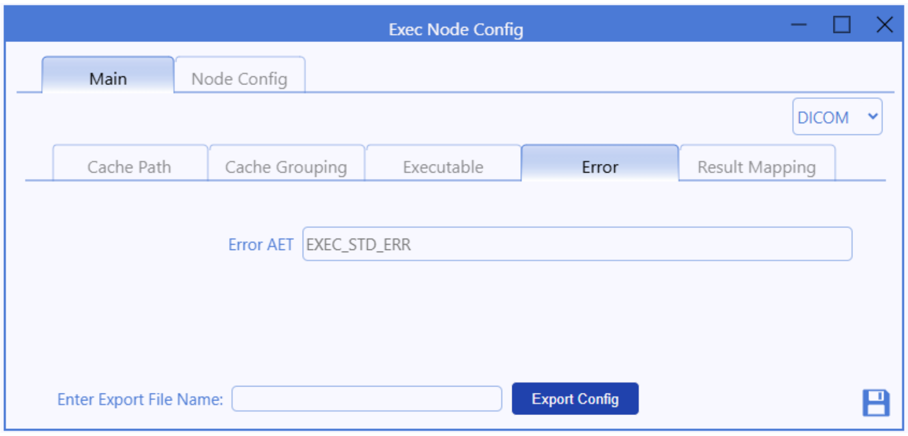
Timeout (s): 300

☑ Auto Delete Cache — ☑ Reconcile DICOM Reults

Enter Export File Name: — Export Config

Figure 104: External executable parameters configuration.

If an error occurs during execution, the DICOM file and any error data can be passed along using a separate AE Title ("Error AET"), allowing it to be routed to an alternative node (e.g., a Notification Node).


Exec Node Config

Main | Node Config

DICOM

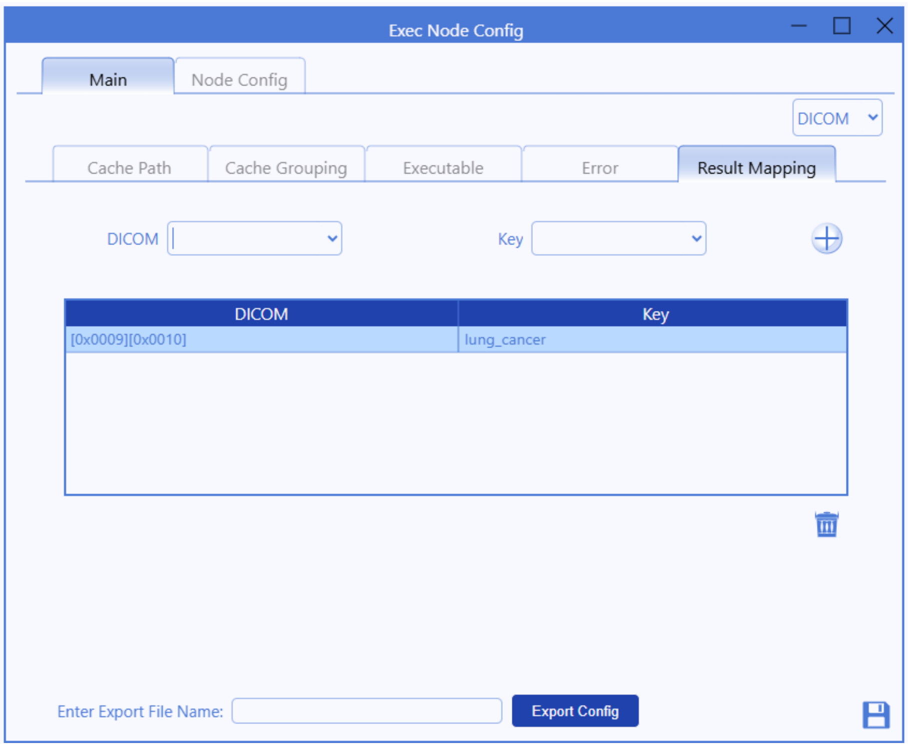
Cache Path | Cache Grouping | Executable | Error | Result Mapping

Error AET: EXEC_STD_ERR

Enter Export File Name: — Export Config

Figure 105: Error AE Title configuration.

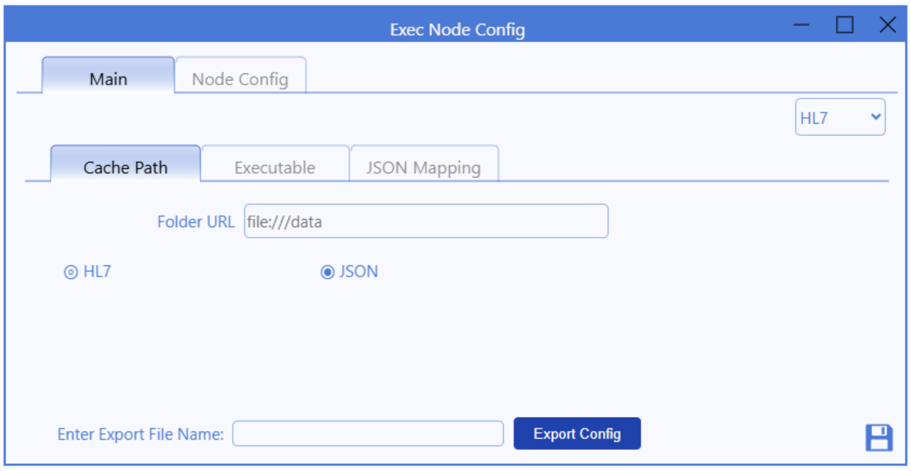
Returned results can be parsed and inserted into DICOM headers using key-to-tag mapping. For plain text files, a key-value separator must be defined. For instance, Figure 106 shows a value associated with the key "lung_cancer" being stored in private tag `[0x0009][0x1010]`.



DICOM	Key
[0x0009][0x0010]	lung_cancer

Figure 106: Mapping result keys to DICOM tags.

The Exec Node also supports execution triggered by HL7 messages. In this case, the HL7 content may be cached as the raw message or transformed into a JSON object, with HL7 fields mapped to key-value pairs. Figure 107 shows the HL7 cache configuration, and Figure 109 shows JSON mapping.

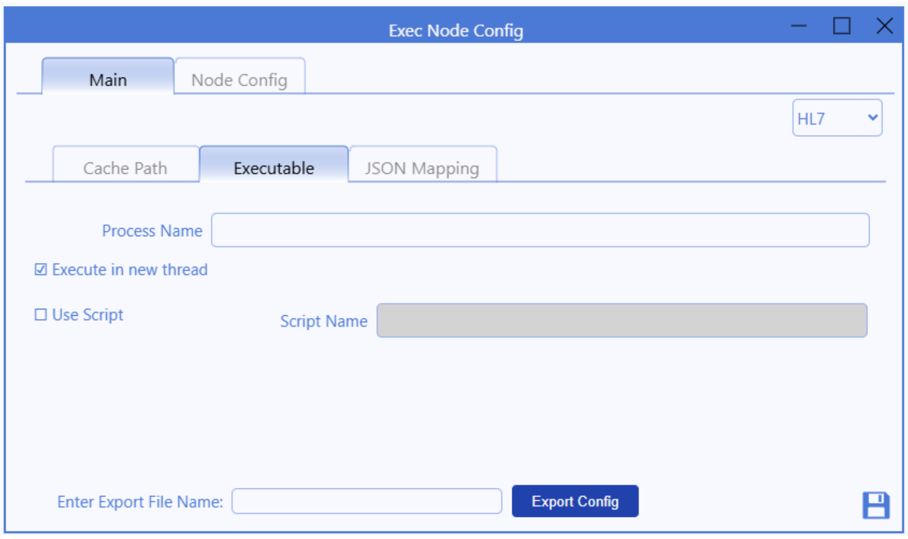


Folder URL: file:///data

HL7 [✓ JSON]

Figure 107: HL7 cache path configuration.

HL7-triggered executions are simpler than those for DICOM and do not expect a result file. The process and execution style are configured similarly, as shown in Figure 108.



Process Name

☑ Execute in new thread

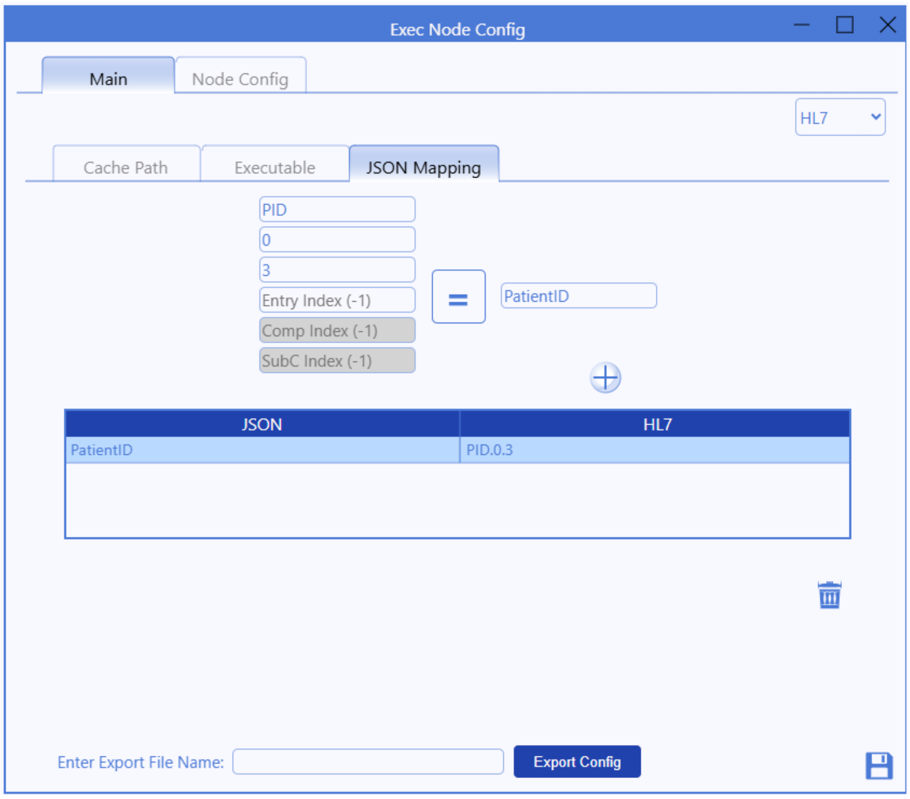
☐ Use Script

Script Name

Enter Export File Name:

Figure 108: HL7 execution configuration.

HL7 to JSON mapping is configured on the final tab. Here, HL7 pointer locations are mapped to custom JSON keys. For example, `PID.0.3.0` might be mapped to the key “PatientID”, as illustrated in Figure 109.



PID

0

3

Entry Index (-1)

Comp Index (-1)

SubC Index (-1)

= PatientID

JSON	HL7
PatientID	PID.0.3

Enter Export File Name:

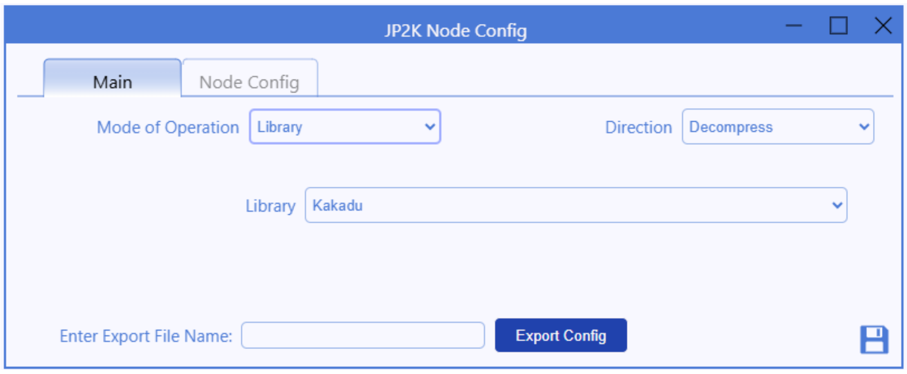
Figure 109: HL7 to JSON output mapping.

5.4.25 JP2K Node

The JP2K node compresses and decompresses the pixel data within a DICOM file using the JPEG2000 format.

The JP2K node is used to compress or decompress pixel data to or from the JPEG2000 format within a DICOM file. The node offers two modes of operation: **Library Mode** and **Executable Mode**.

Library Mode provides integrated decompression capabilities directly within the Nexus platform. This requires a licensed library and is suitable for seamless, embedded processing. Configuration options for this mode are shown in Figure 110.


JP2K Node Config

Main | Node Config

Mode of Operation: Library

Direction: Decompress

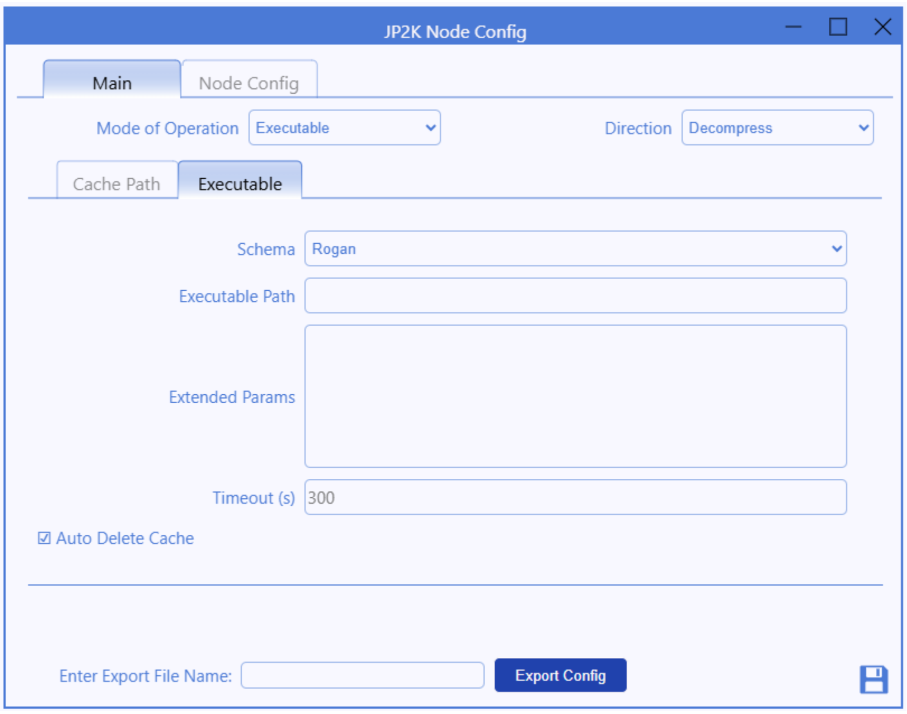
Library: Kakadu

Enter Export File Name:

Export Config

Figure 110: Configuration options for Library Mode.

Executable Mode enables JPEG2000 decompression via a user-supplied executable. This approach is ideal for organisations that already have established codecs or external image processing utilities, as it avoids the need to implement a bespoke processing node. Figure 111 shows the relevant configuration options.


JP2K Node Config

Main | Node Config

Mode of Operation: Executable

Direction: Decompress

Cache Path | Executable

Schema: Rogan

Executable Path:

Extended Params:

Timeout (s): 300

☑ Auto Delete Cache

Enter Export File Name:

Figure 111: Configuration options for Executable Mode.

Depending on your setup and specific requirements, additional third-party libraries or executables may be required to support the chosen mode of operation.

5.4.26 Gzip Node

Compress pixel data using Gzip for network transfer of single-frame images.

The Gzip node is used to compress the pixel data in single-frame DICOM images for efficient network transmission. When a DICOM file passes through this node, the pixel data is compressed using the Gzip algorithm, and the resulting compressed byte stream replaces the original data in the pixel data tag.

In order to distinguish the compressed file format, the SOP Class UID of the DICOM object is updated to a proprietary value. This enables the file to be transmitted via a DICOM output node in its compressed form.

It is important to note that the output from this node is not compatible with standard DICOM systems and is only intended to be used within a Nexus pipeline. The receiving system must use the corresponding Gunzip node to decompress and restore the original image data.

The Gzip node supports only single-frame DICOM objects and cannot be used with already compressed SOP Classes (e.g. JPEG or JPEG2000 encoded images). It works exclusively with uncompressed formats such as Implicit or Explicit Little Endian.

There are no user-configurable settings for this node.

5.4.27 Gunzip Node

Decompress Gzip-compressed pixel data within a DICOM file.

The Gunzip node reverses the operation of the Gzip node by decompressing the pixel data and restoring it to its original byte array form in the pixel data tag. It also resets the SOP Class UID to its original value, thereby restoring full DICOM compatibility for further processing or archival.

This node is intended to be used as the receiving counterpart to the Gzip node within a Nexus workflow. It ensures that DICOM files, which were temporarily compressed for internal network transfer, are correctly reverted to their standard uncompressed format before being processed by downstream systems.

There are no user-configurable settings for this node.

5.4.28 Nuclear Medicine Fixup Node

Fixes issues in Nuclear Medicine DICOM headers for compatibility.

The Nuclear Medicine (NM) Fixup Node is used to correct known DICOM non-conformities typically found in legacy Nuclear Medicine data. These issues often stem from improper conversions from older systems, such as ADAC, that do not conform fully to the DICOM standard.

This node automatically applies internal correction routines to amend attributes such as frame vectors and values referenced by the `Frame Increment Pointer`. These adjustments ensure improved compatibility and consistency across the dataset.

There are no user-configurable options for this node; all corrections are applied based on built-in logic addressing known DICOM compliance issues.

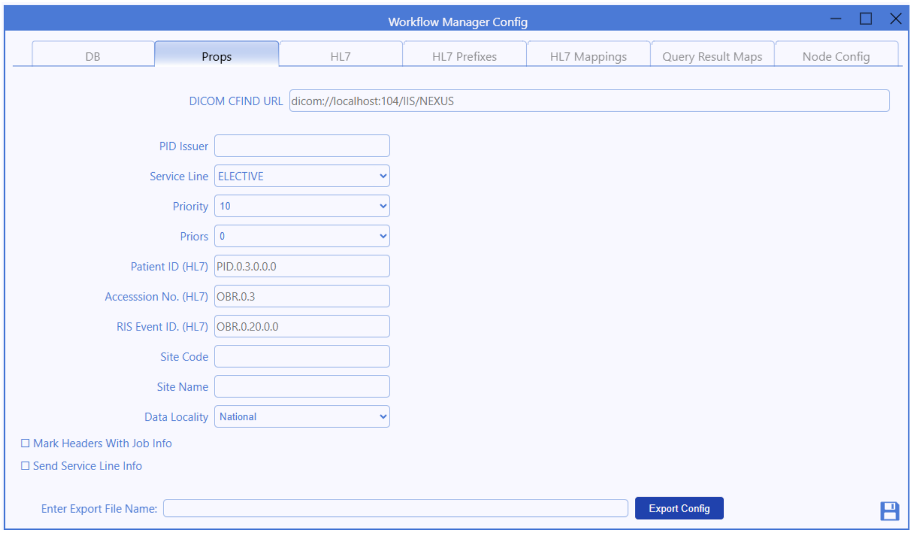
5.4.29 Workflow Management Node

Drive workflow processes from incoming DICOM files and HL7 messages.

The Workflow Management Node is used to drive workflow processes based on incoming DICOM files and HL7 messages. It operates in conjunction with the Workflow Service and serves multiple functions. The first of these is to trigger the creation of “Workflow Jobs” when data arrives. This process can automate the retrieval of prior imaging for patients based on primary DICOM files processed by this node. The import of primary studies may also be initiated when the node receives a HL7 message. Furthermore, the node can generate HL7 messages to indicate the receipt of DICOM files and can function as a CFIND/CMOVE handler. It communicates with the Workflow Service via a standard database connection.

Figure 112 shows the main configuration panel for the Workflow Node. The concept of a "Service Line" is employed to differentiate how data is handled—such as the number of prior studies to retrieve. The "Service Line" entry must match the rules configured in the Workflow Service. "Priority" defines the level of urgency for job creation, ranging from 1 (lowest) to 10 (highest). Different priorities can be handled using separate Workflow Management Nodes. The "Priors" setting specifies how many prior studies should be fetched for the patient. Rules within the Workflow Service define how the most relevant priors are selected.

HL7 properties are used to determine where the Patient ID, Accession Number, and RIS ID reside in the HL7 message. "Site Code" and "Site Name" may be set to add job identifiers. The "Data Locality" option flags the job for whether the associated data may be sent across locations. If CFIND interception is configured, the "DICOM CFIND URL" defines the remote server for query forwarding and result modification.



DICOM CFIND URL	dicom://localhost:104/IIS/NEXUS
PID Issuer	
Service Line	ELECTIVE
Priority	10
Priors	0
Patient ID (HL7)	PID.0.3.0.0.0
Accesssion No. (HL7)	OBR.0.3
RIS Event ID. (HL7)	OBR.0.20.0.0
Site Code	
Site Name	
Data Locality	National

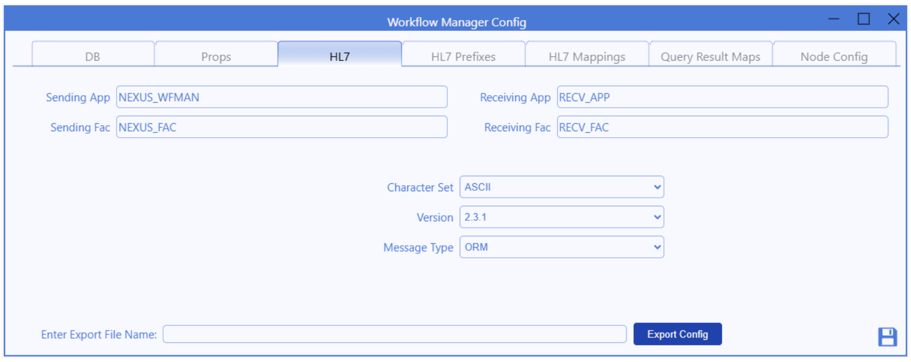
☐ Mark Headers With Job Info

☐ Send Service Line Info

Enter Export File Name:

Figure 112: Primary workflow node configuration for service line options and CFIND interception.

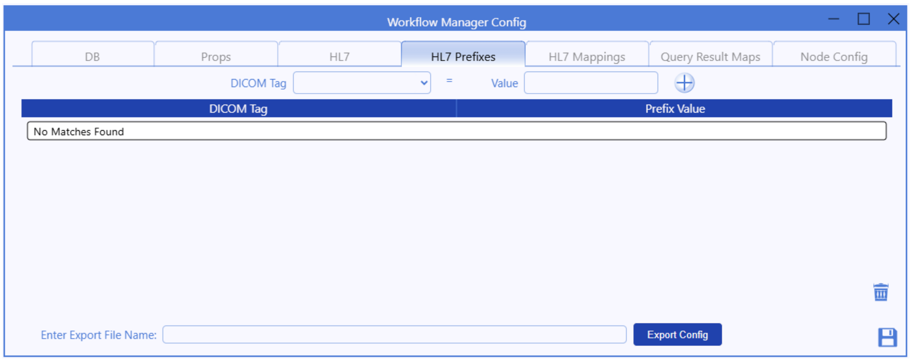
When unsolicited DICOM files are received, the node can generate HL7 messages to notify external systems. Figure 113 displays the basic HL7 connection and version settings. In order for HL7 messages to be forwarded, a time dependent channel out of the bottom of the Workflow Node should be created down which the generated messages will be sent and from there they can be sent on using a HL7 Output Node.



Sending App: NEXUS_WFMAN
Receiving App: RECV_APP
Sending Fac: NEXUS_FAC
Receiving Fac: RECV_FAC
Character Set: ASCII
Version: 2.3.1
Message Type: ORM
Enter Export File Name:

Figure 113: Basic HL7 output configuration.

For HL7 message generation, DICOM header values may need to be prefixed to indicate data origin. Figure 114 shows configuration options to define which DICOM tags should be prefixed and what prefixes should be applied.

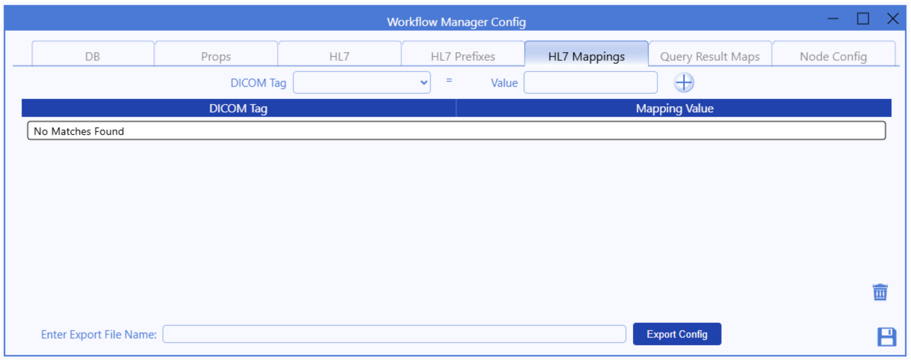


DICOM Tag	Prefix Value
No Matches Found	

Enter Export File Name:

Figure 114: Configuration options for prefixing DICOM tag values used in HL7 messages.

The “HL7 Mappings” tab, shown in Figure 115, lets users assign fixed values to DICOM tags. These fixed values are then included in the generated HL7 messages.

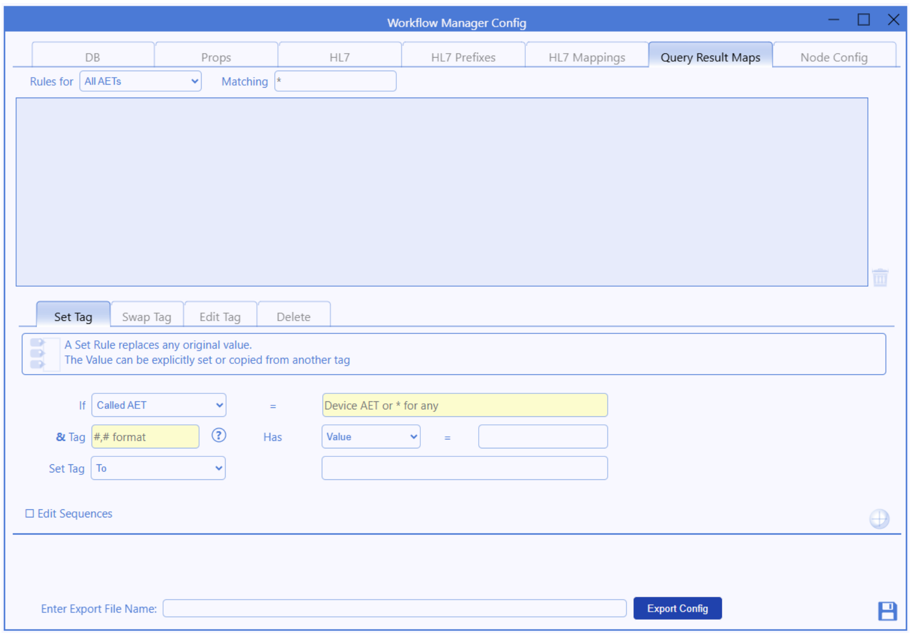


DICOM Tag	Mapping Value
No Matches Found	

Enter Export File Name:

Figure 115: DICOM tag mappings for fixed values in HL7 output.

When intercepting CFIND requests, query responses may need adjustment (e.g., prefixing a Patient ID). The mapping interface—shown in Figure 116—follows the same logic as the DICOM Tag Mapping Node, allowing for “Set”, “Edit”, “Swap” or “Delete” operations on the DICOM query response.


Workflow Manager Config

DB | Props | HL7 | HL7 Prefixes | HL7 Mappings | Query Result Maps | Node Config

Rules for: All AETs   Matching: *

Set Tag | Swap Tag | Edit Tag | Delete

A Set Rule replaces any original value.
The Value can be explicitly set or copied from another tag

If: Called AET = Device AET or * for any
& Tag: #,# format   Has: Value =
Set Tag: To

☐ Edit Sequences

Enter Export File Name: Export Config

Figure 116: DICOM tag mapping configuration for modifying intercepted CFIND results.

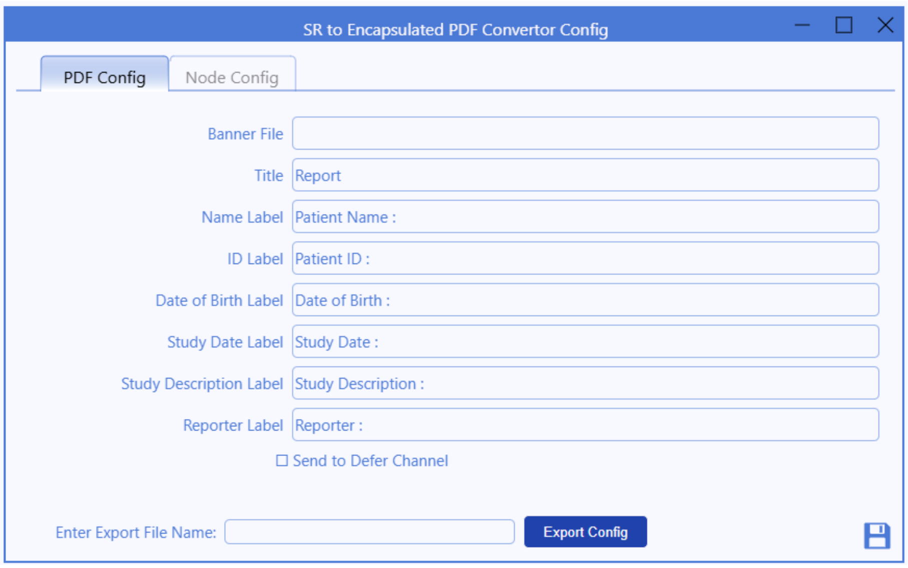
5.4.30 SR to PDF Node

Convert a DICOM SR file to a DICOM encapsulated PDF file.

The SR to PDF Node converts a DICOM Structured Report (SR) into a DICOM encapsulated PDF file using the contents of the SR header. Figure 117 illustrates the available configuration options for the node.

An optional banner image (in JPEG format) can be included at the top of the resulting PDF to serve as a header. The layout of the PDF is fixed, with key headings used to identify patient and study details near the top of the document. These label texts can be customised within the node configuration settings. The report content itself is included as plain text below the headings and their corresponding values.

The node also includes an option to route the generated DICOM file via the time-dependent output channel instead of the standard one. This can be enabled by selecting the corresponding check-box, and may be useful if the output file requires alternative routing behaviour within the pipeline.


