ICW Master Patient Index 3.0

Medical Service Bus 3.0

Administration Manual

Document version: 1.0 | Status: Approved

Security level: sensitive

Imprint

InterComponentWare AG

Altrottstr. 31

69190 Walldorf, Germany

Tel.: +49 (6227) 385 – 0

Fax: +49 (6227) 385 – 199

E-Mail: info@icw.de

Document ID: 20e94e87-8bee-2d10-c2a3-bc56e5382c3e

Document version: 1.0

Document language: en-US

Security level: sensitive

Product name: Medical Service Bus

Product release: ICW Master Patient Index 3.0

Last change: January 2011

Table of Contents

1 Preface

This document is intended as a manual for the installation and maintenance of the ICW Medical Service Bus (MSB) application. It addresses the topics of configuration, monitoring and troubleshooting.

This document is intended for use by integration specialists and administrators who are responsible for installing and maintaining communication interfaces between the MPI application and data processing systems operated in the healthcare sector.

1.1 Required Prior Knowledge

This manual assumes the reader already possesses the following knowledge and skills:

- Good administrator knowledge about the operating system on which the MSB server will be run (Windows or UNIX/Linux)
- Good understanding of the purpose and functionalities of the MPI application
- Knowledge about the data interface used, that is the transport protocol and data formats of HL7 2.x

1.2 Document Overview

This document is organized in the following manner:

- Section 2 [page 11] contains fundamental information about MSB.
- Section 3 [page 14] summarizes the necessary preconditions and preparatory steps for installing MSB.
- Section 4 [page 17] walks the reader through the installation procedure.
- Section 5 [page 20] discusses the file directory structure of the installed application in more detail.
- Section 6 [page 22] describes configuration options for individual components.
- Section 7 [page 61] explains methods for starting and ending the application.
- Section 8 [page 63] deals with monitoring the application at runtime.
- Section 9 [page 71] provides an introduction to the Flow Manager and the replay mechanism using the JConsole.
- Section 10 [page 79]describes updates and migration.
- Section 11 [page 104] shows you how to operate the MPI 3.0 together with the VMR from PXS 3.1.2

- Section 12 [page 105] explains some specific technical aspects of MSB implementation.
- Section 13 [page 114] gives general information about the Flow Manager.
- Section 14 [page 118] contains a complete user manual for the Integrated Platform Manager application. Among other uses, the Integration Manager is used for Flow Management.
- Section 15 [page 130] Flow Manager in IPM

2 Overview of MSB and its Components

Medical Service Bus (MSB) is the integration module of the ICW Master Patient Index (MPI). It mediates the exchange of data between the application modules of MPI and external client systems.

MSB comprises several message-based communication interfaces. Each of these is represented by its own *component*. MSB components are based on an integration platform (IPF) that is implemented with the help of the open-source integration software Apache Camel.

The integration platform also contains a management server. It enables MSB components to be monitored and controlled at runtime. Figure 1 shows the role of MSB within the context of the overall MPI application.

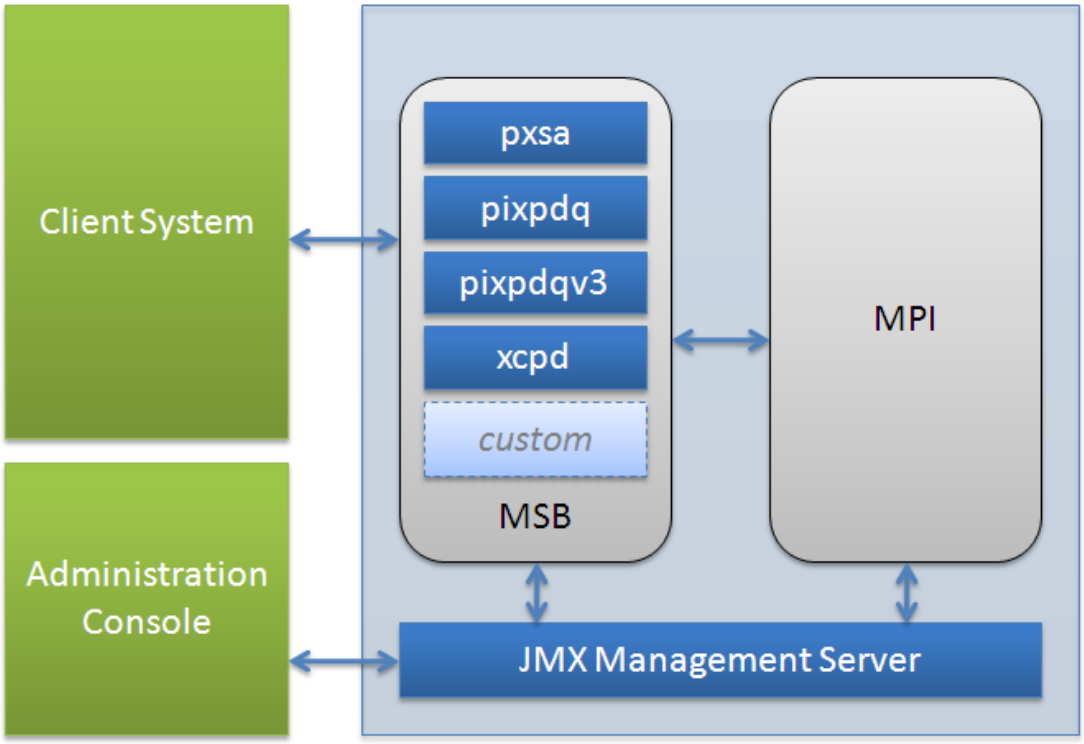


Figure 1: Interactions between client systems

2.1 MSB Components

Client systems that are connected to MPI may include hospital information system (HIS) that communicate over the HL7 v.2 protocol. MSB is foremost an adapter for message formats and transport protocols.

MSB transforms incoming message into an internal data format that can be understood by the MPI application modules. Outgoing messages are transformed into external data formats that can be understood by connected client systems.

MSB consists of four independent interface components. Table 1 describes these components in detail.

Component name	Description
pxsa	The pxsa (MPI adapter) component is the primary interface for hospital integration scenarios. With the help of pxsa, patient data sets (including case and movement information) and documents can be sent to the MPI. Under HL7 2.x, this corresponds to the message types ADT. Supported transport protocols for data exchange are File, MLLP and HTTP.
pixpdq	The pixpdq (IHE PIX/PDQ) compnent expresses an interface that is compatible with the IHE Patient Identifier Cross Referencing (PIX) and Patient Demographics Query (PDQ). In this scenario, MPI acts as the server, that is as the Cross Reference Manager or the Demographics Supplier, respectively. Although the IHE specification does not specify the use of this communication interface in any specific use case, it is typically used in hospital computing as an alternative or supplement to pxsa. As with pxsa, pixpdq also uses HL7 2.x as a communication protocol, but here only in combination with the MLLP protocol.
pixpdqv3	The pixpdqv3 (IHE PIX/PDQ v3) component expresses an interface that is compatible with the IHE Patient Identifier Cross Referencing V3 (PIXv3) and Patient Demographics Query V3 (PDQv3) profiles. While covering the same use cases as pixpdq, the message payload is expressed as HL7v3 messages in XML, and the messages are sent and received as HTTP or HTTPS web services.
xcpd	The xcpd (IHE XCPD) component expresses an interface that is compatible with the IHE Cross Community Patient Discovery profiles. This component implements a *Receiving Gateway* – it received cross-community patient demographic queries and responds with matching patients from the MPI database. The message format is very similar to pdqv3 except for a few attributes necessary for cross-community queries.

Table 1: Overview of MSB Compnents

In this document, an MSB instance running the pxsa, pixpdq, pixpdqv3 and xcpd components is called an *MSB server*, or just *MSB* for short.

2.2 MSB Administration with Management Server

The MSB's built-in Management Server is based on the industry-standard Java Management Extensions (JMX). Find more information under:

http://java.sun.com/javase/6/docs/technotes/guides/management/overview.html

The Management Server enables access to the running Java virtual machine and to application specific services (MBeans).

Two client applications provide user interfaces (administration consoles) for JMX.

- JConsole. This tool provides a graphic display of the status of important system resources such as working memory, including garbage collection and active threads. MSB-specific tasks, especially flow management, can be performed using a general-purpose user interface for MBeans, see section 8.3.
- The Integrated Platform Manager (IPM) is a separate client application from the open-source IPF project. It has a general-purpose screen for MBeans (JMX Explorer) and a specialized GUI for flows (Flow Manager).

The following table summarizes the properties of these tools.

Property	JConsole	Integration Platform Manager
Availability	Part of the Java installation, and is automatically present once MSB is installed.	Available as a free download from http://repo.opene-health.org/confluence/display/ipf/Home
Scope of functions	All Java Virtual Machine operational parameters can be displayed, some in graphical form. Numeric MBean attributes shown as progressive curves.	Separate installation. No monitoring of system parameters. MBean attributes displayed as text, tables and lists.
Flow Management	Flow Management available over the general MBeans GUI.	Flow Management over a consistent and user-friendly GUI.

Table 2: Advantages and disadvantages of available administrative tools

NOTE JConsole

Because JConsole is in wider use and has a broader spectrum of functions, this tool is the one that is officially supported for MSB. Interested users can find the IPM manual in Appendix E.

3 Hardware, System, and Software Requirements

3.1 Server

The basic system requirements for operating the MSB server are described in detail in the document MPI Hardware and Software Requirements in the section dealing with the communication server.

To install the application you need to have the following software installed on your server:

- Windows or Linux operating system
- Java JRE 1.6.0_18 or higher

3.2 Workstation Computer for Remote Monitoring and Management

- Windows operating system
- Java JDK 1.6.0_18 or higher

3.3 System Installation Requirements

- A user account is registered with the operating system under which MSB is installed and operated. It is also presumed that the name of this user is *msb*.
- The JAVA_HOME environment variable is set correctly.

3.3.1 Open File Limits

The limit of the allowed number of open files for the application user (for example, msb) has to be set to 20000. The default value is 1024 on most Linux systems, which is not sufficient.

3.3.1.1 Open File Limits on Linux

1. As root user, add the following lines to */etc/security/limits.conf*:

```
# Raise open files limit
* soft nofile 20000
* hard nofile 20000
```

2. After a reboot, use the ulimit command to ensure that the limit has been set correctly. At the command prompt, enter:

```
# ulimit -n
```

```
20000
```

3.3.1.2 Open File Limits on Windows Server

On Windows Server systems, you can use the Windows *System Resource Manager*, which has similar features in that you can set CPU and memory limits for a process, user or session. Please contact your system administrator for details.

3.3.2 TCP Keepalive settings

NOTE

This setting is only necessary if in this installation the MSB sends messages to external systems.

If the MSB leaves a connection to an external destination system open without sending messages for a longer period of time, the MSB needs to send keepalive packets to ensure that the connection is not dropped by firewalls and other network components between the MSB and the destination system.

The Java runtime environment utilizes the corresponding operating system settings; here the default delay until sending the first keepalive packet usually is 2 hours. This parameter needs to be changed to 3-5 minutes, depending on the timeout configuration of the network components. Please consult the responsible network administrators for the required configuration.

3.3.2.1 Setting TCP Keepalive on Linux

1. As root user, add the following lines to the file */etc/sysctl.conf*:

```
# Setting delay until first keepalive to 5 minutes (300 seconds),
# probing 5 times every 5 seconds. Depends on network config.
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_keepalive_intvl = 5
net.ipv4.tcp_keepalive_probes = 5
```

2. Install the new settings. Depending on the Linux distribution, enter in the command shell:
   - On SuSE Linux:

     ```
     # chkconfig boot.sysctl on
     ```

- On RedHat Linux:

  ```
  # chkconfig sysctl on
  ```

3. After a reboot, use the sysctl command to ensure that the parameters have been set correctly

   ```
   # sysctl -a | grep tcp_keepalive
   ```

   ⇨
   ```
   net.ipv4.tcp_keepalive_intvl = 5
   net.ipv4.tcp_keepalive_probes = 5
   net.ipv4.tcp_keepalive_time = 300
   ```

3.3.2.2 Setting TCP Keepalive on Windows Server

1. From the Registry Editor, open the following set of parameters: **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters.**
2. Add or change the following keys and values. Note that *KeepAliveTime* and *KeepAliveInterval* have to be specified in milliseconds.

   ```
   KeepAliveTime = 300000
   KeepAliveInterval = 5000
   TcpMaxDataRetransmission = 5
   ```

3. Reboot the system so that the parameters become effective.

4 System Installation

4.1 Installing the MSB

NOTE

If you are already running an earlier version of MSB, please check the chapter on Upgrade and Migration [page 79] for additional migration instructions after the MSB 3.0 has been installed.

4.1.1 Installing MSB under Linux

- ▹ An existing user account under which the MSB is supposed to run, for example *msb*. optionally, the MSB can be installed under a different user, for example *icwInstall*, however, the users must belong to the same user group, for example *icw*.
- ▹ If you want to install the MSB as a service, you need an administrator account (root).

1. Log in to the operating system using the appropriate username. For example as the user *icwInstall*.
2. Copy the *msb-<Version>-bin.zip* file into your chosen installation directory. From now on, this directory will be referred to as *<ICW_HOME>*.
3. Unpack the .zip file in the *<ICW_HOME>* directory. This will expand the application into two subdirectories:
   - <ICW_HOME>/product/msb/<version> (referred to as <MSB _HOME>)
   - <ICW_HOME>/data/msb (referred to as <MSB_DATA_HOME>)
4. Set the execution rights of the startup script in the *<MSB_HOME>* directory:
   ```
   icwInstall@server:~> chmod ug+x *.sh
   ```
5. Create the following symbolic link named data inside the <MSB _HOME> directory, pointing to <MSB_DATA_HOME>:
   ```
   icwInstall@server:~> ln -s <MSB_DATA_HOME> data
   ```
6. Make sure that the user that runs the MSB has permissions to write into <MSB_DATA_HOME>. To achieve this, assign the common user group to <MSB_DATA_HOME> and make the directory writable and executable for this group.
   ```
   icwInstall@server:~> chgrp -R icw <MSB_DATA_HOME>
   icwInstall@server:~> chmod -R g+wx <MSB_DATA_HOME>
   ```

7. In order to start up the MSB as a service during runlevel initialization, create the following symbolic link (requires root permissions):

   ```
   root@server:~> ln -s <MSB_HOME>/msb.sh /etc/init.d/msb
   ```

8. Check the script <MSB _HOME>/msb.sh and adapt user and paths corresponding to the installation. Finally, register the script to the run during startup (requires root permissions, see example below for RedHat Enterprise Linux):

   ```
   root@server:~> cd /etc/init.d
   root@server:~> chkconfig --add msb
   root@server:~> chkconfig --level 35 msb on
   ```

⇨ The MSB binaries are now installed. However, before you can actually use the MSB you need to customize its configuration.

4.1.2 Installing the MSB under Windows

1. Log in to the operating system as administrator.
2. Copy the `msb-<Version>-bin.zip` file into your chosen installation directory, for example `C:\icw`. From now on, this directory will be referred to as `<ICW_HOME>`.
3. Unpack the .zip file in the `<ICW_HOME>` directory. This will expand the application into two subdirectories:
   - `<ICW_HOME>/product/msb/<version>` (referred to as `<MSB _HOME>`)
   - `<ICW_HOME>/data/msb` (referred to as`<MSB_DATA_HOME>`)
4. Unlike Linux, Windows does not support symbolic links and the MSB does not let you configure the data directory yet. So you have to manually move the `<MSB_DATA_HOME>` into the `<MSB_HOME>` directory.

   ```
   C:\> move C:\icw\data\msb C:\icw\product\msb\3.0\data
   ```

5. If you plan to run the MSB as a Windows service (recommended), edit the file `<MSB_HOME>/conf/wrapper.conf` and replace `%JAVA_HOME%` with the actual Java Home directory, e.g. `C:/java/jdk1.6.0_21` in both of these properties:
   - `wrapper.java.command`
   - `wrapper.java.classpath.1`
6. Register the MSB as a Windows service by running

   ```
   <MSB_HOME>/Install_msb-NT.bat
   ```

⇨ The MSB binaries are now installed. However, before you can actually use the MSB you need to customize its configuration.

4.2 Administration Console Installation

JConsole is a standard part of the Java SDK distribution. It enables Java applications to be monitored and controlled at runtime. It is operated as a remote maintenance tool from the monitoring computer, a Windows workstation (see the section Workstation Computer for Remote Monitoring and Management [page 14]).

4.2.1 JConsole setup

In order to be able to use *Flow Manager*, JConsole also needs a number of additional program libraries.

- ▹ Check the network and operation system levels to ensure that the monitoring computer can connect with the MSB server. The TCP ports of the firewall that need to be enabled are listed in section 8.1.

1. Copy the file <MSB_HOME>/lib/commons-flow-<Version>.jar from the MSB server to the <JDK_HOME>/lib folder.
2. Copy the file <MSB_HOME>/start-console.bat from the MSB server to the <JDK_HOME>/bin folder.

⇨ Once the installation has completed successfully you can start JConsole and connect to a running MSB server.

5 MSB Installation Directory Structure

The directory structure of an MSB installation is listed in following table.

Directory	Content
<MSB_HOME>	MSB installation directory
<MSB_HOME>/conf	Configuration directory
<MSB_HOME>/conf/msb	Configuration files for MSB
<MSB_DATA_HOME>	Data directory that will be created at runtime
<MSB_HOME>/data/activemq1	Temporary storage for queue data from ActiveMQ
<MSB_HOME>/data/audit...	Directory where message traffic is tracked (audit). The subdirectories depend on interface component and audit time.
<MSB_HOME>/data/flowmgr...	Database directory for Flow Manager
<MSB_HOME>/data/log	Directory for the MSB log files
<MSB_HOME>/data/in…	Directory for importing files in file-based data exchange
<MSB_HOME>/lib	Program libraries

Table 3: MSB directory structure

The actual directory structure may vary depending on the configurations. Directories that contain subdirectories are indicated by ellipses (...) in the MSB directory structure table.

NOTE **Directory creation**

The system sets up parts of the directory structure during installation, but some subdirectories will not be set up until the MSB is started for the first time.

CAUTION

In the directories `<MSB_DATA_HOME>/log` and `<MSB_DATA_HOME>/audit` , protocol and audit files accumulate during system operation. Please make sure that these files are regularly backed up and removed afterwards to avoid running out of disk space. See the section Monitoring Audit files [page 69] for an example.

6 System Configuration

6.1 Overview of MSB Configuration and Customization

The behavior of MSB can be controlled at a variety of levels that are listed in the table *Overview of customization options*. This document describes the *Context-Properties* and *Code System Mappings* mechanisms in detail. Some of these mechanisms lie outside the realm of normal administration and are not discussed in this manual. Advanced customization options for the program are discussed in the *Customization Reference*.

Customization instrument	Customization level	Valid for	Relevant resource(s)	Described in...
Context properties	Configuration	General	`<MSB_HOME>/conf/msb/context-<component-name>.properties`	Administration manual (this document)
Code System Mappings	Configuration	General	`<MSB_HOME>/conf/msb/default-<component-name>-mappings.map`	Administration manual (this document)
Context file	Configuration	General	`<MSB_HOME>/conf/msb/context-<component-name>.xml`	Customization Reference
Start script	Modification or extension	General	`<MSB_HOME>/start-node.*`	Customization Reference
Pre-processing	Extension	pxsa	`<MSB_CUSTOM_HOME>/*`	Customization Reference

Table 4: Overview of customization options

6.2 Summary of Context Properties

Context properties are simple, key-value pairs, that are read into the Spring context of MSB. Context properties can be set at two different levels:

- Common level - in the file `<MSB-HOME>/conf/msb/context-common.properties`
- Component level - in the file `<MSB_HOME>/conf/msb/context-<component-name>.properties`

All property files are extensively documented for each individual key-value pair, most of the time beyond the description in this documentation.

NOTE **Priority of Context Properties**

Context properties at the component level have priority over the context properties in the common configuration context-common.properties.
Context properties are static. Changes become active when you restart the affected MSB component.

6.2.1 Changing context properties

1. Ensure the MSB is not running. To stop the process, see the section Stopping MSB [page 61]
2. Open the context-<component>.properties or context-common.properties file in a text editor and make your changes to the content. Make sure there are no blank spaces at the end of the line - these are usually undesirable.
3. Save the modified file.
4. Restart the MSB, see the section Starting MSB [page 61].