SR to Encapsulated PDF Convertor Config

PDF Config | Node Config

Banner File	
Title	Report
Name Label	Patient Name :
ID Label	Patient ID :
Date of Birth Label	Date of Birth :
Study Date Label	Study Date :
Study Description Label	Study Description :
Reporter Label	Reporter :

☐ Send to Defer Channel

Enter Export File Name:

Export Config

Figure 117: Configuration options for converting a DICOM SR to an encapsulated PDF.

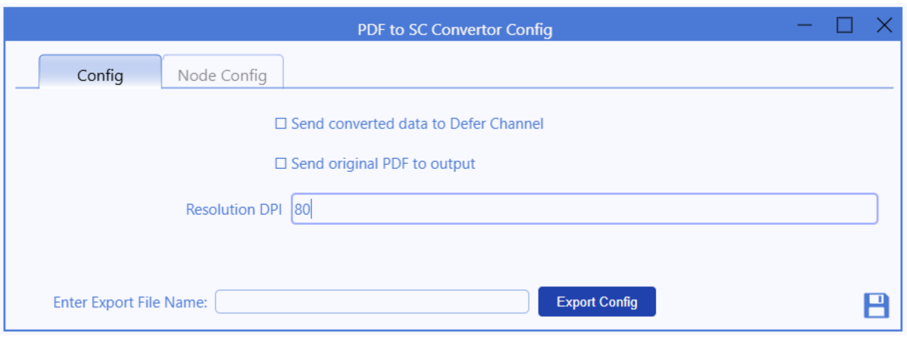
5.4.31 PDF to Secondary Capture Node

Convert a DICOM encapsulated PDF file to a DICOM secondary capture file.

A DICOM encapsulated PDF file can be converted into a DICOM secondary capture (SC) file by rendering the PDF content as an image and using that image as the pixel data in the output file. Figure 118 shows the configuration options for this node.

The resolution of the output image is specified in dots per inch (DPI) and can be set via the corresponding entry field. Since the resulting SC files are uncompressed, a high resolution setting may significantly increase the file size. Users should adjust the resolution to achieve an acceptable balance between image quality and file size.

There are two routing options available in the configuration. Firstly, the user can choose to route the newly created secondary capture file through the time-dependent (defer) output channel by enabling the corresponding checkbox. Secondly, the original encapsulated PDF file can either be discarded or retained and forwarded through the standard output channel, depending on the user's selection in the configuration interface.



PDF to SC Convertor Config

Config | Node Config

☐ Send converted data to Defer Channel

☐ Send original PDF to output

Resolution DPI: 80

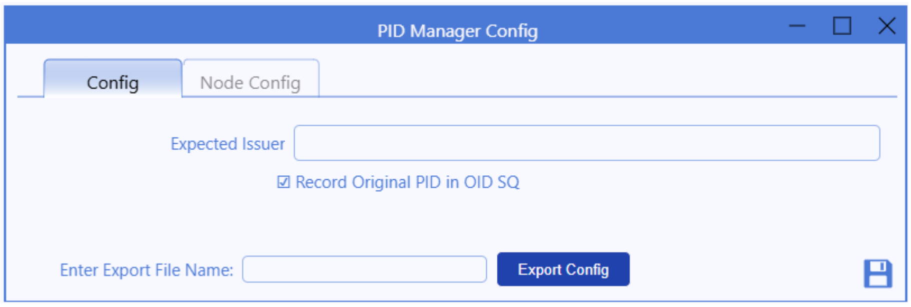
Enter Export File Name:

Figure 118: Configuration options for converting a DICOM encapsulated PDF to a DICOM secondary capture file.

5.4.32 PID Manager Node

Modify the primary ID in the DICOM header.

Figure 119 shows the configuration options for the PID Manager Node.



PID Manager Config

Config | Node Config

Expected Issuer:

☑ Record Original PID in OID SQ

Enter Export File Name:

Figure 119: Configuration options for replacing the primary Patient ID tag in a DICOM header.

When a `C-MOVE` operation is initiated from a PACS for a specific Study Instance UID, and multiple providers share the PACS, the primary Patient ID in the DICOM header may vary depending on which provider originally acquired the data. This can result in studies for the same patient arriving with differing primary Patient IDs.

The PID Manager Node enables harmonisation of this variation by replacing the current Patient ID in the header with one that matches the preferred Patient ID Issuer. It achieves this by searching the `Other Patient IDs Sequence` in the header and substituting the existing primary ID with the appropriate value from that sequence.

An additional configuration option (via checkbox) allows the original Patient ID and Issuer to be added into the `Other Patient IDs Sequence` following the substitution, if they are not already present. This ensures the provenance of the original identifiers is retained for downstream systems.

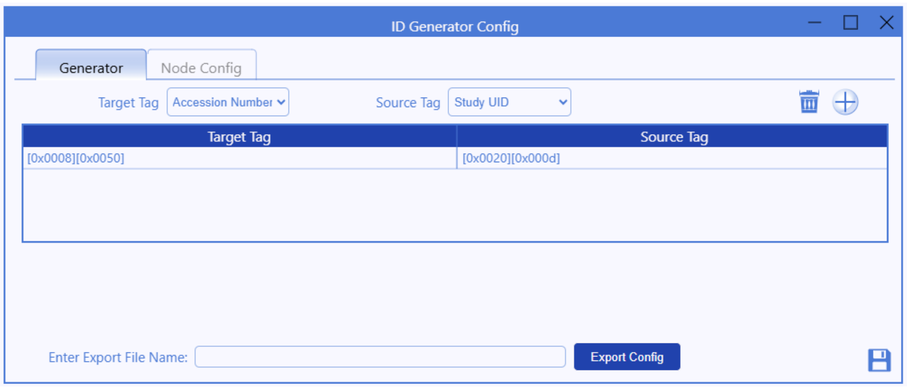
5.4.33 ID Generator Node

Generate a new ID for DICOM files.

Figure 120 shows the configuration options for the ID Generator Node. This node is typically used to generate an accession number for studies where one is not already present in the DICOM header.

The node works by taking a hash of the Study Instance UID and using that hash value as the basis for the new ID. This approach ensures that every file associated with the same Study UID receives the same generated ID. Crucially, this eliminates the need for maintaining a database of generated IDs and performing lookups, thereby improving efficiency and consistency.

The example shown in the figure demonstrates how the accession number can be regenerated using the hash of the Study UID.



Target Tag: Accession Number
Source Tag: Study UID

Target Tag	Source Tag
[0x0008][0x0050]	[0x0020][0x000d]

Enter Export File Name:

Figure 120: Configuration options for generating new IDs.

5.4.34 Multiframe Splitter Node

Splits multi-frame DICOM files into single-frame images.

The Multiframe Splitter Node is used to convert a multi-frame DICOM object into individual single-frame images, retaining them within the same series. This function is particularly useful when processing multi-frame secondary capture files that need to be split into separate secondary capture (SC) images. Multi-frame Nuclear Medicine (NM) images are also supported and can be split into individual files using this node.

There are no configuration options for this node; it operates automatically when a multi-frame input is detected.

5.4.35 Robotic Process Automation Node

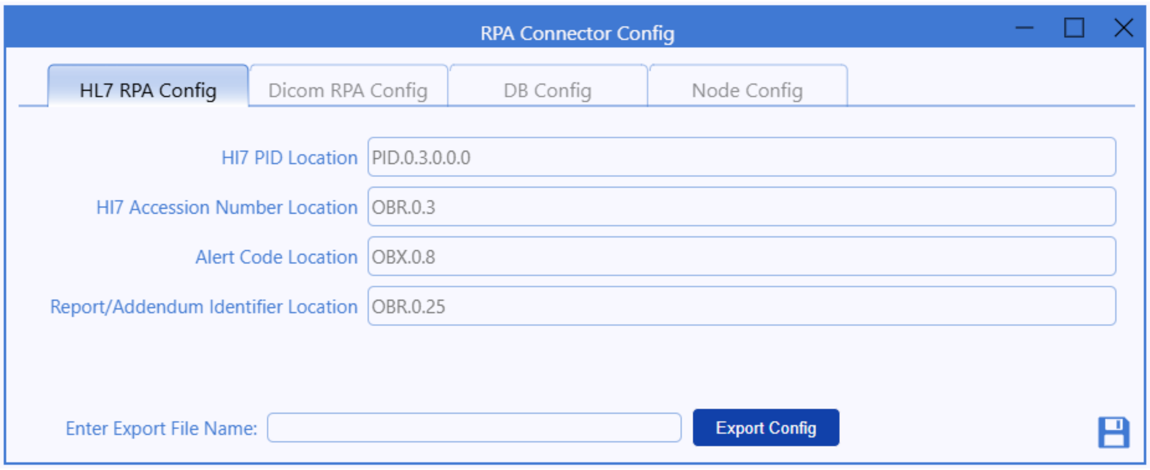
Creates RIS RPA Jobs from incoming DICOM and HL7 files.

The RPA node takes inbound DICOM or HL7 objects and uses the data from the inbound items to create a task to be communicated to an RPA service. The node does not operate on it's own but requires a secondary service to be configured which picks up the tasks created by the node and processes them via API communications with the RPA service.

Responsibilities of the RPA Node includes caching DICOM headers that are streamed through the node. This enables the node to match responses to the cached data and use the cache instead of performing a new DICOM CFind. Naturally, if the DICOM header is not cached then a CFind will be initiated as required. The RPA Node is responsible for parsing both Hl7 Messages and DICOM headers to gather the required information in order to create RPA Jobs.

HL7 Messages that come through the RPA node are used to create HTTP POST requests of the report/addendum the HL7 message holds. Whether it is a report or a addendum is defined in the message as the message type and the alert code within the HL7 message defines the action type for the report/addendum. Once these details, along with other essential details such as the patient id and accession number, the RPA job is created and send to the RPA engine.

After the RPA engine receives this HTTP POST, the engine will direct the report/addendum to where it needs to go and respond with a feedback message. The feedback message comes back to the RPA Connector Service, gets matched with the request job and proceeds to construct a HL7 message based on whether the report was send through successfully or not.



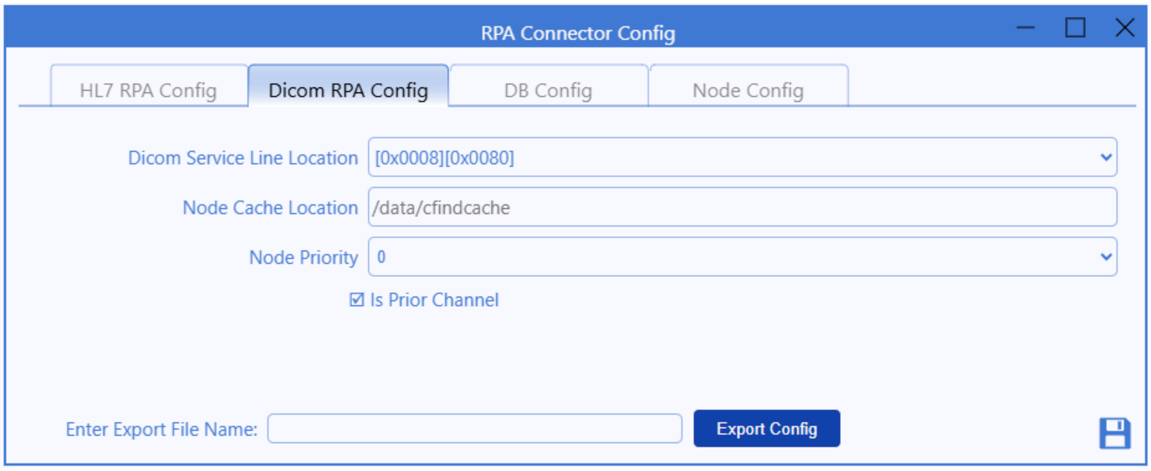
Hl7 PID Location	PID.0.3.0.0.0
Hl7 Accession Number Location	OBR.0.3
Alert Code Location	OBX.0.8
Report/Addendum Identifier Location	OBR.0.25
Enter Export File Name:	

Figure 121: RPA HL7 Mapping Configuration.

Figure 121 shows the HL7 mappings used to select the key values from the HL7 message. The values are recorded in the RpaJob table and passed in the API requests to the RPA. The key demographic values are patient ID and accession number. By default the patient ID is taken from the first component in the first entry in PID field 3 and the accession number is the full value of OBR field 3. The alert (urgency) code for the report is taken from OBX field 8 by default and is typically a single character to indicate an urgency code. Addenda are identified by a flag in the HL7 OBR segment which by default is field 25.

DICOM files will generate RPA jobs, the type of which is based on whether the study should be treated as a prior or a primary. DICOM to be considered a prior will trigger a job for a prior report whereas primary studies will create jobs to look up order request details. The RPA node can be set to primary or prior within it's node configuration.

For both these requests, the RPA Node gathers the required information to generate HTTP GETs to the engine. The responses depends on the order types. If it were a primary order, the RPA engine will respond with the primary, i.e. the clinical history of the patient. As well as this, if there are any documents associated with the primary such as jpegs or pdfs, then these will be sent to the RPA Connector Service as well. If it was a prior order, then the RPA engine will respond with the prior history of the patient/study. All the primary and prior responses are converted to HL7 messages and sent to a destination defined by the RPA service configuration. Whereas the documents are wrapped first into a DICOM file before being sent to a DICOM destination defined in the service configuration.

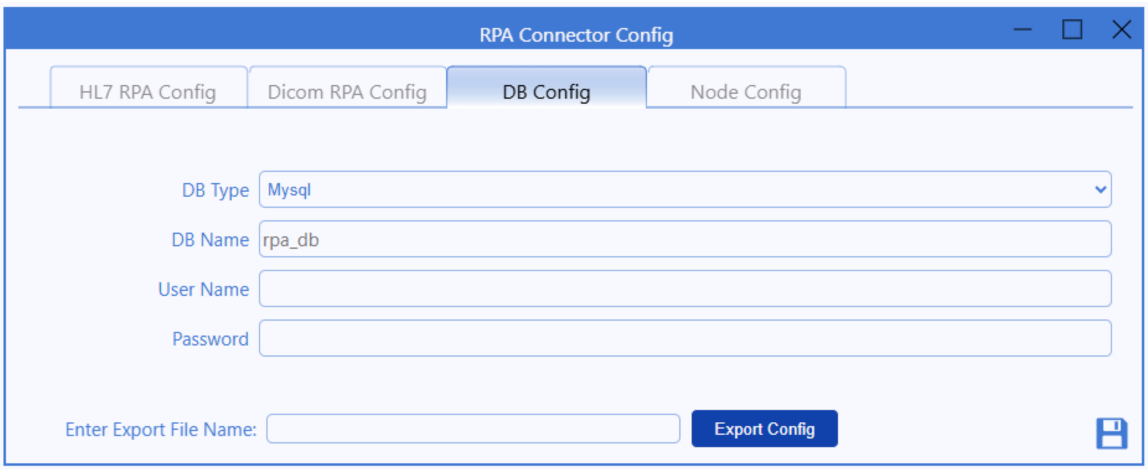


Dicom Service Line Location [0x0008][0x0080]
Node Cache Location /data/cfindcache
Node Priority 0
☑ Is Prior Channel
Enter Export File Name:

Figure 122: RPA Config 2.

Figure 122 shows the RPA configuration used for pulling key values from the DICOM header when creating an RPA job. By default, the "service line" is taken from the Institution Name tag in the header. The value in here is added to the HL7 message in OBR 16. The cache location for the DICOM header defaults to /data/cfindcache. The node priority is used to set a priority on the RPA job to ensure that higher priority jobs are passed to the API first. The node needs to be configured as either a prior or primary job handler and is configured using the tick box on the DICOM configuration panel.

The RPA node adds jobs to the database which must be the same database that the secondary service is configured to use to that jobs created by the node are picked up by the service.



DB Type: Mysql

DB Name: rpa_db

User Name:

Password:

Enter Export File Name:

Figure 123: RPA Config 3.

Figure 123 shows a standard database configuration panel used to define the RPA node database.

5.5 Output Nodes

Output nodes are end points in the network, either to a file system or another network connection.

5.5.1 Terminator Node

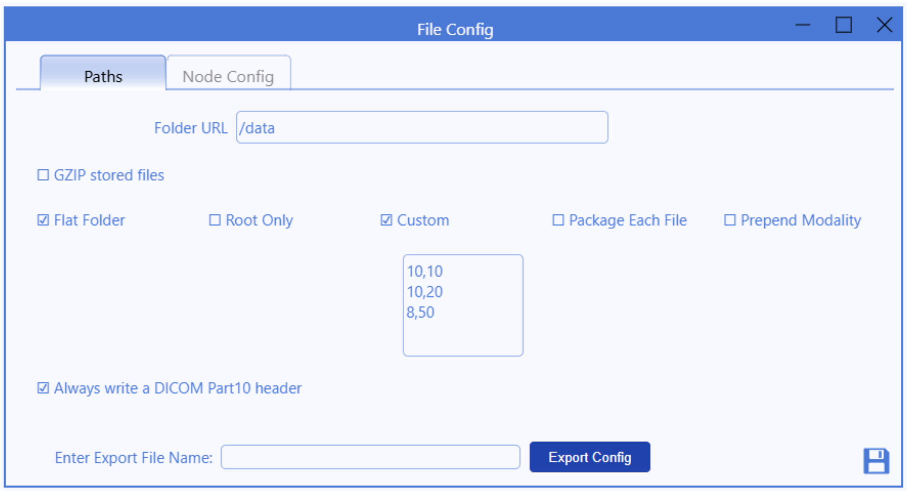
A terminator discards the stream rather than sending it to either a network or file output.

A Terminator Node effectively discards the stream. It can be used for testing data as it passes through the network without having to write it out to any destination.

5.5.2 DICOM File Output Node

A DICOM file output node writes the contents of the stream to a specified file system location.

A file output node is simply a local file system reference to where data is written.


File Config

Paths | Node Config

Folder URL: /data

☐ GZIP stored files

☑ Flat Folder ☐ Root Only ☑ Custom ☐ Package Each File ☐ Prepend Modality

```
10,10
10,20
8,50
```

☑ Always write a DICOM Part10 header

Enter Export File Name:

Export Config

Figure 124: Configuration options for writing DICOM files to a file output node.

The Folder URL is the local file system defined as a URL prefix of file:// followed by the absolute path e.g., /data or D:\data. The output files can be optionally GZIP'ed to compress the data. The flat folder option changes the folder hierarchy of the output data. Traditionally, the output for the DICOM files is of the form:

```
{root}/study/{called_aet}/{calling_aet}/{study_uid}/{series_uid}/
```

With the image files stored in the series UID folder. The flat folder option writes all files in the study in the same folder and the folder is labelled using the patient name, ID and date of birth. The example in Figure 124 show how the flat folder format can be customised to write the files out to folder created from values in tags in the DICOM header. The example shows how the custom option is used to write the data to a custom folder which will split writing at the study level using the accession number (tag 8,50). The data would be written into a folder based on the below format.

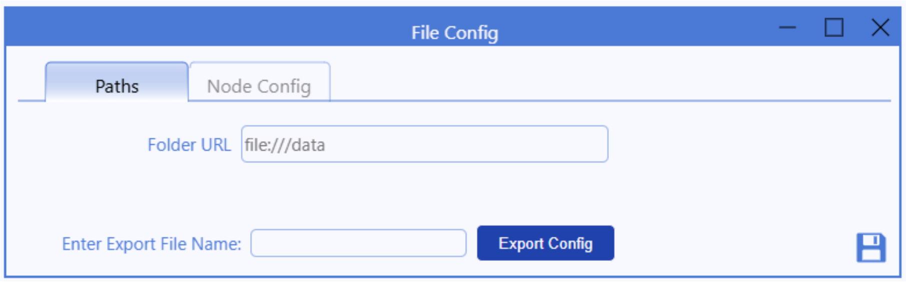
```
    {root}/study/{patientName_patientID_accessionNumber}/
```

The final configuration field configures the system to always write a DICOM Part 10 header, even for implicit data. When using the flat system, it is optionally possible to put each file in its own folder and to prefix the filename with the modality type.

5.5.3 HL7File Output Node

Writes HL7 Messages to a specified files system location.

The HL7 file output node simply writes the contents of an input HL7 message to a .dat file on the file system within the specified folder URL as shown in Figure 125.



Folder URL: file:///data

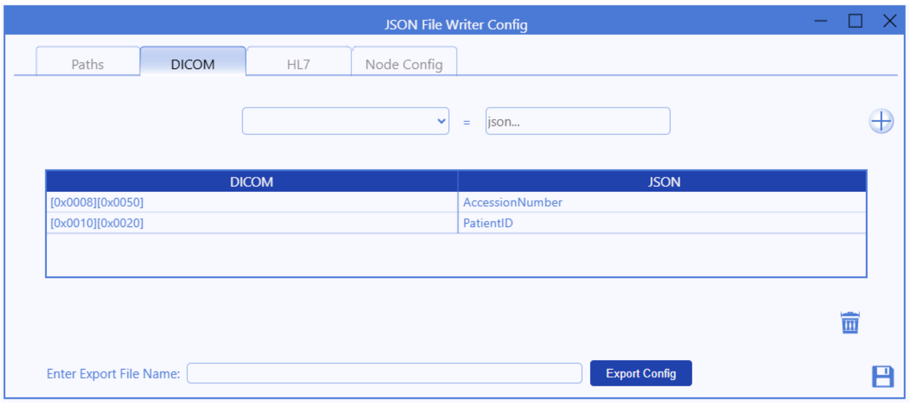
Enter Export File Name:

Figure 125: Configuration options for writing HL7 files to a file location.

5.5.4 JSON File Output Node

Writes JSON text files to a specified files system location.

The JSON file output node writes the contents of an input HL7 field or DICOM tag values to a .json file on the file system within the specified folder URL. The path location is set the same way as for HL7 messages as shown in Figure 125. Figure 126 show how the values in the accession number and patient ID tags can be written to a JSON file with the keys in the JSON file being AccessionNumber and PatientID.

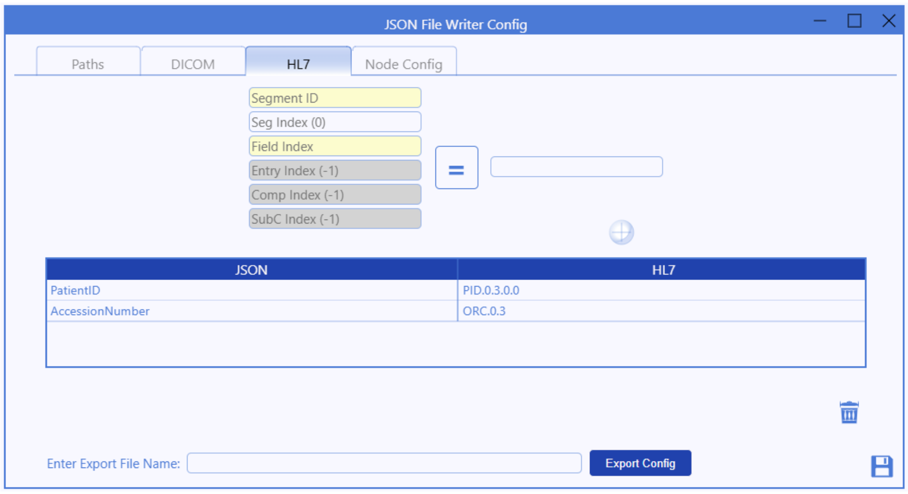

JSON File Writer Config

DICOM	JSON
[0x0008][0x0050]	AccessionNumber
[0x0010][0x0020]	PatientID

Enter Export File Name:

Figure 126: Configuration options for writing JSON files to a file location with the values taken from a DICOM header.

Figure 127 show how the same information can be obtained by mapping values from a HL7 message


JSON File Writer Config

JSON	HL7
PatientID	PID.0.3.0.0
AccessionNumber	ORC.0.3

Enter Export File Name:

Figure 127: Configuration options for writing JSON files to a file location with the values taken from a HL7 message.

5.5.5 DICOM Output Node

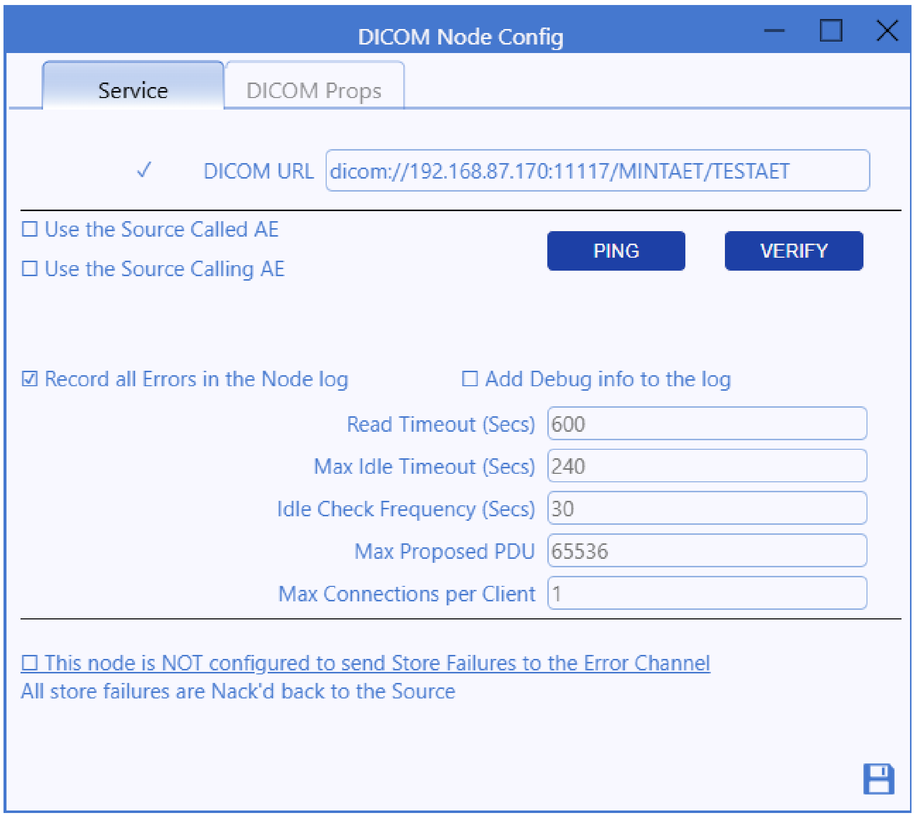
A DICOM stream output node writes the contents of the stream to a network DICOM destination.

The DICOM Output node allows the data stream to be written to any DICOM entity.

The node is configured using the dialogue as shown in Figure 43. The DICOM URL is the basic definition of the output destination and should be in the form:

```
    dicom://{ipaddress}:{port}/{called_aet}/{calling_aet}
```

Where “ipaddress” and “port” should be replaced with the IP address (hostname) and DICOM port of the destination. The called AET and calling AET values should be set to default values but these will be overridden if the option to use the source called AE or source calling AE are selected. In this case, the AETs from the input stream after any rules have been applied will be used to communicate with the output stream.



✓ DICOM URL: dicom://192.168.87.170:11117/MINTAET/TESTAET

☐ Use the Source Called AE

☐ Use the Source Calling AE

☑ Record all Errors in the Node log

☐ Add Debug info to the log

Read Timeout (Secs)	600
Max Idle Timeout (Secs)	240
Idle Check Frequency (Secs)	30
Max Proposed PDU	65536
Max Connections per Client	1

☐ This node is NOT configured to send Store Failures to the Error Channel

All store failures are Nack'd back to the Source

Figure 128: Configuration options for writing DICOM files to a DICOM URL destination.

On the right, there are two buttons, ”PING” and ”VERIFY”. Both of these buttons aims to establish a DICOM association. The ”VERIFY” button only appears when both options of using either the source called AE and source calling AE is unticked. In a ”PING” request, the system initiates a socket connection by checking for the IP address and DICOM port. An invalid entry for either the IP address or the port will result in a connection refusal. In a ”VERIFY” request, the system additionally takes into account the calling and called AE. Similarly, the association checks for the validity of calling and called AE, as according to the DICOM port config.

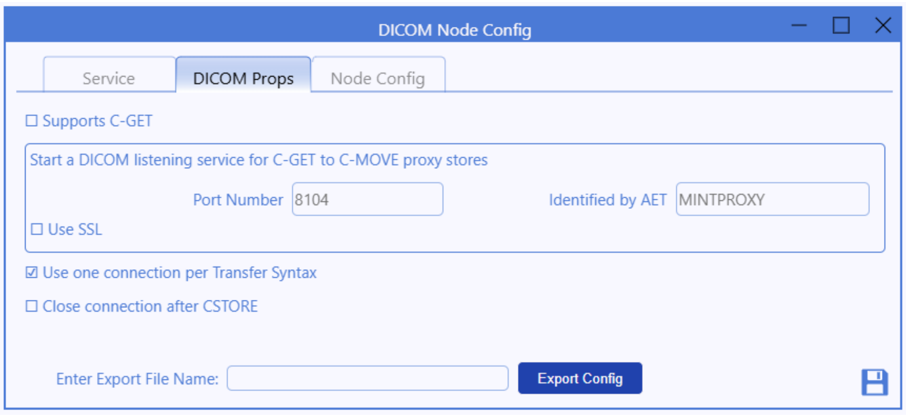
The read timeout option specifies the time the node should wait when reading data on the output connection.

The “Max Connections per Client” option specifies the number of connections that will be opened to an output source for a given input stream. For efficiency, an output node may try to keep an association open in order to stream through data. If the SOP class of the data changes, this will trigger a new connection to open. The old connection can be kept open for further data that is of the original SOP class, but this can cause some systems to reject new communications because of limits set on the number of concurrent connections. Setting this to 1 will mean any existing connection will be closed before a new one is opened. For values greater than 1, the oldest (time since last used) connection will be closed.

Max idle time is the maximum amount of time in seconds that an association will be kept open before it is closed if there is no more activity using that association. This should be set to below the read timeout of the receiving DICOM node.

The debug option, when selected will result in the DICOM association details being logged in detail when the DICOM output port opens a connection to another system.

If the NACK option is ticked, any errors will be passed back down in the input pipeline to the source rather than the error being processed via an error channel.



☐ Supports C-GET

Start a DICOM listening service for C-GET to C-MOVE proxy stores

Port Number: 8104

Identified by AET: MINTPROXY

☐ Use SSL

☑ Use one connection per Transfer Syntax

☐ Close connection after CSTORE

Enter Export File Name:

Figure 129: Extended configuration options for writing DICOM files to a DICOM URL destination.