6.2.2 Relevant context properties

MSB is configured initially in the following file:

```
<MSB_HOME>/conf/msb/context-<componentenname>.properties
```

Change the parameter only when the default settings cannot satisfy your system requirements.

CAUTION **Setting Parameters**

Incorrectly set parameters may degrade the system to complete failure. Avoid making changes to parameters when the possible consequences are not known.

The following table lists the relevant configuration areas for each component. Read the sections relating to all of the interfaces and then the sections that relate to the MSB components you want to configure.

Interface	Relevant properties
<all interfaces>	See section, Logging: Log levels [page 24] See section, Console output and log file: log4j.xml [page 25] See section, Connection to the MPI core application [page 25] See section, ActiveMQ as internal JMS service [page 27]
pxsa	See section, pxsa settings [page 35]
pixpdq	See section, pixpdq settings [page 43]
pixpdqv3	See section, pixpdqv3 settings [page 55]
xcpd	See section, xcpd settings [page 58]

Table 5: Summary of configuration options for MSB

6.3 General Configuration Options

6.3.1 Logging: Log levels

Apache log4j, a third-party software product, is integrated for the purpose of writing log files. A priority (the log level) is noted for each entry in a log file. The size of the log file is limited by means of log level restrictions. The table *Log levels in log4j* describes the values allowed for log levels.

Log level	Meaning	As log entry type	Meaning in terms of explicitness
OFF	Deactivated	<N/A>	No log entries.
FATAL	Fatal error	Entry refers to a fatal error	Logs entries with log level FATAL only.
ERROR	Error	Entry refers to an error	Logs entries with log level FATAL, ERROR only.
WARN	Caution	Entry refers to a warning	Logs entries with log level FATAL, ERROR, WARN only.
INFO	Information	Entry provides information	Logs entries with log level FATAL, ERROR, WARN, INFO only.

Log level	Meaning	As log entry type	Meaning in terms of explicitness
DEBUG	Debugging information	Entry contains debugging information	Logs entries with log level FATAL, ERROR, WARN, INFO, DEBUG only.
TRACE	Detailed debugging information	Entry contains detailed debugging information	Logs entries with log level FATAL, ERROR, WARN, INFO, DEBUG, TRACE only.
ALL	All entries	<N/A>	Logs all entries.

Table 6: Log levels in log4j

6.3.2 Console output and log file: log4j.xml

Use the control file <MSB_HOME>/conf/msb/log4j.xml to control the level of detail of the output. One of the principal loggers is Camel. Use the value attribute to change the log level.

```
<!-- Logger for all native Camel output -->
<logger name="org.apache.camel">
<level value="INFO"/>
</logger>
```

More options are described in the *Customization Reference*.

6.3.3 Connection to MPI Core Application

All MSB components forward messages to the MPI core application instance. This instance has a common configuration for all MSB components. The HTTP connection parameters listed in the table must match the configuration of the corresponding MPI core application so that MSB and MPI can communicate with each other (see also, the MPI System Administration Manual) .

MSB communicates with the server of the MPI core application (target server) as a client over the HTTP protocol. Fundamentally, authentication over HTTP Basic Authentication is used. If the target server does not use authentication, the parameters `pxs.http.userName` and `pxs.http.password` can be left without values.

Parameter	Meaning	Default value	Probability of change
`pxs.http.url.base`	Connection protocol, host and port number of the target application in URL notation.	`http://localhost:8080`	Medium
`pxs.http.url.localpart`	Context path in MPI to which messages are sent. Appended to `pxs.http.url.base`	`/mpi/exporter`	low
`pxs.http.socket.timeout`	Time limit for the completion of the HTTP request (in milli-seconds).	`30000`	low
`pxs.http.userName`	Username for basic authentication for the target application. Optional.	`<not shown here>`	high
`pxs.http.password`	Password for basic authentication for the target application.	`<not shown here>`	High

Table 7: MPI HTTP connection parameters

6.3.3.1 Handle unavailability of the target server

MSB has several mechanisms to compensate for temporary disruptions of the network infrastructure or disruptions on the recipient side. One of these is the redelivery policy. When the first attempt to deliver a message to the target system fails, MSB will cache the message and attempt to re-send the message later (see section, Connection from MPI Core Application). The number of attempts and the time interval between attempts are both configurable.

Parameter	Meaning	Default value	Probability of change
`pxs.redelivery.maximumRedeliveries`	Number of repeated attempts. The initial attempt is not included in this number.	6	low
`pxs.redelivery.initialRedeliveryDelay`	Wait interval between the initial attempt and the first repeated attempt (in milliseconds).	5000	low
`pxs.redelivery.backOffMultiplier`	Multiplier for the wait intervals for each additional attempt. Relevant when exponential backoff is active, which is the default for pxsa and pixpdq. When the value is "1" the interval remains constant.	2	low

Table 8: Message redelivery parameters

NOTE **Log file size**

In the case of frequent and constantly unsuccessful redelivery attempts the log file can reach considerable size because of the error messages. In this case ensure you have adequate disk storage space.

You will find detailed information about the redelivery policy in the Customization Reference and in the Camel documentation.

6.3.4 ActiveMQ as an internal JMS service

Persistent JMS in front of the transmitting processes compensate for any system downtime or deviations in the processing speeds of the sending MSB components and the receiving eMPI application modules. The utilized queue service, ActiveMQ from Apache, is compatible with the industry standard Java Message Service (JMS). Messages are stacked in the JMS queues and simultaneously stored so that nothing will be lost if the system goes down. Movement of messages to and from the queue is synchronous and transactional. When MSB components are started, messages remaining in the queues will be automatically processed.

NOTE

The PIX Feed, PIX Query and PDQ profile's transactions require a synchronous request-response communication pattern. Therefore, ActiveMQ is not used for these interfaces, and the settings described below do not apply.

The embedding of ActiveMQ in MSB is controlled using the following parameters, see the table Active MQ configuration parameters:

Parameter	Meaning	Default value	Probability of change
`activemq.network.connector.uri`	URL of an ActiveMQ network. Only required if the JMS Broker is accessible from outside the MSB process.	`static:(tcp://localhost:60002)`	low
`activemq.transport.connector.uri`	URL of the ActiveMQ transport connector. Only required if the JMS Broker shall be accessible from outside the MSB process.	`tcp://localhost:60001?daemon=true`	low
`activemq.connection.broker.url`	URL of the ActiveMQ broker.	`vm://broker1?create=false`	low
`activemq.broker.name`	Name of the ActiveMQ broker.	`broker1`	low
`activemq.data.dir`	Directory where JMS messages are temporarily persisted.	`data/activemq1`	low

Table 9: Active MQ configuration parameters

6.3.5 Configuring auditing

All interface components are able to log every message they receive, forward or send back as response. The messages are logged into one file per message. If the MSB works under high load, this might result in millions of message files in the file system. Thus, these files must be split into manageable chunks.

The MSB offers configuration items that limit the number of audit files and distributes them into subdirectories.

Parameter	Meaning	Default value	Probability of change
`audit.enabled`	Turn auditing on or off completely.	`true`	low
`audit.enabled.for`	Transactions for which auditing shall be enabled. Effective only if audit.enabled is set to true. Comma-separated list without whitespaces.	`pxsa, pixfeed, pixquery, pdq, pixupdate, pixsource pixfeedv3, pixqueryv3, pdqv3, pixupdatev3, xcpd, sthl hirslanden, backtransfer`	medium
`audit.enabled.when`	Points of interest for which auditing shall be enabled. Effective only if audit.enabled is set to true. Comma-separated list without whitespaces.	`request, internal-request, internal-response, response, ping, error notification-request, notification-response, notification, translated, postprocess-request, postprocess-translated`	medium

Parameter	Meaning	Default value	Probability of change
`audit.time.based.directory`	Whether a date/time-dependent subdirectory shall be created for the requests.	`true`	medium
`audit.time.based.granularity`	DateTime pattern that determines the time intervals at which new audit directories are created. The default value would create a new directory every hour.	`yyyy-MM-dd-HH`	medium

6.3.6 Flow Manager settings

The Flow Manager is an IPF service that:

- Registers incoming messages.
- Persists message metadata and content in a database.
- Tags messages on either a successful delivery or on an error condition with an appropriate marker.
- Allows registered messages to be replayed at a later point in time in case of system failures.
- Cleans up persisted messages after successful delivery, at an appropriate time interval.

You will find details about flow management in the section Flow Manager in JConsole [page 71]. Storage for database files is configurable. Because the embedded database communicates with MSB over internal channels, no other system resources such as port numbers need be configured.

Parameter	Meaning	Default value	Probability of change
`flowManager.enabled`	Turn Flow manager on or off	`true`	medium
`flowmgr.data.dir`	Database directory	`data/flowmgr`	Low
`flowmgr.database.dialect`	Hibernate SQL dialect class depending on the data source.	`org.openehealth.ipf.commons.flow.derby.DerbyDialect`	Low

Table 10: Flow Manager control parameters

NOTE **Flow Manager**

The PIX Feed, PIX Query and PDQ profile's transactions require a synchronous request-response communication pattern. The client receives an immediate status report on requests that are sent. As the client waits for the response, there's no meaningful way of replaying the requests at a later point in time. Therefore, the Flow Manager is not used for PIX Feed, PIX Query and PDQ interfaces.

If an external database is used (for example, an Oracle instance shared with MPI), some other settings become relevant. At the same time, you have to change the database connection configuration in the*context-flowmgr.xml* file by uncommenting one of the connection configurations.

Parameter	Meaning	Default value	Probability of change
`flowmgr.database.name`	Name of the database.	`flowmgr`	Medium
`flowmgr.database.host`	Hostname of the database server.	`localhost`	Medium
`flowmgr.database.port`	Port of the database service.	`1527`	Medium
`flowmgr.database.host2`	Hostname of a failover database server.		Low
`flowmgr.database.port2`	Port of a failover database service.		Low

Parameter	Meaning	Default value	Probability of change
`flowmgr.database.username`	Database user name.		Medium
`flowmgr.database.password`	Database password.		Medium

Table 11: Flow Manager control parameters for external data sources

6.3.7 Transport-level Security settings

You secure the HTTP and MLLP interfaces exposed by the MSB by activating transport level security (TLS). A key prerequisite is the availability of appropriate certificates in keystores. With Java applications you set these parameters by starting the application with the system properties as shown in the table TLS system properties.

Parameter	Meaning	Probability of change
`-Djavax.net.ssl.keystore` `-Djetty.ssl.keystore`	Keystore file (.jks), contains the MSB's server certificate. Used for incoming TLS connections.	high
`-Djavax.net.ssl.keystorePassword` `-Djetty.ssl.password`	Password to the keystore file.	high
`-Djavax.net.ssl.truststore` `-Djetty.ssl.truststore`	Truststore file containing the MSB's trusted certificate issuers. Used for outgoing TLS connections.	high
`-Djavax.net.ssl.truststorePassword` `-Djetty.ssl.trustPassword`	Password to the truststore file.	high
`-Dhttps.protocols`	Secure protocol to be used (e.g., TLSv1, SSLv2).	medium
`-Dhttps.cipherSuites`	Cipher suites to be used.	low

Table 12: TLS system properties

Instead of providing the parameters as Java options, you can also specify them in the *context-common.properties* file:

Parameter	Meaning	Default value	Probability of change
`set.certificate.stores`	Whether the settings below should be active.	`False`	low
`keystore.path`	Keystore file (.jks) containing the MSB's server certificate. Used for incoming TLS connections.	`/home/pxs/keystore-ICW.jks`	high
`keystore.password`	Password to the keystore file.	`changeit`	high
`truststore.path`	Truststore file containing the MSB's trusted certificate issuers. Used for outgoing TLS connections.	`/home/pxs/keystore-ICW.jks`	high
`truststore.password`	Password to the truststore file.	`changeit`	high
ssl.protocol	Allowed Server-side Security Protocols	TLSv1	low
ssl.cipherSuites	Allowed Encryption algorithms	SSL_RSA_WITH_NULL_SHA, TLS_RSA_WITH_AES_128_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA	low

Table 13: TLS parameters

6.3.8 Debugging Transport-level Security

To debug TLS connection failures carry out these steps.

1. Start the MSB with the parameter
   `-Djavax.net.debug=<value>`
2. To turn on all debugging set `<value>` to `all`
   or

To turn on SSL debugging set `<value>` to `ssl`

3. Redirect you console output into a file for further analysis as the debugging output is only visible on the console.

The following parameters are used with `ssl`:

- `record`: enable per-record tracing
- `handshake`: print each handshake message
- `keygen`: print key generation data
- `session`: print session activity
- `defaultctx`: print default SSL initialization
- `sslctx`: print SSLContext tracing
- `sessioncache`: print session cache tracing
- `keymanager`: print key manager tracing
- `trustmanager`: print trust manager tracing
- `pluggability`: print pluggability tracing

Handshake debugging can be extended with:

- `data`: hex dump of each handshake message
- `verbose`: verbose handshake message printing

Record debugging can be extended with:

- `plaintext`: hex dump of record plaintext
- `packet`: print raw SSL/TLS packets

6.3.9 Common parameters

Parameter	Meaning	Default value	Probability of change
`hl7.charset`	Character Set for incoming and outgoing HL7v2 messages.	`ISO-8859-1`	low

Table 14: Common parameters

6.3.10 Incoming HTTP connections

As of MSB version 2.9, all incoming HTTP connections are received over the same port. Instead, the local part of the URL is used to differentiate between the individual interface components. The table below shows the parameters that influence the construction of the URL.

Parameter	Meaning	Default Value	Probability of change
`msb.http.port`	Port for incoming messages over HTTP. If set to 0, the port is disabled.	`8484`	Medium
`msb.https.port`	Port for incoming messages over HTTPS. If set to 0, the port is disabled.	`0`	Medium
`msb.http.context.path`	Servlet context path common to all HTTP component interfaces.	`msb`	low
`msb.http.request.path`	Request path for simple HTTP requests.	`plain`	low
`msb.http.webservice.path`	Request path for webservice HTTP requests.	`ws`	low

The resulting HTTP URLs are constructed from the host name, port, context path, request path and the component specific extension. Using the defaults, for example this would result in `http://myhost:8484/msb/plain/receiver` for an incoming pxsa HTTP connection. More details are given in the specific interface component sections.

6.4 Pxsa context properties

6.4.1 Enable/disable settings

The pxsa component interface can be enabled or disabled.

Parameter	Meaning	Default value	Probability of change
`pxsa.enabled`	Enable or disable the pxsa interface.	`true`	Low

Table 15: Pxsa enable/disable

6.4.2 Performance-related settings

The following pxsa settings optimize message thoughput.

Parameter	Meaning	Default value	Probability of change
`pxsa.threads.to.pxs`	Number of parallel connections between pxsa and MPI.	`5`	medium
`pxsa.validate.translated.message`	Enable or disable validation of translated HL7v2 messages. Enable this to identify erroneous messages before they are sent to MPI.	`false`	low

Table 16: Pxsa performance parameters

6.4.3 URLs for incoming messages

The parameters shown in the table control the endpoints for incoming messages.

Parameter	Meaning	Default value	Probability of change
`input.url.file`	Receiving pxsa file endpoint.If the value is a null character string the endpoint is deactivated.	`file:data/in/pxsa ?delete=true &include=.*txt &delay=5000 &sortBy=file:name`	medium
`input.url.mllp`	Receiving pxsa MLLP endpoint. If the value is a null character string the endpoint is deactivated.	`mina: tcp://0.0.0.0:8777 ?sync=true &codec=#hl7codec`	medium
`input.http.url`	Receiving pxsa HTTP endpoint. If the value is a null character string the endpoint is deactivated. Using the default values, the resulting URL would be `http://myhost: 8484/msb/ plain/receiver`	`servlet:///receiver`	low

Table 17: Pxsa-specific URL control parameters for incoming messages

6.4.3.1 Connection from MPI Core Application

Upon certain events in the MPI core application, messages are also sent back to the MSB. The settings in the MSB HTTP connection parameters table define the endpoints that are exposed by the MSB.

Parameter	Meaning	Default value	Probability of change
`pxs.update.uri`	Camel endpoint for messages sent by MPI. Using the default values, the resulting URL would be `http://myhost:8484/msb/plain/update`	`servlet:///update`	Low
`ping.uri`	Camel endpoint for the MSB status page. Using the default values, the resulting URL would be `http://myhost:8484/msb/plain/ping`	`servlet:///ping`	Low
`msb.http.context.options`	Whether the web container will enable session handling and/or basic security for incoming HTTP connections. 0 = no sessions, no security 1 = sessions, no security 2 = no sessions, security 3 = sessions, security	`0`	Medium
`jetty.http.userName`	User name for basic authentication of incoming HTTP requests. Used if msb.http.context.options is set to 2 or 3.	`not shown here`	High
`jetty.http.userName`	Password for basic authentication of incoming HTTP requests. Used if msb.http.context.options is set to 2 or 3.	`not shown here`	High

Table 18: MSB HTTP connection parameters

NOTE **Basic authentication**

Basic Authentication for http is *not* enabled by default. You have to add a security handler to the jetty uri : `jetty:http://0.0.0.0:8484/update?handlers=`**#jettySecurityHandler**
Setting a security handler for a Jetty endpoint will by default affect all endpoints accepting messages on the same port! You can change this behavior by adding further `constraintMapping` beans in the *context-msb.xml* file

6.4.3.2 Other HTTP-related parameters

You can configure the size of the HTTP connection pool. For performance reasons, HTTP connections are not released after use but remain open. Client threads borrow connections from the pool and release them once they send their payload.

Parameter	Meaning	Default value	Probability of change
`http.max.connections.per.host`	Number of pooled connections for a specific remote http host.	`5`	Medium
`http.max.total.connections`	Total number of pooled connections.	`10`	Medium

Table 19: HTTP connection parameters

6.4.4 Behavior on negative MPI response

There are two failure classes when the process of sending messages to MPI fails:

MPI receives the message but is not able to process it due to failing authentication, bad message format, server overload etc. In this case a respective HTTP response code is returned.

MPI cannot receive the message due to network failure or because MPIis simply not running. In this case the request will time out and no response is available.

In the latter case, the messages are always redelivered for a certain period of time.

In the case of a negative response code, you can configure on which response codes the failing message is redelivered.

Parameter	Meaning	Default value	Probability of change
`pxsa.redeliver.on.statusCodes`	A space-separated list of HTTP response codes	`302 401 503`	low

Table 20: Redelivery on Http Response Codes

6.4.5 Supported Messages

The pxsa interface handles a selected set of HL7 messages, forwards them to MPI and returns an ACK message. All other messages are blocked and a NAK (negative acknowledgement) is returned. For application customizations that explicitly handle other messages it is necessary to extend this set of messages.

Parameter	Meaning	Default value	Probability of change
`pxsa.supported.trigger.ADT`	Supported trigger events of the ADT domain.	`A01 A04 A08 A14 A18 A28 A31 A34 A40`	low
`pxsa.supported.version.ADT`	Supported message versions for ADT messages.	`2.2 2.3 2.3.1 2.4 2.5`	low

Table 21: Supported messages in pxsa

6.4.6 Pxsa context properties for ISO OIDs

The properties primarily deal with ISO OIDs for global identifiers.

Parameter	Meaning	Default value	Probability of change
`hl7.translate.language.communication`	Determine patient's preferred language (Field PID-15) - yes or no?	`true`	low
`hl7.language.codesystem`	Code system for patient's preferred language. Relevant only when `hl7.translate.language.communication = true`	`2.16.840.1.113883.6.100`	low
`hl7.national.id.root`	Code system for national patient identifiers (Field PID-19).	`2.16.840.1.113883.4.1`	medium
`hl7.insurance.enable`	Process insurance data (segment IN1) - yes or no?	`false`	low
`hl7.insurance.insured.id.variable.root.enable`	Should the the patient ID root (see `hl7.insurance.insured.id.root)` be appended to the numeric insurer ID from Fields IN 1-3 - yes or no?	`true`	low
`hl7.insurance.insured.id.root`	Base identifier (root) for the insurance membership number from Field IN 1-49. Relevant only when `hl7.insurance.enable = true`	`1.2.3`	high
`hl7.insurance.company.id.root`	Base identifier for the insurer (Field IN1-3). Relevant when `hl7.insurance.enable = true`	`1.2.3`	high

Table 22: Pxsa-specific parameters dealing with ISO OIDs

6.4.7 Enabling secure transport

The MSB is able to use TLS over HTTP and TLS over MLLP for secure message transport. This section summarizes the necessary changes in the configuration. The main tasks are to generate proper server certificates for the MSB (keystore) and to obtain the issuer's certificates for trusting secured connection to remote hosts (trust-store).

The certificate files must be pointed to either by a Java system property or by configuration in *context-common.properties* file. For examples, see the section TLS settings [page 32] and the `#TLS configuration` part of the *context-common.properties* file.

6.4.7.1 Receiving data over https

HTTP connections are managed globally by the MSB for all incoming HTTP connections. This also applies when using HTTP with transport-level security. Setting the `msb.https.port` property in `context-common.properties` to a non-null value (for example, 8485) enables incoming HTTPS connections for PixUpdate on the specified port.

6.4.7.2 Receiving data over mllps

In `context-pxsa.properties`, change the `input.url.mllp` property to reference the security filter defined in the `context-msb` application context file. For example,
`mina:tcp://0.0.0.0:8777?sync=true&codec=#hl7codec&filters=#sslFilter`

6.4.7.3 Sending data to MPI over https

In `context-common.properties`, change the `pxs.http.url.base` property to contain https as the protocol. For example, `https://localhost:8080`

6.5 PIXPDQ context properties

The function of the pixpdq component is explained in detail in the Interface references.

6.5.1 Enabling/disabling pixpdq

The pixpdq component interface can be partly enabled or disabled.

Parameter	Meaning	Default value	Probability of change
`pix.enabled`	Enable/disable all PIX related interfaces (Feed, Query, Update)	`true`	low
`pdq.enabled`	Enable/disable pdq	`true`	low

Table 23: pixpdq-specific control parameters for general settings

6.5.2 Receiving endpoints

This part of the configuration is where exposed endpoints are configured. If a port is set to 0, the endpoint is disabled.

Parameter	Meaning	Default value	Probability of change
`pixFeed.mllp.port`	MLLP server for receiving ADT messages (PIX feed).	`3700`	medium
`pixQuery.mllp.port`	MLLP server for receiving QBP message QBP^Q23 (PIX query).	`3710`	medium
`pdqQuery.mllp.port`	MLLP server for receiving QBP message QBP^Q22 (PIDQ query).	`3750`	medium
`pixFeed.mllps.port`	MLLP secure server for receiving ADT messages (PIX feed).	`3705`	medium
`pixQuery.mllps.port`	MLLP secure server for receiving QBP message QBP^Q23 (PIX query).	`3715`	medium
`pdqQuery.mllps.port`	MLLP secure server for receiving QBP message QBP^Q22 (PIDQ query).	`3755`	medium

Table 24: pixpdq-specific control parameters for receiving endpoints

6.5.3 Performance-related settings

Some PIXPDQ settings are related to optimizing message thoughput.

Parameter	Meaning	Default value	Probability of change
`validate.request`	Enable/disable validation of both received and translated HL7v2 messages. Enable this to identify erroneous messages before they are sent to MPI.	`true`	medium
`validate.response`	Enable/disable XML schema validation of XML messages returned by MPI.	`true`	medium

Table 25: pxsa parameters

6.5.4 ATNA auditing properties

The PIX and PDQ transactions produce IHE ATNA-compliant audit entries to a Syslog server on behalf of the MPI backend.

Parameter	Meaning	Default value	Probability of change
`atna.audit.enabled`	Enable/disable ATNA auditing.	`false`	medium
`allow.incomplete.audit`	Write audit logs even in the case of incomplete source data.	`true`	medium
`atna.audit.host`	ATNA repository host	`localhost`	high
`atna.audit.port`	ATNA repository port	`514`	high
`atna.audit.sourceId`	Source ID identifier	`localhost`	
`atna.audit.enterpriseSiteId`	Enterprise ID identifier	`localhost`	

Table 26: ATNA auditing properties

6.5.5 Handling of Identifiers

The parameters in the table determine how secondary patient IDs and sender and receiver information are handled in PIX and PDQ interfaces.

Parameter	Meaning	Default value	Probability of change
`national.patient.id.root`	The OID of the national patient ID such as a Social Security Number (SSN). IDs with this OID are suppressed if `suppress.national.patient.id` is true.	`2.16.840.1.113883.4.1`	high
`suppress.national.patient.id`	Whether national patient IDs such as a SSN number shall be removed from the PID-3 identifier list in a PIX-Query/PDQ response.	`true`	medium
`mpi.patient.id.root`	The OID of the MPI patient ID. IDs with this OID are suppressed if `suppress.mpi.patient.id` is true.	`2.16.840.1.113883.3.37.4.1.1.2.1.1`	low
`suppress.mpi.patient.id`	Whether the MPI patient ID of MPI shall be removed from the PID-3 identifier list in a PIX-Query/PDQ response.	`false`	medium
`suppress.account.number`	Whether configured account numbers shall be removed from the PID-3 identifier list in a PIX-Query/PDQ response.	`true`	low
`mpi.message.id.root`	Configuration of MPI message OID. Must correspond with MPI device settings.	`2.16.840.1.113883.3.37.4.1.1.2.1.4`	low
`mpi.system.id.root`	Configuration of MPI system OID. Must correspond with MPI device settings.	`2.16.840.1.113883.3.37.4.1.1.2`	low
`mpi.system.id.extension`	Configuration of MPI system extension. Must correspond with MPI device settings.	`1`	low

Parameter	Meaning	Default value	Probability of change
`sending.application.name`	Name of the application that sends MPI messages for PixUpdate/PixFeed.	`ICW_MPI`	medium
`sending.assigning.authority`	Name of the application that sends MPI messages for PixUpdate/PixFeed.	`ICW_MPI`	medium
`sending.facility.name`	Name of the organization that sends MPI messages for PixUpdate/PixFeed.	`ICW`	high
`suppress.universal.id`	Whether the OIDs of identifiers in outgoing PIX/PDQ messages are removed (i.e. only an assigning authority name is transferred) or not.	`false`	medium

Table 27: Settings for PIX/PDQ responses

6.5.6 Enabling secure transport

The MSB is able to use TLS over HTTP and TLS over MLLP to secure message transport. This section summarizes the necessary changes in the configuration. The main task is to generate proper server certificates for the MSB (keystore) and to obtain the issuer's certificates for trusting secured connection to remote hosts (truststore).

The certificate files must be pointed to either by a Java system property or by configuration in context-common.properties. Please see section 6.3.8 and the bottom of this property file for examples.

6.5.6.1 Receiving data over mllps

In `context-pixpdq.properties`, enable or disable the `*.mllps.port` properties. If you comment out the `*.mllp.port` properties, only the TLS transport remains open.

6.5.6.2 Receiving data over https from MPI

HTTPS is enabled by setting the `msb.https.port` to a value greater 0 (e.g. 8485).

6.5.6.3 Sending data to subscribers over mllps

In `subscriber-pix.properties`, add a `?secure=true` parameter to the `feedUrl` and/or `updateUrl` properties, for example, `remotehost:8081?secure=true`.

If you want to ensure the usage of a certain SSL protocol or certain encryption algorithms add an `sslProtocols` and/or an `sslCiphers` parameter.

For example:

```
s5.feedUrl=localhost:3805?secure=true&sslProto-
cols=TLSv1,SSLv3&sslCi-
phers=SSL_RSA_WITH_NULL_SHA,TLS_RSA_WITH_AES_128_CBC_SHA
```

6.5.6.4 Sending data to MPI over https

In `context-common.properties`, change the `pxs.http.url.base` property to contain https as the protocol. For example, `https://localhost:8080`.

6.5.7 Other settings

Parameter	Meaning	Default Value	Probability of Change
`pdq.response.name.typeCode`	Name type code to be used for names in PDQ responses (PID-5-7). If empty no name type code is entered. Valid type codes are defined in HL7 Table 200. Default value “L” means “Legal”.	L	Low
`send.birthname.in.PID6`	Whether the birth name shall be sent in PID-6 if the MPI acts as PIX Source. Otherwise the birth name is sent in PID-5(1).	true	Low
`pixpdq.pxs280Compatibility`	Set this to true if the MSB should work with PXS 2.8.0 and 2.7.x. Set this to false if the MSB should work with MPI 2.8.1 or PXS 2.9.x or MPI 3.0	false	Low

6.6 Code System mappings

Code System Mappings encapsulate part of the translation logic in theintegration program in a simple and easily modifiable form.

A mapping represents the unique correspondence between two sets and is somewhat like Java's Map or Dictionary data structures. In MSB, this mechanism is represented in the form of mapping scripts, which are made up of individual mappings. Their general structure is shown in Figure 2.

Among other functions, the mapping example shown transforms the value O (on the left side of the mapping) to the value U (on the right side). If the left side or the right side correspond to a standardized code system, the code system identifiers (OIDs) will be noted at the beginning of the mapping entry (option). You can find more information about the use of mappings in the Customization Reference.

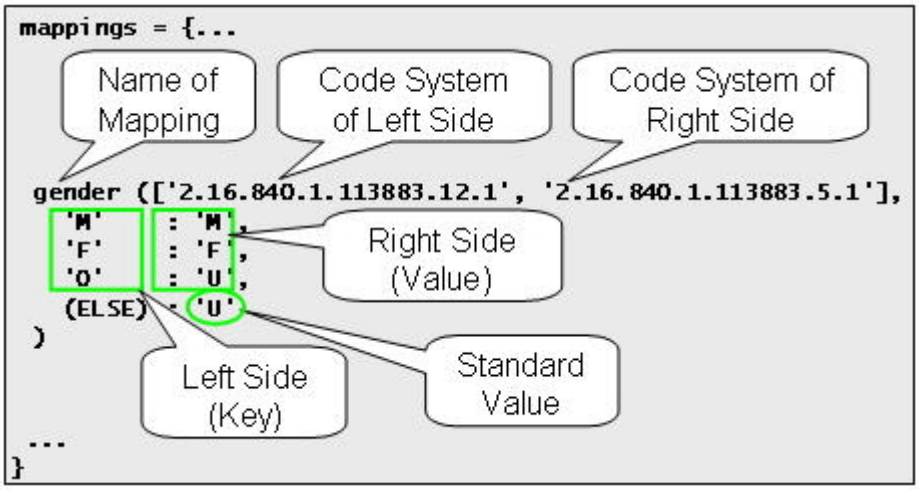


Figure 2: Structure of a mapping script in MSB

6.6.1 Bidirectional mappings

Mapping tables in MSB are used either unidirectionally or bidirectionally. The properties of these two mapping types are summarized in the table **Differences between unidirectional and bidirectional mappings**.

Mapping type	Naming convention	Restriction on the left side	Restriction on the right side	Syntax for standard values (example)
unidirectional	map...	Same keys not permitted.	<none>	`(ELSE) : 'value'`
bidirectional	bdm…	Same keys not permitted.	Same values not permitted.	`(ELSE) : 'L->R', 'L->R' : (ELSE)`

Table 28: Differences between unidirectional and bidirectional mappings

In bidirectional mappings a unique one-to-one correspondence must exist between the elements on the left and right sides of the mapping. A bidirectional mapping can have one standard value (‚ELSE') for each direction.

6.6.2 Compound mapping entries

Compound values can be used on both the left and the right side of the mapping instead of simple strings, and the mapping service will treat these like a list of strings. The syntax for compound entries is: `<element 1>~<element 2>...<element N>`.

EXAMPLE **Compound mapping example**

The entry `'SAP-ISH~HZL': '2.16.840.1.113883.3.37.4.1.1.2.411.2'` has a compound value on the left side consisting of the strings *SAP-ISH* and *HZL*. Since the ~ character serves as a separator it cannot be used as an element of a compound entry.

6.7 Pxsa mappings

The primary purpose of changing pxsa mappings is to map the sending facility and application to ISO OIDs that are used by MPI to recognize the IDs that are local to the source system. The default pxsa mappings are defined in the file `default-pxsa-mappings.map`.

Mapping Name	Description	Example
patientID_OID	Maps the combination "sending application + sending facility" to the namespace of the Patient ID.	SAP-ISH~HZL' : 2.16.840.1.113883.3.37.4.1.1.2.411.1'
patientID_namespace_OID	Maps the combination "sending application + sending facility+ namespace ID" to the namespace of the Patient ID.	'SAP-ISH~HZL~HOSPIS' : '2.16.840.1.113883.3.37.4.1.1.2.411.1.0'
messageID_OID	Maps the combination "sending application + sending facility" to the namespace for messages.	'SAP-ISH~HZL' : '2.16.840.1.113883.3.37.4.1.1.2.411.4'
organization_OID	Maps the sending facility to the namespace for organizations.	'HZL' : '2.16.840.1.113883.3.37.4.1.1.1.100.1~HZL'
device_OID	Maps the combination "sending application + sending facility" to the namespace for systems (applications).	'PRG~ICW' : '2.16.840.1.113883.3.37.4.1.1.2~2'

Table 29: Mappings for instance IDs in pxsa

You can overwrite all mappings or individual mappings by modifying the `pxsa-mappings.map` file. It is intended to be used for simple custom adaptions of the mapping definitions.

NOTE

When adding or overwriting translation rules, make sure to specify all of the other translations of the respective default mapping as well.

6.8 pixpdq mappings

6.8.1 Translation of patient IDs

Patient identifiers are critically important for all transactions invoolving the IHE PIX and PDQ profiles. HL7 2.3.1 and 2.5 represent patient IDs as a combination of ID and its office of assigning authority. The assigning authority is specified either as a namespace ID (alphanumeric string) or as a universal ID (ISO OID).

Data type CX (Extended Composite ID with Check Digit)

	Patienten-ID			
	ID	Assigning Authority		
		Namespace ID	Universal ID with Type	
			Universal ID	Type of Universal ID
Example				
	12345678	NMS1	1.2.3	ISO
HL7 2.x	12345678^^^NMS1&1.2.3&ISO			

Figure 3: Patient ID structure for PID/PDQ using an example in HL7 notation

Translation of HL7 notation into an MPI compatible form occurs in several steps.

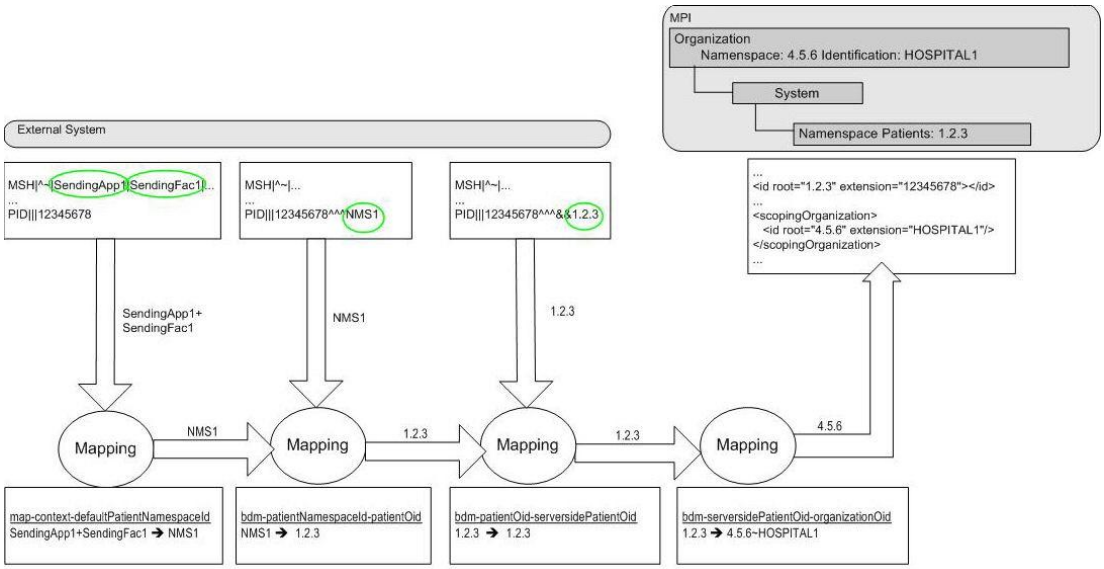


Figure 4: Significance of mappings in pixpdq

- If neither the namespace ID nor the universal ID is given, the namespace ID will be derived from the combination of the Fields MSH-3 (sending application) and MSH-4 (sending facility).
- If the namespace ID is known, another mapping linking the namespace ID with the universal ID will be used.
- Later on during processing, an option exists to map this universal ID (ISO OID) to a different ISO OID.
- Additionally, queries require that each assigning authority be associated with an organization.
- The table below contains a summary of the mappings that pertain to patient IDs. Compound mapping entries are used. The default pxsa mappings are defined in the file `default-pixpdq-mappings.map`.

Mapping	Description
`map-context -defaultPatientNamespaceId`	Represent namespace ID as combination of the MSH-3 and MSH-4 message fields.
`bdm-patientNamespaceId -patientOid`	Represent namespace ID as pertinent universal ID (ISO).
`bdm-patientOid -serversidePatientOid`	Represent universal ID as identifier from MPI. If identifiers are freely selected in MPI a one-to-one mapping is recommended.
`bdm-serversidePatientOid -organizationOid`	For queries: Maps a universal ID to an organization ID.

Mapping	Description
`bdm-context -accountNumberNamespaceId`	Analogous to `map-context -defaultPatientNamespaceId`. Billing numbers are used here instead of a patient ID.
`bdm-accountNumberNamespaceId -accountNumberOid`	Analogous to `bdm-patientNamespaceId -patientOid`. Billing numbers are used here instead of a patient ID.
`bdm-accountNumberOid -serversideAccountNumberOid`	Analogous to `bdm-patientOid-serversidePatientOid`. Billing numbers are used here instead of a patient ID.

Table 30: Mappings for patient IDs

You can overwrite all or individual mappings by modifying the `pixpdq -mappings.map` file. It is meant for simple custom adaptions of the mapping definitions.

NOTE **Translation rules**

When adding or overwriting translation rules, make sure to specify all of the other translations of the respective default mapping as well.

6.8.2 Subscriber for the transaction update notification

The PIX profile enables external systems to subscribe to update notifications from the Cross-Reference Manager. Subscribers specify in advance which range (assigning authorities or domains) of patient IDs they want to receive update notifications for. Update notification subscribers are configured in the file `subscribers-pix.properties`, see the **Subscriber configuration for update notification** table .

Parameter	Meaning	Default value	Probability of change
`subscribers`	List of subscribers to receive notifications.	`s0 s1 s2`	medium
`<subscriber-name>.name`	Symbolic name of the subscriber, freely selected.	`Subscriber1`	medium

Parameter	Meaning	Default value	Probability of change
`<subscriber-name>.feedUrl`	Host and port number of a Pix Feed consumer that receives MPI index patient feed messages. Remove this setting to disable sending feed messages.	`localhost:8900`	
`<subscriber-name>.updateUrl`	Host and port number of a Pix Update consumer that receives MPI index patient update messages. Remove this setting to disable sending update messages.	`localhost:8900`	high
`<Subscriber-name>.receivingApplication`	Value of the "receiving application" message field (MSH-5) that is expected from the subscriber.	`System1App`	high
`<Subscriber-name>.receivingFacility`	Value of the "receiving facility" message field (MSH-6) that is expected from the subscriber.	`System1Fac`	high
`<Subscriber-name>.matchingNamespaceIds`	Regular expression - specifies assigning authorities (as namespace ID), that interest this subscriber. Empty value means no assigning authority.	`^.*$`	medium
`<Subscriber-name>.matchingUniversalIds`	Regular expression - specifies assigning authori-ties (as universal ID) that interest this subscriber. Empty value means no assigning authority.		medium

Table 31: Subscriber configuration for update notification

6.9 PIXPDQv3 context properties

PIX/PDQv3 requires the PIX/PDQv2 interfaces to be enabled and correctly configured, because PIX/PDQv3 is implemented as a bridge that translates the client's HL7v3 requests into PIX/PDQv2 messages and vice versa.

Edit the `context-pixpdqv3.properties` file for configuring the PIX/PDQv3 interfaces.