The second tab on the DICOM output port configuration allows the user to define if the system being connected to supports CGET. If so, then if the network is proxying a retrieve request, it will try to use CGET in order to obtain the database before sending it to the requested AET. As most systems won't support CGET, the option allows the user to define a receiving DICOM listening port, the details for which should be used by the system which will receive the CMOVE request this node will send. When a CMOVE results in data returning to this port, the originating input port for the CMOVE request will attempt to open a CSTORE to the destination AET. This requires therefore that the destination AET be defined as a known node in the input port configuration.

The "Use one connection per Transfer Syntax" setting in the figure defines how the output node tries to negotiate an association with the destination. When ticked, the output port will only open one association (per inbound connection) per transfer syntax. This means the output node will propose all possible storage classes when it opens the connection. The association is then kept open and new data, provided it was an accepted SOP class can be sent on using the existing association. When this option is unticked, the output node will open (and keep open) a separate connection for each combination of transfer syntax and storage SOP class. Each SOP class will be negotiated individually for the inbound syntax.

The option to stream DICOM tags in queries allows the user to specify that a query can be sent over multiple PDVs, effectively streaming the tags. The default is to buffer the query and send in a complete packet (provided it is less than the Max PDU value.

5.5.6 DICOM Caching Output Node

DICOM Output Node with built in file caching.

This node is very similar to the standard DICOM Output Node, in that it sends DICOM files to a DICOM URL. This node however, combines the DICOM Output Node with a DICOM File Writer node to temporarily write the files out as they arrive. These files are deleted when the data is successfully sent to the destination. The primary benefit of this mode of operation is that, in the event that an error response is returned from the destination, rather than propagating that response back to the originator, this node will send an OK acknowledgement and send the failed data to the error channel. This is different from the standard output node, which can only send to the error channel if the initial connection fails.

The configuration for the node is a combination of the DICOM Output Node shown in Figure 43 and the DICOM File Writer Node shown in Figure 124. There is one extra bit of configuration, which is a tick box for “Parallel Storage”. If this option is ticked, then the node will write the data to both the cache and destination at the same time. If the option is not selected, the node will write everything in the file to the cache location and once everything is received for that instance, the node will try and send it to the destination. In either case, if anything fails, the cached file will be sent to the error channel, visualised in the network as the pipe out of the bottom of the output node.

5.5.7 HL7 Output Node

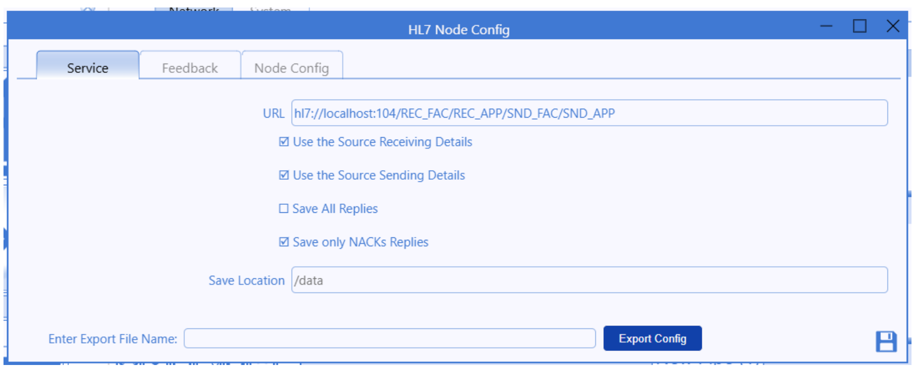
Opens HL7 communications with a specified receiver and sends messages out and reads replies.

The HL7 Output node allows the data stream to be written to any HL7 entity.

The node is configured using the dialogue as shown in Figure 130. The HL7 URL is the basic definition of the output destination and should be in the form:

```
    hl7://{ipaddress}:{port}/{rec_fac}/(rec_app}/{snd_fac}/{snd_app}
```

Where “ipaddress” and “port” should be replaced with the IP address (hostname) and HL7 port of the destination. The “Receiving Facility and Application” and “Sending Facility and Application” values should be set to default values but these will be overridden if the option to use the source values are selected. In the case that this option is used, the message values for these parameters from the input stream after any rules have been applied will be used to communicate with the output stream.


HL7 Node Config

Service | Feedback | Node Config

URL: hl7://localhost:104/REC_FAC/REC_APP/SND_FAC/SND_APP

☑ Use the Source Receiving Details

☑ Use the Source Sending Details

☐ Save All Replies

☑ Save only NACKs Replies

Save Location: /data

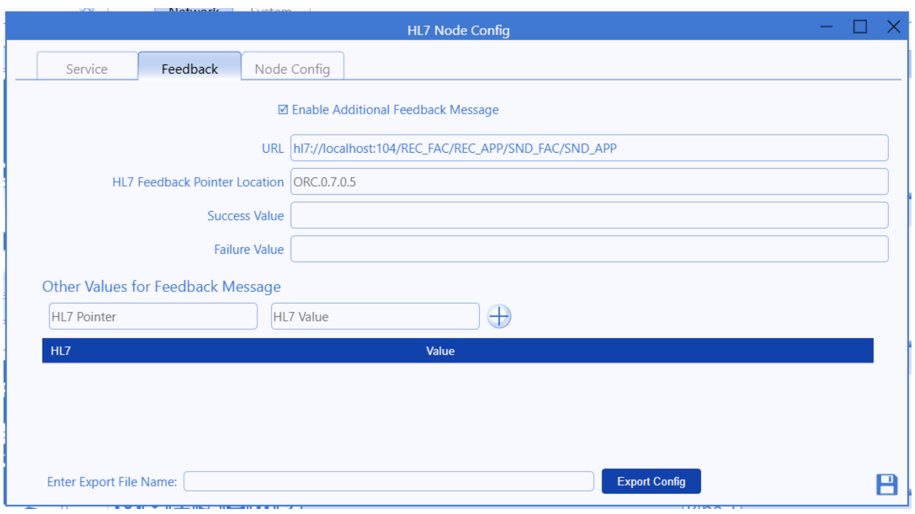
Enter Export File Name:

Export Config

Figure 130: The HL7 output node configuration panel.

There is the option to define the save location of the replies that are received from the HL7 messages sent. By default, only the NACK responses are saved but that can be changed to saving all replies or if unticked then to not save any replies.

That is the main process of the HL7 Output Node. An additional feature to this node is the "Feedback" tab where users can configure the HL7 Output node to send an additional "Feedback" message once an ACK or an NACK has been received from the streamlined HL7 message. The typical use case is for a status update on systems as well as customised use within Nexus. Feedback configuration is configured in the same config dialog but under the "Feedback" tab as shown in Figure 131. Initially the tab just shows a tickbox to enable the Feedback messages (which is off by default). However once enabled, the full config is shown and can be configured.


HL7 Node Config

Service | Feedback | Node Config

☑ Enable Additional Feedback Message

URL: hl7://localhost:104/REC_FAC/REC_APP/SND_FAC/SND_APP

HL7 Feedback Pointer Location: ORC.0.7.0.5

Success Value:

Failure Value:

Other Values for Feedback Message

HL7 Pointer | HL7 Value

HL7	Value

Enter Export File Name:

Export Config

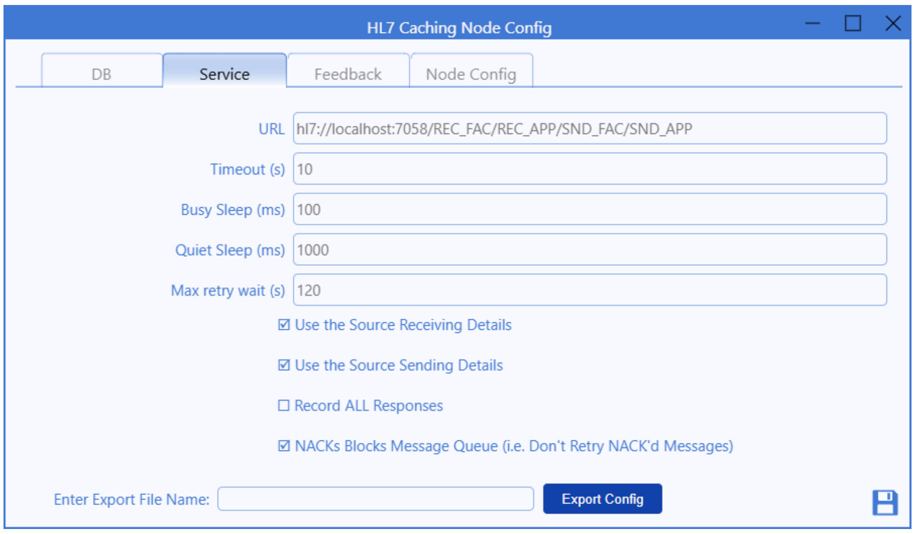
Figure 131: The HL7 output node Feedback tab configuration panel.

Once the feedback is enabled, the HL7 "URL" can be set similarly to how the node output url was set and this will become the destination for the Feedback message. The "HL7 Feedback Pointer Location" is the location within the HL7 Feedback message that would contain the "Success Value" or the "Failure Value" which is based on the ACK received for the intial HL7 message or if the HL7 message failed to send. Finally, "Other Values for Feedback Message" is a table where additional modifications can be added that is up to the user. To add to this, just enter a valid HL7 pointer location with the value that it'll contain for all Feedback messages. The HL7 Value for this can be blank.

5.5.8 Caching HL7 Output Node

Opens HL7 communications with a specified receiver and sends messages out from a cached queue.

The Caching HL7 Output Node works the same way as the standard HL7 Output Node works, but rather than streaming the messages straight to the output and propagating any errors back to the send. The node will write the message into an outbound queue in the database and send an OK to the sender. The node uses a standard file cache and database configuration with an extra service configuration piece as shown in Figure 132


HL7 Caching Node Config

URL hl7://localhost:7058/REC_FAC/REC_APP/SND_FAC/SND_APP
Timeout (s) 10
Busy Sleep (ms) 100
Quiet Sleep (ms) 1000
Max retry wait (s) 120
☑ Use the Source Receiving Details
☑ Use the Source Sending Details
☐ Record ALL Responses
☑ NACKs Blocks Message Queue (i.e. Don't Retry NACK'd Messages)
Enter Export File Name:

Figure 132: The caching HL7 output node configuration panel.

The service configuration uses a HL7 URL to define the destination and a timeout on the connection in seconds. Each time messages in the queue are sent, the service will loop and ask for more jobs. If the service just processed jobs, it will sleep for the configured “Busy Sleep” period. If there were no jobs to process, it will sleep for the “Quiet Sleep” period. This means if there are no new messages arriving the system can sleep for longer but if there is data coming in, it will prevent delays in moving the data on. The messages are then sent out in the order they arrive. In the event that one fails, the node will sleep and then try again. Each time it fails, it will sleep a little longer, backing off the retries up to a configurable maximum wait. The “Max retry wait” is the maximum time the system will wait before retrying to send the message. The tick boxes determine whether the source and sending details in the URL should be replaced with the incoming ones in the message. If they are unticked, the relevant fields in the MSH segment of the message will be replace with the ones defined in the URL.

There is an option for the node to record all replies (rather than just NACKs). If enabled, the replies will be saved in the DB.

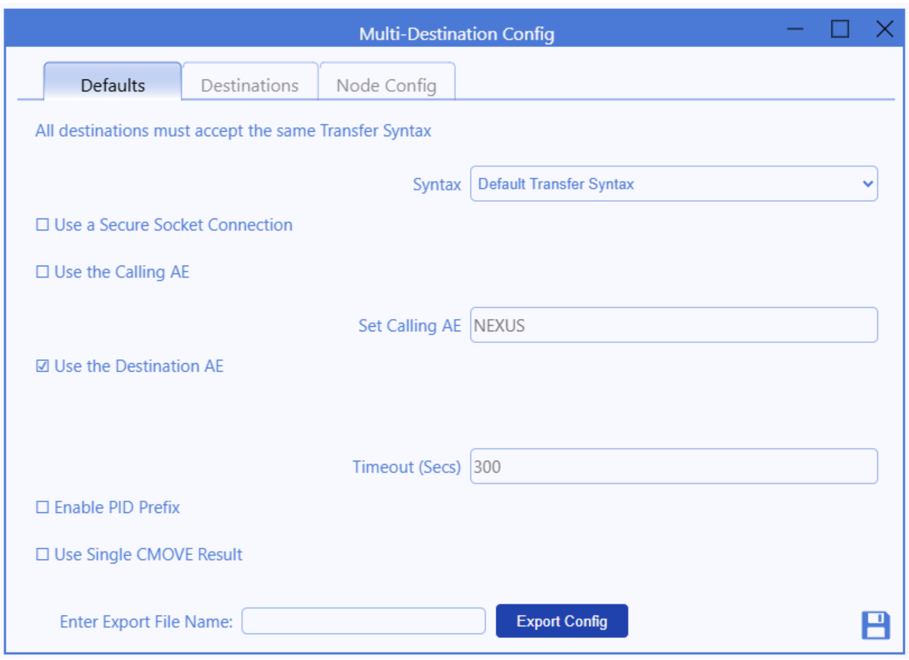
By default, any failures or rejected messages will cause the queue to back up until the blockage is cleared. There is an option to not block the queue if the node successfully sends the message but receives a negative acknowledgement back from the receiver. When ticked, the node will move on to the next message and log the NACK as a failure and add an alert to the error management to be cleared or dealt with manually.

Similary to the HL7 Output node, the HL7 Caching Output node also has the feature of sending an additonal “Feedback” message based on the messages that has been streamed through. The configuration is the same and can be seen in 5.5.7. The only unique behaviour to the Feedback messages on the HL7 Caching output is that only one Feedback message is sent per HL7 message that is streamed through unless on a retry, an ACK is received. Then the successful Feedback message is sent to indicate that the HL7 message has now been sent successfully.

5.5.9 Multi Destination DICOM Output Node

A multiple DICOM stream output node writes the contents of the stream to multiple network DICOM destinations.

The multi DICOM output node is a special case which is directly associated with the DICOM proxy node and is used to pass CFIND and CMOVE requests to multiple DICOM destinations. The requests are sent to each channel in sequence and the results are combined and sent back to the requesting system. In the case of CMOVE requests, the moving of the data is handled directly between the destination of the request and the associated destination AET within that request. Figure 133 shows the defaults that are used for the communications.



All destinations must accept the same Transfer Syntax

Syntax: Default Transfer Syntax

☐ Use a Secure Socket Connection

☐ Use the Calling AE

Set Calling AE: NEXUS

☑ Use the Destination AE

Timeout (Secs): 300

☐ Enable PID Prefix

☐ Use Single CMOVE Result

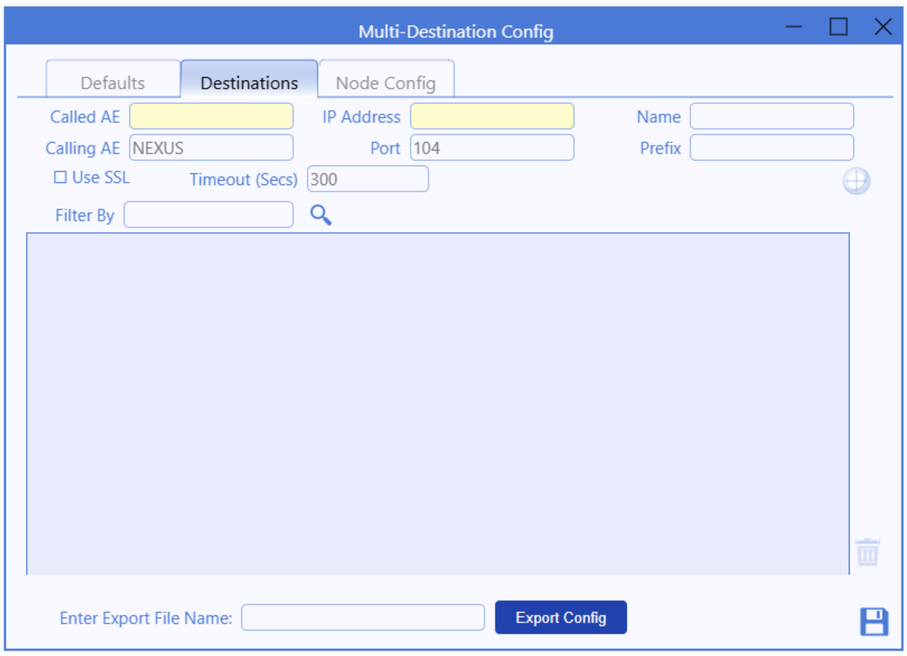
Enter Export File Name:

Figure 133: Multi DICOM output node configuration for default settings.

All communications are done using the default transfer syntax, as these just involve sending CFIND and CMOVE requests. There is an option to select that the communications should be performed over a secure connection, although most systems will not support this option. The default calling AET can be defined in the top text field and the timeout below. In the event that no specific value is given for an output node, these defaults will be used. If the option to use the calling AE is selected, the calling AE from the node into this one will be used in the communications and could therefore vary based upon the source of the input. If $SYSID is used anywhere in the configuration for the calling AET, it will be changed by the server to the configured value for the system ID before being used in any communications.

By default, the system will use the destination AET in the CMOVE request as the destination AET in the forwarded request, as that is the AET where the requesting system wants the data sent. It is possibly to override this by unticking the “Use the Destination AE” option. The specified destination AET will then replace that set by the originating system. If the “Enable PID Prefix” is set, rather than the have the system send the request to all channels, the patient ID tag will be compared with the prefix set for each channel in Figure 134 and only route the request to channels where the patient ID tag value starts with the prefix set in the prefix field.

When handling a CMOVE request, by default, the system will send the CMOVE request to all channels, and each one would respond in turn by performing a CSTORE of the requested data to the destination AET. By selecting the “Use Single CMOVE Result” option, the system will stop sending the CMOVE request to the channels once it receives a response from the destination to indicate that it has a match for the request and it is moving data. This can prevent the same data being sent by multiple system that each have a copy of the data.



Called AE

IP Address

Name

Calling AE NEXUS

Port 104

Prefix

☐ Use SSL

Timeout (Secs) 300

Filter By

Enter Export File Name:

Figure 134: DICOM output configuration.

Figure 134 shows how to add destinations to the multiple outputs on the node. The called AET, IP address and port are required as they define the system that will be connected to. The other settings are optional and if not filled in, the default defined on the first tab will be used. When all of the required settings have been entered, the node can be added to the list be selecting the (+) icon. Nodes can be edited by selecting them from the list. They can be deleted by selecting the entry and then the trash icon at the bottom right. The list can be searched using the “Filter By” option. This will list only the destinations with a matching name.

5.5.10 JPEG Output Node

A JPEG file output node writes the pixel contents of the stream to a specified file system location as a JPEG.

A file output node is simply a local file system reference to where data is written. The JPEG node configuration is a standard file writer configuration, similar to that shown in Figure 124. When a DICOM file is sent into the JPEG file output node, the pixel data is read and converted to a JPEG. This image is written to the specified location on disk in a folder that is comprised of the patient ID, name and date of birth. Multi-frame data is saved as individual images, with one image per frame. Files are labelled with the instance UID and in the case of multi-frame data, the files are also indexed with the frame number.

5.5.11 PDF Filer Node

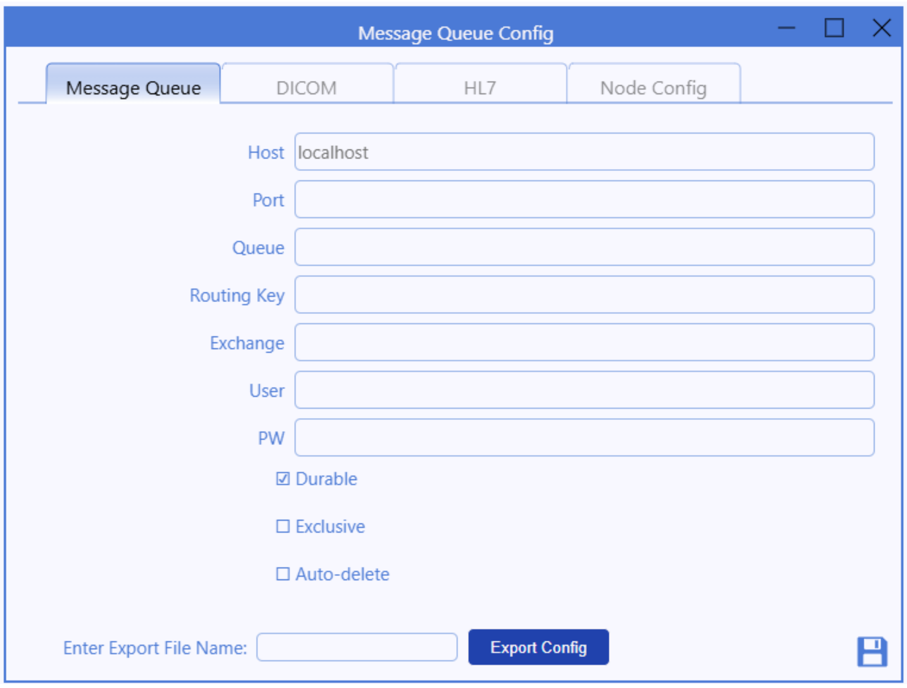
A PDF file output node writes the content tag of the stream to a specified file system location as a PDF.

This node uses a standard file writer configuration to define where the output files are written. This node is used to convert Encapsulated PDF files back to a normal PDF file. The output file is written with a filename of the SOP instance UID from the DICOM header with a “pdf” extension.

5.5.12 Message Queue (MQ) Output Node

Convert DICOM or HL7 to JSON and submit to a message queue.

The Message Queue Output Node provides a way to push information from a DICOM or HL7 file into a message queue with a JSON payload. The node uses the same configuration options as the JSON File Writer Node shown in Figure 126 and Figure 127. These configurations determine the content of the payload to the message queue. The configuration details for the connection to the queue are shown in Figure 135. Only Rabbit MQ has been supported by this mechanism. The fields correspond with the required connection details for the message queue and should be completed according to the queue configuration.


Message Queue Config

Message Queue | DICOM | HL7 | Node Config

Host: localhost
Port:
Queue:
Routing Key:
Exchange:
User:
PW:
☑ Durable
☐ Exclusive
☐ Auto-delete

Enter Export File Name:
Export Config

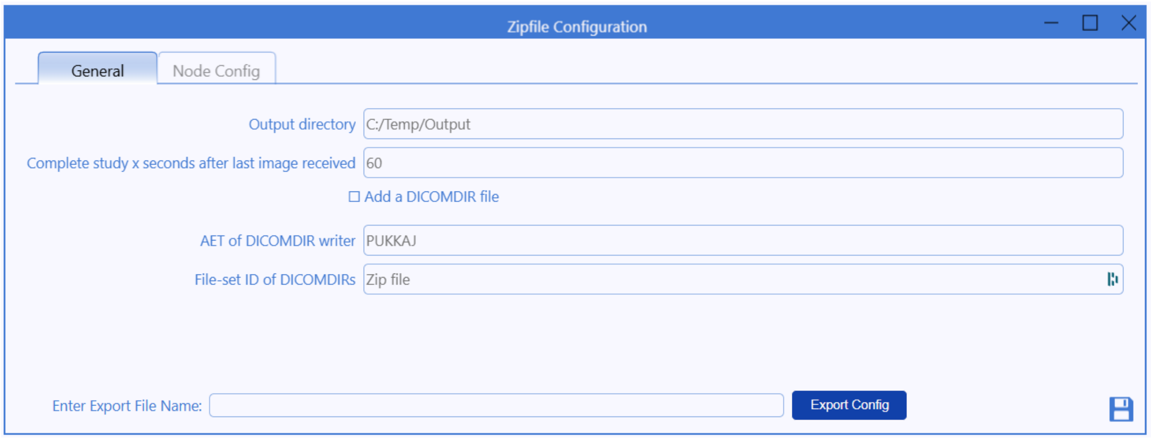
Figure 135: Rabbit MQ queue configuration options.

5.5.13 DICOM Study Zip Node

Creates a zip archive of DICOM output files.

The DICOM Zip Output node writes out DICOM files to the file system in a similar way to the DICOM File Writer node but has an extended process which monitors the output directory for new files and when no more files have arrived for a set period, the output folder is zipped up at the study level.

The configuration for the node is shown in Figure 136.


Zipfile Configuration

General | Node Config

Output directory: C:/Temp/Output

Complete study x seconds after last image received: 60

☐ Add a DICOMDIR file

AET of DICOMDIR writer: PUKKAJ

File-set ID of DICOMDIRs: Zip file

Enter Export File Name:

Export Config

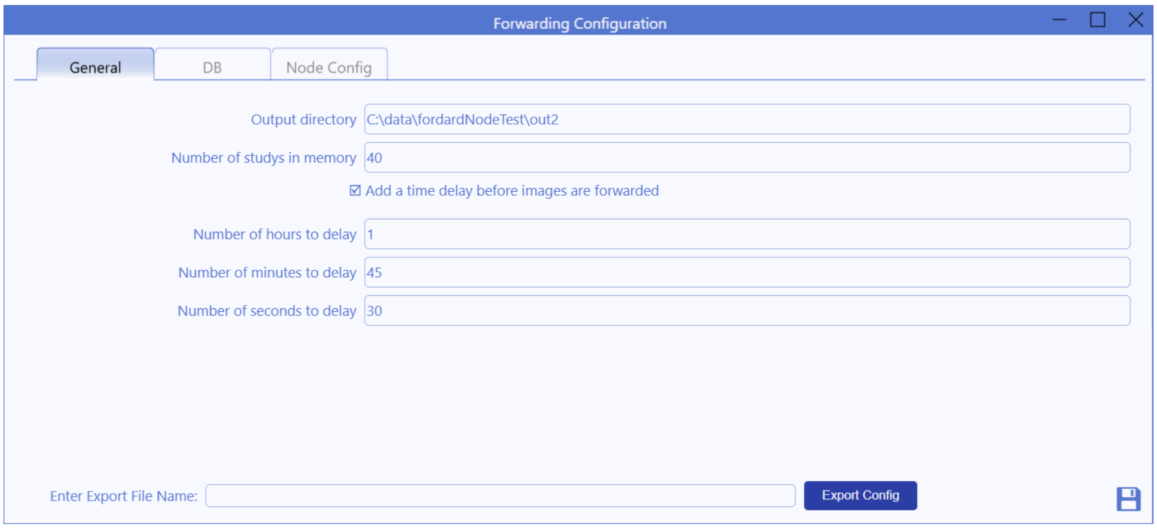
Figure 136: DICOM study zip output node configuration.

The output directory is the location where the files are written to with the subfolder name being the DICOM study UID of the images. The images are cached in a hidden folder (.cache) in the output location and the study folder is zipped up once no more images have been written into the output location with the time delay defined by the second field in the figure. A DICOMDIR index file can be optionally written as part of the output to create an index that a DICOMDIR reader can use to list the contents of the directory. The last two settings relate to the DICOMDIR, where the saving AET can be set along with the “File set ID” which is a description of the media and is used in the DICOMDIR.

5.5.14 Forwarding Node

Caches files and adds a forwarding job to a database.

The Forwarding Node saves the inbound DICOM file to the directory specified in the *General* configuration tab (figure 138) and adds a forwarding *job* to the database specified in the *DB* configuration tab. The database specified in the configuration is automatically created if it does not already exist. The node does not control the eventual destination of the DICOM data, it merely saves it, rather a *Database Forwarding Node* (see section 5.3.13) reads these jobs from the database and allows an entire pipeline to be configured for handling the workflow. Multiple Forwarding Nodes can co-exist and write to either the same, or different databases. The input Database Forwarding Node, however, has exclusive access to a given database (see section 5.3.13).



Output directory: C:\data\fordardNodeTest\out2
Number of studys in memory: 40
☑ Add a time delay before images are forwarded
Number of hours to delay: 1
Number of minutes to delay: 45
Number of seconds to delay: 30
Enter Export File Name:

Figure 137: Forwarding output node configuration.

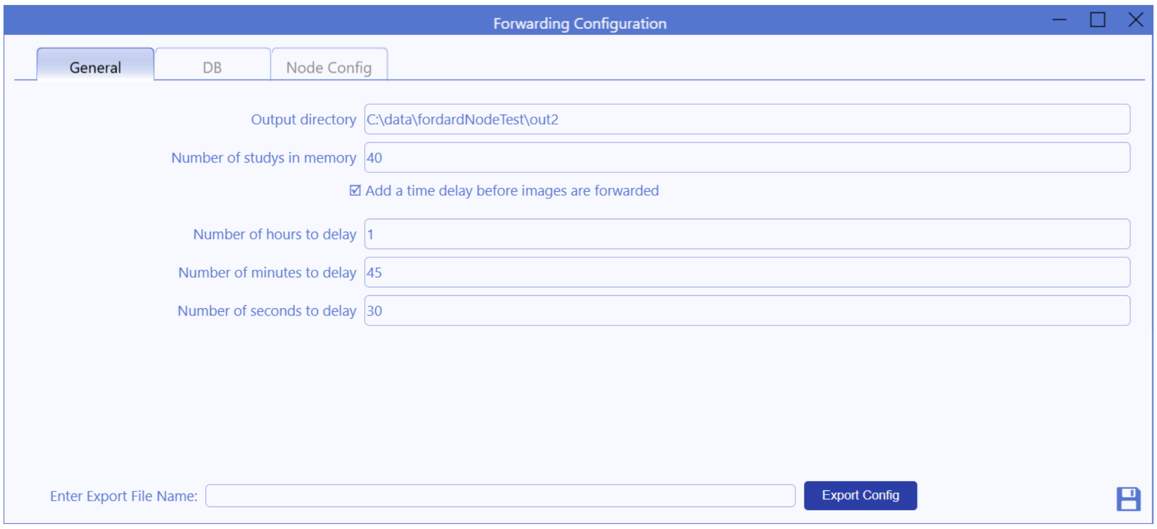
The node configuration allows the user to control the number of studies kept in memory and to add an optional delay before forwarding. The number of studies kept in memory defaults to 50, but can be adjusted to suit the needs of the user; a higher number will result in reduced database calls at the expense of increased memory usage and vise versa. There is a tickbox to add a delay between the images arriving and the earliest time an image could be collected for forwarding by a *Database Forwarding Node*. When this is ticked, the times in the three delay fields (hours, minutes, seconds) are added together and used to created a scheduling time in the database. The configuration shown in figure 138 would schedule images to be sent no earlier than 1 hour, 45 minutes and 30 seconds after arrival.

5.5.15 DICOM Web Node

Handles connections to a DICOM Web RESTful API.