6.9.1 Enabling/disabling pixpdqv3

Parameter	Meaning	Default Value	Probability of change
`pixv3.enabled`	Enable/disable all PIXv3 related interfaces (Feed, Query, Update)	true	medium
`pdqv3.enabled`	Enable/disable PDQv3	true	medium

6.9.2 Receiving endpoints

HTTP connections are managed globally by the MSB for all incoming HTTP connections. The URL is composed by the global settings in context-common.properties. Using the default settings, all IHE-related web services are exposed under the following URLs, where XX holds the corresponding transaction number in the IHE ITI domain: `http://<host>:8484/msb/ws/itiXXService`

The following table summarizes these URLs:

IHE Web Service transaction	Default URL
PIX Feed V3	http://<host>:8484/msb/ws/iti44Service
PIX Query V3	http://<host>:8484/msb/ws/iti45Service
PDQ V3	http://<host>:8484/msb/ws/iti47Service
XCPD Receiving Gateway	http://<host>:8484/msb/ws/iti55service

You can also obtain the Web Service URLs using a HTML browser by entering the following address: `http://<host>:8484/msb/ws/?wsdl`

6.9.3 Settings related to pixpdqv3 to pixpdqv2 translation

In general these settings should not be changed as the pixpdq component's default configuration matches the translation settings as described below.

Parameter	Meaning	Default Value	Probability of change
`pixpdqv3.use.sender.device.name`	If `true`, the translator will use `//sender/device/name/` for MSH-3. If the value of the name is empty or the parameter is set to `false`, the translator will use a concatenation of `//sender/device/id/@root` and `//sender/device/id/@extension` instead.	`true`	Low
`pixpdqv3.use.receiver.device.name`	If `true`, the translator will use `//receiver/device/name/` for MSH-3. If the value of the name is empty or the parameter is set to `false`, the translator will use a concatenation of `//receiver/device/id/@root` and `//receiver/device/id/@extension` instead.	`true`	Low
`pixpdqv3.copy.email.as`	Name of the field for storing email addresses. On 'PID-13-1' emails are copied to PID-13-1 before forwarding to the PIX/PDQv2 interface. On 'PID-13-4' emails are copied to PID-13-4 before forwarding to the PIX/PDQv2 interface. Otherwise email information is not copied to output.	`PID-13-1`	Low

Parameter	Meaning	Default Value	Probability of change
`pixpdqv3.copy.accountNumber.as`	Name of the field for storing the account number. On 'PID-18' account number (recognized as such by the means of accountNumberRoot) will be copied to PID-18 before forwarding to the PIX/PDQv2 interface. On other values account number will not be copied.	`PID-18`	Low
`pixpdqv3.accountNumber.root`	Patient ID with this root found either under `//patient/id` or `//patient/asOtherIDs` will be handled as assigning authority for account numbers.	`1.2.3`	Low
`pixpdqv3.copy.national.identifier.as`	Name of the field to store the national identifier. On 'PID-19' account number (recognized as such by the means of `nationalIdentifierRoot`) will be copied to PID-19 before forwarding to the PIX/PDQv2 interface. On other values account number will not be copied.	`PID-19`	Low
`pixpdqv3.copy.birthname.as`	Name of the field to store the birth name. On 'PID-5' birth name will be copied into the output as a repetition of PID-5 with PID-5-7 set to 'B'. On 'PID-6' birth name will be copied into the output as the only repetition of PID-6. Otherwise birth name will be ignored. In general this setting corresponds with send.birthname.in.PID6, which defines how the MSB as Pix Source handles birth names.	`PID-6`	Low

Parameter	Meaning	Default Value	Probability of change
`pixpdqv3.copy.other.ids`	If true, patient IDs listed under "asOtherIDs" will be copied before forwarding to the PIX/PDQv2 interface. Otherwise "asOtherIDs" will be ignored.	`true`	Low
`pdqv3.output.result.total.quantity`	If true, `<resultTotalQuantity nullFlavor="UNK/>` will appear in the output. Otherwise the element will not be present.	`true`	Low
`pdqv3.output.result.remaining.quantity`	If true, `<resultRemainingQuantity nullFlavor="UNK/>` will appear in the output. Otherwise the element will not be present.	`true`	Low
`pdqv3.default.match.quality`	Predefined constant value of `<code>//patient/queryMatchObservation/value/@value`	`100`	Low

6.10 XCPD context properties

Edit the `context-xcpd.properties` file for configuring the XCPD Receiving Gateway interface.

XCPD Receiving Gateway requires the PDQv3 interfaces to be enabled and correctly configured, because XCPD is implemented as a bridge that translates the client's requests into PDQv3 messages and vice versa.

6.10.1 Enabling/disabling xcpd

Parameter	Meaning	Default Value	Probability of change
`xcpd.enabled`	Enable/disable XCPD Receiving gateway	`true`	Medium

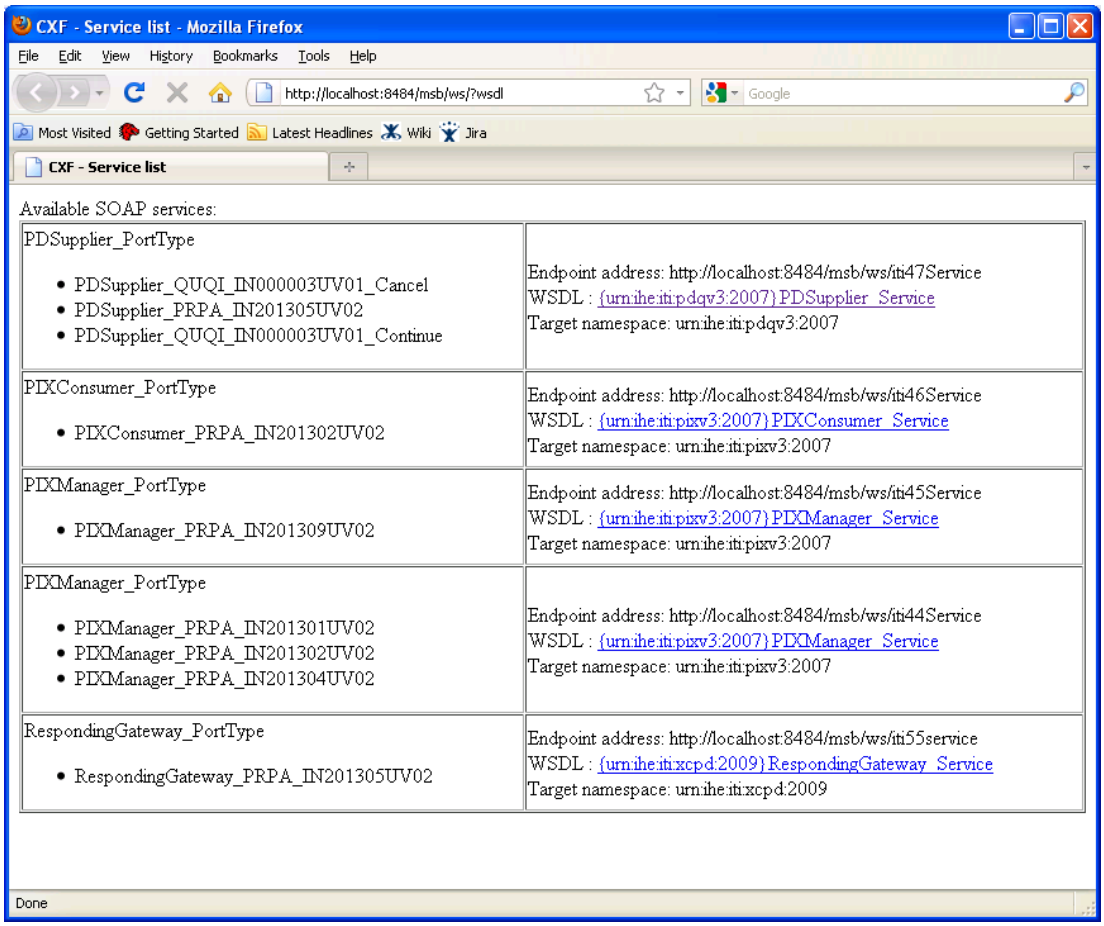
6.10.2 Receiving endpoints

HTTP connections are managed globally by the MSB for all incoming HTTP connections. The URL is composed by the global settings in `context-common.properties`. Using the default settings, all IHE-related web services are exposed under the following URLs, where XX stands for the corresponding transaction number in the IHE ITI domain: `http://<host>:8484/msb/ws/itiXXService`.

The following table summarizes these URLs:

IHE Web Service transaction	Default URL
PIX Feed V3	http://<host>:8484/msb/ws/iti44Service
PIX Query V3	http://<host>:8484/msb/ws/iti45Service
PDQ V3	http://<host>:8484/msb/ws/iti47Service
XCPD Receiving Gateway	http://<host>:8484/msb/ws/iti55service

You can also obtain the Web Service URLs using a HTML browser and the following address: `http://<host>:8484/msb/ws/?wsdl`



6.10.3 Other XCPD configuration items

Parameter	Meaning	Default Value	Probability of change
`xcpd.allow.multiple.patients`	This configures how to proceed if the query returned more than one matching patient. If set to `true`, all patient matches are sent back to the Initiating Gateway. If set to `false`, an empty result is sent back to the Initiating Gateway.	`false`	Medium
`xcpd.query.id.root`	OID to be used for the internal PDQv3 query intentifier.	`1.2.3`	High
`xcpd.home.community.id.root`	OID of the home community of the receiving gateway.	`1.2.3.4.5`	High

7 Starting and Stopping MSB

NOTE **Configuration**

For the MSB to execute correctly the system must be correctly configured, see System Configuration [page 22]. Without proper configuration, the MSB may start up, but messages are not translated and correctly forwarded to MPI.

7.1 Starting MSB under Linux

7.1.1 As service

This is the recommended way to run the MSB. The MSB automatically starts when the server boots up. To do this manually, execute:

```
root@server:~> /etc/init.d/msb start
```

7.1.2 As script

Execute the start script located in <MSB_HOME>:

```
msb@server:~> nohup <MSB_HOME>/start-node.sh &
```

7.2 Stopping MSB under Linux

7.2.1 As service

The MSB automatically stops when the server shuts down. To do this manually, execute:

```
root@server:~> /etc/init.d/msb stop
```

7.2.2 As script

1. Get the process number of the MSB process. You can get this by entering the following command:

   ```
   msb@server:~> ps -efa | grep msb | grep -v grep
   ```

2. Send a KILL signal to the MSB process

   ```
   msb@server:~> kill <process number>
   ```

3. The process has finished stopping when the response to a ps command is empty, see step 1.

7.3 Starting MSB under Windows

7.3.1 As service

This is the recommended way to run the MSB. The MSB automatically starts when the server boots up. To achieve this manually:

1. Open the Service Control Panel (**Control Panel | Administrator Tool |Service**)
2. Double-click the installed service `MSB`
3. Click **Start**

Alternatively you can enter the following command into a command window:

```
net start MSB
```

7.3.2 As script

Execute the start script located in <MSB_HOME>:

```
<MSB_HOME>/start-node.bat
```

7.4 Stopping MSB under Windows

7.4.1 As service

1. Open the Service Control Panel (**Control Panel | Administrator Tool | Service**)
2. Double-click the installed service `MSB`
3. Click **Stop**

Alternatively you can enter the following command into a command window:

```
net stop MSB
```

7.4.2 As script

1. Hit **Ctrl-C** in the command window in which the MSB was started
2. Confirm by entering **Y**

8 Monitor MSB

8.1 Connecting to MSB with JConsole

JConsole monitors Java VM processes and enables the administration of applications using MBeans (Managed Beans) in accordance with Java Management Extensions (JMX) specifications. MBeans are edited at runtime using a JMX client, such as JConsole. The system does not need to be restarted. JConsole is a component of the Java SDK.

NOTE

For selected monitoring and management functions as described in sections 8.1, 8.2.2, 8.2.3 and 8.3, the specialized management application IPM can be used instead of the standard tool JConsole, for more information see Integration Platform Manager [page 118].

This section describes how to monitor MSB from a workstation computer.

1. Check whether the *<JDK_HOME>/bin* directory has been set in your PATH environment variable. If not, switch to this directory.
2. Enter `start-console` at the command prompt.
   ⇨ JConsole starts. A **New Connection** dialog will open.
3. Select **Remote Process**.
4. Enter the JMX connection parameters. By default, the string to be entered is `service:jmx:rmi://<server>:1100/jndi/rmi://<server>:1099/jmxrmi/camel`
5. Click **Connect**.

   NOTE

   Check whether the MSB really is running. The firewall settings must allow a server connection over the port used.

⇨ JConsole connects with the corresponding MSB interface. The main **Overview** window opens. Use this window to monitor system-related resources such as required memory, threads, CPU utilization, and so on. You will find more information about this topic in the JConsole online help and in this manual in the section on Monitoring the MSB [page 64].

8.2 Monitoring the MSB

Select the MBean tab, see Figure 6. This tab contains a directory tree with MBeans that are being managed by JMX at different levels of the system.

The following functions help you to monitor the system.

See JVM [page 64].

See Messaging server (ActiveMQ) [page 65].

See Service container (Apache Camel) [page 68].

See Flow Manager [page 114].

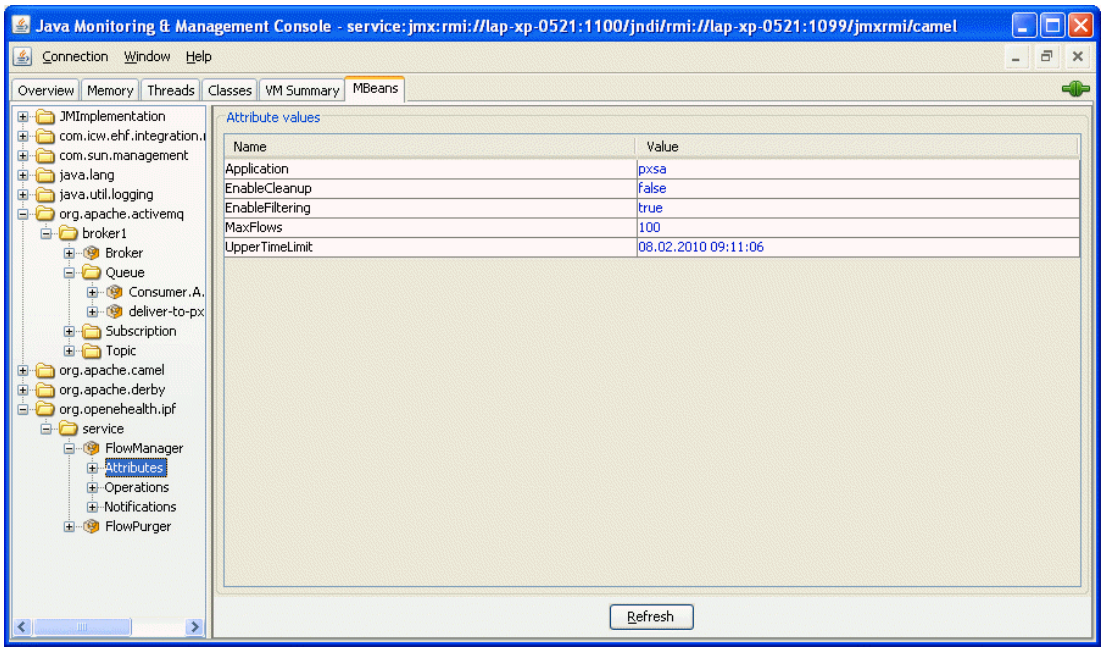


Figure 5: Hierarchy of MBeans being administered

8.2.1 Monitoring JVM

The monitoring of a Java process is supported by MBeans over the nodes `java.lang` and `com.sun.management`. Other monitoring views are located on the *Overview*, *Memory*, *Threads* and *Classes* tabs.

The figure below shows the **Memory** view during a performance test. In this example, 5000 messages are being send in parallel from five transmitters to MSB for processing. The **Heap Memory Usage** section shows your storage. It is allocated during processing and released again once the test is ended. The maximum required memory is about 100 MB.

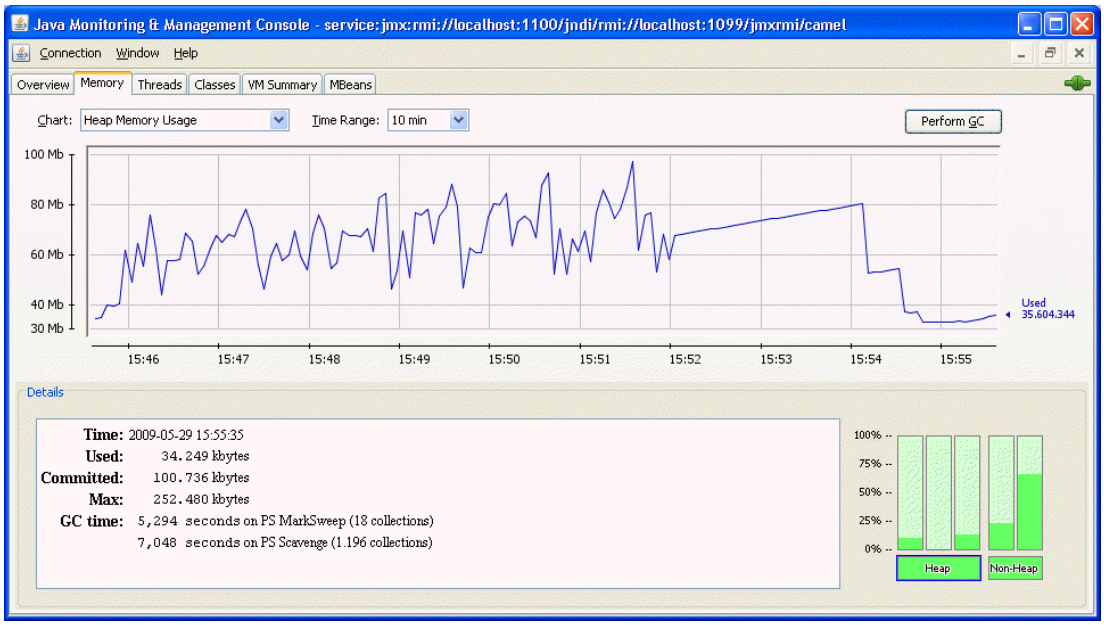


Figure 6: Memory requirements during message processing – pxsa example

8.2.2 The Messaging Server

The MSB components bridge and pixpdq each use one instance of Active MQ Messaging Server to support internal message processing.

You can monitor how full the queue is and how many messages have been put into or retrieved from it in JConsole.

8.2.2.1 Monitoring the delivery queue

1. In the **MBeans** view, select the node `org.apache.activemq/broker1/Queue/delive-to-pxs/Attributes`.
2. Choose a value to monitor:
   - *EnqueueCount* and *DequeueCount* show the number of messages put into or retrieved from the queue.
   - *QueueSize* shows the number of messages currently stored in the queue.
3. Click one of the values.

⇨ A progress curve for that value will be displayed.

NOTE

This action applies only to the Read-Only Attributes shown in black.