The Forwarding Node saves the inbound DICOM file to the directory specified in the *General* configuration tab (figure 138) and adds a forwarding *job* to the database specified in the *DB* configuration tab. The database specified in the configuration is automatically created if it does not already exist. The node does not control the eventual destination of the DICOM data, it merely saves it, rather a *Database Forwarding Node* (see section 5.3.13) reads these jobs from the database and allows an entire pipeline to be configured for handling the workflow. Multiple Forwarding Nodes can co-exist and write to either the same, or different databases. The input Database Forwarding Node, however, has exclusive access to a given database (see section 5.3.13).

The node configuration allows the user to control the number of studies kept in memory and to


Forwarding Configuration

General | DB | Node Config

Output directory: C:\data\fordardNodeTest\out2

Number of studys in memory: 40

☑ Add a time delay before images are forwarded

Number of hours to delay: 1

Number of minutes to delay: 45

Number of seconds to delay: 30

Enter Export File Name:

Figure 138: Forwarding output node configuration.

add an optional delay before forwarding. The number of studies kept in memory defaults to 50, but can be adjusted to suit the needs of the user; a higher number will result in reduced database calls at the expense of increased memory usage and vise versa. There is a tickbox to add a delay between the images arriving and the earliest time an image could be collected for forwarding by a *Database Forwarding Node*. When this is ticked, the times in the three delay fields (hours, minutes, seconds) are added together and used to created a scheduling time in the database. The configuration shown in figure 138 would schedule images to be sent no earlier than 1 hour, 45 minutes and 30 seconds after arrival.

5.6 Network Information and Management

When the network has been successfully created and applied to the server, input nodes will start listening on their specified ports or will start processing any resources they have been configured to access. Data is passed from the input nodes to the next node in the network, with each node acting on the data based on what it is configured to do. When data is passed to a split in the network, a copy is passed down each path sequentially. Output nodes write the data out or send it to another system. Each node tracks its own progress with metrics which are displayed above each node in the network to indicate the status of the node. Figure 139 shows two networks each configured and applied. Above each node, the first metric of the status information is displayed. By left-clicking with the mouse, all the available status information for that node is displayed.

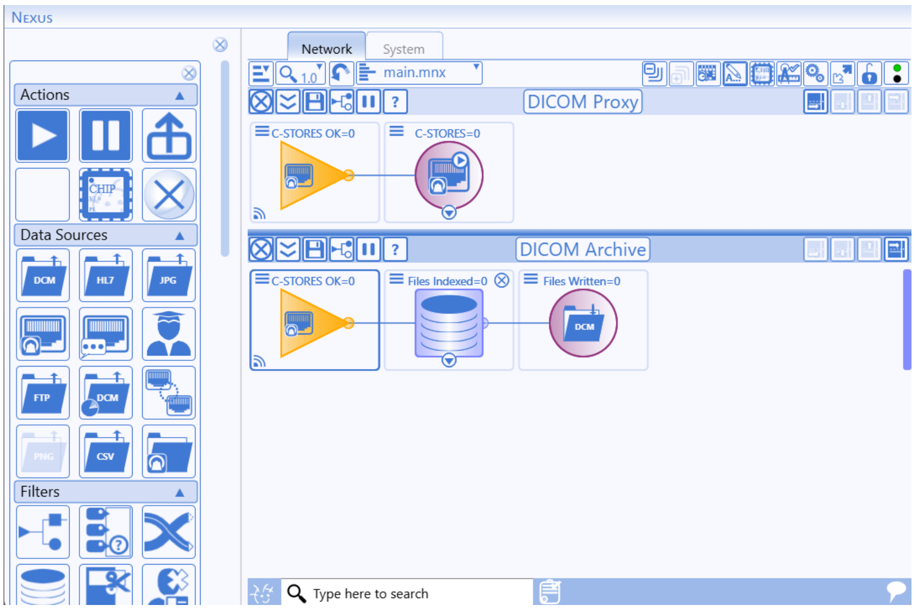


Figure 139: Multiple configured networks.

Using the mouse scroll wheel whilst holding down shift on the main network display zooms the display in and out and using the wheel over the scroll bar on the side scrolls up and down.

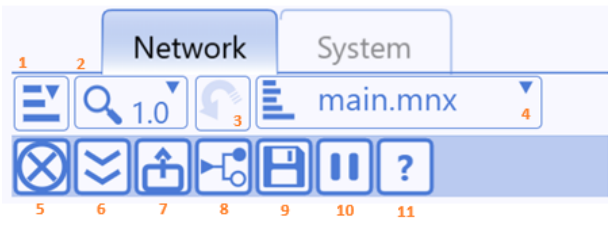


Figure 140: Network controls.

Figure 140 shows the primary controls of the network (top row) and the controls available on an individual pipeline.

1. Network Menu.

2. Network Zoom.
3. Undo Change.
4. Network Name.
5. Remove Pipeline.
6. Collapse Pipe.
7. Save Pipe Config.
8. Copy Pipe Config.
9. Apply Individual Pipe Config.
10. Pause Pipe.
11. Edit Pipe Comments.

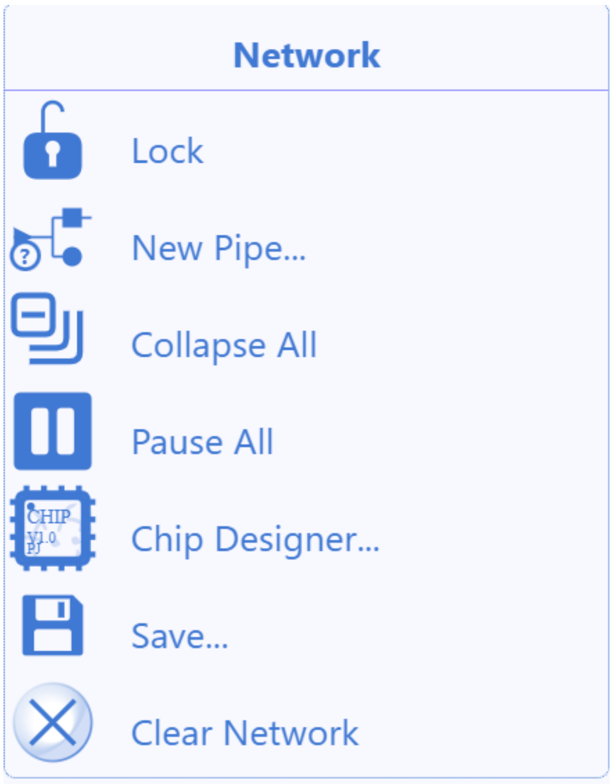


Figure 141: Network menu options.

The network menu option shows some menu configuration and control options as shown in Figure 141. The options are self explanatory, but the "New Pipe" option expands to provide a list of input node options from which to create a new pipe and set the input node.

The network zoom option applies a zoom level to the network. The zoom can also be applied by holding down the shift key and using the mouse scroll wheel.

The undo option reverts the last change made to the network.

The network name label is used if the network is exported to define the name of the file that is saved to the download location.

The close option on a pipeline removes that pipeline from the network.

The collapse option changes the pipeline view to a summary view showing some key status values.

The save pipeline option displays a dialogue showing the serialised configuration for the pipeline. The configuration can be saved by either copying the text or by selecting the save option in the dialogue which will store a file in the download location using the name of the pipeline defined in the network. The name of the pipeline should be defined in advance by double clicking on the pipeline title to edit it and pressing “Enter” or “Return” to apply the change to the pipe configuration. A previously saved pipeline can be added to a network by dragging and dropping the file onto an unlocked Nexus network.

The copy network option create a copy of the pipeline in the network. If the input node is set to listen on a port, the port number must be updated before applying the configuration change otherwise a port bind exception will occur.

The pause option can be used to pause the specific pipeline. In the case of a file reader which is multi-threaded, the pause option will pause once each worker has finished the current task.

The pipeline comments are edited by selecting the comments option which displays a dialogue. The comments can be used to save notes on the function of the pipeline and there is an option point comments at specific nodes in the pipeline.

There are option buttons at the top right of the network configuration as shown in Figure 142.

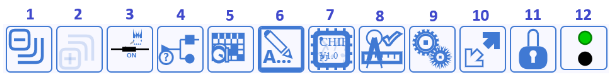


Figure 142: Network configuration options.

The options are explained via tooltip if hovered over with the mouse. They are:

1. Collapse all pipelines.
2. Expand all pipelines.
3. Switch grouping editor.
4. Pipe template editor.
5. Network configuration history.
6. Show/hide annotations.

7. Open the chip designer.
8. Show scales.
9. Open the service option dialogue.
10. Fullscreen.
11. Lock/unlock.
12. Status indicator.

The collapse all option collapses all the pipelines in the network and likewise, the expand all option will expand any that are collapsed.

For designing pipelines and importing example pipelines, a system is available to convert nodes and pipelines in a network into templates that can later be edited and managed. The template manager can be accessed by clicking the templates menu button on the top right of the network graph buttons.

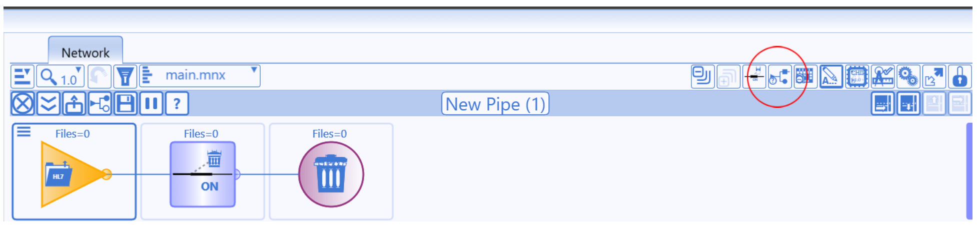


Figure 143: Template Manager Button

Templates can be published from nodes or pipes after they have been added to the network. Right-clicking on a pipe and clicking "Publish As Template" will open a dialog window prompting for a name and description. After the name and description is confirmed, it will be added to the templates menu automatically. The same can be done for any node in the network.

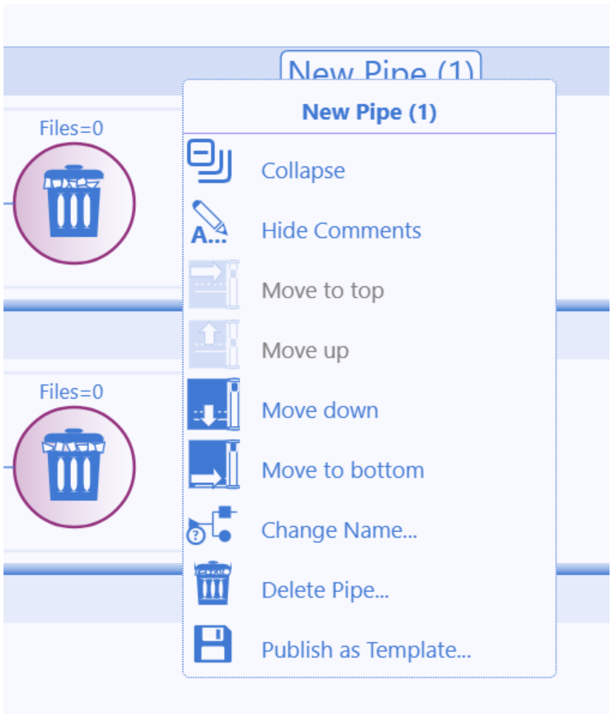
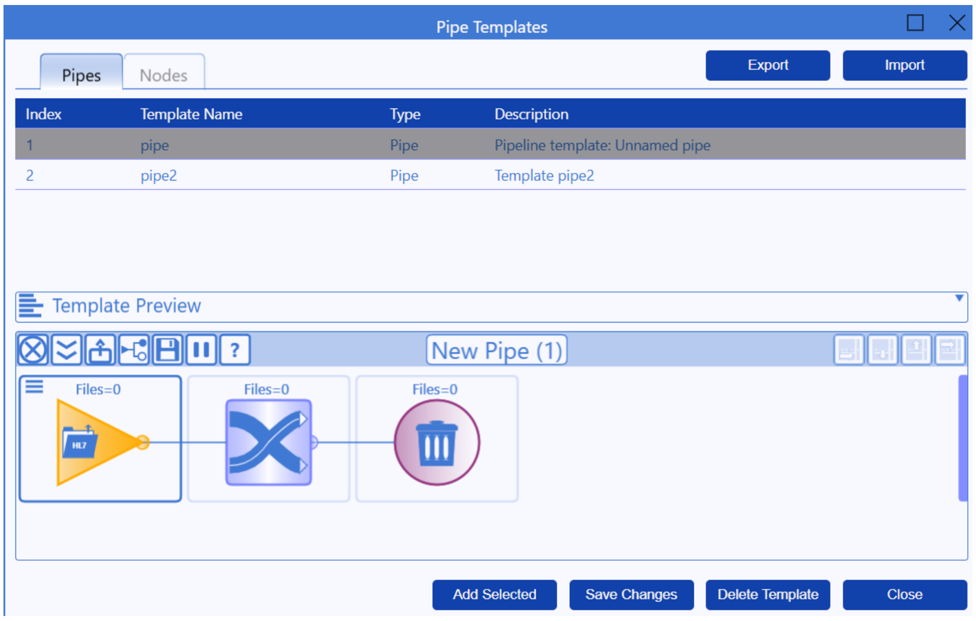


Figure 144: Adding a Pipe Template

The template manager is divided into template nodes and pipes. A template can be added by selecting a template in the manager and selecting the ”Add Selected” button. If a node template is being added, you must have selected a valid node location in the network before opening the template manager, such as a valid input node for an input node template. Templates can be removed from this library using the ”Delete Template” button while the correct template is selected. This will permanently delete them.

The template previewer underneath shows the node or pipe template that is currently selected. Node configurations can be edited using this previewer by double-clicking the node to open the configuration and then clicking the bottom right save icon, much like a regular pipeline in the network. After this, the ”Save Changes” button in the template menu must be selected to save any changes to the node or pipe template.



Index	Template Name	Type	Description
1	pipe	Pipe	Pipeline template: Unnamed pipe
2	pipe2	Pipe	Template pipe2

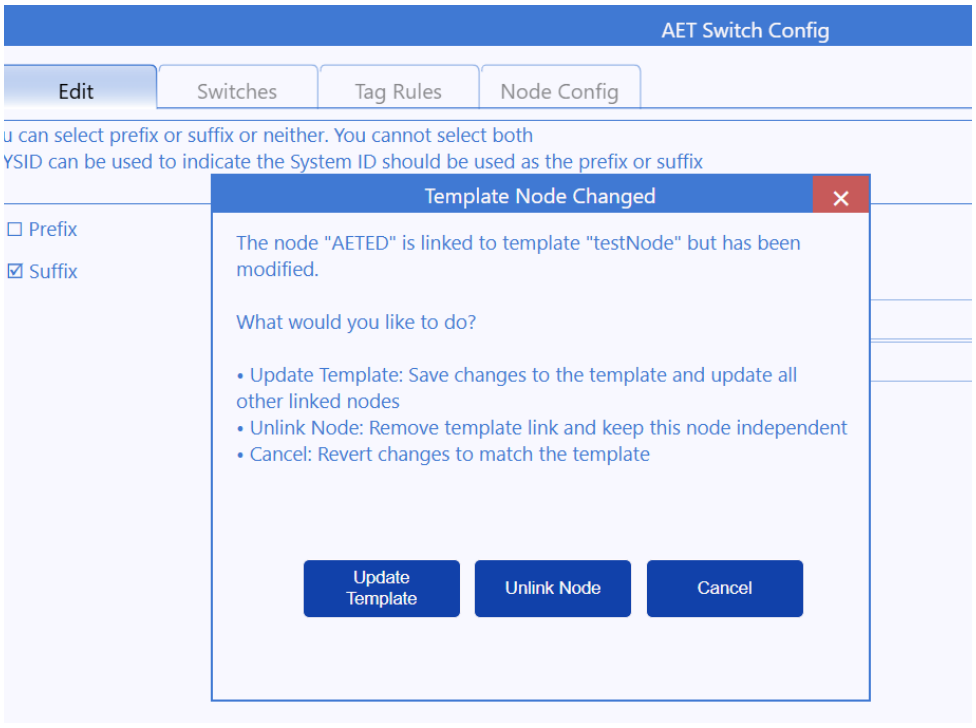
Figure 145: Template Menu

The ”Export” button can be selected to export all current pipe and node templates into a Json file which will be saved locally. This file can be imported using the adjacent ”Import” button, which will automatically add all listed templates into the menu after a valid Json file is selected. Any repeated templates will be skipped.

An example of the Json file’s structure for a single template is as follows:

```
{
  "exportDate": "2025-08-18T14:56:34.121Z",
  "templates": [
    {
      "date": "20250723",
      "name": "pipe",
      "description": "Pipeline template: Unnamed pipe",
      "time": "165821.0",
      "id": "1.2.826.0.1.3680043.2.526.11.30.
      17531093348798504.285.3789917183",
      "type": "PIPE",
      "user": "admin",
      "config": "2099030847171,IN,HL7FILES,New Pipe (1)
      {}[[2098954741725,PIPE,
      AETED{TMPL:
      eyJ0ZW1wbGF0ZVNvdXJjZSI6InRlbXBfMTc1MzEwOTQ2MTc4Nl9xcWRlc2k
      iLCJ0ZW1wbGF0ZU5hbWUiOiJub2RlIiwidGVtcGxhdGVWZXJzaW9uIjoxfQ==}
      [[2098954741734,OUT,DNULL{}[+]]+]]+]"
    },
    ]
}
```

The template menu can also be opened by right-clicking on a network node and clicking ”Apply Template”. This will open a separate version where only a node can be selected, and they will be filtered for nodes of the selected type of node. For instance, if an input node was right-clicked, it will only show templates of input nodes. Instead of adding the node into the network, it will convert the selected node into the node shown in the selected template by pressing the ”Replace Node” option.


AET Switch Config
Edit
Switches
Tag Rules
Node Config
u can select prefix or suffix or neither. You cannot select both
YSID can be used to indicate the System ID should be used as the prefix or suffix
☐ Prefix
☑ Suffix
Template Node Changed
Update Template
Unlink Node
Cancel

The node "AETED" is linked to template "testNode" but has been modified.

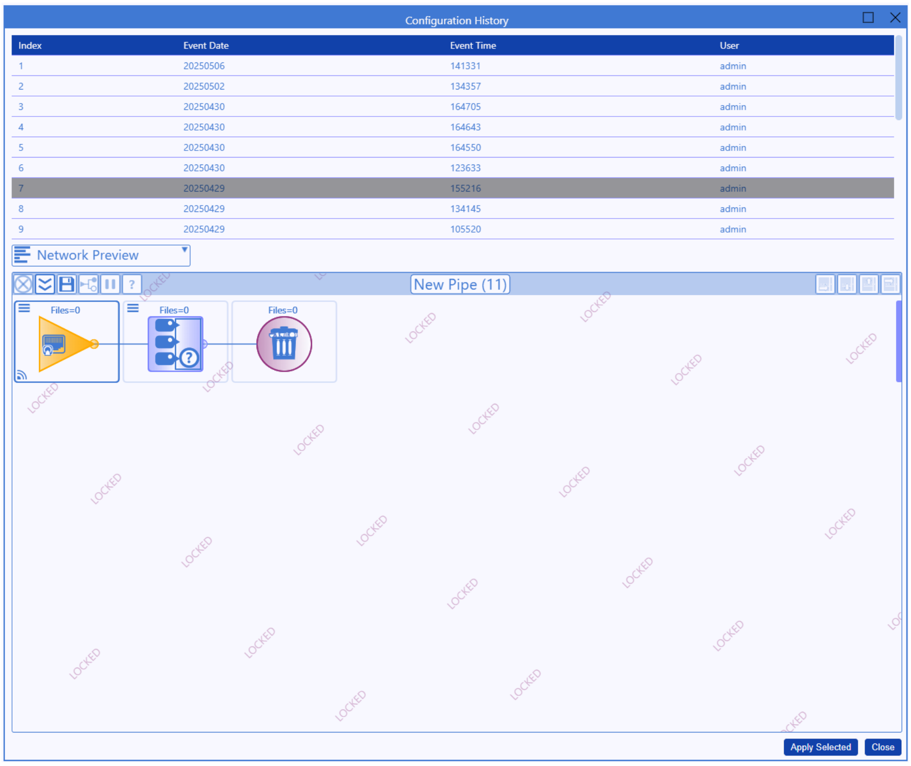
What would you like to do?

• Update Template: Save changes to the template and update all other linked nodes
• Unlink Node: Remove template link and keep this node independent
• Cancel: Revert changes to match the template

Figure 146: Template Menu

By default, all nodes added from a template are considered linked. This means that if a change is made to the node’s configuration and then the save button is clicked, the user will be prompted to apply the same changes to all other nodes created from the original template. This includes the original template itself. A prompt will be shown whenever this occurs asking if the user wants to update all linked nodes and templates, or instead to unlink the current node from the template so that it can be edited separately.

The network history tool allows previous configurations of the network to be restored. When a configuration is changed, the configuration file is backed up on the server side. The history display allows the recent changes to be displayed.



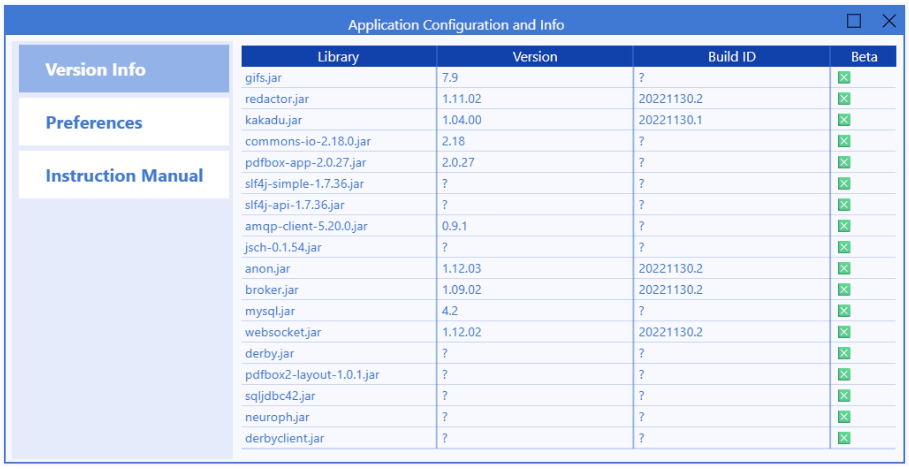
Index	Event Date	Event Time	User
1	20250506	141331	admin
2	20250502	134357	admin
3	20250430	164705	admin
4	20250430	164643	admin
5	20250430	164550	admin
6	20250430	123633	admin
7	20250429	155216	admin
8	20250429	134145	admin
9	20250429	105520	admin

Figure 147: Network configuration history.

Each backed up configuration can be previewed in the user interface and can be restored by selecting the “Apply Selected” option at the bottom right of the dialogue.

The show annotations shows/hides any annotations that have been created on the network. The chip button is a shortcut to bring up the chip creation tool and the “show scales” option simply shows the background cell layout of the nodes and the network.

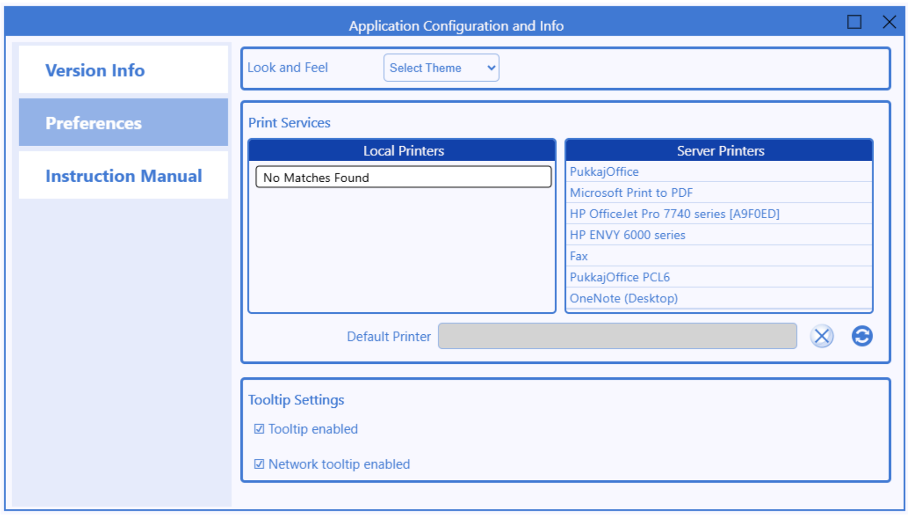
The service option dialogue shows the version numbers of the libraries in use as shown in Figure 148. If the library supports the version numbering, the version information is displayed in the table along with the beta status.



Library	Version	Build ID	Beta
gifs.jar	7.9	?	☒
redactor.jar	1.11.02	20221130.2	☒
kakadu.jar	1.04.00	20221130.1	☒
commons-io-2.18.0.jar	2.18	?	☒
pdfbox-app-2.0.27.jar	2.0.27	?	☒
slf4j-simple-1.7.36.jar	?	?	☒
slf4j-api-1.7.36.jar	?	?	☒
amqp-client-5.20.0.jar	0.9.1	?	☒
jsch-0.1.54.jar	?	?	☒
anon.jar	1.12.03	20221130.2	☒
broker.jar	1.09.02	20221130.2	☒
mysql.jar	4.2	?	☒
websocket.jar	1.12.02	20221130.2	☒
derby.jar	?	?	☒
pdfbox2-layout-1.0.1.jar	?	?	☒
sqljdbc42.jar	?	?	☒
neuroph.jar	?	?	☒
derbyclient.jar	?	?	☒

Figure 148: Network library version information.

There are some user preferences that are configurable here as shown in Figure 149.


Application Configuration and Info

Version Info
Preferences
Instruction Manual

Look and Feel: Select Theme

Print Services

Local Printers	Server Printers
No Matches Found	PukkajOffice
	Microsoft Print to PDF
	HP OfficeJet Pro 7740 series [A9F0ED]
	HP ENVY 6000 series
	Fax
	PukkajOffice PCL6
	OneNote (Desktop)

Default Printer

Tooltip Settings

☑ Tooltip enabled

☑ Network tooltip enabled

Figure 149: Network user prefs.

The look and feel allows the user to choose the colour scheme to use. This information is saved as a cookie on the PC. There are two types of tooltip on the application. Standard tooltips and tooltips related to the network. Both can be toggled by the user and saved as cookies. The default printer can also be chosen here.

This manual can be launched from the “Instruction Manual” section provided that the manual is available in the docs folder as “Nexus-Manual.pdf”.

The option buttons at the top right of each pipeline is for ordering the network in the display.

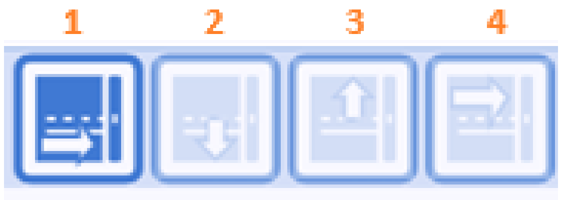


Figure 150: Network ordering.

Option 1 is to move the pipeline to the bottom for the networks, option 2 moves the pipeline down, option 3 moves the pipeline up and option 4 moves the pipeline to the top.

5.7 Task Manager

There are some processes that require a standalone methodology, rather than a pipeline, such as transferring all images within a certain date from an external PACS. Nexus uses a task manager system to create new tasks to drive these jobs. Task manager processes are managed by the taskbar which also includes a search feature to find nodes in the network.

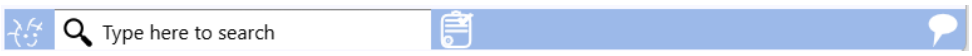


Figure 151: Nexus Taskbar

A task can be created by selecting the clipboard icon at the bottom left of the network page and selecting the New option. This opens the new Task Selector dialogue as shown in Figure 152. When the type of controller has been selected, the add icon at the top left of the dialogue opens the configuration dialogue for the selected controller.

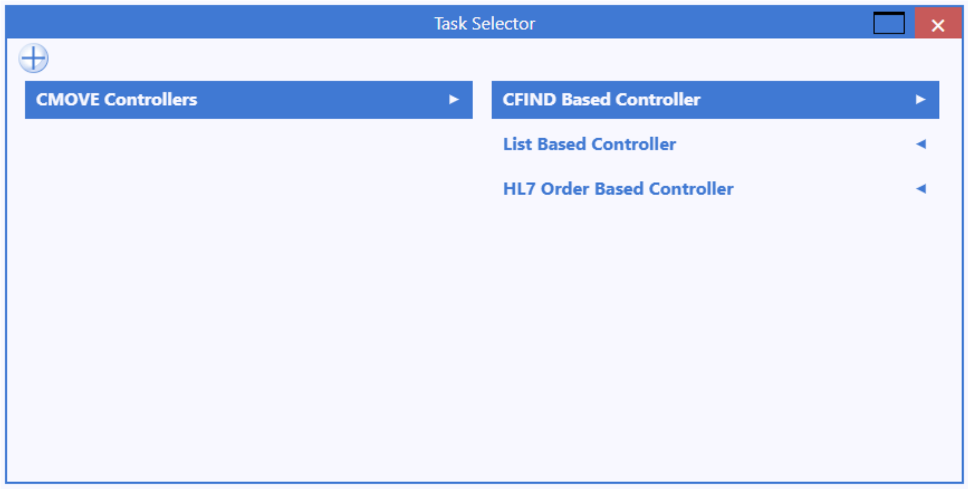
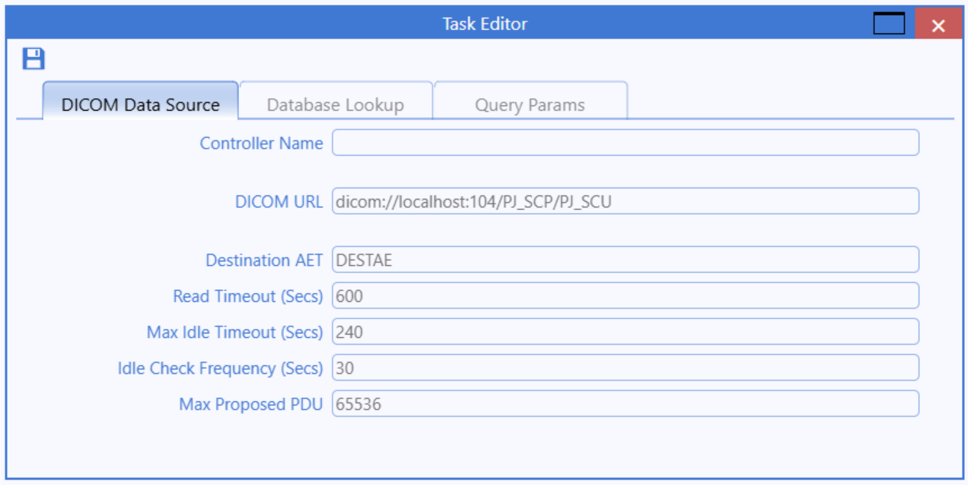


Figure 152: Task selection of a CMOVE controller that uses a CFIND process.



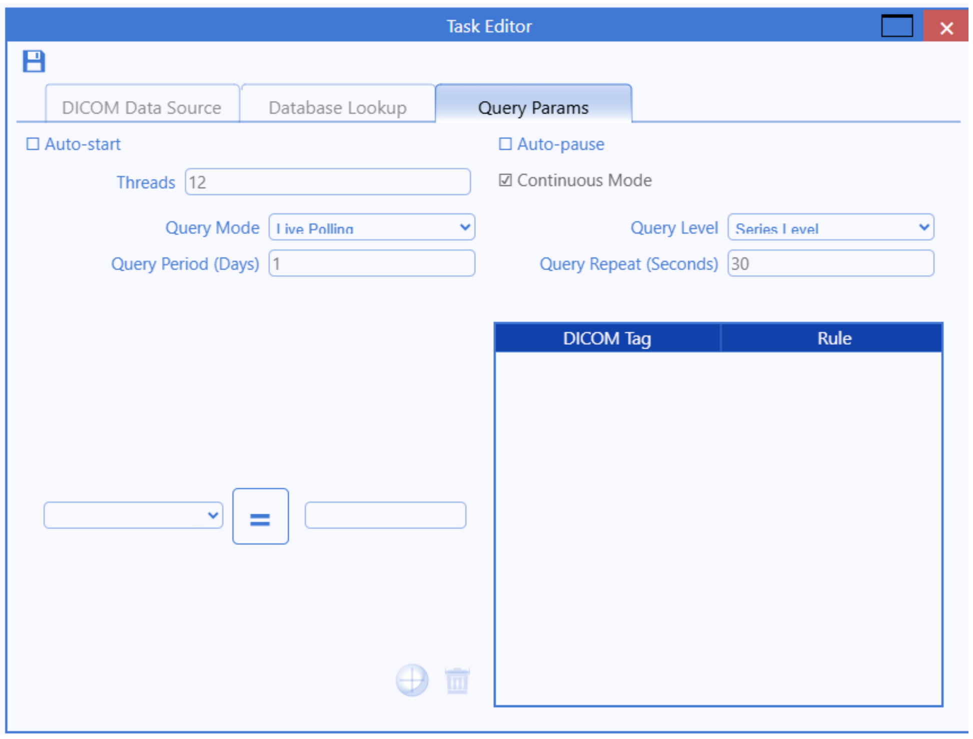
Controller Name	
DICOM URL	dicom://localhost:104/PJ_SCP/PJ_SCU
Destination AET	DESTAE
Read Timeout (Secs)	600
Max Idle Timeout (Secs)	240
Idle Check Frequency (Secs)	30
Max Proposed PDU	65536

Figure 153: DICOM Configuration for a CFIND Based Move Controller.

Figure 153 shows the DICOM definition of the CMOVE Controller Task. A destination AET is required in order to specify where the data should be transferred to and should be a known AET by the system specified in the DICOM URL. The DICOM URL is the IP/host address of the system to be queried and the port on which it is to be connected on. The called AET and the calling AET are specified after the port number.

In order to not move data that is already known the controller, a database lookup configuration is defined. This will check the database for any data that has already been transferred. In order to update the database as CMOVEs progress, it is required that the data pass through a Nexus network with a database node using the specified database to update it as new data arrives. The database definition uses the typical database settings options.

In the case of the CFIND Based CMOVE Controller, there are three modes of operation; Live Polling, Historic Backwards and Historic Forwards. Live polling will continuously look for new data arriving that day and move that data. The historic modes will perform a CFIND over a period of time broken up into specified smaller periods in order to move all data within the defined data range. The data can be filtered further using the tag filter to define what query parameters will be used in the CFIND request.

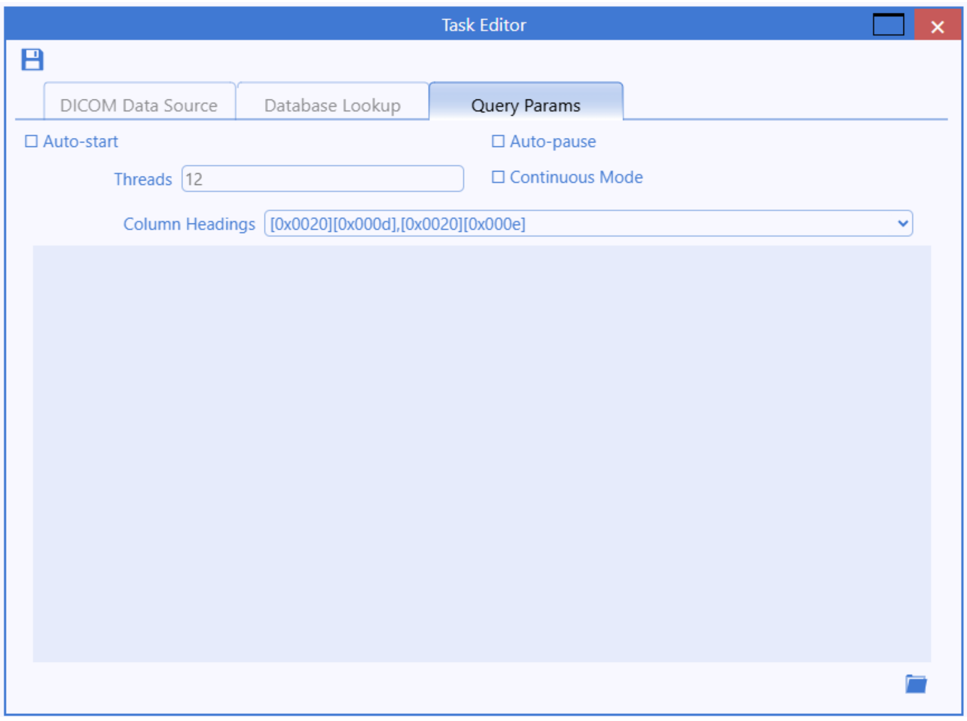


☐ Auto-start
☐ Auto-pause
Threads 12
☑ Continuous Mode
Query Mode Live Polling
Query Level Series Level
Query Period (Days) 1
Query Repeat (Seconds) 30

Figure 154: CFIND Based Move Controller query parameters and mode selection.

The auto-start option will indicate that the jobs should start when the Nexus service starts or the configuration is applied. Auto-pause will configure the task to halt if it encounters an error. The number of threads specifies how many workers are assigned to the task. The number specified defines the number of individual CMOVE processes that will run in parallel once the list of jobs has been established. Continuous mode indicates that once the list of jobs has been completed, the process will start over.

As an alternative to driving the CMOVE process with a general query-based method, a specific list of studies to be transferred can be used to drive the process. Figure 154 show the dialogue used to select a CSV file. The folder button at the bottom right opens the file selection dialogue. The column headings are a pre-set list of options (study UID, series UID or Patient ID, accession number) but the heading list can be customised using a comma separated list of DICOM tag values which should match the columns in the CSV file.


Task Editor

DICOM Data Source | Database Lookup | Query Params

☐ Auto-start
☐ Auto-pause
Threads: 12
☐ Continuous Mode
Column Headings: [0x0020][0x000d],[0x0020][0x000e]

Figure 155: List Based Move Controller tag definition and list selection.

When all the settings have been specified, selecting the save button at the top left of the dialogue will push the configuration to the server and a task icon will be displayed on the task bar as shown in Figure 156. Right-clicking on the task icon will display the control menu for the task where it can be started/stopped etc.

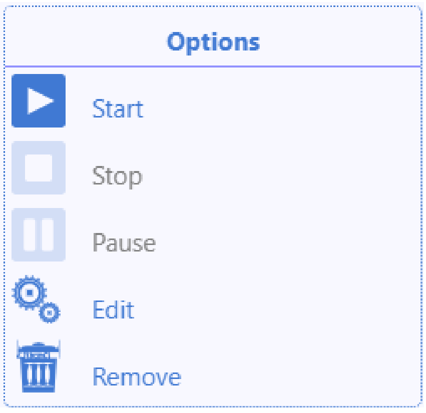


Figure 156: Task menu

6 Extended Services

There are secondary services that can be started as part of a Nexus service which can add extra applications if you have the correct license. Typically, these services are enabled in the deployment properties file in the install folder and may require additional configuration. Detailed manuals for some these services are provided separately but an overview of the functionality and configuration is provided in the following sections.

The launchable apps are controlled using the apps.json config file.

```
    {install_location}/docs/config/server/apps/apps.json
```

The apps.json file defines which application entry points are available to users when the apps.html page is used.

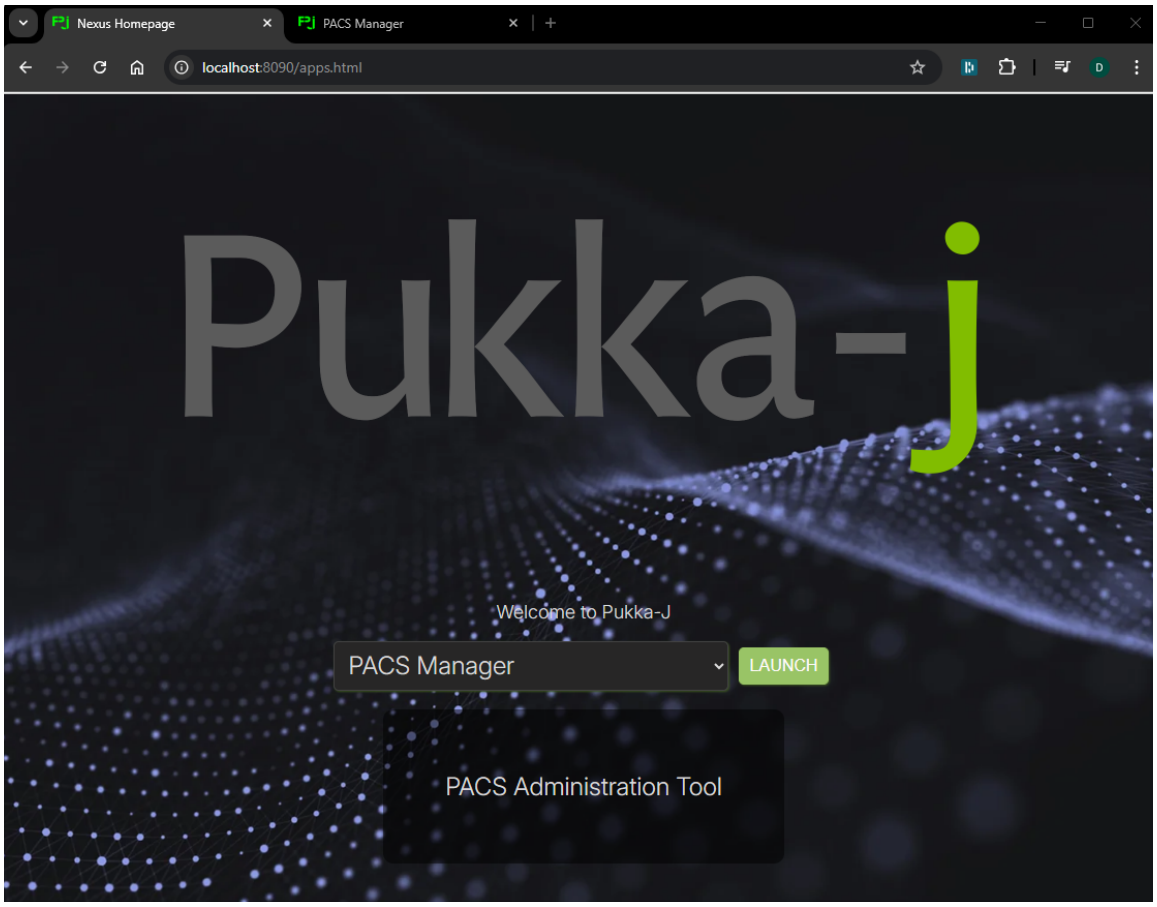


Figure 157: Pukka-j Application Launcher Homepage.

Figure 157 shows the homepage served out by the apps.html file. The drop down options are defined by the apps.json file. Selecting an option and pressing the launch button will open the selected app on a new tab. The example shows the PACS Manager app being selected and the tab which appears when launched.

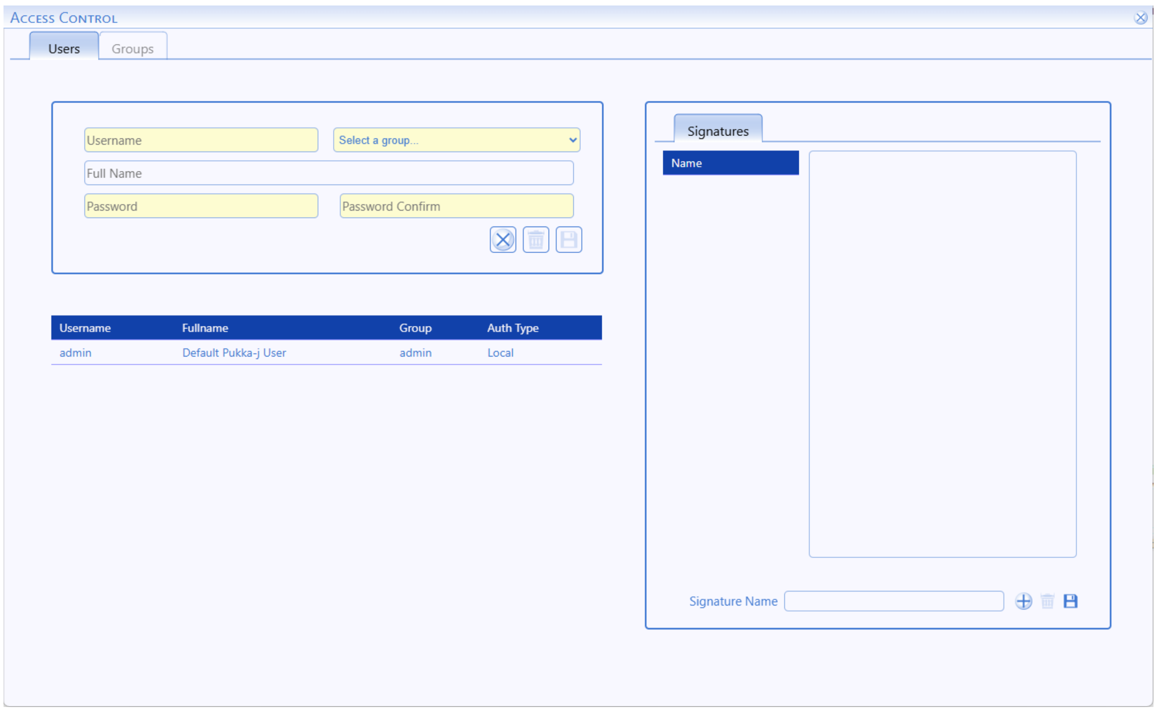
6.1 User and Groups Manager

The user and user groups are managed by a management panel that can be accessed as a stand-alone application typically from the App Launcher home page, or by browsing to the users.html page on the host where Nexus is installed, e.g.

```
    https://your-nexus-hostname/users.html
```

The user management may also be embedded within certain other applications for convenience.

There are two tabs to the application, one called “Users” and another “Groups” within which the users reside.



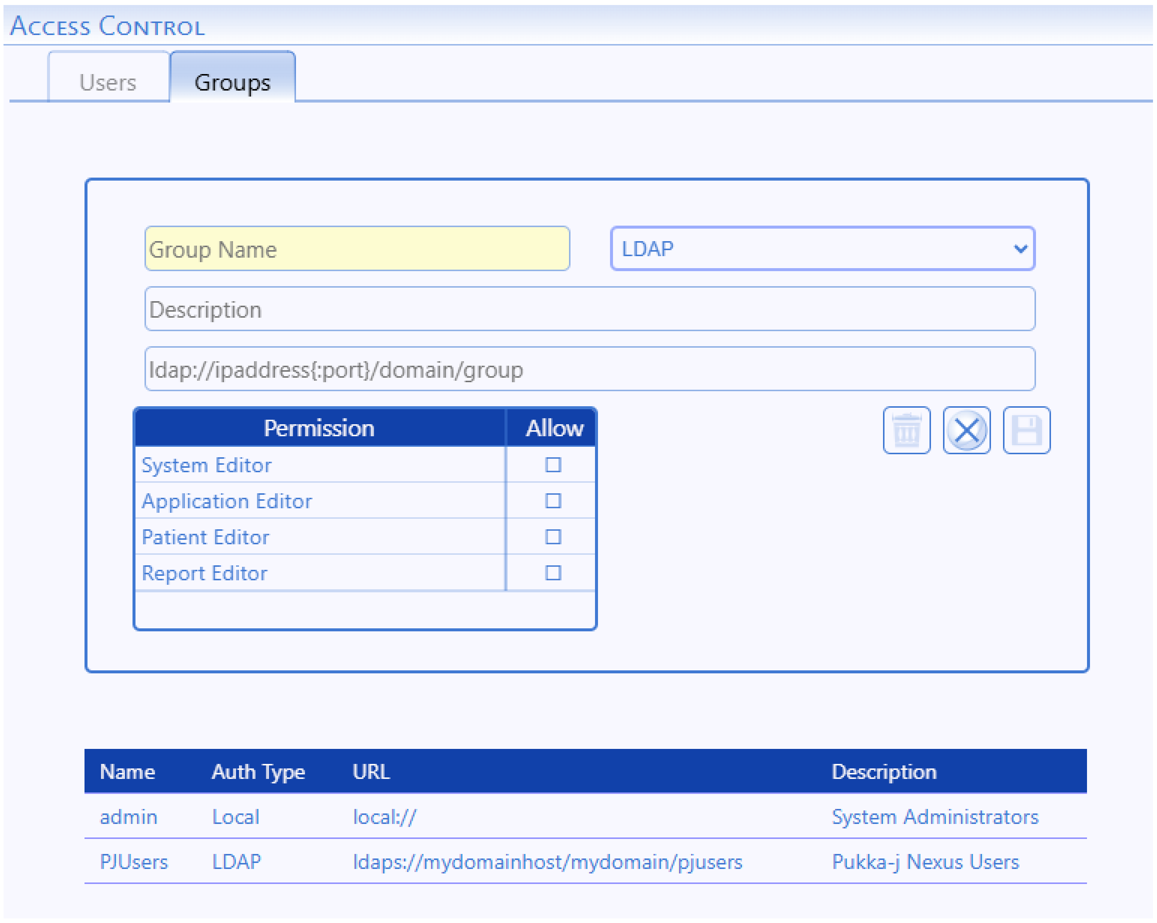
Username	Fullname	Group	Auth Type
admin	Default Pukka-j User	admin	Local

Figure 158: Nexus users management tab.

Figure 158 displays the administrator view, which includes a table of current users and a section for adding or editing user accounts. Users with basic rights do not have access to this user or group management interface; they can only update their own password here, and only if they are local users.

To add a new user, enter the username in the top-left field and select a group from the existing list. Optionally, you can provide the user’s full name. If the user is part of a local group, a password must be entered and confirmed. For externally authenticated groups (e.g., via LDAP), the password fields are not required.

Once all necessary details are filled in, click the “save” icon located below the password fields to add the user to the list. To edit a user, select the corresponding entry in the table and update the fields as needed. You can filter the user table by typing into the username field, which dynamically narrows the list. To delete a user, select the desired entry and click the “trash” icon below the password section.



Permission	Allow
System Editor	☐
Application Editor	☐
Patient Editor	☐
Report Editor	☐

Name	Auth Type	URL	Description
admin	Local	local://	System Administrators
PJUsers	LDAP	ldaps://mydomainhost/mydomain/pjusers	Pukka-j Nexus Users

Figure 159: Nexus group management tab.

Figure 158 shows the administrator view for managing authentication groups. There are two types of groups supported: **LOCAL** and **LDAP**.

Local groups are managed in the system database, and passwords are stored and validated by the system. **LDAP** groups delegate authentication to an external LDAP server, where password management is handled outside the application.

When creating a group, only a group name is required for local groups. For LDAP groups, a `URL` must be defined to redirect authentication to the correct Active Directory (AD) server. The required URL format is shown below:

```
    ldap(s)://ldap-host(:port)/domain/group
```

The protocol prefix should be either `ldap` or `ldaps`, depending on whether secure LDAP is enabled or required. The `ldap-host` should point to the IP address or hostname of the domain controller that provides the LDAP service. If the service is hosted on a non-default port, you can specify it explicitly using the optional `:port` portion in the URL. For example:

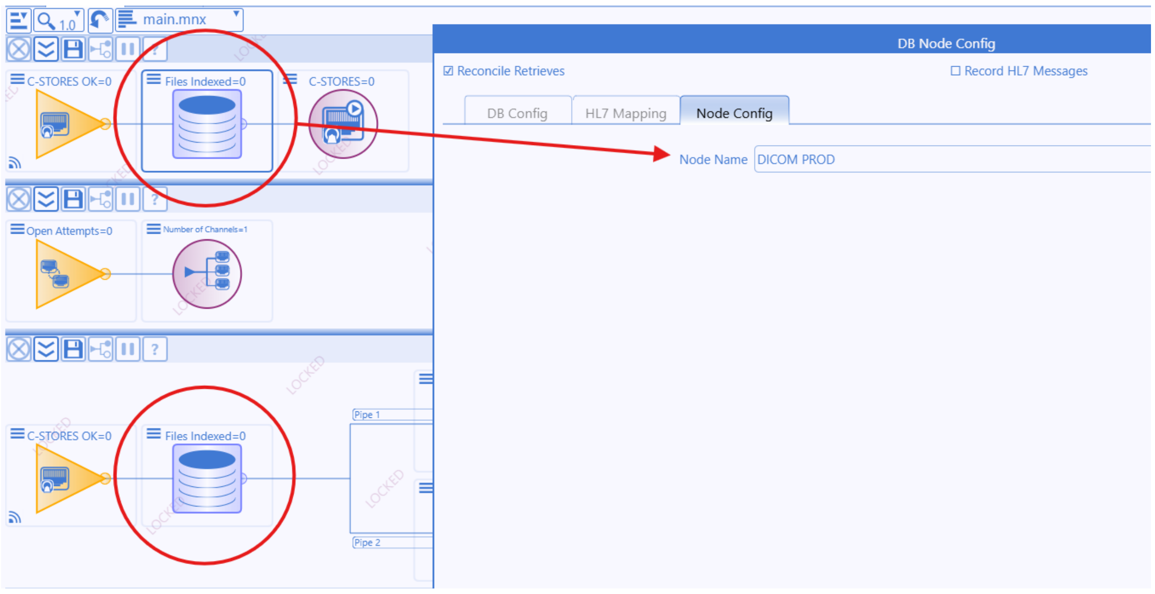
```
ldaps://ldap.example.com:636/mydomain/mygroup
```

If no port is specified, the default port is assumed: `389` for `ldap` and `636` for `ldaps`.

6.2 PACS Manager

The **PACS Manager** application allows users to manage stored DICOM image data and their associated databases.

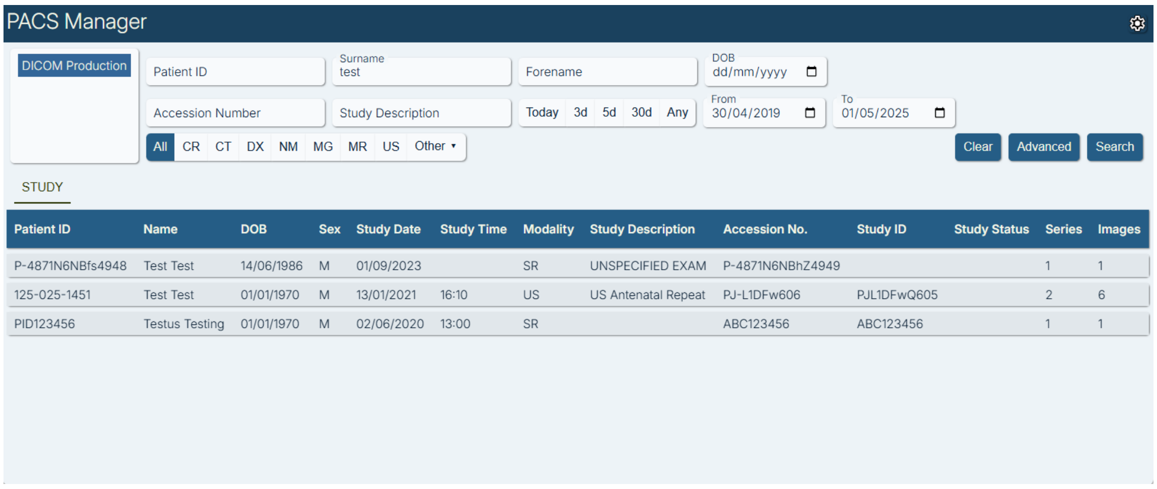
The list of available databases is shown in the top-left panel of the page. These databases correspond to the database nodes created within **Nexus** and are displayed here for management purposes. When creating a database node in Nexus, a node name can be defined in its configuration, which is then shown in this list, as illustrated in Figures 160 and 161.



Node Name: DICOM PROD

Figure 160: Nexus database node and its configured node name.

Once a database is selected, you can query it using the query bar, with options such as date range, modality, and more. An advanced search is also available for filtering by Study ID, Study UID, or Procedure Code. Figure 161 shows the query interface of the PACS Manager application.



Surname: test
DOB: dd/mm/yyyy
From: 30/04/2019
To: 01/05/2025

Patient ID	Name	DOB	Sex	Study Date	Study Time	Modality	Study Description	Accession No.	Study ID	Study Status	Series	Images
P-4871N6NBfs4948	Test Test	14/06/1986	M	01/09/2023		SR	UNSPECIFIED EXAM	P-4871N6NBhZ4949			1	1
125-025-1451	Test Test	01/01/1970	M	13/01/2021	16:10	US	US Antenatal Repeat	PJ-L1DFw606	PJL1DFwQ605		2	6
PID123456	Testus Testing	01/01/1970	M	02/06/2020	13:00	SR		ABC123456	ABC123456		1	1

Figure 161: PACS Manager database query interface.

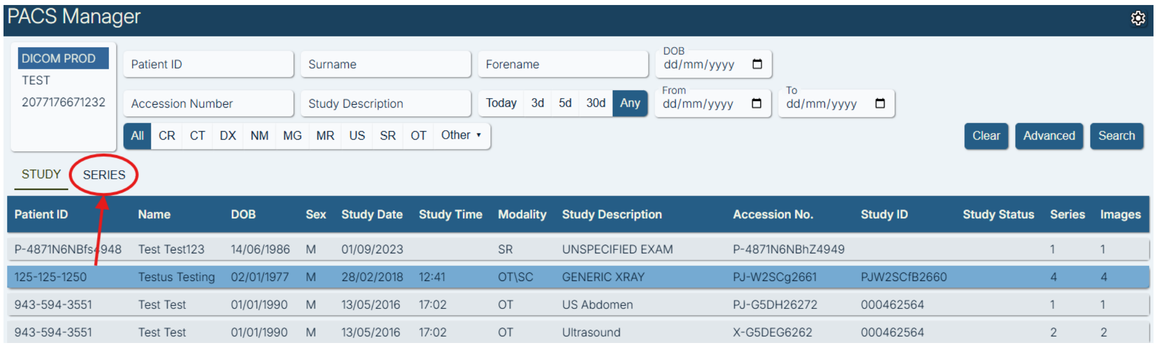
Navigation

Once a search is performed on a selected database, the **Study Table** tab appears, displaying all studies that match the criteria. Each database includes two additional tabs: the **Series Table** and the **Image Table.** Each table provides its own set of modification options, which are discussed in Section 6.2.

To navigate:

- Select a study in the Study Table to access its corresponding Series Table.
- From the Series Table, select a series to access the Image Table.
- Alternatively, double-clicking a study or series will open the respective next level directly.

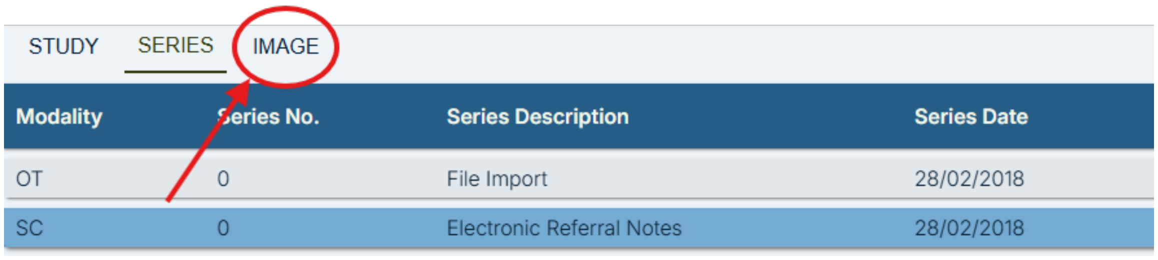
Figures 162 and 163 illustrate the Study and Series Table views.



DOB: dd/mm/yyyy
From: dd/mm/yyyy
To: dd/mm/yyyy

Patient ID	Name	DOB	Sex	Study Date	Study Time	Modality	Study Description	Accession No.	Study ID	Study Status	Series	Images
P-4871N6NBfs4948	Test Test123	14/06/1986	M	01/09/2023		SR	UNSPECIFIED EXAM	P-4871N6NBhZ4949			1	1
125-125-1250	Testus Testing	02/01/1977	M	28/02/2018	12:41	OT\SC	GENERIC XRAY	PJ-W2SCg2661	PJW2SCfB2660		4	4
943-594-3551	Test Test	01/01/1990	M	13/05/2016	17:02	OT	US Abdomen	PJ-G5DH26272	000462564		1	1
943-594-3551	Test Test	01/01/1990	M	13/05/2016	17:02	OT	Ultrasound	X-G5DEG6262	000462564		2	2

Figure 162: Study Table view with a selected study.



Modality	Series No.	Series Description	Series Date
OT	0	File Import	28/02/2018
SC	0	Electronic Referral Notes	28/02/2018

Figure 163: Series Table view with a selected series.