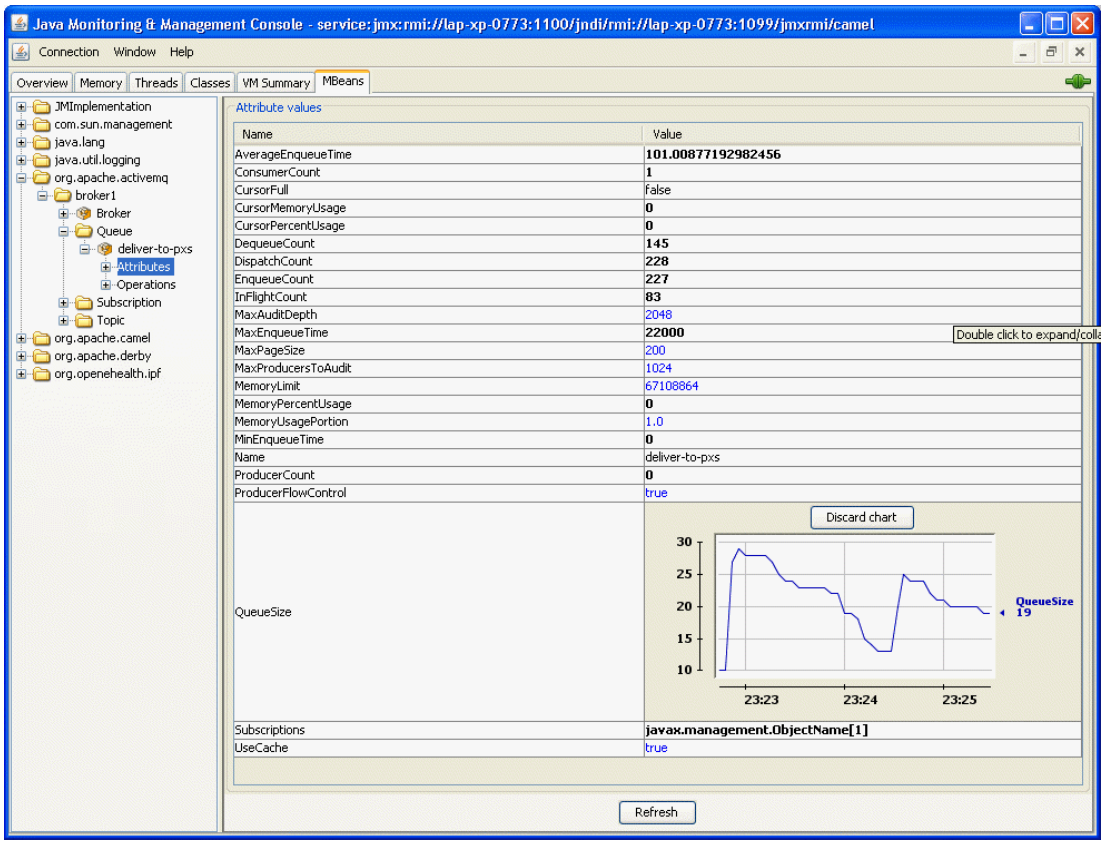


Figure 7: *Monitoring QueueSize* with JConsole. Example of a standard route

The Figure: *Monitoring QueueSize* shows an example of a progress curve for *QueueSize*. This value has risen slightly over the course of processing a few hundred messages, showing that the target application has been able to accept the messages quickly enough.

Very large or constantly growing queue sizes indicate a problem with the subsequent message processing. That means either the target application is unavailable or that it is working too slowly.

Temporarily rising queue sizes indicate a higher load on the input side. No intervention is required. The size of the queue should diminish when the load decreases.

An example of rising queue sizes is shown the Figure *Typical behavior*. In this case, the MPI core application was shut down, so that messages could not be delivered. *EnqueueCount* and *QueueSize* rise accordingly, but *DequeueCount* does not. This example shows a temporary flattening of the DequeueCount curve between 23.51 and 23.54.

Once the MPI application becomes available again the delivery of the buffered messages will be begin after a period of time, the duration of which depends on the message redelivery settings. For more information see the table Message redelivery pa-

rameters in the section Handle unavailibility of the target server [page 26]. From 23:54, *DequeueCount* begins to rise and *QueueSize* correspondingly drops to null.

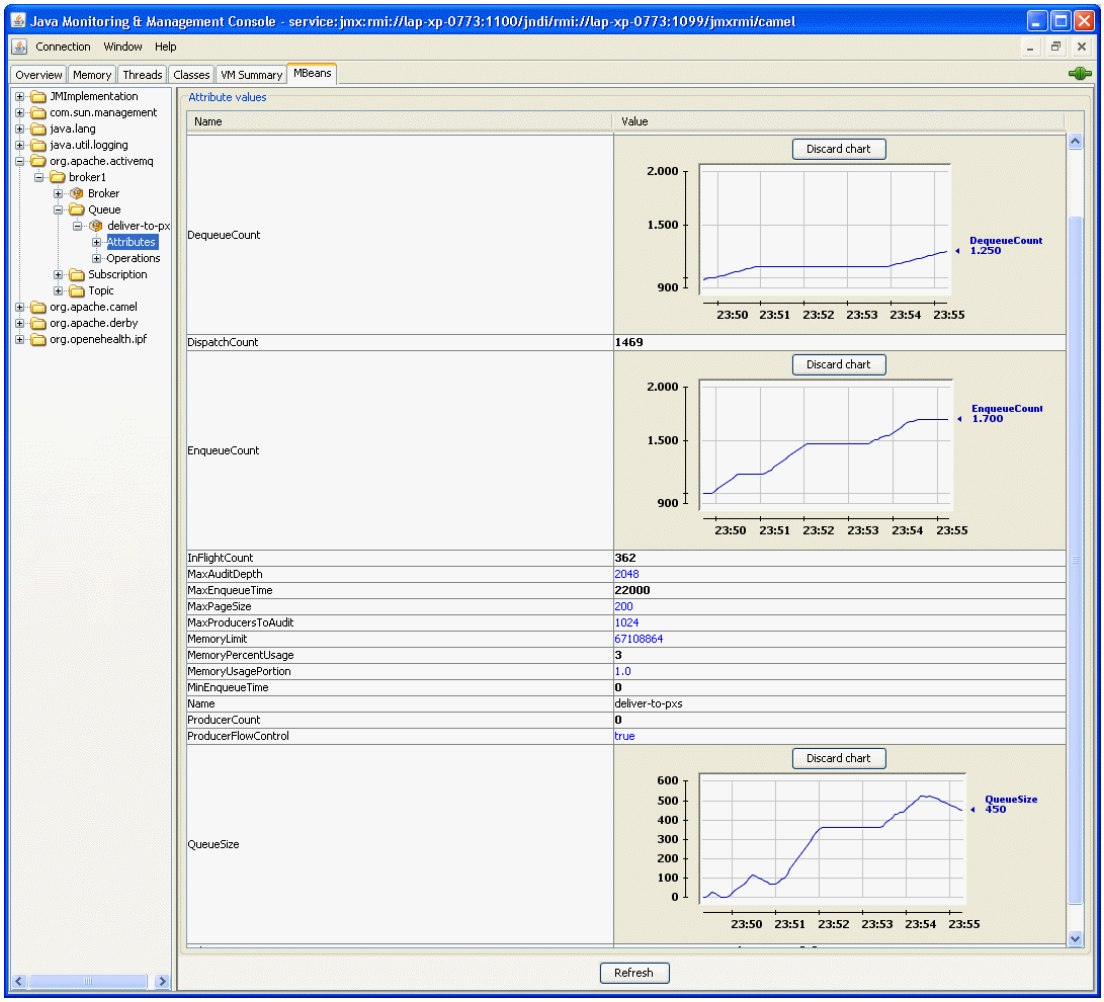


Figure 8: *Typical behavior* of the queue parameters when the target application is temporarily unavailable

NOTE **Redelivery settings**

Depending on the redelivery settings, messages that cannot be delivered will be removed from the queue after a specified period of time and stored in the error log. Use the replay function of the Flow Manager to deliver them later, see Flow Manager in JConsole [page 71].

8.2.3 Service Containers

In MSB, not only can you monitor message processing using the Messaging Server, you can also view the individual steps of a process in the Service Container of Apache Camel. This is usually only necessary if one or more errors occur.

NOTE

Analyzing message processing in the Service Container demands exact knowledge about the processing sequence, which may be specific to the project or version. In case of an error, contact your Support representative.

1. In the **MBeans** view, select a subnode of:
   `org.apache.camel/routes/<HostName>/camelContext`
2. Click the **Attributes** view.
   ⇨ The total of successfully and unsuccessfully processed messages is displayed as well as the processing time in milliseconds for each route, see the Figure below.

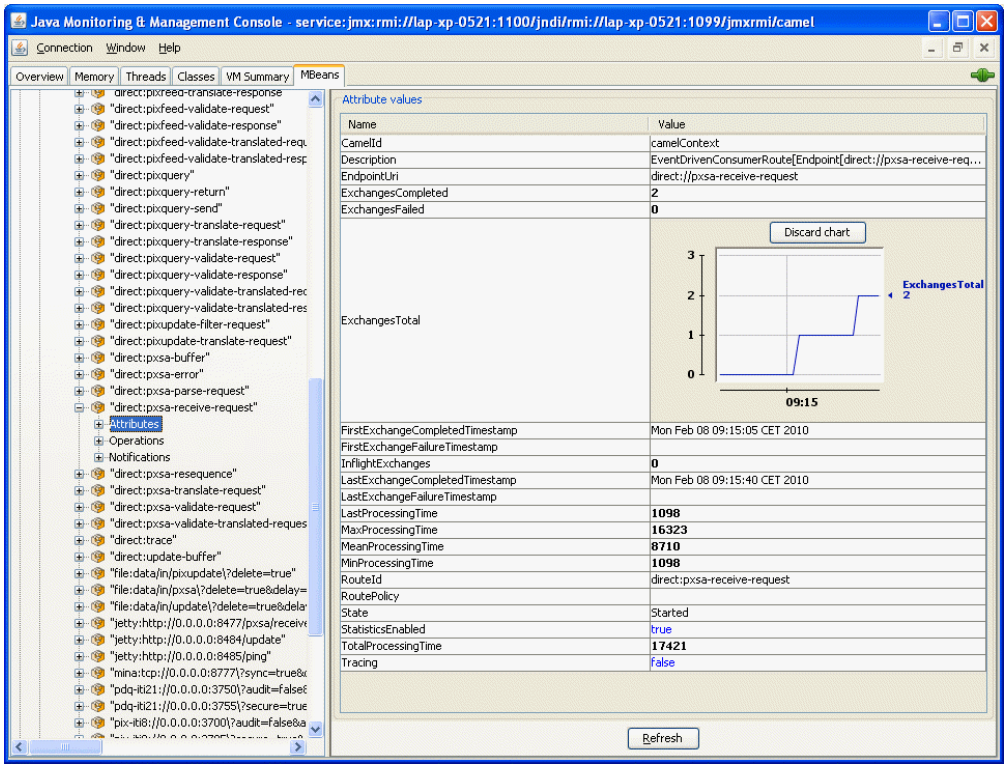


Figure 9: Monitoring single processing steps in the Service Container (example)

3. Double click a value.

⇨ A progress curve will be displayed for that value. This is only possible for *Read Only Attributes* shown in bold.

8.3 Monitoring Audit files

In addition to system-level logging and using the Flow Manager, MSB writes the messages it processes to the file system.

The audit files are in `<MSB_HOME>/data/audit/…`

Each IHE transaction and the pxsa interface writes their audit files into a separate directory. The files created all have names with the following semantics:

```
<process ID>-<host name>-<message number>-<audit type>.txt
```

Example:

7768-msbhost-000000002319-internal-response.txt

Each message is audited several times, this is indicated by the audit type:

Audit type	Meaning	Written for interface
request	The message from the source system as it is received by the MSB	all
internal_request	The translated message as it is sent to MPI	Pix feed, Pix query, Pdq query, pxsa
internal_response	The response from MPI as it is received by the MSB	Pix feed, Pix query, Pdq query
response	The response as it is sent back to the source system	Pix feed, Pix query, Pdq query
notification	Notification sent by MSB to a consumer system (when MPI acts as source system)	Pix Update, Pix Feed
error	Erroneous message (wrong message type, content, and/or internal errors). Error audits also contain the error message and a Java strack trace for further analysis.	all

Table 32: Audit message types

NOTE

Audit files do accumulate, which can cause disks or inodes (Linux) to become full, as for every incoming message, two small files are written. Take care to regularly clean or compress the audit directory.

On Linux you can define a cron job executing the following command (single line), which regularly copies audit files older than 3 days into a Zip file with a time-stamped name and removes the original files if the Zip file is consistent:

```
find $MSB_HOME/data/audit -name "*.txt" -mtime +3 | xargs -n 500 zip -g -m -T -q $MSB_HOME/data/audit/$(date +"\%y\%m\%d-\%H\%M\%S")-audits.zip -@
```

For example to execute the command every 3 days at 04:00, use this statement:

```
0 4 */3 * * find $MSB_HOME/data/audit -name "*.txt" -mtime +3 | xargs -n 500 zip -g -m -T -q $MSB_HOME/data/audit/$(date +"\%y\%m\%d-\%H\%M\%S")-audits.zip -@
```

9 Flow Manager in JConsole

The *Flow Manager* is an important infrastructure service. It is displayed in the JConsole via the MBean `org.openehealth.ipf/service/FlowManager`. Alternatively, you can use Flow Manager via the Integration Platform Manager [page 118]. Flow Manager monitors and controls the application-specific flow of messages. It stores incoming messages and their processing status in a database and updates the corresponding `MessageFlow` object as soon as a message has been delivered.

When a message has been processed and successfully forwarded to the target system, the system will save an acknowledgement (ACK) along with this specific Message Flow. If the message cannot be processed successfully, the Message Flow will be stored along with a negative acknowledgement.

One important function of the Flow Manager is to replay messages. This enables the following options:

- **Dealing with unreliable, external processing steps**. When an external service loses a message it is possible to replay it.
- **Dealing with target systems that are temporarily unavailable**. If the target system is temporarily unavailable message deliveries, including all re-delivery attempts will fail. Should this occur, administrators can manually initiate retransmission when the target system becomes available again.

Additional information is stored for each message, including processing time or number of times reset.

9.1 The Flow Manager JMX Interface

The sections below describe the Flow Manager JMX interface and the operations it supports.

The Flow Manager MBeans support:

- Searching for and displaying message flows [page 72] in a defined period of time.
- Replaying negatively confirmed message flows (NAK) [page 74] in a specified period of time.
- Replaying unconfirmed message flows [page 74] in specified period of time.
- Replaying message flows with a defined flow identifier [page 74].

You can find information on how to Parameterize the operations listed above, in the section Flow Manager Parameters [page 75].

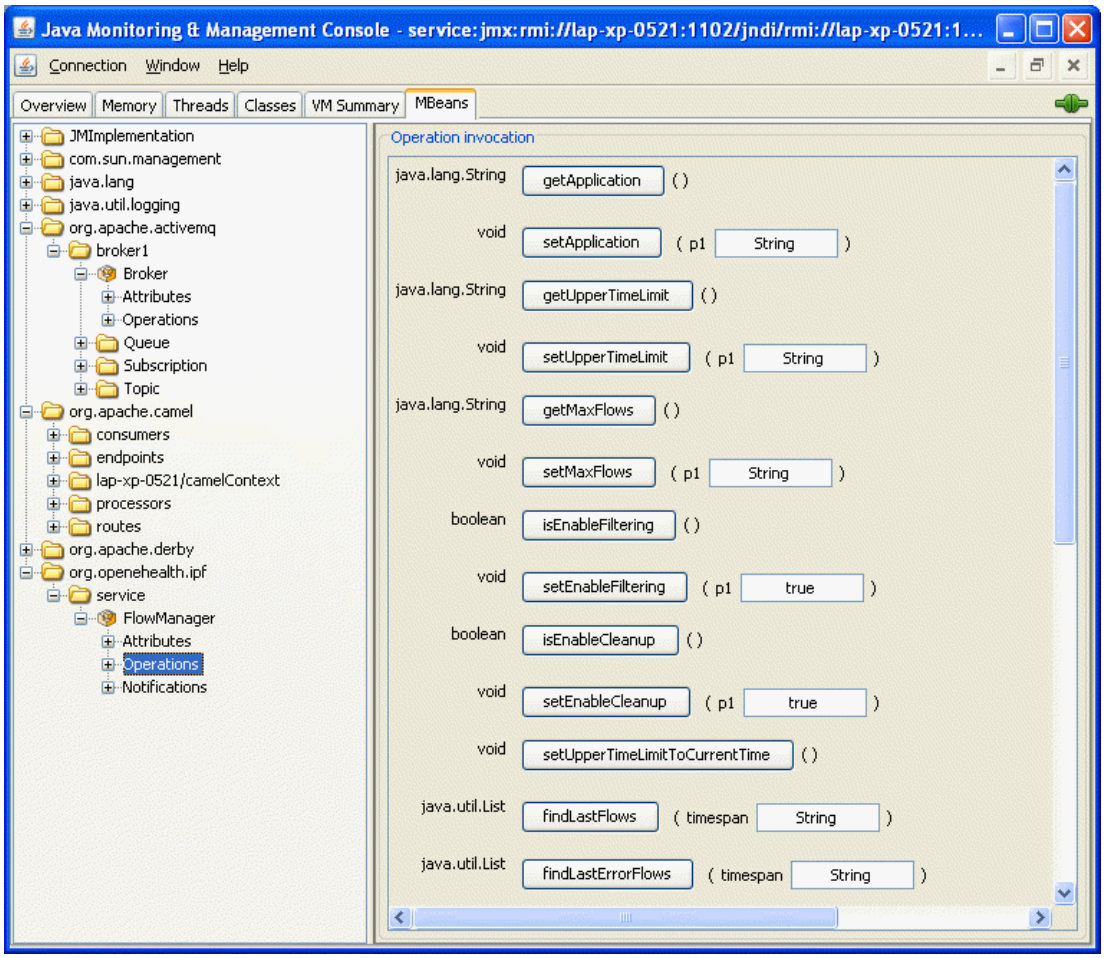


Figure 10: Flow Manager JMX interface

9.1.1 Searching for and displaying message flows

Search for message flows either over a given period of time, by acknowledgement type (ACK or NAK), or both.

- Timeframes are defined in terms of milliseconds (no symbol), seconds (s), minutes (m) and hours (h).

  Examples:

  - 2000 = 2000 milliseconds
  - 4s = four seconds
  - 3m = 3 minutes
  - 1h = one hour

Search according to timeframe

1. In the field next to the **findLastFlows** button, enter an argument.

2. Click **findLastFlows**.
3. A list of the most recently performed flows will be displayed as a result, see Error! Reference source not found..

Example: Enter the value 3m as the argument for the findLastFlows button. Flow Manager will now display all message flows for the last three minutes.

You can configure an upper limit for the timeframe, see section Flow Manager Parameters [page 75].

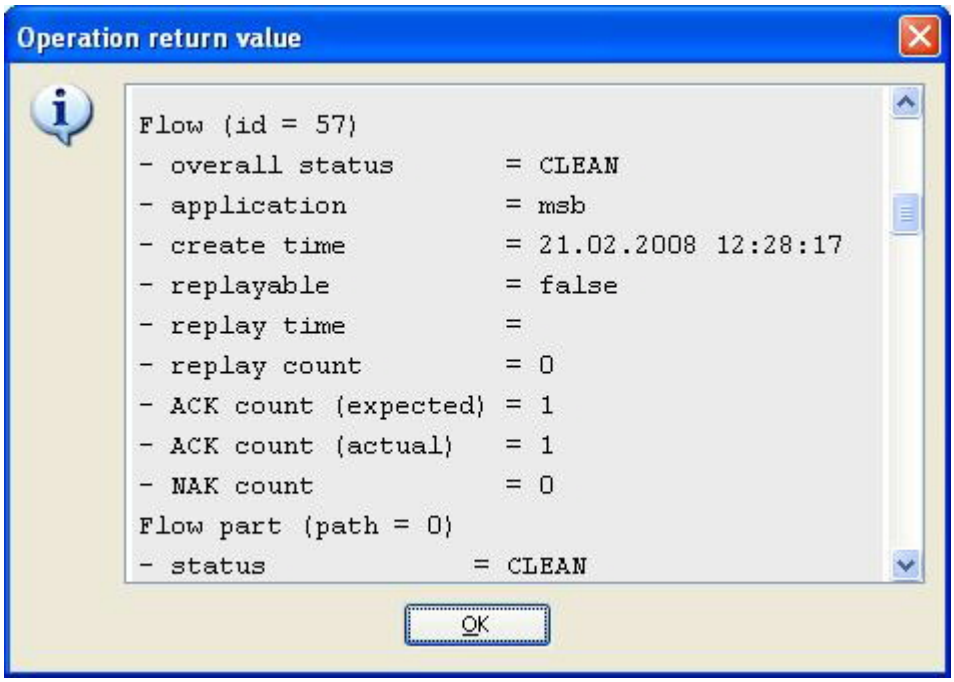


Figure 11: Results for the operation findLastFlows

The latest message flow that ran will be at the top of the list of results. In the figure above this message flow has the identifier 57.

Properties of a message flow:

- *Flow - overall status*: CLEAN or ERROR
- *Application*: The name of the integration application (component) associated with this flow.
- *Create time*: Time stamp when the message was forwarded to MSB.
- *Replayable*: Shows whether the message can be replayed.
- *Replay time*: Time stamp when the message was last replayed.
- *Replay count*: Number of times the message was replayed.
- *ACK count (expected)*: Expected number of acknowledgements for a message

- *ACK count (actual)*: Number of acknowledgements actually received for a message (number of flow parts whose status is CLEAN).
- *NAK count*: Number of negative acknowledgements for each message.