Options

Each table supports right-click context menus with various options depending on the selection:

Multi-selection When multiple rows are selected in any table, the only available action is to delete them, as shown in Figure 164. Once confirmed, this will remove the entries from the database and delete their associated files (if stored).

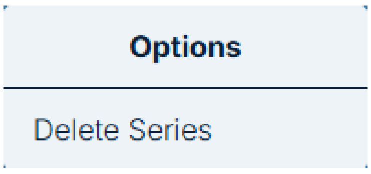


Figure 164: Delete option for multiple row selection.

Study Table Options When a single study is selected, the following options are available:

- **Edit patient details**: Opens a dialog to modify patient information. An option is provided to update the image files themselves with the new details.
- **Edit study**: Allows editing of study-level metadata with an optional update to the underlying files.
- **Set study status**: Change the status of a study by selecting applicable statuses. Status indicators are displayed in the Study Table.
- **Delete study**: Prompts a confirmation dialog to permanently remove the study.
- **Open audit view**: Displays a log of all actions performed on the selected study.

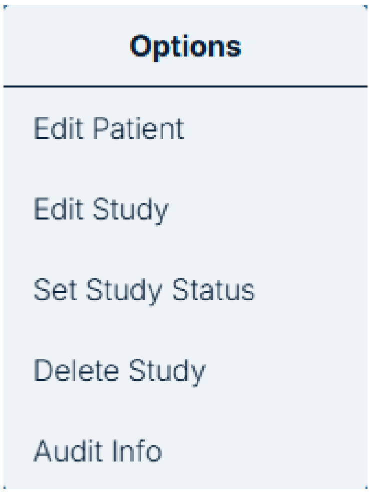
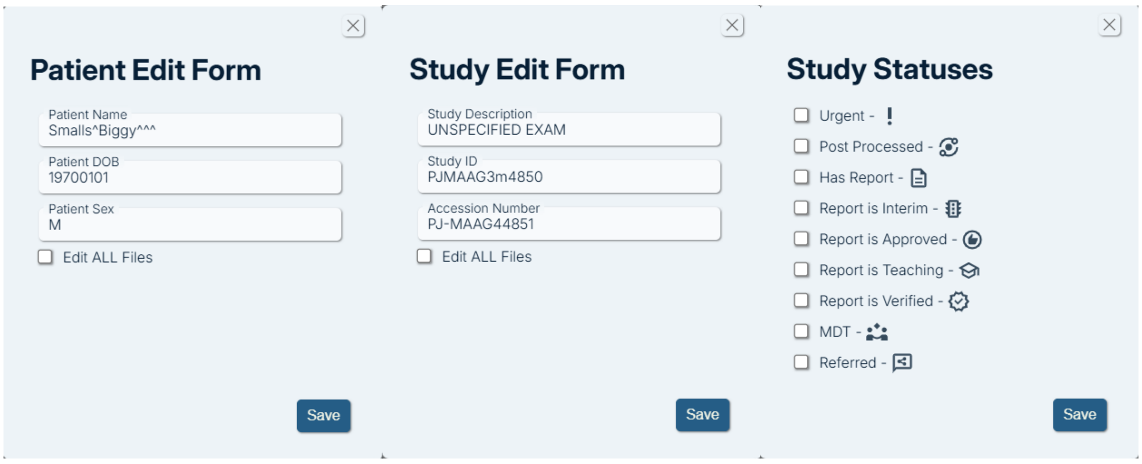


Figure 165: Study options context menu.

Edit dialogs are illustrated in Figure 166.



Patient Edit Form

Patient Name: Smalls^Biggy^^^
Patient DOB: 19700101
Patient Sex: M
☐ Edit ALL Files

Study Edit Form

Study Description: UNSPECIFIED EXAM
Study ID: PJMAAG3m4850
Accession Number: PJ-MAAG44851
☐ Edit ALL Files

Study Statuses

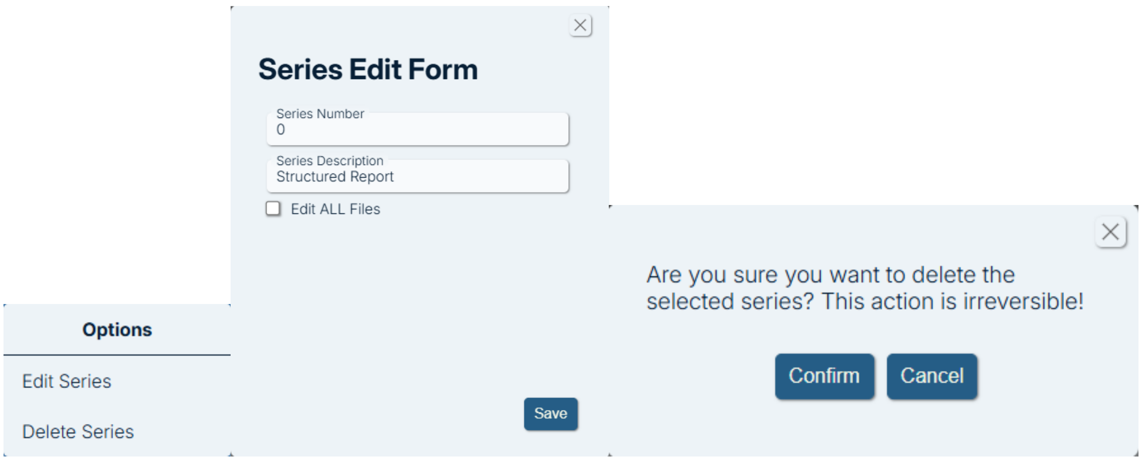
☐ Urgent -
☐ Post Processed -
☐ Has Report -
☐ Report is Interim -
☐ Report is Approved -
☐ Report is Teaching -
☐ Report is Verified -
☐ MDT -
☐ Referred -

(a) Edit Patient Dialog *(b) Edit Study Dialog* *(c) Study Status Dialog*

Figure 166: Edit and status dialogs for a selected study.

Series Table Options The Series Table provides a subset of the Study Table options:

- **Edit series**: Allows modification of series-level details, with an option to update the files.
- **Delete series**: Prompts a confirmation dialog to remove the selected series and associated files.



(a) Series Options Context Menu *(b) Edit Series Dialog* *(c) Delete Confirmation Dialog*

Figure 167: Options and dialogs for a selected series.

Image Table Options In the Image Table, the only available action is to delete one or more selected images. This action also requires confirmation.

6.3 Image Transfer Application

The **Image Transfer** application enables users to search for studies and transfer or export a selected study to customisable destinations. The stand-alone Image Transfer page is configured to transfer studies from a configurable DICOM source that supports DICOM Q/R. An extended version of this application, the **Nexus PACS** application (shown in Section 6.5) allows transfers from any database, as well as the main configurable DICOM source used in Image Transfer.

The main interface of the Image Transfer page is shown below in Figure 168, featuring a DICOM search bar at the top and a central table displaying all results available for transfer. Configuration settings for destinations and the DICOM URL can be accessed via the settings icon in the top-right corner.

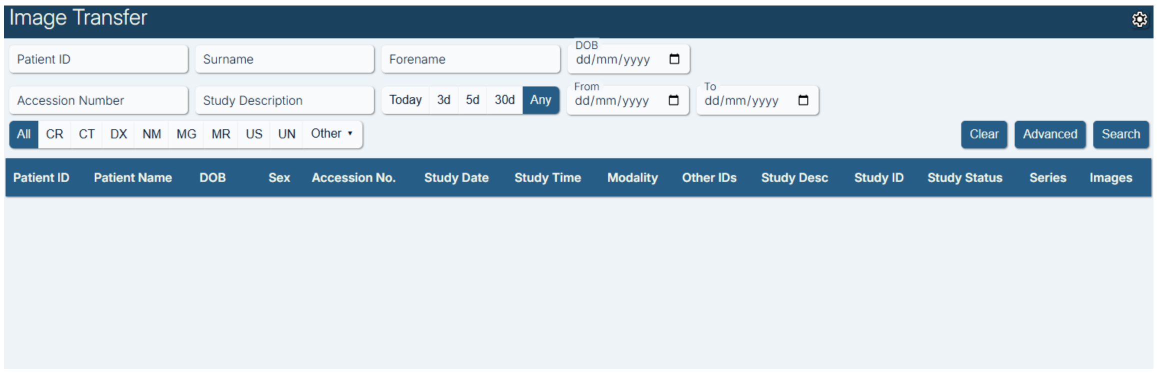


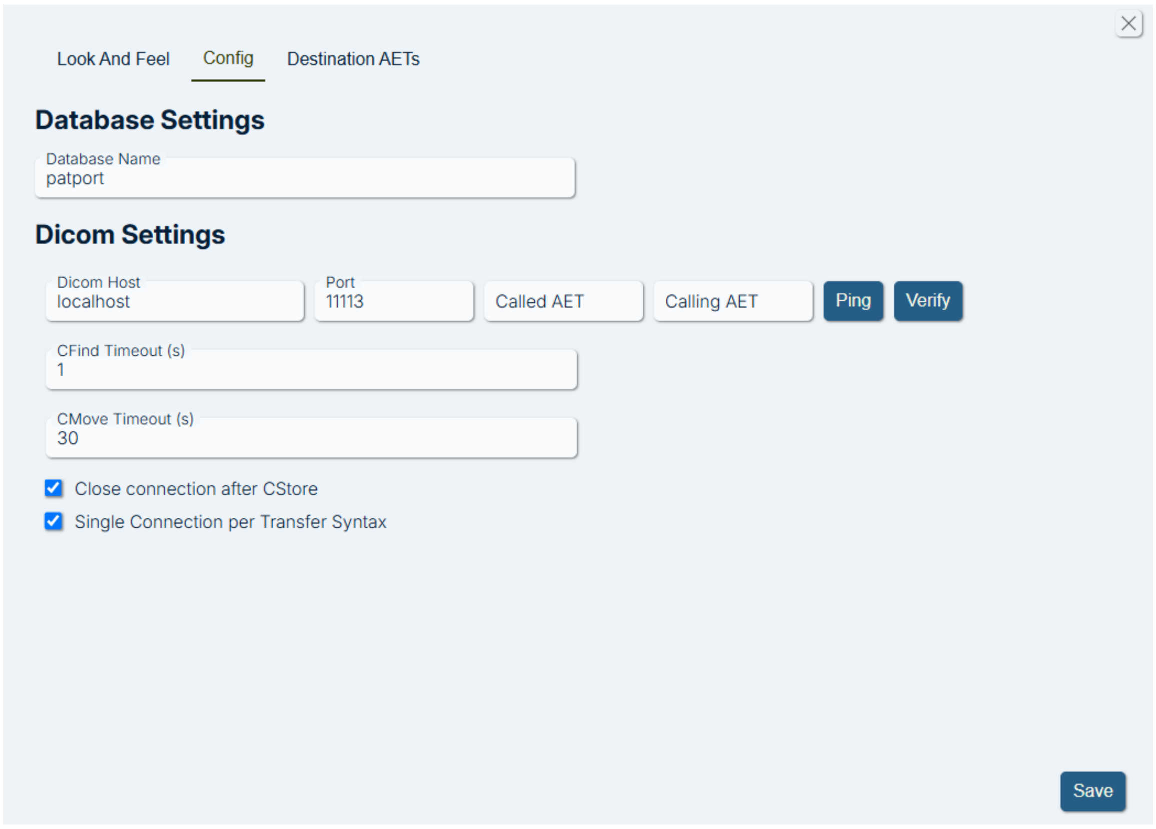
Figure 168: Image Transfer main search page.

6.3.1 Configuration and Set-Up

To set up the Image Transfer application, the service must be enabled in the Nexus *deployment* properties file. Additionally, if exporting to PacsPost or other destinations where ZIP transfer is required, the **File Uploader** service must also be enabled and configured to handle the appropriate destination URL(s).

```
service.imgtransfer.enabled=true
# Below is optional, for exporting
service.fileupload.enabled=true
```

The rest of the configuration can be completed through the frontend user interface, without needing to manually edit Nexus configuration files. Figure 169 displays the main configuration tab for the Image Transfer service. Additional configuration for PacsPost exports is detailed in Section 6.11.1 of the File Upload Service.



Database Settings

Database Name: patport

Dicom Settings

Dicom Host: localhost
Port: 11113
Called AET
Calling AET
CFind Timeout (s): 1
CMove Timeout (s): 30
☑ Close connection after CStore
☑ Single Connection per Transfer Syntax

Figure 169: Image Transfer Configuration Settings

The following is a summary description of each configuration item:

Database name By default, this is the Nexus database. If exporting is required, the database must match that used by the File Uploader service.

DICOM host and IP port Mandatory address items for DICOM query-retrieve.

Called and Calling AET Fill these in to set the AETs on the outbound communication. If left empty, the system will default to called AET *IIS* and calling AET *MINT*. When the Called and Calling AETs are defined, the *Ping* and *Verify* buttons are enabled for checking routing and DICOM association.

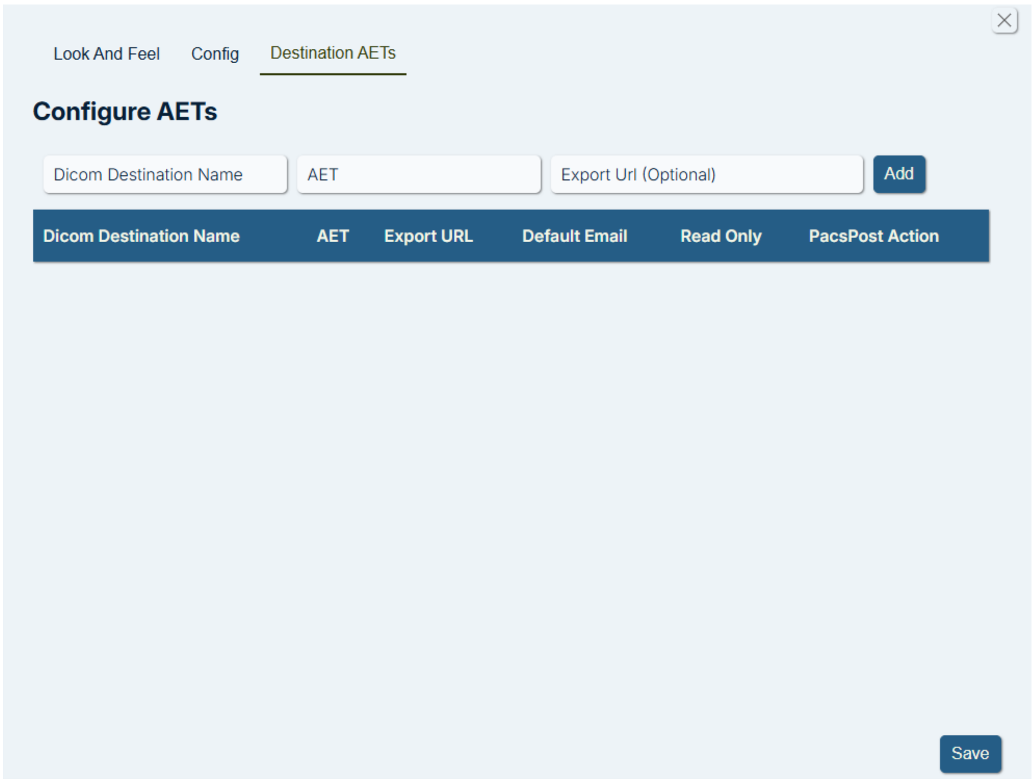
C-FIND timeout Duration in seconds before a query times out.

C-MOVE timeout Duration in seconds before a move request times out.

Closed connection after CStore Forces the output channel to close the association after storing a SOP. When unticked, the output channel attempts to keep the association open (up to the timeout limit) for efficiency.

Single Connection per Syntax Forces the outbound DICOM channel to renegotiate the DICOM association for each different SOP class and transfer syntax (described in *Nexus DICOM Output Node* in Section 5.5.5.

Once the DICOM URL is configured, the Image Transfer application can query studies available for transfer or export. In order to define where the studies can be sent, destinations need to be defined in the "Destination AETs" tab in the settings dialogue, as shown below. These configured AETs must also be known on the destination system (e.g., a Nexus node or a PACS) to ensure compatibility.

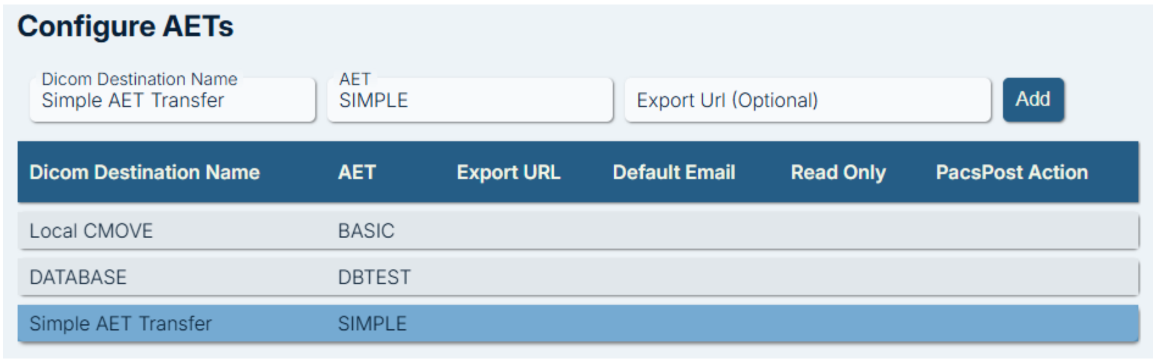


Dicom Destination Name	AET	Export URL	Default Email	Read Only	PacsPost Action

Figure 170: Destination AETs Configuration

There are three categories of destinations that can be set up: **Simple Transfer AET Destinations**, **Custom Export Destinations**, and **PacsPost Export Destinations**.

The simplest setup is a Simple Transfer AET, requiring only a name and destination AET. Transfers are conducted via the configured DICOM destination, which may point to a PACS or a Nexus node, such as the DICOM Proxy (ideal for multiple Q/R targets). This type of transfer uses a standard DICOM CMOVE request to a destination AET to trigger the selected study to be sent to the destination AET.



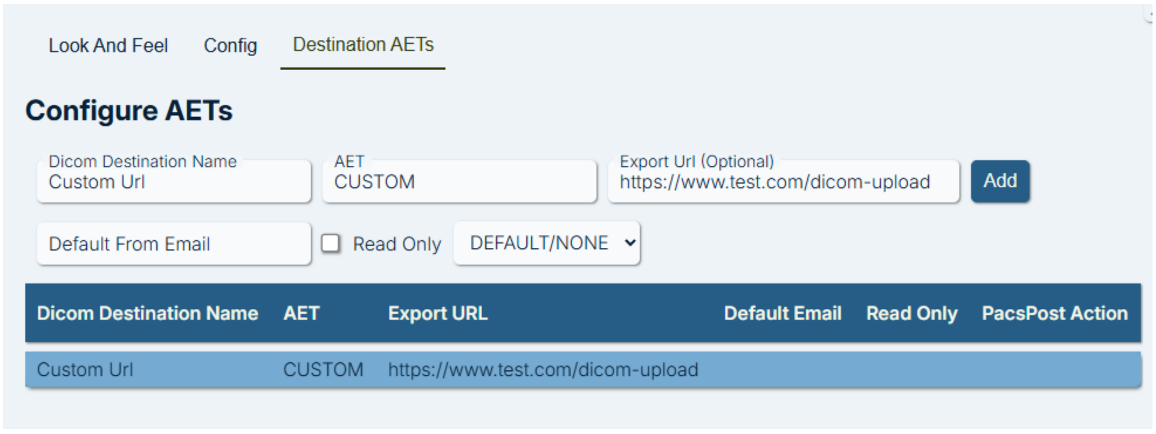
Configure AETs

Dicom Destination Name: Simple AET Transfer
AET: SIMPLE
Export Url (Optional)
Add

Dicom Destination Name	AET	Export URL	Default Email	Read Only	PacsPost Action
Local CMOVE	BASIC				
DATABASE	DBTEST				
Simple AET Transfer	SIMPLE				

Figure 171: Simple Transfer AET Example

An extension to Simple Transfers are exports, which require the File Uploader service and a Custom Export Destination configured as shown in Figure 172. This process uses the same CMOVE request as a standard DICOM transfer but the destination AET should point to a pipeline with a Study Zipper. The zipped study is exported to the URL configured in the File Upload service. The upload destination URL must be included in the ‘additionalUrls‘ setting, along with any required keystore files or certificates if using SSL. Full details are provided in Section 6.11.1.



Configure AETs

Dicom Destination Name: Custom Url
AET: CUSTOM
Export Url (Optional): https://www.test.com/dicom-upload
Add
Default From Email
☐ Read Only
DEFAULT/NONE

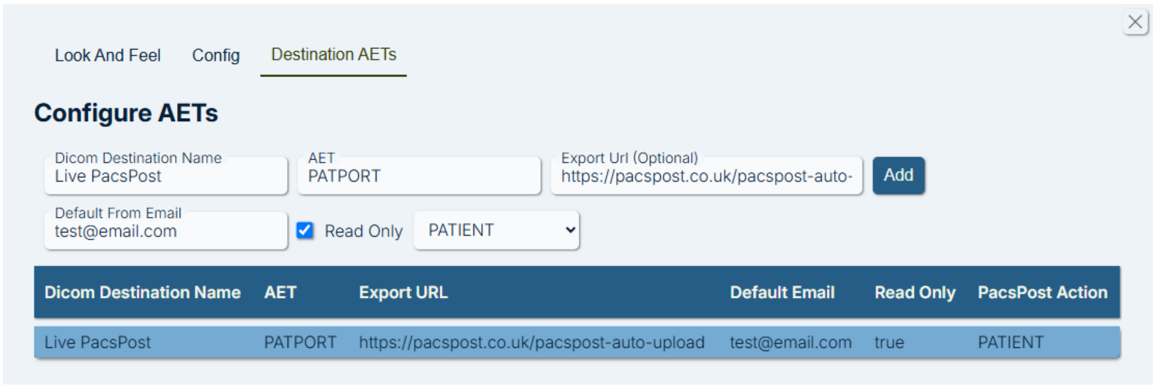
Dicom Destination Name	AET	Export URL	Default Email	Read Only	PacsPost Action
Custom Url	CUSTOM	https://www.test.com/dicom-upload			

Figure 172: Custom Export Destination Example

To configure a PacsPost Export Destination (Figure 173), which extends a Custom Export Destination, additional fields are required to be sent to PacsPost, such as the “From Email” address, read-only settings, and the type of upload action. These options can be customised in the Image Transfer JSON as shown:

```
{
    "appName" : "Image Transfer",
    "pacspostActions" : [["","DEFAULT/NONE"], ["PATIENT", "PATIENT"],
        ["REFERRER", "REFERRER"], ["OTHER", "OTHER"]]
}
```

The upload action defines what actions are available to the recipient when the data is uploaded. Selecting the **Read Only** option fixes the "from address" in the transfer to prevent it from being altered at the point of export.



Look And Feel | Config | Destination AETs

Configure AETs

Dicom Destination Name: Live PacsPost
AET: PATPORT
Export Url (Optional): https://pacspost.co.uk/pacspost-auto-
Add

Default From Email: test@email.com
☑ Read Only
PATIENT

Dicom Destination Name	AET	Export URL	Default Email	Read Only	PacsPost Action
Live PacsPost	PATPORT	https://pacspost.co.uk/pacspost-auto-upload	test@email.com	true	PATIENT

Figure 173: PacsPost Export Destination Example

6.3.2 Nexus Set-Up Example

An example setup template for configuring the Image Transfer application with PacsPost exporting in Nexus is shown in Figure 174.

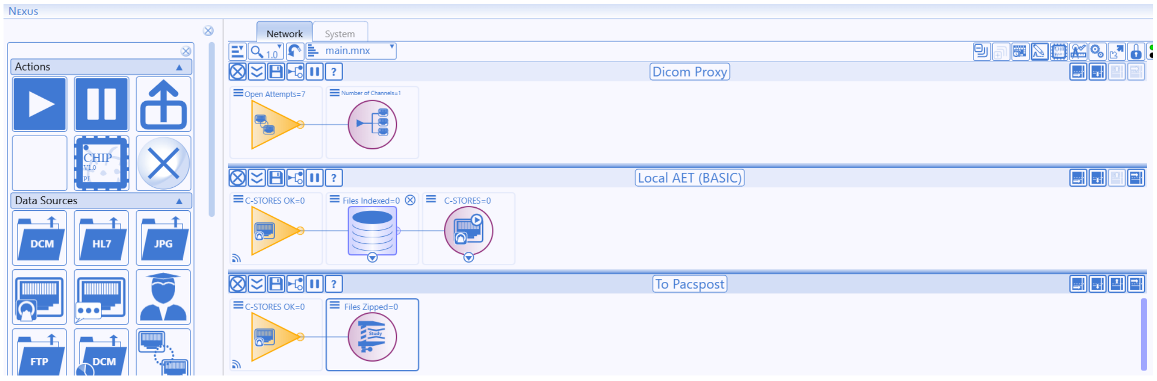


Figure 174: Example Nexus Set-Up for Image Transfer

The following summaries give descriptions of each pipeline in the network :

- The top pipeline is a DICOM Proxy that can connect to multiple DICOM sources, including internal pipelines or external PACS. The DICOM URL configuration in the Image Transfer application can point to this proxy to give a single source of studies. The DICOM proxy passes any CMOVE (transfer) requests to the destinations defined in the proxy output node. The connected systems must know of any destination AET sent to them. See Section **??** for more details on how to set this up.

- The second pipeline demonstrates a basic DICOM listener that stores received DICOM data into a database. In the example, the destination AET "BASIC" is used, and all proxy destinations are configured to use this AET and point to this pipeline's port. Any destination AET must be defined in the configurations section of the input node.
- The third pipeline is configured with the destination AET "PACSPOST", which is used for export jobs. Transferred studies are zipped using the Study Zipper node and then sent via the File Uploader service to PacsPost. Systems must have the IP and Port for the input node on this pipeline as a destination AET and that destination AET is used to trigger the data to be sent to this pipeline and thus uploaded.

6.3.3 Transferring/Exporting

Once set up, studies can be selected for transfer through a simple search and click. Figure 175 shows a selected study with a list of configured destinations.

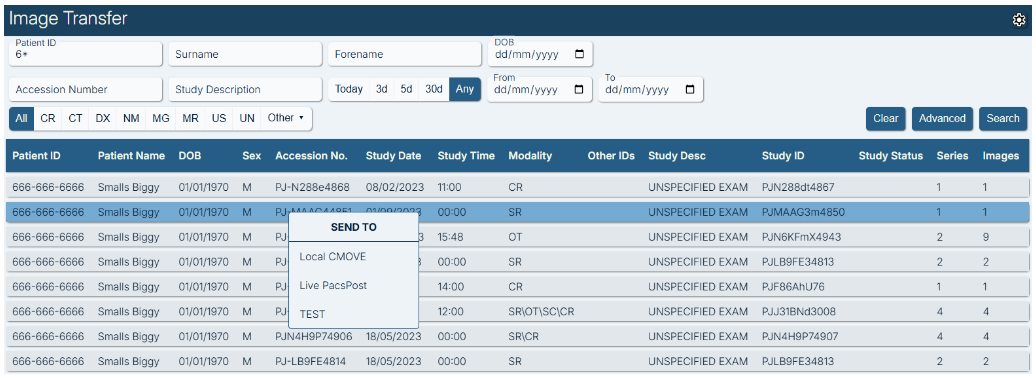


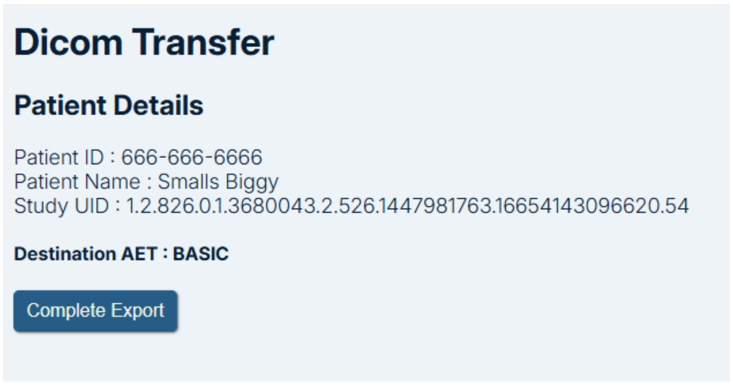
Image Transfer

Patient ID: 6*
Surname
Forename
DOB: dd/mm/yyyy
Accession Number
Study Description
Today 3d 5d 30d Any
From: dd/mm/yyyy
To: dd/mm/yyyy
All CR CT DX NM MG MR US UN Other
Clear Advanced Search

Patient ID	Patient Name	DOB	Sex	Accession No.	Study Date	Study Time	Modality	Other IDs	Study Desc	Study ID	Study Status	Series	Images
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-N288e4868	08/02/2023	11:00	CR		UNSPECIFIED EXAM	PJN288dt4867		1	1
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-MAAG44851	01/09/2023	00:00	SR		UNSPECIFIED EXAM	PJMAAG3m4850		1	1
666-666-6666	Smalls Biggy	01/01/1970	M	PJ	[illegible]	15:48	OT		UNSPECIFIED EXAM	PJN6KFmX4943		2	9
666-666-6666	Smalls Biggy	01/01/1970	M	PJ	[illegible]	00:00	SR		UNSPECIFIED EXAM	PJLB9FE34813		2	2
666-666-6666	Smalls Biggy	01/01/1970	M	PJ	[illegible]	14:00	CR		UNSPECIFIED EXAM	PJF86AhU76		1	1
666-666-6666	Smalls Biggy	01/01/1970	M	PJ	[illegible]	12:00	SR\OT\SC\CR		UNSPECIFIED EXAM	PJJ31BNd3008		4	4
666-666-6666	Smalls Biggy	01/01/1970	M	PJN4H9P74906	18/05/2023	00:00	SR\CR		UNSPECIFIED EXAM	PJN4H9P74907		4	4
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-LB9FE4814	18/05/2023	00:00	SR		UNSPECIFIED EXAM	PJLB9FE34813		2	2

Figure 175: Send To Context Menu

If a simple transfer is selected, a confirmation dialogue appears, as shown in Figure 176.


Dicom Transfer

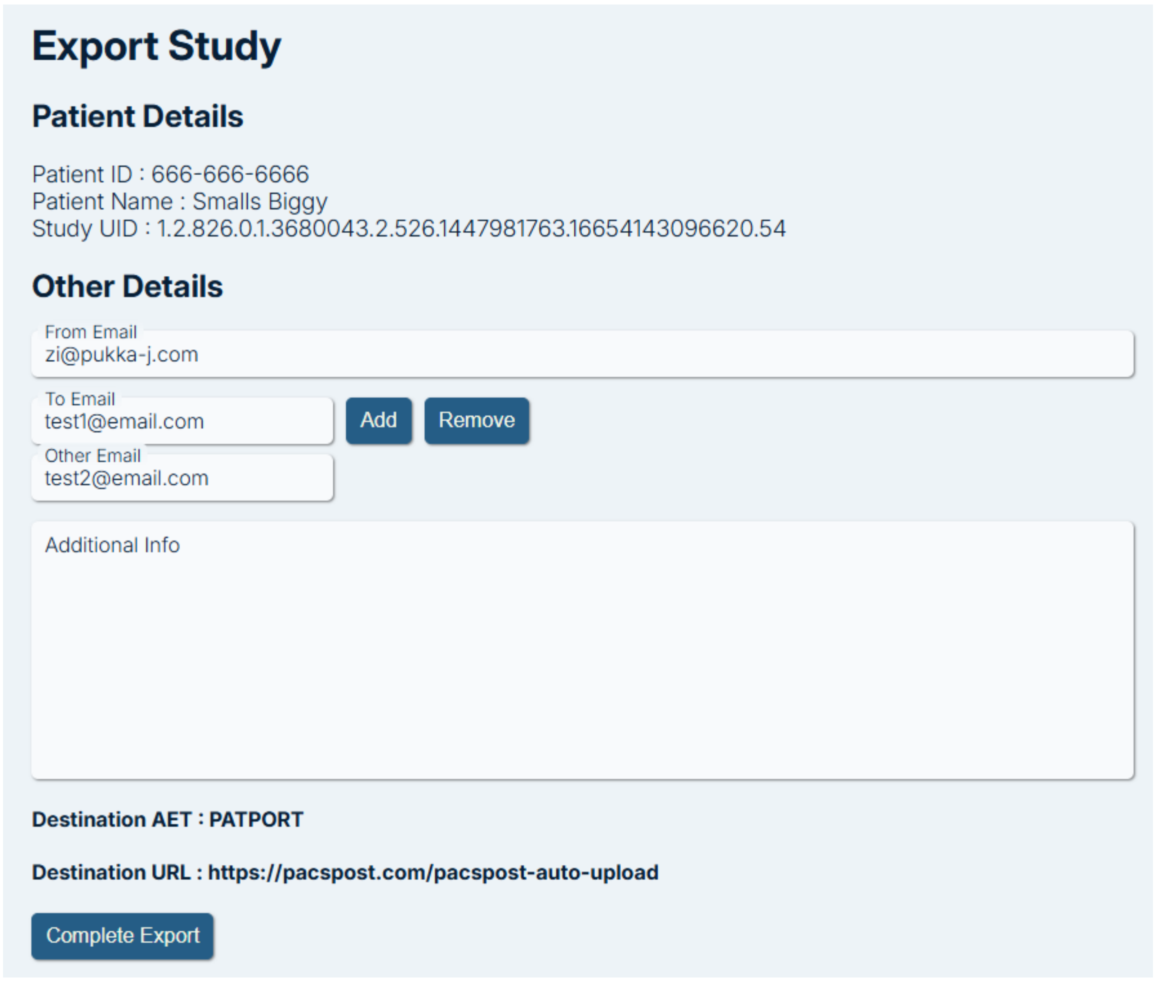
Patient Details

Patient ID : 666-666-6666
Patient Name : Smalls Biggy
Study UID : 1.2.826.0.1.3680043.2.526.1447981763.16654143096620.54

Destination AET : BASIC

Figure 176: Simple Transfer Dialogue

If an export destination is selected, a more detailed form appears (Figure 177). This form, sent as a JWT payload to PacsPost along with the zipped study, allows for optional fields such as **From Email**, **Recipient Details**, and an optional **message**.



Export Study

Patient Details

Patient ID : 666-666-6666
Patient Name : Smalls Biggy
Study UID : 1.2.826.0.1.3680043.2.526.1447981763.16654143096620.54

Other Details

From Email: zi@pukka-j.com

To Email: test1@email.com

Other Email: test2@email.com

Additional Info

Destination AET : PATPORT

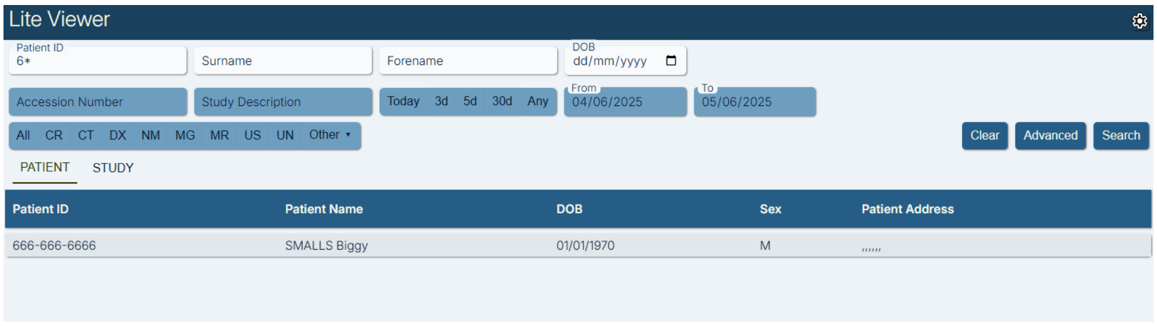
Destination URL : https://pacspost.com/pacspost-auto-upload

Figure 177: Export Transfer Dialogue

This form is designed for PacsPost and can be edited unless the “From Email” was configured as read-only. You may add or remove multiple recipients and enter a description. Once submitted, an export job is created. The zipped study is processed through the Study Zipper pipeline, and the File Uploader service matches the configured URL to initiate the upload to PacsPost. The specified recipients can then view the sent study via PacsPost.

6.4 PukkaJ LiteViewer

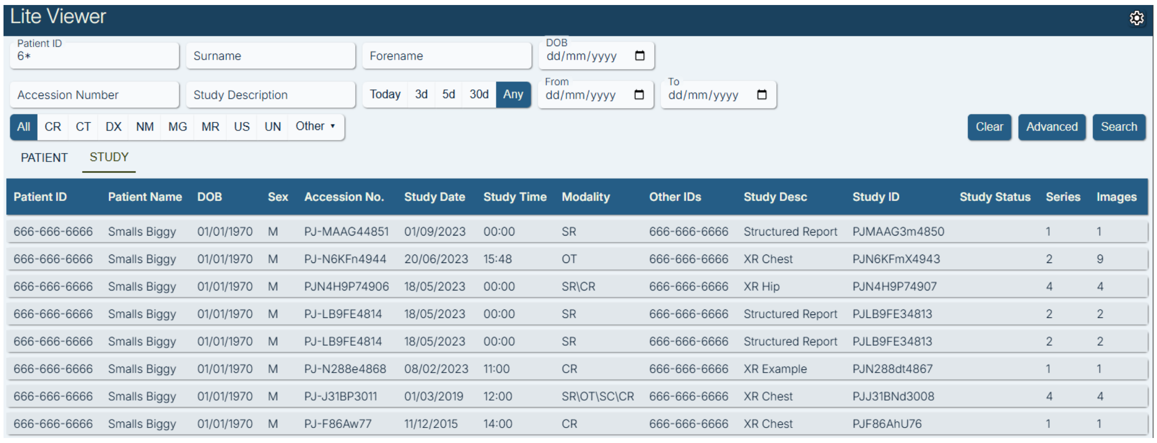
The **PukkaJ Liteviewer** is automatically enabled and utilises the main database configured for Nexus in the *deployment.properties* file. On the main page of the Liteviewer, users can perform a patient search, which returns a list of available patients from the database who have associated images to view (see Figure 178).



Patient ID	Patient Name	DOB	Sex	Patient Address
666-666-6666	SMALLS Biggy	01/01/1970	M	,,,,,,

Figure 178: Liteviewer Patient Search Screen

Additionally, users can select the `Study` tab to search for studies which will return a list of avaliable studies to view (see Figure 179).



Patient ID	Patient Name	DOB	Sex	Accession No.	Study Date	Study Time	Modality	Other IDs	Study Desc	Study ID	Study Status	Series	Images
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-MAAG44851	01/09/2023	00:00	SR	666-666-6666	Structured Report	PJMAAG3m4850		1	1
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-N6KFn4944	20/06/2023	15:48	OT	666-666-6666	XR Chest	PJN6KFmX4943		2	9
666-666-6666	Smalls Biggy	01/01/1970	M	PJN4H9P74906	18/05/2023	00:00	SR\CR	666-666-6666	XR Hip	PJN4H9P74907		4	4
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-LB9FE4814	18/05/2023	00:00	SR	666-666-6666	Structured Report	PJLB9FE34813		2	2
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-LB9FE4814	18/05/2023	00:00	SR	666-666-6666	Structured Report	PJLB9FE34813		2	2
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-N288e4868	08/02/2023	11:00	CR	666-666-6666	XR Example	PJN288dt4867		1	1
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-J31BP3011	01/03/2019	12:00	SR\OT\SC\CR	666-666-6666	XR Chest	PJJ31BNd3008		4	4
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-F86Aw77	11/12/2015	14:00	CR	666-666-6666	XR Chest	PJF86AhU76		1	1

Figure 179: Liteviewer Study Search Screen

6.4.1 JWT Configuration

By default, Nexus includes a pre-configured PukkaJ JWT (JSON Web Token) which the Liteviewer uses. However, if a different JWT is required, the configuration can be updated to use a custom JWT instead. This configuration is located in the default configuration path:

```
/[install_directory]/docs/config/server/scm/scm-config.txt
```

Within this file, you can define the custom issuer name and the JWT expiry time (in seconds). The specified issuer must also exist in the "jwt.json" file, which defines the corresponding key file location and the algorithm to be used. Further details on JWT configuration can be found in Section 6.12.

6.4.2 Image Viewing

After performing a patient search and selecting a patient, the Liteviewer generates a JWT using its configuration, including the necessary user and patient details. A new window is then opened, displaying the patient's studies with the most recent study expanded by default as the primary study.

The remaining studies that this patient may have will be on the left as prior studies. Various series can be selected from any expanded studies from the panel on the left, which are then displayed on the main viewing screen to the right.

Alternatively, if a study search is performed and a study was selected, the Liteviewer will still generate a JWT and open a new window but will have the selected study as the primary study and the other studies that exist for the patient as their priors.

An example is illustrated in Figure 180, where an older study was selected.

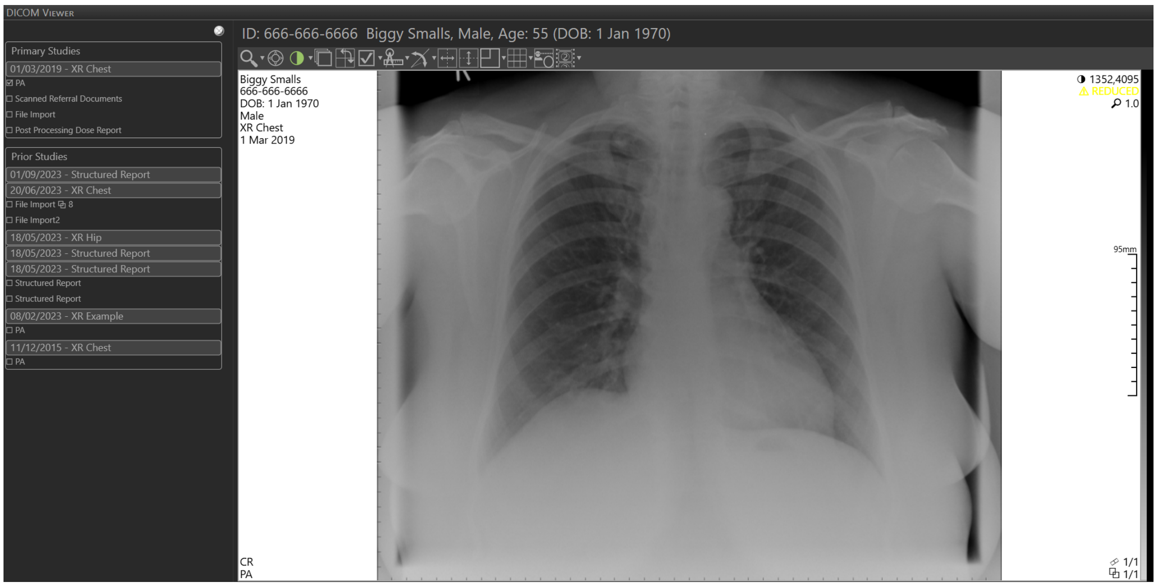


Figure 180: Liteviewer Image Display with some priors expanded

6.4.3 Viewing Tools

The Liteviewer includes a range of image viewing tools, accessible via the toolbar, as illustrated in Figure 181.

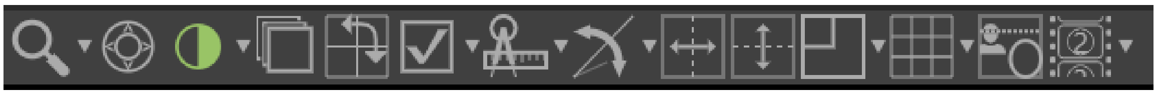
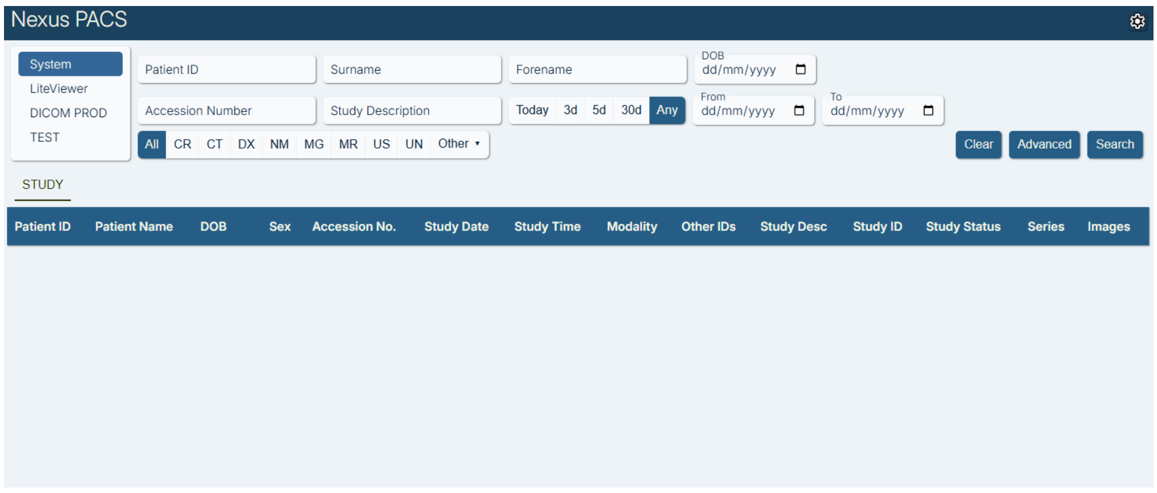


Figure 181: Liteviewer Toolbar

6.5 Nexus PACS

Nexus PACS is a combination of the PACS Manager (see Section 6.2), Image Transfer (see Section 6.3) and the LiteViewer (see Section 6.4). It merges all these functionalities into a unified main interface, providing convenient access for users requiring full capabilities of the named systems.

The configuration for Image Transfer can be set as previously described, via the settings menu in the top-right corner. Once configured, it appears as a data source labelled `System` in the data source list located at the top-left. The `LiteViewer` datasource is also avaliable in this list and is connected to the main database which will provide the option to view any of it's queried patients and studies. Additionally, any database nodes present in Nexus will also appear in this list, as illustrated in Figure 182.



Patient ID	Patient Name	DOB	Sex	Accession No.	Study Date	Study Time	Modality	Other IDs	Study Desc	Study ID	Study Status	Series	Images

Figure 182: Nexus PACS Main Screen

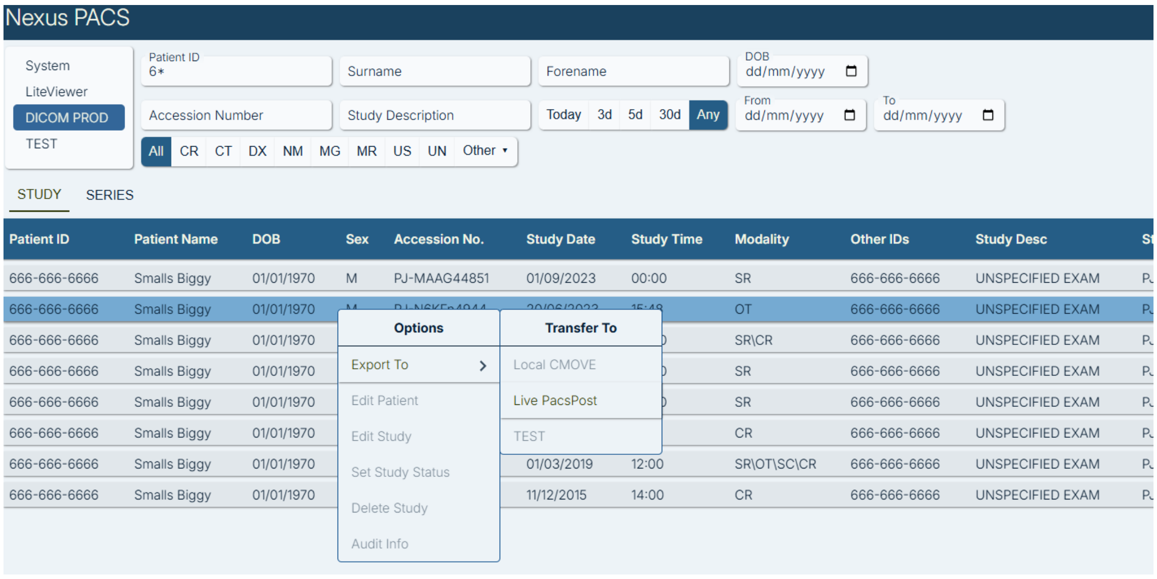
Query searches will be performed on the currently selected data source, and the results will be displayed in a table beneath. The available actions for each result will depend on the capabilities and access permissions associated with the selected source.

6.5.1 Nexus PACS Options

When a data source linked to a Nexus database node is selected — similar to the PACS Manager — the results allow for various operations such as editing, deleting, and viewing related audit logs.

With the exception of the `System` and `LiteViewer` data sources, where `System` is configured to a DICOM URL and `LiteViewer` is configured to the main database for Image Viewering. All other data sources support PACS Manager functionality.

These additional database data sources enable direct access to both Image Transfer and PACS Manager operations, without needing to modify the `System` DICOM destination. All available options are shown in Figure 183.



Patient ID	Patient Name	DOB	Sex	Accession No.	Study Date	Study Time	Modality	Other IDs	Study Desc	St
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-MAAG44851	01/09/2023	00:00	SR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970	M	PJ-N6KFn4944	20/06/2023	15:48	OT	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970					SR\CR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970					SR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970					SR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970					CR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970			01/03/2019	12:00	SR\OT\SC\CR	666-666-6666	UNSPECIFIED EXAM	P
666-666-6666	Smalls Biggy	01/01/1970			11/12/2015	14:00	CR	666-666-6666	UNSPECIFIED EXAM	P

Figure 183: Nexus PACS Options

6.6 Workflow Manager

The workflow manager (WFM) is a service used for triggering imports or moves of data for the purposes of optimising user workflow to ensure data is where it is needed when it is needed. It can be used in conjunction with Nexus pipelines to route data to multiple destinations. The WFM works using the concept of a job, primarily a WorkflowJob in the database. The jobs can be created via different methods, primarily; DICOM, HL7, CSVor manually created. There are some dependant services used by the workflow manager which summarised in the following sections.

The workflow manager service is enabled using the property shown below.

```
    service.wflow.enabled=true/false
```

6.6.1 Pre-processor - Priors Selection

When a job is submitted to the workflow manager, a pre-processor uses the details in the job to look up the patient history on the PACS defined in the config. Using the details for the primary study of the workflow job, the pre-processor uses the configuration to choose the best matching priors available on the PACS and create a DicomImportJob for them along with the primary (if required). The pre-processor can also be configured to send a HL7 stream of the full patient history on the PACS along with the details of which priors were selected and progress updates as each are imported.

6.6.2 RIS Import Service

In addition to the PACS connection, the WFM can use a sub-service to connect to a RIS to look up booking information for the primary job and to also import any previous reports for the patient.

This functionality will depend upon the RIS involved and whether the RIS supports HL7 and HTTP connections. A fully featured install is able to retrieve all HL7 reports and future booking information for a patient and also download and convert to DICOM any scanned documents associated with the primary study/order.

6.7 Tele-Radiology Reporting Services

The tele-radiology service is associated with the workflows for providing a reporting and reporting management solution. Different applications/views on to the platform are possible dependant upon requirements and can be tailored for specific workflows via configuration options.

To enable this service, the property below should be set.

```
service.trad.enabled=true/false
```

6.8 Purge Service

The purge service is a service extension to the Nexus service which allows database and files to be automatically deleted from the system. Multiple services can be configured, one for each database on the system. The deployment.properties file in the install folder should be updated with the following settings to enable purging and define the database(s) on which it should run.

```
service.purge.enabled=true/false
service.purge.databases=dbname1;dbname2
```

Each database has it's own configuration file and the location of these properties files can be defined by the config root property. A prg-dbname-config.properties file should be created for each database to which purging is to be applied.

```
purge.config.root=docs/config/server/purger
prg-{dbname}-config.properties
```

The settings in the specific DB purging config files define the application/service database that will be used to find files on the file system and will have entries removed or updated as part of the purging process. The database name in the properties and the property filename should match. The database properties are listed here.

```
prg_dbtype=Mysql
prg_dbname={dbname}
prg_dbhost=localhost
prg_dbport=3306
prg_dbuser={your_user}
prg_dbpwrd=
prg_dbisnt=
prg_dbsec=false
```

```
    prg_dbautotrust=false
    prg_dbssl=false
```

The purge service can either be configured with it's own rules for what needs to be purged and when, or it can connect to an external system to be notified of studies or orders to be purged. The external connector type that is currently supported is RabbitMQ. A queue can be configured to notify the service that a study UID or a patientID + accession number needs to be removed from the system. The following properties are used to define the connector details for RabbitMQ. The host and login details will depend upon the local configuration of the RabbitMQ service. Port 5672 is the default port used by RabbitMQ.

```
    prg_service_workers=1
    purge-connector-type=RABBIT_MQ

    purge-connector-host={your_host}
    purge-connector-port=5672
    purge-connector-queue={name_of_queue}
    purge-connector-user={user_for_connector}
    purge-connector-pw{pw_for_connector}

    purge-connector-queue-durable=true/false
    purge-connector-queue-exclusive=true/false
    purge-connector-queue-auto-delete=true/false
    purge-connector-queue-declare=true/false
```

When the queue is for simply consuming, typically, the durable, exclusive and auto-delete settings can be ignored (will default to false) and the declare property should also be set to false as it is expected that the queue will already exist and be being populated by another process. This means the queue can be read only for the user defined for the purge service.

The following properties are used to define the local connector/analyser details. DB is the value used for the property purge-connector-type when internal system checking is required. These properties define the rules which will be used to identify if an entry on the system is scheduled for purging. The ages are in days and the properties specify which areas of the system to check; orders, studies and reports. If the use study date and use order date options are true, the system will use the date in the study or order, otherwise the system will use the insert date of the study or order.

```
    check_orders=true
    use_order_date=true
    max_order_age=3650
    check_studies=true
    use_study_date=true
    max_study_age=3650
```

Regardless of the mechanism used to identify entries ready for purging, there are options available to specify which areas of the system to apply the purging to. The following properties show the flags used to turn on or off the different areas for purging. The messages table in the database doesn't maintain a reference to a specific patient or order, so they must be purged based on a maximum age regardless of the connector used to identify items for purging.

```
    purge_orders=true
    purge_notes=true
    purge_reports=true
    purge_hl7_export_jobs=true
    purge_studies=true
    purge_empty_patients=true
    purge_messages=true
    max_message_age=30
```

When the system is running, it will sleep for set periods between runs. By default, this period is set to 10 minutes (600 seconds). The system will also limit the number of jobs to 2000 per each run. These properties can be tailored to suit how aggressive the system should be in purging items. The number of workers specifies the number of threads that run in parallel to delete items on the system.

```
    prg_service_workers=1
    prg_service_sleep=600
    queue_query_limit=2000
```

It is also possible to make the purge service check for the existence of a study UID against an external DICOM source before purging. There are two properties for this process.

```
    use_external_lookup=false
    use_external_url=
```

By default, the external look up use is off (false) and the URL is null. In order to use this function, the lookup property should be set to true and the URL be set to a DICOM URL format of the form:

```
    use_external_url=dicom://host:port/called_aet/calling_aet
```

If the study UID that has matched as a purge candidate is found with images on the external lookup, then the Study UID will be removed from the list of studies or orders to purge.

There is an option for the purge service to operate in an analysis only mode when setting things up. This is configured using the property analysis only in the purge properties file.

```
    analysis_only=false
```

In analysis only mode, the system will simply log the UIDs that would be deleted rather than actually deleting anything from the system.

6.9 Nexus Cleaner Services

Like the *Purge Service* (see section 6.8), the *Nexus Cleaner Service* allows databases and files to be automatically removed. The cleaner service is primarily driven by Nexus nodes or secondary services, which add cleaning jobs into a database. The cleaning priorities and schedules are set in the individual configuration files and nodes of implementing services.

The directory of the cleaning configuration file and the property to enable the cleaning services are defined in the system *deployment.properties* file:

Listing 2: deployment.properties

```
service.cleaner.enabled=true
cleaner.config.root=/example/config/
```

If the root directory property is omitted, the system will use the default location:

```
/[install_directory]/docs/config/server/fileupload/
```

The cleaner service properties file is named *cleaner-config.txt* and defines the database used to store the cleaning jobs:

Listing 3: cleaner-config.txt

```
cleaner_dbtype=MYSQL
cleaner_dbhost=127.0.0.1
cleaner_dbport=3306
cleaner_dbname=example_database_name
```

For other properties, such as setting non-default database username and password, or properties specific to MSSQL, please contact support@pukka-j.com.

Each service that implements a cleaner will have the following settings:

- Cleaner enabled/disabled
- Database purge enabled/disabled
- Item retention time

By default, the cleaner and database purge are disabled. The database purge property does nothing if the cleaner service is disabled, however, the user may wish to only clean files from the system and leave the database intact, so purging the database is an additional option. The retention time is set in either seconds, minutes, hours or days. If the user attempts to set multiple retention times, 2 seconds *and* 2 days for instance, then the system uses the longest retention. For data integrity safety, there is no default retention time; if the user does not set a retention time, the cleaner does not run. The general configuration for a cleaner for service *some_service* is:

Listing 4: someservice-config.txt

```
some_service_cleaner.enabled=true
some_service_dbpurge.enabled=false
some_service_retention.days=2
some_service_retention.hours=2
some_service_retention.minutes=2
some_service_retention.seconds=2
```

Note that in the example, all of the retention properties have been set, so the system would use a retention time of 2 days.

6.10 Database Listening Service

The Database Listening Service allows users to selectively copy database patients from one database to another and optionally fetch a history of medical images. The user must populate their source database with *PatientLevel* information.

To mark a patient to be copied, the user must add an *"n"* character in the *Priv* column of the *PatientLevel* database table. The service copies all of the patient details into the destination database and marks the *Priv* column in the source database with a *c* character, to show that copying is completed. The service then queries a Dicom source for imaging history for that patient, which will populate the destination database *StudyLevel* and *SeriesLevel* tables.

To retrieve images, the user must add a *"g"* in the *Priv* column of the desired items. A *"g"* added to the *StudyLevel* will direct the service to download the whole study, while a *"g"* added to the *SeriesLevel* will direct the service to only download a single series within the study. The *Priv* columns of the destination database are not updated to mark the items completed as all such status information is in the *DBListenerJob* table.

To use the service, add the following to the *deployment.properties* file:

```
service.dblistener.enabled=true
```

A *dbl-config.txt* file needs to be created in *docs/config/server/dblistener/*, which contains the configuration of the source database:

```
dbl_src_dbtype=Mysql
dbl_src_dbname=dicom_src
dbl_src_host=10.2.3.4
dbl_src_port=3036
```

The configuration of the destination database:

```
dbl_dbtype=Mysql
dbl_dbname=dicom_dest
```

```
dbl_host=10.3.4.5
dbl_port=3036
```

And the configuration of the Dicom query-retrieve provider:

```
cfind_port=8085
cfind_cache_root=/cache
cfind_called_aet=MINT
cfind_cache_dcm=true
cfind_cache_text=true
cfind_host=10.1.2.3
cfind_calling_aet=NEXUS
cmove_dest_aet=PJDBL
cmove_calling_aet=NEXUS
cmove_port=8085
cmove_use_single_cmove_association=true
cmove_called_aet=MINT
cmove_host=localhost
cmove_max_connections_per_client=5
cmove_use_single_connection_per_syntax=false
```

6.11 File Upload Service

The File Upload Service monitors a set of configurable directories for new files matching a specified pattern. Matching files are then queued for upload by inserting an *UploadJob* entry into the configured database. This service is specifically designed to work with *PacsPost* and other web-based Pukka-j portal services. Uploads are authenticated using JSON Web Tokens (JWTs), and therefore the receiving service must be configured with the relevant public or pre-shared keys associated with the uploader.

It is possible to configure multiple directories for file monitoring, and each directory may be assigned a unique upload configuration—for example, differing upload URLs or custom sender/recipient usernames for *PacsPost*.