Flow Parts contains information about the history of all of the messages that belong to a specific Flow. Exactly one Flow is initiated for each incoming message. Splitting the incoming message may result in multiple messages belonging to one Flow. These are represented by multiple Flow Parts. When there is only one message per Flow there will also be only one Flow Part.

The properties of a Flow Part

- *Flow part* - status: CLEAN or ERROR
- *Flow duration*: Flow transmission time in milliseconds. This includes the time for a message to be transmitted successfully to the target system.
- *Contribution time*: Time stamp for the acknowledgement (ACK) or negative acknowledgement (NAK) of the Flow Part.
- *Filter time*: Time stamp for filtering the Flow Part during a replay.

The path for each Flow Part is shown. When a message has not been split the path is always 0. If a message has been split in two parts their paths are 0.0 and 0.1. If the message with path 0.1 is again split in two the result is three messages with the paths 0.0, 0.1.0 and 0.1.1.

9.1.2 Replaying Message Flows

Messages will either be replayed automatically following a search, or using a specific identifier.

- **replayLastFlows** button: Starts the replay of all Message Flows inside of a defined timeframe.
- **replayLastErrorFlows** button: Starts the replay of all negatively (NAK) acknowledged Message Flows inside of a defined time frame.
- **replayLastUnackFlow** button: Starts the replay of all messages in a defined time frame that were not acknowledged.
- **replayFlow** button: Starts the replay of the Message Flow with the specified Flow identifier.

Replayed messages will be filtered out by default if they were already successfully transmitted to the target system. The prevents repeated messages to the target system and duplicate data sets. The filter can be also deactivated, see section Flow Manager Parameters [page 75].

9.1.3 Flow Manager Parameters

Parameters may be assigned to Flow Manager in several ways.

In the JConsole *MBeans* tab, select the `org.openehealth.ipf/service/FlowManager` entry and select the *Attributes* view.

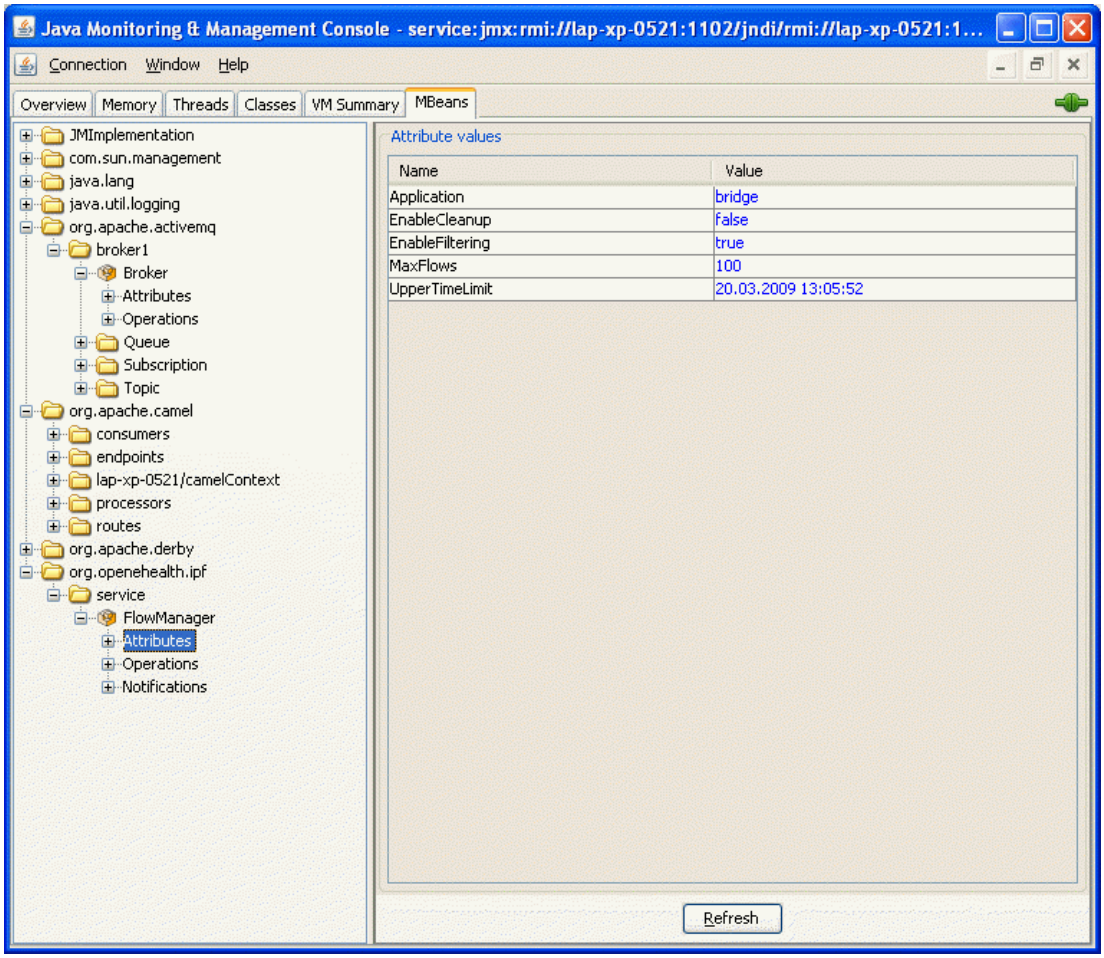


Figure 12: Flow Manager parameters

Set the attribute values as described in the Table *Attributes for the Flow Manager*.

Attribute	Description	Default
`Application`	Name of the integration application. All of the following operations are valid for the messages of only this application. If you want to apply JMX operations to another application you must change the `Application` attribute to the corresponding value.	`MSB`
`EnableCleanup`	Specifies whether a message will be deleted when the number of expected acknowledgements has been received. Cleanup means that the content of the incoming message will be removed from the database. This saves hard disk space while preserving the record in the Flow history. A replay of the exchange of messages will not be possible later.	`False`
`EnableFiltering`	The `EnableFiltering` attribute controls whether duplicate filters are activated or deactivated. For test purposes it may be useful to set the `EnableFiltering` attribute to the value `false`. Set this attribute also when the target application cannot process or store the transmitted data reliably. This means that messages will need to be re-transmitted.	`True`
`MaxFlows`	Limits the size of the result delivered by a Flow Manager search. When a search finds more than `MaxFlows` messages in the database only the `MaxFlows` most recent ones will be shown.	`100`
`UpperTimeLimit`	You can define an upper time limit when defining a time frame. The default limit is the current time. For example, queries that have the same timeframe parameter can differ from one another depending on when the query was performed. The entry format is "DD-MM-YYYY HH:MM:SS". For example for July 31, 2009, at 4:00 p.m. enter "31.07.2009 16:00:00".	

Table 33: Attributes for the Flow Manager

9.1.4 Purging the Flow Manager database

Depending on the number of messages tracked by the Flow Manager, the flow management database may reach a critical size. Besides options for manual database backups and cleanups, MSB also provides a service that purges messages from the flow management database whose age exceeds a certain limit. For example, you can configure the service to remove all messages from the flow management database that are older than 30 days.

Purge schedules are configured with cron expressions. All settings are persistent and purge jobs are re-activated automatically after JVM restarts or crashes.

How to schedule Flow Manager cleanups:

1. In the JConsole MBean view, select the org.openehealth.ipf/service/FlowPurger entry and select the Attributes view.

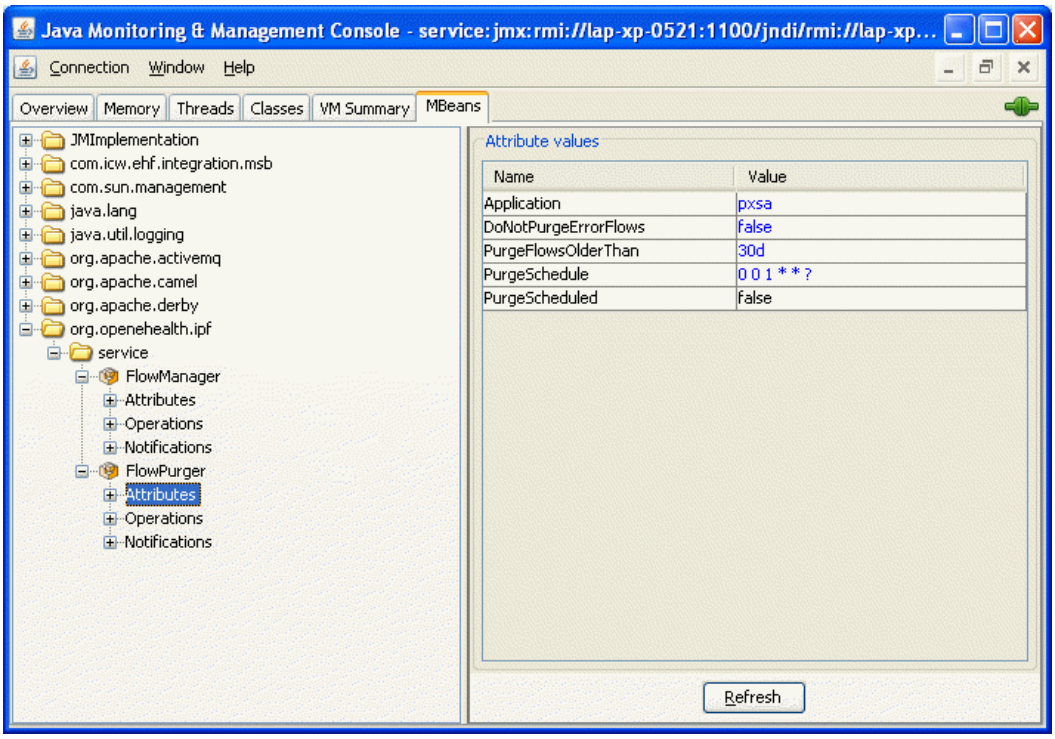


Figure 13: Figure 4: Flow Purger JMX view

2. Set the attributes as shown in the following table:

Attribute	Description	Default
`Application`	Name of the integration application.	`pxsa`
`DoNotPurgeErrorFlows`	Set to `true` if you don't want flows with the status `ERROR` to be removed from the database.	`false`

Attribute	Description	Default
`PurgeFlowOlderThan`	Messages older than the given value will be removed from the database. Valid units are seconds (s), minutes (m), hours (h) and days. Under high load, the default value should be decreased.	`30d`
`PurgeSchedule`	A cron expression that defines when purge operations are executed. By default, the schedule is to run a purge operation every day at 1:00 am.	`0 0 1 * * ?`
`PurgeScheduled`	A read-only attribute that indicates whether the purge scheduler is activated. By default the scheduler is not activated.	`false`

Table 34: Attributes for purging messages from the flow management database

3. To activate the scheduler, change to the the *Operations* view and click the **schedule** button.
4. Now the schedule is active. The *PurgeScheduled* attribute is set to `true`. To pause the scheduler, simply click the **unschedule** button.

NOTE **One-time cleanup**

For a one-time cleanup, you can click the **execute** button, which runs a single flow purge operation immediately for the current application. This operation doesn't influence the *PurgeScheduled* attribute.

10 Upgrade and Migration

10.1 Upgrade MSB 2.9 to MSB 3.0

1. As of MSB version 3.0, the MSB only supports messages that register, update, merge or query patient data, because theICW Master Patient Index 3.0 as a successor of PXS 2.9 does not support patient record information, that is encounters, diagnoses, procedures and so on, anymore. Consequently, HL7v2 MDM message support has been removed as well as the ADT messages that are not relevant for operating an MPI:
   - A02 (transfer patient)
   - A03 (discharge patient)
   - A06 (change outpatient to inpatient)
   - A07 (change inpatient to outpatient)
   - A11 (cancel admit)
   - A12 (cancel transfer)
   - A13 (cancel discharge)
   - A27 (cancel pending admit)
2. The default web application path of MPI has changed from `<host>:<port>/pxs` to `<host>:<port>/mpi`. The `pxs.http.url.localpart` property in `context-common.properties` reflects this change.
3. The directory outline of an MSB installation has been changed and aligned with the directory outline of other ICW applications, for example the MPI 3.0 The directory structure strictly separates application files from runtime data to allow for easier partitioning of disk space.

10.2 Upgrade MSB 2.8 to MSB 2.9

New configuration items are not listed in the upgrade tables. Please refer to the administration documentation and/or the respective property files for details.

10.2.1 General upgrade notes

By including Web Service interfaces into the MSB it became necessary that the MSB is now technically run as embedded web application. All HTTP transports are routed over two Servlets:

- one for the plain HTTP requests (ping, incoming pxsa interface, incoming pix update interface)
- one for web service requests (pixv3, pdqv3, xcpd).

All HTTP interfaces share the same port, by default 8484. The standard URIs are:

- http://<host>:8484/msb/plain/ping
- http://<host>:8484/msb/plain/receiver
- http://<host>:8484/msb/plain/update
- http://<host>:8484/msb/ws/itiXXService (where XX holds the ITI transaction number)

The https port are configured separately. Context path (msb), and the servlet-specific paths (plain, ws) are also configurable.

All MLLP-based transactions remain on the same ports as they were in MSB 2.8.

10.2.2 Upgrading pxsa

- Resequencing of messages has been completely removed. The `pxsa.resequence` option to either include or skip resequencing therefore does not exist anymore.
- You can now append `&minaLogger=true` to the `input.url.mllp` for detailed connection logging.

10.2.3 upgrading pixpdq

- The property `validate.translated.message` has been renamed to `validate.request`. It influences both validation of the incoming v2 request and the internal XML request to MPI.

10.3 Upgrade MSB 2.3 (Camel) to MSB 2.8 (Camel)

The content of messages processed by MSB is temporary and short-lived. Accordingly, there no fundamental reason for storing this data for extended periods of time. For the purposes of troubleshooting, debugging or analysis, it is usually sufficient to save messages for a period of hours or days. When transitioning to a newer version, MSB does not save any message data from the prior installation. MSB upgrades are limited to transforming the existing module configuration.

New configuration items are not listed in the upgrade tables. Please refer to the administration documentation and/or the respective property files for details.

Important Note: The bridge interface component is no longer supported as of MSB 2.8. In case you require this interface together with MPI 2.8, please install MSB 2.3 and run the bridge interface.

10.3.1 Upgrading pxsa

In MSB 2.8 this element is not configurable. The value is fixed.

MSB 2.3	MSB 2.8
Install Property *File* : `Parameter=default value`	Corresponding property *File* : `Parameter=default value`
context-pxsa.properties : `pxsa.audit*`	<fixed value> *Fixed directoy* `data/audit/pxsa`

Table 35: Changed pxsa configuration parameters (fixed)

In MSB 2.8 a corresponding configuration item exists with the same default value. The syntax may vary.

MSB 2.3	MSB 2.8
Install Property *File* : `Parameter=default value`	Corresponding property *File* : `Parameter=default value`
context-pxsa.properties : `pxs.http.url.localpart =/pxs/exporter`	*context-common.properties* : `pxs.http.url.localpart =/pxs/exporter`
context-pxsa.properties : `input.charset=ISO-8859-1`	*context-common.properties* : `hl7.charset=ISO-8859-1`

Table 36: Changed pxsa configuration parameters (same default)

10.3.2 Upgrading pixpdq

In MSB 2.8 this element is not configurable. The value is fixed.

MSB 2.3	MSB 2.8
Install Property *File* : `Parameter=default value`	Corresponding property *File* : `Parameter=default value`

context-pixpdq.properties :
`pixpdq.audit.directory=`
`data/pixpdq/audit/`
	<fixed value> :
	Fixed directories
	`data/audit/pixfeed,`
	`data/audit/pixquery,`
	`data/audit/pdq`

Table 37: Changed pixpdq configuration parameters (fixed)

In MSB 2.8 a corresponding configuration item exists with the same default value. The syntax may vary.

MSB 2.3	MSB 2.8
Install Property *File* : `Parameter=default value`	Corresponding property *File* : `Parameter=default value`
context pixpdq.properties : `hl7.charset=ISO-8859-1`	*context-common.properties* : `hl7.charset=ISO-8859-1`

context-pixpdq.properties :
`pxs.http.url.localpart=`
`/pxs/exporter`
	context-common.properties :
	`pxs.http.url.localpart=`
	`/pxs/exporter`

Table 38: Changed pixpdq configuration parameters (same default)

In MSB 2.8 a corresponding configuration item exists with a different default value. The syntax may vary.

MSB 2.3	MSB 2.8
Install Property *File* : `Parameter=default value`	Corresponding property *File* : `Parameter=default value`

context-pixpdq.properties :
`flowmgr.data.dir=`
`data/pixpdq/flowmgr`
	context-common.properties :
	`flowmgr.data.dir=data/flowmgr`

context-pixpdq.properties :
`pixUpdate.http.url=`
`http://0.0.0.0:8480/pix/update/`
	context-common.properties :
	`pxs.update.uri=`
	`jetty:http://0.0.0.0:8484/update`

MSB 2.3	MSB 2.8

context-pixpdq.properties :
`pdq.mllp.ssl=false`
`pix.mllp.ssl=false`
	context-pixpdq.properties :
	`pixFeed.mllps.port=3705`
	`pixQuery.mllps.port=3715`
	`pdqQuery.mllps.port=3755`
	Setting the *.mllp.port and *.mllps.port properties automatically enables/disables the respective endpoints.

context-pixpdq.properties : `keystorePath, trustStorePath, keyStorePasword, trustStorePassword`	*context-common.properties* : `keystorePath, trustStorePath, keyStorePasword, trustStorePassword`

Table 39: Changed pixpdq configuration parameters (different default)

10.4 Upgrade MSB 1.x (ServiceMix) to MSB 2.8 (Camel)

Perform the following actions:

1. Block new incoming messages - reconfigure the endpoints.
2. Empty the JMS queue (check queue status with JMX).
3. Complete any outstanding flows.
4. Back up the ServiceMix directory.
5. Install MSB 2.8.
6. Parameterize MSB 2.8. Create the configuration for MSB 2.x based on the configuration for MSB 1.x. The next sections will help you to migrate the old parameters.

10.4.1 Naming conventions for directories

The table below compares the directories of MSB 1.x and 2.8 that have the equivalent meanings. This listing presumes a standard configuration, with pxsa standing in place of all other modules.

	MSB 1.x (ServiceMix)	MSB 2.8 (Camel)
Recommended installation directory (MSB_HOME) under Linux/Unix	`/usr/local/servicemix`	`/usr/local/msb`

Configuration files	`<MSB_HOME>/conf`
	`<USER_HOME>/icw/pxsa/conf`
	`<MSB_HOME>/data/…/xbean-platform.xml`
		`<MSB_HOME>/conf/`

JMS queue (ActiveMQ) data	`<MSB_HOME>/data/amq` `<MSB_HOME>/data/derby`	`<MSB_HOME>/data/activemq1`
Flow management data	`<USER_HOME>/icw/esb/data/derby`	`<MSB_HOME>/data/flowmgr`

Table 40: Naming conventions for directories

10.4.2 Upgrading the configuration

The tables below provide a summarized representation of the configuration items of MSB 1.x and MSB 2.8.

10.4.2.1 Configuration items with same default

In Camel a corresponding configuration item exists with the same default value. The syntax may vary. They are shown in the Table: *Transition of parameters from MSB 1.x to MSB 2.x - same default*.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Description	**Install Property** `Parameter= default value`	**Config Parameter** *File*:`Parameter= default value`	**Corresponding property** *File*:`Parameter= default value`	**Comment**
File: Polling interval in milliseconds	<not available>	*pxsa-file. properties* : `pxsa-v2-input.file= 1000`	*context-pxsa. properties* : `input.url. file=file: //./data/in/ pxsa?delete= true&delay= 5000&sortBy= file:name`	Parameter embedded in the endpoint URL
HL7: codeSystem for language code	<not available>	*mappings/v2v3 -codesystem- mappings.xml* : `<domain id= "language- codesystem" … value=" 2.16.840.1. 113883.6.100 " …`	*context-pxsa. properties* : `hl7.language .codesystem= 2.16.840.1. 113883.6.100`	
HL7: if true, the extension of the company will become a part of the root for the member	<not available>	*mappings/v2v3 -codesystem- mappings.xml* : `<domain id= "insurance- insured-id- variable- root-enable" … value= ”false”`	*context-pxsa. properties* : `hl7. insurance. insured.id. variable. root.enable = true`	

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HL7: Mappings for the code system	<not available>	*mappings/v2v3-codesystem-mappings.xml* : …	*pxsa-mappings.groovy* : …	
HL7: Mappings for OIDs	<not available>	*mappings/v2v3-oid-mappings.xml* : `<domain id="patientID-OID"` …	**pxsa-mappings.groovy* : `patientID_OID (...`	Camel-Mappings use a naming convention.
HL7: OID used as root for national identifiers (SSN, AHV...)	<not available>	*mappings/v2v3-codesystem-mappings.xml* : `<domain id="national-id-root"` … `value="2.16.840.1.113883.4.1"`…	*context-pxsa.properties* : `hl7.national.id.root = 2.16.840.1.113883.4.1`	Default corresponds to the SSN in the USA. Swiss AHV numbers have the OID 2.16.840.1.113883.3.37.4.1.5.1.
HL7: root for insurance companies (one for all!)	<not available>	*mappings/v2v3-codesystem-mappings.xml* : `<domain id="insurance-company-id-root"`… `value="100"`…	*context-pxsa.properties* : `hl7.insurance.insured.id.root = 1.2.3`	

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HL7: root for insurance program enrollees	<not available>	*mappings/v2v3 -codesystem-mappings.xml* : `<domain id= "insurance-insured-id-root"..value ="false/">`	*context-pxsa. properties* : `hl7. insurance. company.id. root = 1.2.3`	
HL7: Support of insurance data	<not available>	*mappings/v2v3 -codesystem-mappings.xml* : `<domain id= "insurance-enable" …` `value="false "…`	*context-pxsa. properties* : `hl7. insurance. enable = false`	
HL7: Transmit information about preferred language for communica-tion - yes or no?	<not available>	*mappings/v2v3 -codesystem-mappings.xml* : `<domain id= "language-communica-tion-enable"` `value="true"`	*context-pxsa. properties* : `hl7. translate. language. communica-tion= true`	
HTTP(S): Encoding of requests for the HTTP endpoint that receives messages	`pxsa.v2. input.http. request. encoding= ISO-8859-1`	*pxsa-http. properties*: pxsa-v2-input-http. request Encoding= ISO-8859-1	*context-common. properties* : `hl7.charset =ISO-8859-1`	Common value for receiving endpoints.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HTTP(S): Encoding of responses for the HTTP endpoint that receives messages	`pxsa.v2. input.http. response. encoding= ISO-8859-1`	*pxsa-http. properties*: `pxsa-v2- input-http. response Encoding= ISO-8859-1`	*context-common. properties* : `hl7. charset = ISO-8859-1`	Common value for all endpoints that receive messages. This value is also used for responses.
HTTP(S): Username for authenticating at the target HTTP server (MPI)	`pxsa.v3. output.http. credentials. username= <not shown>`	*pxsa-http. properties*: `pxsa-v3- output-http- credentials. username= system`	*context-common. properties* : `pxs.http. userName= system`	
HTTP(S): URL of the HTTP endpoint that sends messages	`pxsa.v3.out- put.http.url = http:// localhost: 8080/prg- all/ exporter`	*pxsa-http. properties*: `pxsa-v3-out- put-http.lo- catio- nURI=http:// localhost: 8080/prg- all/exporter`	URL is a compound of two parameters: • `context- common. properties :pxs.http. url.base= http:// localhost: 8080` • `context- common. properties :pxs.http. url. localpart = /pxs/ exporter`	The default compound URL is: http:// localhost: 8080/pxs/ exporter

MSB 1.6 ServiceMix			MSB 2.8 Camel	
JMS: Initial redelivery delay (in milliseconds)	<not available>	*pxsa-jms. properties*: `pxsa-jms-redelivery.initialRedeliveryDelay=5000`	*context-common. properties* : pxs. redelivery .initial Redelivery Delay=5000	
JMS: Maximum number of delivery retries	<not available>	*pxsa-jms. properties*: `pxsa-jms-redelivery.maximumRedeliveries=5`	*context-common .properties* : `pxs.redeliv-ery.maximumRedeliveries=6`	
JMS: The backoff multiplier (defines how much the backoff increases)	<not available>	*pxsa-jms. properties*: `pxsa-jms-redelivery.backOffMultiplier=2`	*context-common. properties* : pxs. redelivery. backOff Multiplier=2	

MSB 1.6 ServiceMix			MSB 2.8 Camel	
MLLP: Character encoding of incoming messages transported over MLLP	`pxsa.input .mllp.char. encoding= ISO-8859-1`	*pxsa-hl7bc. properties*: `pxsa-input- mllp.char Encoding= ISO-8859-1`	*context-common. properties* : `hl7.charset = ISO-8859-1`	Common value for all receiving endpoints.
MLLP: Port number of the MLLP endpoint	`pxsa.input. mllp.port= 8777`	*pxsa-hl7bc. properties:* `pxsa-input- mllp.port= 8777`	*context-pxsa. properties* : `mina:tcp:// 0.0.0.0:8777 ?sync=true& codec= #hl7codec`	Parameter embedded in the endpoint URL

Table 41: Transition of parameters from MSB 1.x to MSB 2.x - same default value

10.4.2.2 Configuration items with different default

In Camel a corresponding configuration item exists with a different default value. The syntax may vary.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Description	**Install Property** `Parameter= default value`	**Config Parameter** *File*:`Parameter= default value`	**Corresponding property** *File*:`Parameter= default value`	**Comment**
File: Directory for the import of HL7-2.x messages as files	`pxsa.v2. input. directory= file:${work. root.dir}/ pxsa/work/ v2-input`	*pxsa-file. properties* : `pxsa-v2- input.file= file:${work. root.dir}/ pxsa/hl7/ v2-input`	*context-pxsa. properties* : `input.url. file=file:// ./data/in/ pxsa?delete= true&delay= 5000&sortBy =file:name`	Parameter embedded in the endpoint URL

MSB 1.6 ServiceMix			MSB 2.8 Camel	
File: Encoding of the HL7v2 input files.	`pxsa.v2.input.reader.encoding=UTF-8`	*pxsa-file.properties* : `pxsa-v2-input-reader.encoding=UTF-8`	*context-common.properties* : `hl7.charset = ISO-8859-1`	Common value for all receiving endpoints.
File: File ordering direction	<not available>	*pxsa-file.properties* : `pxsa-v2-input.incrementing=true`	`…&sortBy=reverse:file:name`	Parameter embedded in the endpoint URL
File: A wildcard expression for files to be read	<not available>	*pxsa-file.properties* : `pxsa-v2-input.filenameWildcard=*`	*context-common.properties* : `…&include=.*txt`	Parameter embedded in the endpoint URL

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HTTP(S): Password for authenticating at the target HTTP server (MPI)	`pxsa.v3.output.http.credentials.password=<not shown>`	*pxsa-http.properties*: `pxsa-v3-output-http-credentials.password=hutzlibutzli`	*context-common.properties* : `pxs.http.password=`	
HTTP(S): URL of the HTTP endpoint that receives messages	`pxsa.v2.input.http.url=http://localhost:8477/pxsa/receiver`	*pxsa-http.properties*: `pxsa-v2-input-http.locationURI=http://0.0.0.0:8477/pxsa/receiver`	*context-pxsa.properties* : `jetty:http://0.0.0.0:8477/pxsa/receiver`	

Table 42: Transition of parameters from MSB 1.x to MSB 2.x - different default value

10.4.2.3 Configuration items with fixed values

In Camel this element is not configurable. The value is fixed.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Description	**Install Property** `Parameter= default value`	**Config Parameter** *File*:`Parameter= default value`	**Corresponding property** *File*:`Parameter= default value`	**Comment**
File: Sorting criteria for read-in sequencing	<not available>	pxsa-file. properties : pxsa-v2 -input.order =lastModified	context-pxsa. properties : input.url.file= file://./data/in/ pxsa? delete=true &delay=5000 &sortBy =file:name	Parameter embedded in the endpoint URL
File: Audit directory for the output of transformed messages (as HL7v3).	pxsa.v3.output. directory= file:$ {work.root.dir} /pxsa/work/ v3-output	pxsa-file. properties : pxsa-v3-output. directory=file:/usr/ local/servicemix/ pxsa/hl7/v3-output	<fixed value>	Audit files of outgoing messages are written into data/audit/ pxsa/ process>-< sequenceId >.internal- request.txt
File: Encoding of the HL7v3 output files	pxsa.v3.output. writer.encoding =UTF-8	pxsa-file. properties : pxsa-v3-output -writer. encoding =UTF-8	<fixed value>	Always UTF-8

MSB 1.6 ServiceMix			MSB 2.8 Camel	
File: Directory for HL7 v2 validation error messages	pxsa.v3.error. validation.directory =file:$ {work.root.dir} /pxsa/hl7/error	pxsa-file. properties : pxsa-v3-error-validation.directory =file:/usr/local/ servicemix/pxsa/ hl7/error	<fixed value>	Error Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.error.txt
File: Directory for HL7 v2 to v3 transforma-tion error messages	pxsa.v3.error .transformation. directory=file:$ {work.root.dir} /pxsa/hl7/error	pxsa-file. properties : pxsa-v3-error-transformation. directory =file:/usr/local /servicemix/ pxsa/hl7/error	<fixed value>	Error Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.error.txt
File: Directory for HL7 v3 delivery error messages	pxsa.v3.error. delivery.directory =file:$ {work.root.dir} /pxsa/hl7/error	pxsa-file. properties : pxsa-v3-error-delivery.directory =file:/usr/local/ servicemix/pxsa /hl7/error	<fixed value>	Error Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.error.txt
File: Encoding of validation error files	pxsa.v3.error. validation. writer. encoding =UTF-8	pxsa-file. properties : pxsa-v3-error-validation-writer. encoding=UTF-8	<fixed value>	
File: Encoding of transforma-tion error files	pxsa.v3.error. transformation. writer. encoding =UTF-8	pxsa-file. properties : pxsa-v3-error-transformation -writer. encoding =UTF-8	<fixed value>	

MSB 1.6 ServiceMix			MSB 2.8 Camel	
File: Encoding for delivery error files	pxsa.v3.error. delivery.writer. encoding =UTF-8	pxsa-file. properties : pxsa-v3-error -delivery-writer. encoding =UTF-8	<fixed value>	
File: Audit for inbound messages	<not available>	xbean-platform .xml : <aud:file-auditor id= "pxsaFileAuditor" … outputDirectory="${work.root.dir} /pxsa/data/audit" …	<fixed value>	Incoming Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.request.txt
File: Audit for response messages (for MLLP and HTTP)	<not available>	<not available>	<fixed value>	Response Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.response.txt
File: Audit for response messages from the MPI core application.	<not available>	<not available>	<fixed value>	MPI Response Audit files are written into data/audit/ pxsa/ process>-< sequenceId >.internal-re- sponse.txt

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Resequenc-ing Number of messages the resequencer can hold before blocking	<not available>	pxsa-eip. properties : pxsa-resequencer. capacity=10000	<fixed value>	Behavior is governed by the following entry in the route script: MsbModel Extension. groovy : builder .from ('direct: resequence') .resequence() … .capacity(100) …
Resequenc-ing: Minimum amount of time (milliseconds) the resequensor waits for 'late' successors of a given message.	<not available>	pxsa-eip. properties: pxsa-resequencer. timeout=5000	<fixed value>	Behavior is governed by the following entry in the route script: MsbModel Extension .groovy : builder .from ('direct: resequence') .resequence() … .time-out(4000L)

MSB 1.6 ServiceMix			MSB 2.8 Camel	
MLLP: HL7 encoding of incoming messages transported over MLLP (VB|XML).	pxsa.input.mllp. hl7.encoding=VB	pxsa-hl7bc. properties: pxsa-input -mllp.hl7Encoding =VB	<fixed value>	VB is always used.
JMS: Define whether to use an exponential backoff for redelivery	<not available>	pxsa-jms. properties: pxsa-jms- redelivery. useExponential- BackOff =true	<fixed value>	Always true. Set pxs .redelivery. backOff Multiplier to 1 to effectively disable exponential backoff

Table 43: Transition of parameters from MSB 1.x to MSB 2.x - non-configurable elements

10.4.2.4 Configuration items with no correspondence

No corresponding item exists in Camel. The function has been eliminated, or there is no correspondence with the old implementation.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Description	**Install Property** `Parameter= default value`	**Config Parameter** *File*:`Parameter= default value`	**Corresponding property** *File*:`Parameter= default value`	**Comment**
HL7: Defines whether mapping resources that have been specified but not found shall lead to an error (default is no i.e. missing resources are ignored)	<not available>	pxsa-hl7se. properties: pxsa-mappings. ignore MissingResources =true	<not available>	
HL7: Defines whether ADT preprocessing shall be disabled	pxsa.disable. v2.preprocessing = true	pxsa-hl7se. properties: adt-mapper. identity Transformation =true	<not available>	preprocessing is a separate integration route
HL7: Defines whether MDM preprocessing shall be disabled	pxsa.disable.v2. preprocessing= true	pxsa-hl7se. properties: mdm-mapper. identity Transformation =true	<not available>	preprocessing is a separate integration route

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HL7: Definition of the ADT message category (to be processed by ADT mappers)	<not available>	pxsa-hl7se. properties: adt-category. domain=ADT	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.
HL7: Definition of the ADT message category (to be processed by ADT mappers)	<not available>	pxsa-hl7se. properties: adt-category. event=*	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HL7: Definition of the ADT message category (to be processed by ADT mappers)	<not available>	pxsa-hl7se. properties: adt-category. version=2.1,2.2	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.
HL7: Definition of the MDM message category (to be processed by MDM mappers)	<not available>	pxsa-hl7se. properties: mdm-category. domain=MDM	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
HL7: Definition of the MDM message category (to be processed by MDM mappers)	<not available>	pxsa-hl7se. properties: mdm-category. event=*	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.
HL7: Definition of the MDM message category (to be processed by MDM mappers)	<not available>	pxsa-hl7se. properties: mdm-category. version=2.5	<not available>	In the standard route the combination of message type, trigger event and HL7 version is fixed. In the custom route, this is controlled using the route script.

Table 44: Transition of parameters from MSB 1.x to MSB 2.x - No corresponding item

10.4.2.5 New configuration items

These configuration items are new in Camel.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
Description	**Install Property** `Parameter=default value`	**Config Parameter** *File*:`Parameter=default value`	**Corresponding property** *File*:`Parameter=default value`	**Comment**
HTTP(S): Time until an HTTP connection will be closed by the client.	<not available>	<not available>	context-common.properties : pxs.http.socket.timeout=30000	
Tracing:	<not available>	<not available>		Tracing can be enabled using JMX
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.management.connector.port=1801	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.persistence.cleanup.interval=2000 .broker.url=vm://broker1	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.network.connector.uri=static:(tcp://localhost:60002)	The property was configured at the platform level under ServiceMix.

MSB 1.6 ServiceMix			MSB 2.8 Camel	
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.transport.connector.uri=tcp://localhost:60001 activemq.connection	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.connection.broker.url= vm://broker1	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.broker.name=broker1	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : activemq.data.dir = data/pxsa/activemq1	The property was configured at the platform level under ServiceMix.
ActiveMQ:	<not available>	<not available>	context-common.properties : flowmgr.data.dir = data/pxsa/flowmgr	The property was configured at the platform level under ServiceMix.

Table 45: Transition of parameters from MSB 1.x to MSB 2.x - new elements

11 Connection to PXS 3.1.2

This chapter describes how you operate the MPI with the Virtual Medical Record from PXS 3.1.2. The MPI 3.0 synchronizes the registered index patients including their demographical data and corresponding data sets from the different organizations for the communication with PXS 3.1.2.

1. In MPI 3.0 make sure that in the MBean `com.icw.epr/Context.Beans/eventRulesEnablement` the following three attributes are set to `true`
   - `applicationAcceptAcknowledgeGenerator`
   - `applicationErrorAcknowledgeGenerator`
   - `commitErrorAcknowledgeGenerator`
2. Set PXS 3.1.2 as a subscriber of IHE PIX Feed (ITI-8) messages, by adding an entry for its MLLP entry interface in `<MSB_HOME>/conf/msb/subsciber-pix.properties` and setting the `feedUrl` parameter.
3. In the JMX settings of PXS 3.1.2 check in the MBeans among pxs-patient-import that the URL is set to ITI-8.
4. If you changed the Patient Namespace of Master Patient Index Patients in MPI 3.0, copy the mappings `bdm-patientOid-serversidePatientOid` and `bdm-patientNamespaceId-patientOid` in the file `pixpdq-mappings.map` and change the MPI-OID accordingly. By default this is set to 2.16.840.1.113883.3.37.4.1.1.2.1.1 in MPI 3.0 and MSB 3.0.
5. Change the MPI-Patient-Namespace of PXS 3.1.2 accordingly. For details see the *PXS 3.1.2 System Administration Manual*.
   For the DRR change it with JMX in the MBean `regconfig/jmxRegistryConfigBean/FacilityDomains`
   For the VMR change the file `organizations.xml`.
6. If you want to encrypt the PIX Feed-connection over SSL/TLS, configure a mllps-connection and corresponding Keystores and Truststores.
   a. In `<MSB_HOME>/conf/msb/subsciber-pix.properties` set the `feedUrl` parameter `secure` to `true`.
   b. Configure Keystores and Truststores of MSB, see [page 32].
   c. Configure Keystores and Truststores of MPI 3.1.1. For details see the *PXS 3.1.2 System administration Manual*.

12 Other Relevant Aspects of Implementation

This section brings together a number of specific technical aspects of MSB implementation that are important from an administration perspective. You will find more details in the reference interface description.

12.1 Technical Aspects of pxsa

12.1.1 Troubleshooting in pxsa

Each message is subjected to a number of tests during processing, depending on the processing step that is running.

Directly following receipt of a message, a test is performed to determine whether its structure complies with the HL7 v2 standard.

Then, a determination is made on the basis of the combination of trigger event (field MSH-9-2) and HL7 version (field MSH-12) whether pxsa is appropriate to process the message.

If that is the case, additional test must be performed prior to translation into XML format (HL7 v3):

- Are all segments of the massage valid in the context of the corresponding trigger event?
- Are all required field present and filled out (see MPI documentation [2])?

The existence of invalid segments may prevent required fields from being recognized.

Once translation is successful, an XML schema-based validation routine is performed to determine whether the structure of the freshly generated message complies with the requirements of the HL7 v3 standard.

The final test occurs after the HL7 v3 message is delivered to the MPI core application. As soon as the message is accepted there *pxsa* receives a corresponding response.

Errors occurring during communication with the target application are similarly dealt with.

12.1.1.1 Unresolved errors:

The URL of a *pxsa* endpoint is syntactically incorrect.

A *pxsa* endpoint could not be initialized, because the specified TCP port is occupied by another application.

The JMS component could not be initialized, for example because a TCP port has been assigned twice, the path given for the persistent storage of messages was incorrect, or because there were problems when physically accessing the database.

The Flow Management component could not be initialized - possible causes are similar to those mentioned for the JMS component.

A runtime error occurred during translation of the message from HL7 v2 to HL7 v3 format.

Error when initializing the XML validator: The corresponding XML schema document could not be opened or is syntactically incorrect.

The MPI endpoint cannot be reached, cannot receive the message, or returns an error (HTTP error code 4xx or 5xx). Potential causes, aside from purely technical issues, include: incorrect MPI endpoint URL, username or password.

Audit entries and log files cannot be saved, because the path does not exist or is not writable.

12.2 Technical Aspects of pixpdq

12.2.1 Core Application Settings

Correct configuration of the MPI application components is indispensible if the pixpdq integration component is to function correctly. How to configure the MPIcore applicaton is described in the MPI administration manuals.

12.2.1.1 PIX feed settings

The following settings are required to enable the pixpdq component to accept IHE transactions ITI-8 (Patient Identity Feed).

PROPERTY_ CONFIGURATION	JMX MBean	Required value
message.incoming. application.accept. acknowledge	com.icw.epr > Context.Beans. prg-all.eventRuleEnablement > applicationAcceptAcknowledge Generator	true
message.incoming. application.error. acknowledge	com.icw.epr > Context.Beans. prg-all.eventRuleEnablement > applicationErrorAcknowledge Generator	true
message.incoming. accept.error. acknowledge	com.icw.epr > Context.Beans. prg-all.eventRuleEnablement > commitErrorAcknowlege Generator	true

Table 46: PIX feed settings

12.2.1.2 Settings for PIX update notification

The following settings are required to enable the pixpdq component to support IHE transaction ITI-10 (Patient Identifier Cross-Reference Update Notification).