6.11.1 Configuring the Uploader

To enable the File Upload Service, define the following property in your *deployment.properties* file:

```
service.fileupload.enabled=true
```

There are two configuration files associated with the service:

- A basic configuration file: *fileupload-config.txt*

- An extended configuration file: *fileupload.json*

You may optionally specify the locations for these configuration and log files:

```
fileupload.config.root=/example/config/
fileupload.logs.root=/example/logs/
```

If these paths are not explicitly defined, the default location is assumed to be:

```
/[install_directory]/docs/config/server/fileupload/
```

Basic Configuration

The basic configuration requires you to define:

- The upload URL
- SSL/TLS keystore details
- The database connection for storing upload jobs
- A JWT *issuer* for authorising uploads (see Section 6.12)

Example: *fileupload-config.txt*:

Listing 5: fileupload-config.txt

```
fileupload_url=https://pacspost.net/pacspost-auto-upload
fileupload_keystore.file=config/ks/security/keystore.jks
fileupload_keystore.password=very_strong_password
fileupload_keystore.type=JKS
fileupload_dbtype=MYSQL
fileupload_dbhost=127.0.0.1
fileupload_dbport=3306
fileupload_dbname=example_database_name
fileupload_issuer=issuerNameFoundInJwtConfigFile
```

However, at this stage the service still lacks any paths to scan for uploads. These are specified in *fileupload.json*, which defines:

- Directories to search
- File patterns to match
- Directories to ignore
- Upload-specific metadata (e.g., sender, recipient, and messages)

Example: *fileupload.json*:

Listing 6: fileupload.json

```
"searchpath": [
    "C:/data/somezips",
    "C:/data/morezips"
],
"ignorednames": [
    ".cache",
    "test"
],
"searchpattern": [
    "*.dcm",
    "*.zip"
],
"directories": [
    {
        "directory": "C:/data/somezips",
        "message": "DICOM upload from someZips",
        "sendto": "test1@pukka-j.com",
        "sentfrom": "some@pukka-j.com"
    },
    {
        "directory": "C:/data/morezips",
        "message": "DICOM upload from moreZips",
        "sendto": "test2@pukka-j.com",
        "sentfrom": "more@pukka-j.com"
    }
]
```

Advanced Configuration

Further configuration options can be added to *fileupload-config.txt*:

Set the minimum age (in seconds) that a file must exist before being uploaded (default: 10 seconds):

```
fileupload_minageseconds=10
```

Define the maximum number of upload attempts before giving up (default: 3):

```
fileupload_maxtries=99
```

Set fallback values for sender name and upload message when not specified in *fileupload.json*:

```
fileupload_username=myname.example.com
fileupload_uploaddescription=Testing your file upload service
```

Define target directories for successfully uploaded or failed files (defaults: `processed`, `error`):

```
fileupload_processed=completed
fileupload_error=failures
```

Control upload chunk sizes for optimal performance across networks. Use standard units: `B`, `K`, `M`, `G` (e.g. `10M` = 10 megabytes):

```
fileupload_splitsize=10M
```

Image Transfer Export Configuration

If the Image Transfer service has been configured for export (see Section 6.3), the File Upload Service can be extended to handle export jobs by consuming extra metadata from DICOM transfer forms.

Enable this support in *fileupload-config.txt*:

```
fileupload_export.enabled=true
```

Define the dedicated export path and a list of supported upload destinations in *fileupload.json*:

Listing 7: fileupload.json with export configured

```
"exportpath": "C:/ExportPathName/PacsPost",
"additionalUrls": [
    {
        "url": "https://pacspost.com/pacspost-auto-upload",
        "file": "C:/{path to URL certificate}",
        "password": "{keystore password}",
        "type": "{keystore type}",
        "issuer": "{issuer name in jwtconfig}"
    }
    // Additional URLs can be added
]
```

The service will monitor the specified `exportpath`. When new studies appear, it matches the export job's URL against `additionalUrls` to determine the appropriate keystore and JWT credentials. The export payload includes form details in the JWT, and the study is transmitted to the destination.

6.11.2 Automating Cleanup

The File Upload Service supports automated file and database cleanup using the Nexus Cleaner utility (see Section 6.9).

To remove successfully uploaded files older than 190 days:

Listing 8: fileupload-config.txt

```
some_service_cleaner.enabled=true
some_service_retention.days=190
```

To also purge old upload job entries from the database:

Listing 9: fileupload-config.txt

```
some_service_cleaner.enabled=true
some_service_dbpurge.enabled=true
some_service_retention.days=190
```

6.12 JSON Web Tokens

Many of the Pukka-j Nexus features require authorisation when manual username and password entry is unfeasible. This is often achieved using JSON Web Token (JWT) authentication. To enable Nexus services to authenticate inbound JWTs, the system configuration in *deployment.properties* must be updated to explicitly enable JWT support and define the location of the JWT configuration file.

Listing 10: deployment.properties

```
[JWT]
service.jwt.configfile=/path/to/jwt.json
```

The JWT configuration file is a JSON document consisting primarily of an array of token *issuers*. Each issuer may define one or more associated keys. Each key can include its own encryption and verification details. Although the format allows for complex scenarios, typical configurations are straightforward, requiring just a token issuer and the file path to a public or pre-shared key:

Listing 11: jwt.json

```
{
    "iss": [
        {
            "iss": "hs256.pukka-j.com",
            "file": "/path/to/pj_example_hs256.key"
        }
    ]
}
```

Keys must be stored as plain text on disk, either in PEM format or as base64-encoded text (essentially PEM without header/footer lines). Only one key per file is permitted.

If the service is required to generate JWTs, the signing algorithm must be specified. Currently, only HMAC with SHA-256 (HS256) and RSA with SHA-256 (RS256) are supported. Please contact support@pukka-j.com if you require support for additional algorithms. A basic configuration using a pre-shared key is shown below:

Listing 12: jwt.json

```
{
    "iss": [
        {
            "iss": "hs256.pukka-j.com",
            "alg": "HS256",
            "file": "/path/to/pj_example_hs256.key"
        }
    ]
}
```

If the system is required to sign a JWT with a private key and verify it with a public key, a separate encryption file must be provided. Multiple configurations can be included in a single file, provided each issuer is uniquely defined:

Listing 13: jwt.json

```
{
    "iss": [
        {
            "iss": "hs256.pukka-j.com",
            "alg": "HS256",
            "file": "/path/to/pj_example_hs256.key"
        },
        {
            "iss": "rs256.pukka-j.com",
            "alg": "RS256",
            "file": "/path/to/pj_example_rs256.pem",
            "enc_file": "/path/to/pj_example.pub"
        }
    ]
}
```

In cases where multiple keys are accepted from the same issuer, each key may be identified with a *key ID* (`kid`). This is often used in conjunction with a JWK (JSON Web Key) service (see Section 6.12.1). Each key ID may specify its own key file and algorithm, but if not explicitly listed, the system falls back to the issuer's default key:

Listing 14: jwt.json

```
{
    "iss": [
        {
            "iss": "example.com",
            "alg": "HS256",
            "file": "/path/to/example_default.key",
            "kids": [
                {
                    "kid": "12345",
                    "alg": "HS256",
                    "file": "/path/to/example_12345.key"
                },
                {
                    "kid": "67890",
                    "alg": "RS256",
                    "file": "/path/to/example_67890.pem"
                }
            ]
        }
    ]
}
```

6.12.1 JSON Web Keys

If JWTs are to be authenticated against a JSON Web Key (JWK) set published by a remote server, then optional keystore settings for SSL/TLS verification must be added to the *deployment.properties* file. For example:

Listing 15: deployment.properties

```
[JWT]
service.jwt.configfile=/path/to/jwt.json
service.jwk.keystore.file=config/ks/security/keystore.jks
service.jwk.keystore.password=very_strong_password
service.jwk.keystore.type=JKS
```

The JWK endpoint must then be configured in the JWT JSON file, referenced against the corresponding issuer:

Listing 16: jwt.json

```
{
    "iss": [
        {
            "iss": "example.com",
            "jwk": "https://example.com/common/discovery/keys"
        }
    ]
}
```

6.12.2 Testing Payload Claims

Although a full explanation of JWT structure is beyond the scope of this documentation, it is important to note that JWTs include a set of *claims* within their payload. Some of these may be validated by individual Pukka-j Nexus services.

If the user wishes to enforce specific claims, these may be defined in the configuration under the appropriate issuer. For instance, to enforce that the claim `use` must always equal `test`, the following can be added:

Listing 17: jwt.json

```
{
    "iss": [
        {
            "iss": "example.com",
            "jwk": "https://example.com/common/discovery/keys",
            "claims": [
                {
                    "claim": "use",
                    "op": "=",
                    "val": "test"
                }
            ]
        }
    ]
}
```

Alternatively, to ensure only that the claim `use` exists (i.e. is not an empty string), the `!=` operator can be used:

Listing 18: jwt.json

```
{
    "iss": [
        {
            "iss": "example.com",
            "jwk": "https://example.com/common/discovery/keys",
            "claims": [
                {
                    "claim": "use",
                    "op": "!=",
                    "val": ""
                }
            ]
        }
    ]
}
```

Text comparisons support the following operators:

- `=` (case-sensitive equality)
- `!=` (case-sensitive inequality)
- `*=` (case-insensitive equality)

Numeric comparisons support standard operators: `<`, `<=`, `>`, `>=`, `=`, and `!=`.

By convention, date and time values in JWTs are Unix timestamps (i.e. numeric). A special dynamic value, `$now()`, is available for comparing against the current time. For example, to enforce expiry and not-before claims:

Listing 19: jwt.json

```
{
    "iss": [
        {
            "iss": "example.com",
            "jwk": "https://example.com/common/discovery/keys",
            "claims": [
                {
                    "claim": "exp",
                    "op": ">",
                    "val": "$now()"
                },
                {
                    "claim": "nbf",
                    "op": "<",
                    "val": "$now()"
                }
            ]
        }
    ]
}
```

7 Appendix

7.1 Example Networks

7.1.1 DICOM Proxy

Figure 184 show a DICOM proxy network configuration that simply takes a DICOM input and passes it to a configured DICOM output node.

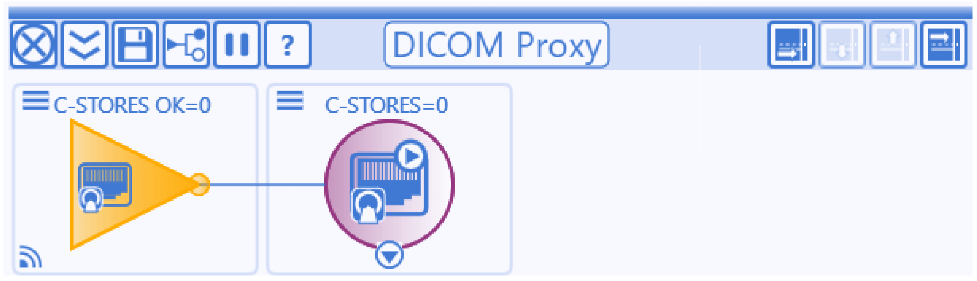


Figure 184: DICOM proxy network configuration.

The input port should be configured to run on a port which other systems wishing to query or retrieve are configured to connect to. Such systems will need to have a DICOM AET configuration entry for the proxy server IP and the input port. If the input port is not configured to be promiscuous for incoming calls, then connecting systems will have to call it by the configured Primary AET or an alias.

Any connecting systems should be listed with their AETs, IP address and port in the connections tab of the input node. In the case of proxying CGET, this may not be required for the calling system as the results will be returned on an existing connection but for proxying CMOVE requests, this node will need to be configured with the details it needs in order to open a connection to the destination AET.

The communications are proxied by the output node to the ultimate destination. The basic configuration uses a DICOM URL to define the destination. If the tick options to use the source AETs are selected, any AET defined in the URL will be overridden by that which is in the incoming communication. If the PACS being queried does not support CGET (and most don't), then the output port will need to start another DICOM listener to receive the results. By deselecting the "supports CGET" option, a port can be specified and an AET. The PACS will use CMOVE to process the request and it is this port which will receive the data. It is therefore these details that should be configured on the PACS (in order for it to send the data to the correct system) for the destination AET in the CMOVE request. Figure 78 shows the proxy listening service for receiving CMOVE responses. The system that will handle the request will need to be configured with a DICOM AET entry for the value specified in the "Identified by AET" field and should be configured with the IP address or host of the proxy server and the port configured in the "Port Number" field. When results are received on the proxy port, they are passed back to the input port which will either send them back to the requester using the established connection (in the case of CGET) or open a new connection to the destination AET (CMOVE).

7.1.2 DICOM CMOVE Proxy

Figure 185 shows a more basic implementation of a DICOM proxy than that of the previous section.

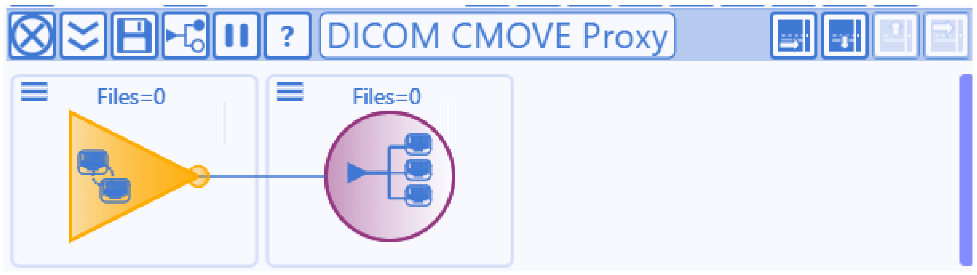


Figure 185: Simple DICOM proxy network.

The input to the network is a DICOM listener but it only responds on the communication sockets that are opened to it, it doesn't open its own sockets and therefore has no configuration for known DICOM nodes. This input node accepts CFIND, CGET and CMOVE requests which are passed to the output node. In the case of CMOVE, unlike the previous network, this one will not expect the CMOVE files, it only expects to read the responses. The responding DICOM service(s) are expected to open the CSTORE request directly to the destination AET defined in the CMOVE request.

This particular output node for this network can handle more than one destination. This allows for combined query results from more than one system, or for example, a combined worklist to a modality from multiple worklist providers. The request is sent in turn to each of the defined outputs and the results combined when all responses have been read. The combined response is then returned to the requesting system on the established socket held by the input node. Care may be required when sending a CMOVE request to multiple destinations, because if more than one has a record of the requested data, then multiple copies could be received concurrently by the destination AET.

7.1.3 Notification Routing

Figure 186 shows how a splitter can be used in conjunction with a notification node in order to route data with specific values down a second pipeline.

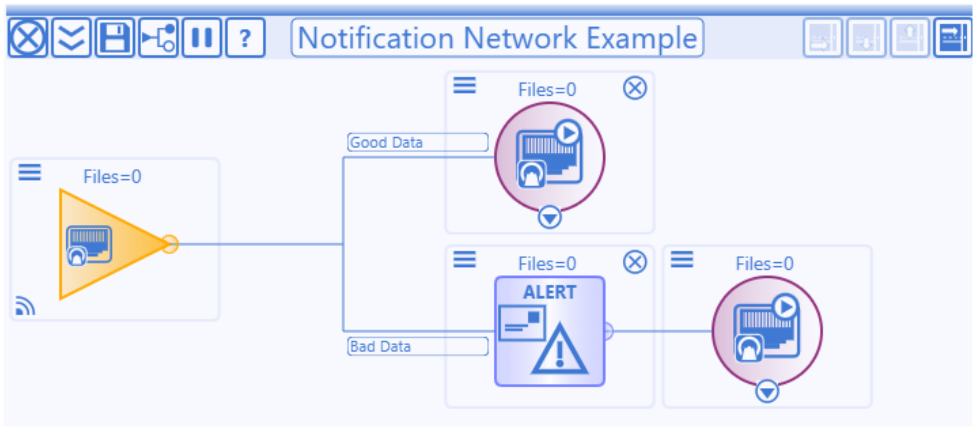


Figure 186: Notification node used with splitter to inform when unexpected values in the data are found.

In this case, the outputs at the end of the pipelines would likely both connect to the same DICOM peer, but because any data taking the second path would pass through the notification node, users can be notified about this data. The specifics of what data goes down the second pipeline would be determined by the settings defined in the splitter. For example, if a tag is missing a value that is expected, or is out of range, the splitter can send that data down the pipeline with the notification node and thus notify the user.

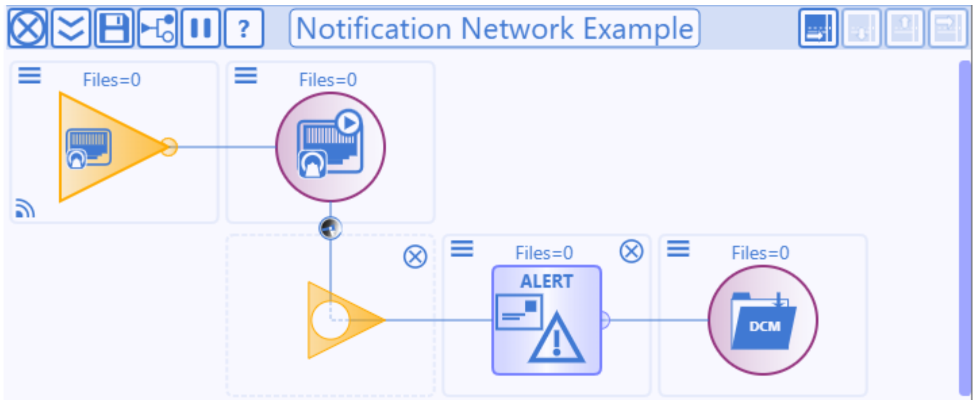


Figure 187: Error channel notifications.

Figure 187 shows how the notification node can be used to inform users if there was an error during sending to the output node such that the data went to the error channel. Only when an error occurs at the output will data take this route and may be a point in the workflow where an administrator wants to be notified of the issue ASAP.

7.1.4 De-identification and Re-identification

When using the anonymisation service to transfer data to an external DICOM system, it may be the case that the outcome that results from the processing of the anonymous data needs to result in that outcome being available with the original study in the originating system. In order to achieve this, the resulting object needs to be able to be re-identified when it is sent back to the originating system. This is possible within Nexus by a combination of different nodes in the network. Figure 188 is an example network of how a DICOM object can be de-identified when passing from one DICOM system to the next and then it, or a derived DICOM object can be re-identified when passing back into the Nexus service.

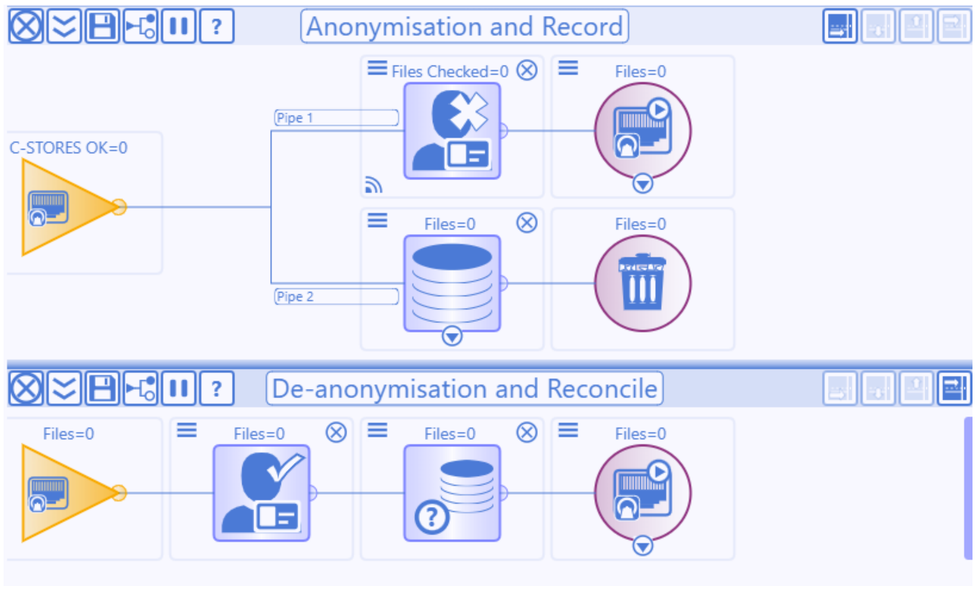


Figure 188: Anonymisation (de-identification) and de-anonymisation (re-identification) example network.

When data is sent to the external system, the anonymisation service keeps track of the anonymised primary demographics and any UIDs that are changed. If study level information needs to be restored as part of the re-identification, the original details for the objects should be recorded in a database node. This is shown in the top network above, where the database node is in the second branch below the anonymisation node. A splitter is used because it makes a copy of the original information. If the database was placed in-line with the anonymisation node, the anonymised data would be recorded in the database. The original files don't need to be stored locally, so the output from the database node is just a terminator (dev null). The output from the anonymisation node is the external system that requires the anonymised data.

When a result needs to be returned to the original study, the DICOM object should be sent to the input node of the second network shown in the bottom of the figure. This network shows the data passing first through the de-anonymisation node which restores the primary demographics and the UIDs and secondly through a reconciliation node which restores study level attributes such as Accession Number, Study ID, Study Date, Study Time that may need to match the original study in order for a successful import to the originating system to occur.

7.1.5 Decoupled Input/Output

To minimise the delay sending across a high latency network connection, the transfer of a single study can be made multi-threaded in order to maximise the use of available bandwidth. This is done by de-coupling the input and output DICOM nodes using file writer and reader nodes as shown in Figure 189.

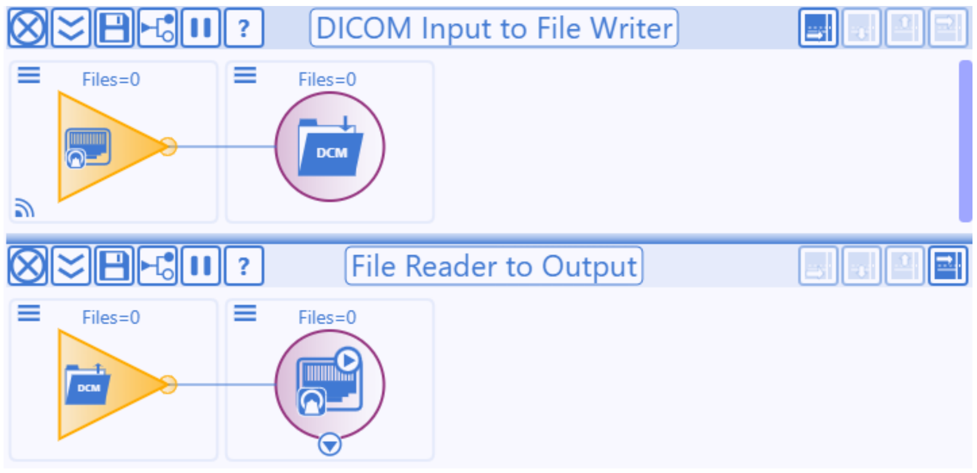
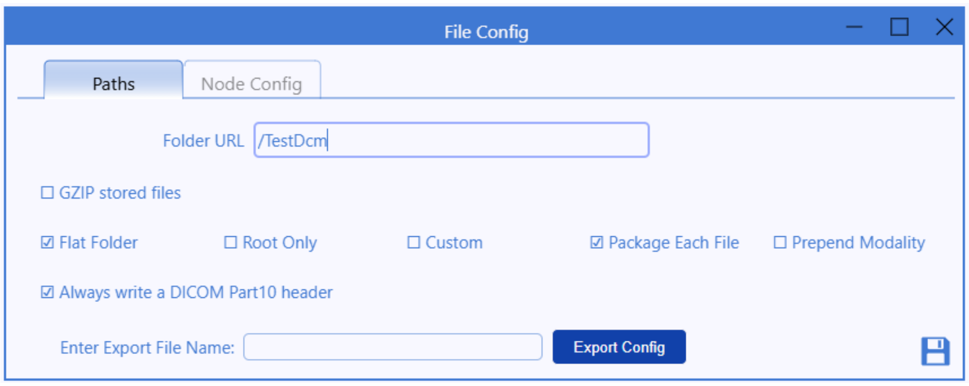


Figure 189: De-coupled DICOM storage network.

By de-coupling the DICOM Input and Output nodes, data can be written quickly to disk using the file output node. In order to be able to maximise the threading capabilities of Nexus, the configuration on the output should be set up as shown in Figure 190. By writing each image into its own folder in a flat file store, the study can be read by multiple threads to send it in parallel over the high latency network.



Folder URL /TestDcm

☐ GZIP stored files

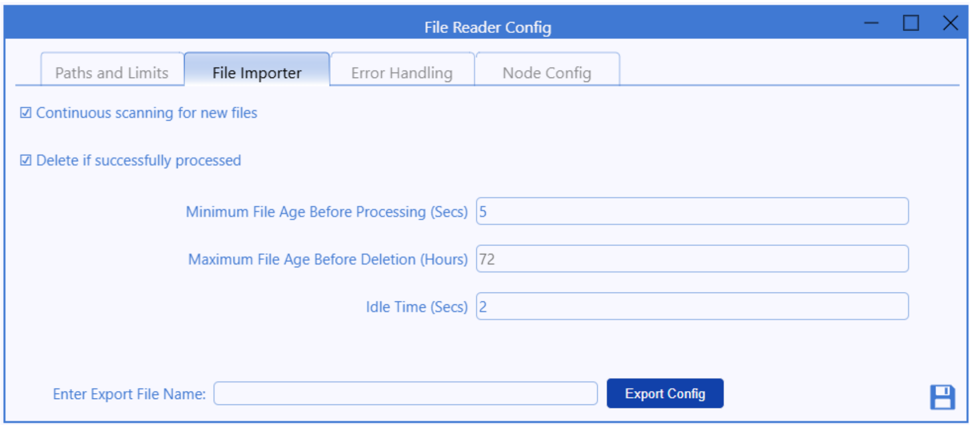
☑ Flat Folder ☐ Root Only ☐ Custom ☑ Package Each File ☐ Prepend Modality

☑ Always write a DICOM Part10 header

Enter Export File Name:

Figure 190: File Output Node config for decoupled network.

In order to send new data as it arrives, the Dicom File Reader node needs to be configured as a file import in continuous mode. Files should be deleted as they are processed. The minimum age and idle time can be tweaked to optimise the transfer rate as shown in Figure 191



☑ Continuous scanning for new files

☑ Delete if successfully processed

Minimum File Age Before Processing (Secs)	5
Maximum File Age Before Deletion (Hours)	72
Idle Time (Secs)	2

Enter Export File Name:

Figure 191: Dicom File Reader Node file importer configuration.

7.1.6 DICOM to HL7 Gateway

To create a HL7 message output based on a DICOM input, the DicomToHL7 Node can be used. This example shows how to configure such a pipeline and to incorporate error notifications.

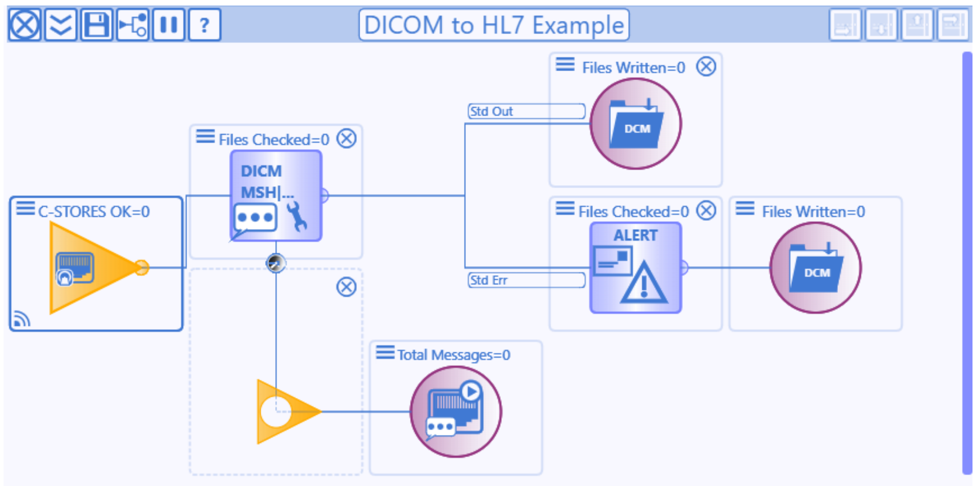


Figure 192: Example DICOM to HL7 Node with Error Notifications.

A DICOM Input node is used to receive DICOM data on a specified port. By sending using different AETs, the DICOM stream could be split based on the AETs used in order to allow different messages to be created based on the AETs. The example is configured with all data going to a single DicomToHL7 node. The primary output from the node allows the incoming DICOM files to pass through to either be sent on, save or discarded. The primary output is also the route that DICOM error files take when sent by the node. These files are sent with a configurable AET and as such can be routed to a notification node in order to send an email alert to a specified email address.

The deferred channel from the bottom of the node in the figure is the route that the generated HL7 messages take and this pipeline should be used to send on the messages using a HL7 output node.

7.1.7 Exec Node DICOM and HL7 Gateway

An Exec Node can be used to call external processing applications and read, process and distribute the results from the called process.

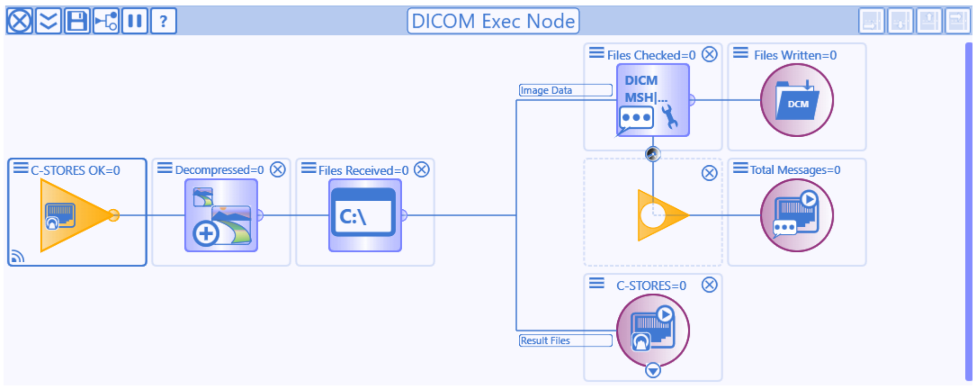


Figure 193: An example Exec Node network configuration.

The figure illustrates how data may be sent into a Nexus pipeline and passed to an Exec Node, possibly (as in this case) with some pre-processing e.g. decompression. The Exec Node is configured to call an external script and read a result file with an output from the called application. The Exec Node maps some or all of the data in the result into specifically mapped DICOM tags in the header of the inbound message before the message is passed once more down the pipeline. This example shows how auxiliary DICOM data can also be sent on. After the Exec Node in the pipeline, there is a splitter in the network for filtering new DICOM result data from the original inbound data, so that it can be distributed to a destination system using a DICOM Output Node. The original data with the result mapping in the header is passed to the top branch where the results are mapped from DICOM into a HL7 file using the DicomToHL7 Node. The DICOM files can be cached as shown or sent on, or simply discarded. The HL7 messages are distributed using a HL7 Output Node. An Exec Node can be used to call any executable external process which may be for example, dose analysis on image data, image processing / edge detection, machine learning for image labelling, image conversion (e.g. NIfTI files).