PROPERTY_ CONFIGURATION	JMX MBean	Required value	Remark
esb.message outgoing. success. http.enabled	com.icw.epr > Context. Beans.prg-all.eventRule Enablement > esbOutgoing EventDispatcher	true	
docIndex.enabled	com.icw.epr > Context. Message.Receiver. Device > prg-all - > defaultPatientInformation ReceiverDevice > Enabled	false	Suppress transmission of irrelevant notifications.

PROPERTY_ CONFIGURATION	JMX MBean	Required value	Remark
piccDevice.enabled	com.icw.epr > Context. Message.Receiver.Device > prg-all > piccDevice > Enabled	true	
esb.docIndexserviceUrl	com.icw.epr > Context. MessageReceiver.Remote Proxy > prg-all > esb > esbMpiMessageRecipient	http://<msb-host>:8480/pix-update/receiver/	(Standard: http://localhost:8480/pix-update/receiver/)

Table 47: Settings for PIX update notification

12.2.1.3 Troubleshooting in pixpdq

For some errors, the IHE specification [6] demands from the server application a specific behavior. This requirement is satisfied by the pixpdq component, see [page 106]. The error message is returned with the confirmation message. Additionally, component-specific error conditions are recognized and reported in response messages, see [page 107]. Other errors are also processed, but these result in the return of unspecific error messages whose interpretation requires extensive diagnostics.

Error	Report contained in the confirmation message (example)
HL7 version number (MSH-12) invalid	`ERR|||207^Application internal error^HL70357^^Can't process message of version '10.3' - version not recognized|E||| Can't process message`
HL7 version number (MSH-12) does not comply with the specification	`ERR|^^^203&Unsupported version id&HL70357&&Unsupported version`
Invalid message type (MSH-9-1)	`ERR|^^^200&Unsupported message type&HL70357&&Unsupported message type`

Error	Report contained in the confirmation message (example)
Invalid trigger event (MSH-9-2)	`ERR|||201^Unsupported event code^HL70357^^Unsupported event code|E|||Unsupported event code`
For PIX queries: The requested patient ID contains an invalid assigning authority (QPD-3-4)	`ERR||QPD^1^3^1^4|204^Unknown key identifier^HL70357^^Unknown assigning authority in QPD-3: \S\1.2.3.4\S\ISO|E|||Unknown assigning authority in QPD-3: \S\1.2.3.4\S\ISO`
For PIX queries: Invalid entries exist under the requested assigning authorities (QPD-4).	`ERR||QPD^1^4^2~QPD^1^4^3|204^ Unknown key identifier^HL70357 |E||Unable to resolve requested domains in QPD-4: \S\\S\\S\NONEXISTING, \ S\\S\\S\\T\1.2.3.4\T\ISO`
For PIX queries: Requested patient number contains a correct assigning authority, but is unknown to the target system.	`ERR||QPD^1^3^1^1|204|E||Unknown patient ID`
For PDQ queries: Invalid entries exist under the requested assigning authorities (QPD-8)	`ERR||QPD^1^8^2~QPD^1^8^3|204^ Unknown key identifier^HL70357 |E||Unable to resolve requested domains in QPD-8: \S\\S\\S\NONEXISTING, \ S\\S\\S\\T\1.2.3.4\T\ISO`

Table 48: Errors according to the IHE specification

Pixpdq also has rules governing the handling of component-specific errors

Error	Report contained in the confirmation message
For PIX query, PDQ query: Required segment QPD missing	`ERR|||101^Required field missing^HL70357^^Required element QPD-1-1 empty|E|||…`
For PDQ queries: No value for parameter in QPD-3	`ERR||QPD^1^3^0^2|207^Application internal error^HL70357^^Missing parameter value in QPD-3(0)-2| E|||…`
For PDQ queries: A parameter code occurs in the request more than once	`ERR||QPD^1^3^1^1|207^Application internal error^HL70357^^Search parameter '@PID.5.1' or its synonym used more than once!| E|||…`
Processing in the eMPI core application failed. The message contains a syntax error. Check the log file of the MPI core application [2].	`ERR|^^^207&Application internal error&HL70357&&HTTP operation failed with statusCode: 400, status: HTTP/1.1 400 Bad Request…`
Processing in the MPI core application failed. Unspecific error. Check the log file of the eMPI core application [2].	`ERR|^^^207&Application internal error&HL70357&&HTTP operation failed with statusCode: 500, status: HTTP/1.1 500 Internal Server Error…`
Processing in the MPI core application failed.Check the log file of the MPI core application [2].	`ERR|^^^207&Application internal error&HL70357&&Application module returned empty response…`

Table 49: Component-specific handling of errors in pixpdq

12.2.1.4 Filtering of index patient patient IDs

The PIX/PDQ IHE profiles do not provide for any hierarchy of assigning authorities (identifier domains). In this model all assigning authorities are equal. In contrast, MPI accords a special position to index patients. Their standard assigning authority is

2.16.840.1.113883.3.37.4.1.1.2.1.1. For the purposes of *pixpdq*, patient IDs of index patients are filtered out of all messages and so are not expressed.

12.2.1.5 Special features of the PIX implementation

12.2.1.5.1 Inadequacy of the update notification transaction

According to the current state of the specification (IHE ITI Vol. 2), the PIX transaction ITi-10 has some limitations:

- Only patient IDs are communicated, demographic data are missing.
- There is not provision to delete patient IDs.

The pixpdq component support the ITI-10 transaction in accordance with the specification. These limitations must be kept in mind during productive operations.

12.2.1.6 Special features of the PDQ implementation

12.2.1.6.1 Index patients as a source of demographic data

Index patients are the only source of demographic data. It is not possible to call demographic data for individual in-house patients (data sets) over the PDQ interface. In this sense, the patient IDs of in-house patients (data sets) do no count as demographic data and so are accessible over a PDQ query in accordance with the IHE specification.

12.2.1.6.2 Representation of accounting numbers in pixpdq

According to the specification of the PDQ profile, the use of account numbers as query criteria is supported. MPIrepresents account numbers by means of a generic mechanism, so-called global profiles. For each system that uses account numbers (field PID-18) over pixpdq a system with the profile global is set up and configured in the MPI core application. Interaction between the MPI core application and the mapping tables in pixpdq is analogous to the procedure tor patient IDs.

12.2.1.6.3 Supported parameter codes for PDQ queries

For PDQ queries, parameters are given in the QPD-3 field, represented in an IHE-specific syntax: `<code11>^<value1>~< code2>^<value2>~...` Using a parameter more than once is not allowed and will cause the query to be rejected. The parameter codes specify elements in the HL7 message, for example @PID.7 designates the field PID-7 and means patient date of birth. Although the PDQ specification as code/value only allows primitive data types, the *pixpdq* component also supports some additional

codes, see the following table. When not otherwise noted in the table, query parameters are simply compared when a search is performed (exact match). For some text fields, the wildcard * is allowed in the sense of a string of any kind of characters. Other HL7 fields that use the same semantics (i.e. synonyms) are added using the corresponding mapping `map-queryParam-synonym`.

Parameter codes	Meaning
@PID.3.1 @PID.3.4.1 @PID.3.4.2 @PID.3.4.3	Patient number, e.g.: @PID.3.1^12345678~@PID.3.4.2^NMS1
@PID.5.1 @PID.5.1.1	Patient's family name. Wildcard „*“ stands for any number of any kind of character. Example (QPD-3): @PID.5.1^Petersen
@PID.5.2	Patient's first name. Wildcard „*“ stands for any number of any kind of character. Example (QPD-3): @PID.5.1^Henry Example (QPD-3): @PID.5.1^H*
@PID.5.3	Patient's middle name. Wildcard „*“ stands for any number of any kind of character. Example (QPD-3): @PID.5.1^Profittlich~@PID.5.2^Markus~@PID.5.3^Maria
@PID.7, @PID.7.1	Patient's date of birth. Example (QPD-3): @PID.7.1^20041231
@PID.8	Patient's gender. Example (QPD-3): @PID.8^M
@PID.11.1 @PID.11.1.1	Patient's street address. Example (QPD-3): @PID.11.1.1^Hauptstrasse 22
@PID.11.3	Patient's city or town. Wildcard „*“ stands for any number of any kind of character. Example (QPD-3): @PID.11.3^Berlin Example (QPD-3): @PID.11.3^*e*
@PID.11.5	Patient's postal code. Example (QPD-3): @PID.11.5^69124

Parameter codes	Meaning
@PID.11.6	Patient's country. Example (QPD-3): @PID.11.6^DE
@PID.18.1 @PID.18.4.1 @PID.18.4.2 @PID.18.4.3	Accounting number: When filling the subcomponents the usual rules apply in accordance with the technical framework [6] . @PID.3.1^AN87654321~@PID.3.4.2^NMS2

Table 50: Component-specific handling of errors in pixpdq

Table 46:

12.2.1.6.4 Multi-part query results (query continuation)

Because PDQ queries can contain wildcards in their parameter specifications, the results of a query may be very extensive. For this reason, the IHE specification foresees the option to split query responses into multiple messages. This mechanism is called *query continuation*.

For query continuation in *pixpdq*, you must pay attention to the following points:

- Each partial result is based on the momentary status of the MPI database at the time the query was made. Consistency cannot be ensured because of the time lapse between partial results. Changes to the database during the time between two partial results may shift the datasets in the set of results. This may cause a dataset to occur in more than one partial result, or cause it to be missing in all of them. Brief intervals between individual partial queries minimize this risk.
- The final partial result may be empty. When, for example, the size of the partial result is limited to 10 datasets and the result contains exactly 10 datasets, the first partial response will contain besides the 10 datasets a reference (query continuation pointer) to a second partial result. But this second partial result will contain 0 datasets.

13 Flow Manager

Flow Manager is a server-based service for monitoring and managing the message transfer on the integration platform. A flow represents the path of a message within the Integration Platform from a starting point to one or more end points. In Flow Manager, message flows are represented as flow objects.

A flow is comprised of one or more flow parts, which all have their own end point assigned. A flow is considered complete as soon as all flow parts have been worked.

Flow Parts contains information about the history of all of the messages that belong to a specific flow, exactly one flow being initiated for each incoming message. Each inbound message initiates exactly one flow. Because the inbound message may have been split, several messages may correspond to one flow. These are represented by multiple flow parts. When there is only one message per flow there will only be one flow part.

13.1 Flow Manager basics

13.1.1 Properties of a flow

The following properties of a flow are relevant for the Integration Platform Manager:

- Flow status: CLEAN or ERROR - applies to flow parts. For more information about the states completed flows may assume, see section States of a flow [page 115]
- **Application**: Name of the application that processes the flow
- **Create time**: Time when a message was delivered to the integration platform.
- **Replay time**: Time the message was last replayed.
- **Replay count**: Number of times the message was replayed
- **Replayable**: Replayability of message. (`true` or `false`)
- **ACK count (expected)**: Expected number of acknowledgements
- **ACK count (actual)**: Actual number of acknowledgements (*CLEAN Acknowledged*) see section States of a flow [page 115]
- **NAK count**: Number of invalid acknowledgements. see section States of a flow [page 115]

The properties of a Flow Part

- **Flow part status**: CLEAN or ERROR
- **Flow duration**: Flow transmission time in milliseconds. This includes the time for a message to be transmitted successfully to the target system.

- **Contribution time**: Time stamp for the acknowledgement (ACK) or negative acknowledgement (NAK) of the Flow Part.
- **Filter time**: Time stamp for filtering the Flow Part during a replay.

The path for each Flow Part is shown. When a message has not been split the path is always 0. If a message has been split in two parts their paths are 0.0 and 0.1. If the message with path 0.1 is again split into two parts then the result is three messages with the paths 0.0, 0.1.0 and 0.1.1.

13.1.2 States of a flow

The state of a flow depends on the number of defined end points the message was delivered to and with which result it was delivered.

A Flow Part can in principle have the state CLEAN or ERROR. The state of a completed flow is the sum of the states of its flow parts. If a single flow part is for example faulty, this state applies also to the completed flow, see Figure Example of a flow.

The Flow Manager distinguishes between three states for flows:

- **CLEAN (Acknowledged)**: The message was delivered to all end points.

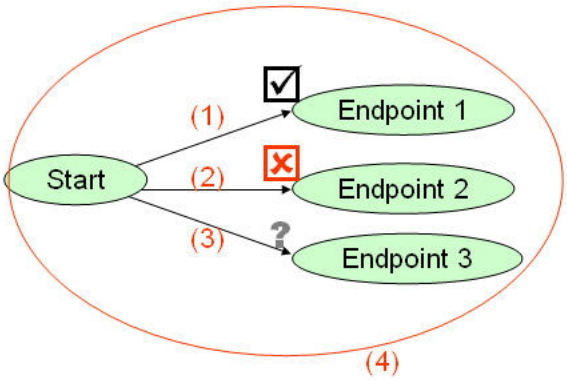
  A flow is Acknowledged if all flow parts are Acknowledged.

  If a flow part is registered as Acknowledged, the ACK property of the flow increases by 1.
- **ERROR**: An error occurred with at least one delivery.

  At least one flow part is invalid.

  If one flow part is registered as invalid, the NAK property of the flow increases by 1.
- **CLEAN (Unacknowledged)**: The message was not yet delivered to all end points. There are no known errors in the deliveries conducted so far.

  A flow has the Unacknowledged status, if at least one flow part is neither Acknowledged nor invalid.



	Status
(1) Flow Part 1	CLEAN (Acknowledged)
(2) Flow Part 2	ERROR
(3) Flow Part 3	CLEAN (Unacknowledged)
(4) Complete Flow	**ERROR**

Figure 14: Example of a flow

13.1.3 Interpreting of the properties of a flow

The flow states (explained in the section States of a flow [page 115]) – ERROR, CLEAN (Acknowledged) und CLEAN (Unacknowledged) – are derived from the NAK and ACK values.

The constant `EXP`, which is defined once when the flow route is defined and can no longer be edited in the Flow Manager, applies for the property ACK count (expected).

The relationships between ACK, NAK and EXP define the flow states and the behavior of the Flow Manager:

In principle:

- A flow is only then CLEAN, when its NAK value equals 0:

  CLEAN <=> (ACK = 0)

For the states CLEAN (Unacknowledged) or CLEAN (Acknowledged) the relations are more complex:

EXP > 0

For the most commonly used case EXP > 0:

- CLEAN (Unacknowledged) <=> (NAK = 0) && (ACK < EXP)
- CLEAN (Acknowledged) <=> (NAK=0) && (ACK = EXP)
- When cleanUp is active and a flow reaches the expected number of expected parts (ACK=EXP) all message data will be removed from the flow.

For all completed flows ACK and NAK are limited by the system:

When EXP > 0, then always (invariant):

0 <= ACK <= EXP; 0 <= NAK <= EXP; ACK + NAK = EXP

EXP = - 1

In the case EXP = -1 no cleanup will be run for the flows and the message will be deleted. Therefore, the default value for EXP is:

- CLEAN (Unacknowledged) <=> (ACK = 0) && (NAK = 0)
- CLEAN (Acknowledged) <=> (NAK = 0) && (ACK > 0)

  For the flow message no `cleanup` will be run (Cleanup <=> false)

For all completed flows ACK and NAK are limited by the system:

When EXP = -1 then always (invariant):

0 <= ACK; 0 <= NAK; ACK + NAK > EXP

EXP = 0

The case EXP=0 is not defined.

14 Integration Platform Manager

14.1 Introduction

The current version of the documentation for the Integration Platform Manager (IPM) is available under http://repo.openehealth.org/confluence/display/ipf/IPF+reference+-+single#IPFreference.

In Platform Manager, use `<hostname>:<portnumber>` for the connection with MSB.

The port number is the same as for ActiveMQ, see [page 27]. If the monitoring computer is identical to the MSB server and the standard setting for the port number was retained the connection is `localhost:1801`.

14.1.1 Overview

The **Integration Platform Manager** is an Eclipse-based application for integration platform management. Processes can be monitored and parameters be modified on the Integration Platform using a dedicated graphic user interface.

14.1.2 System Requirements

- A Java Virtual Machine (JVM): JDK or JRE, Version 1.6 or later is installed.
- The respective installation package, depending on the operating system on which the Integration Platform is to be installed.
  - Windows: *ipm–win32.zip*
  - Unix: *ipm–linux.zip*
- The system has a network interface (LAN or WLAN).
- Direct access to Java Virtual machines (JVM) you desire to administer. The JVM or their JMX ports must be accessible from the local network.
- Host and port name of the desired JVM need be known in advance.

Monitoring and managing the application on the JVM is possible by using the Java Management Extension (JMX) tool. The Integration Platform Manager connects to the server (MBean server) on the Java Virtual Machine that hosts the Integration Platform. JMX via Remote Method Invocation (RMI) is used as the communications protocol.

14.2 Installation

14.2.1 Fast Installation

Installation and configuration of Integration Platform Manager proceeds through the following steps:

1. Check whether JVM 1.6 is set up, see section Requirements [page 119].
2. Install the Integration Platform Manager, see section Installing the application [page 120].
3. Set environment variables, see section Environment variables [page 120].
4. Start application, see section Start application [page 120].
5. Setup connections, see section Setting up a connection [page 125].

14.2.2 Requirements

The following requirements apply when using the Integration Platform Manager.

Prerequisites	Test
A Java Virtual Machine, Version 1.6 or later is installed. The directory of this Java installation will be referred to as <JAVA_HOME> in the following.	The content of the environment variables <JAVA_HOME> can be displayed by entering the command: ● echo %JAVA_HOME% (on Windows) or ● echo %JAVA_HOME% (on Linux) The output contains a valid path.
The corresponding Java installation is included in the system path.	The content of the system path can be viewed in the command line with the command: echo %PATH% (for Windows) or echo $PATH (for Linux) The output text contains the path <JAVA_HOME>/bin.
The corresponding Java installation will be used by the system as standard.	The command java -version on the command line returns which version of Java is installed.

Table 51: IPM installation requirements

14.2.3 Installing the Integration Platform Manager

14.2.3.1 Installing the application

NOTE **Install Location**

Because of security restrictions on most servers it is not possible to operate Integration Platform Manager on them. For that reason, IPM is usually installed on one of the adminstrator's local computers.

Installing on Windows

1. Extract the archive for windows of the Integration Platform Manager to a folder.
2. Start the executable file <IPM_HOME>\ipm.exe. <IPM_HOME> is a placeable for the directory to which you extracted the Integration Platform Manager to.

Installing on Unix

1. Extract the archive for Unix of the Integration Platform Manager to a folder.
2. Start the executable file <IPM_HOME>/ipm. <IPM_HOME> is a placeable for the directory to which you extracted the Integration Platform Manager to.

14.2.3.2 Environment variables

Set the environment variable:

- **Windows**: %JAVA_HOME% for the directory <JAVA_HOME>
- **Unix**: $JAVA_HOME for the directory <JAVA_HOME>

14.2.3.3 Start application

1. Start the Integration Platform Manager:
   - **Windows**: Double-click the application file `<IPM_HOME>/ipm.exe`
   - **Unix**: Enter the command `<IPM_HOME>/ipm` in the command shell.

⇨ The application starts.

14.3 User Interface – The Basics

14.3.1 Definition of User Interface Elements

The following types of objects and control elements are relevant:

- **Connection**: This object describes the target application that is to be operated or the connection parameters to access the target application. Functions of the Integration Platform Manager can only be used in the context of a connection.
- **View**: Component of an Eclipse-based user interface for visualizing objects.
- **Editor**: Component of an Eclipse-based user interface where objects may be edited.
- **Perspective**: Logical collection of views and editors in Eclipse-based user interfaces whose task it is to make logical access to the user interface possible.

NOTE

In this document only application-specific functions of the user interface of the Integration Platform Manager are described. Additional options for adjusting Eclipse-based interfaces to personal preferences, such as changing perspectives or specifically placing views (docking), is described in detail in http://download.oracle.com/docs/cd/E17476_01/javase/1.5.0/docs/guide/management/overview.html.

14.3.2 Launching the Integration Platform Manager

1. Start the Integration Platform Manager by:
   - Windows: Double-clicking the application file <IPM_HOME>/ipm.exe
   - Unix: Enter the command <IPM_HOME>/ipm in the command shell.

⇨ The application starts.

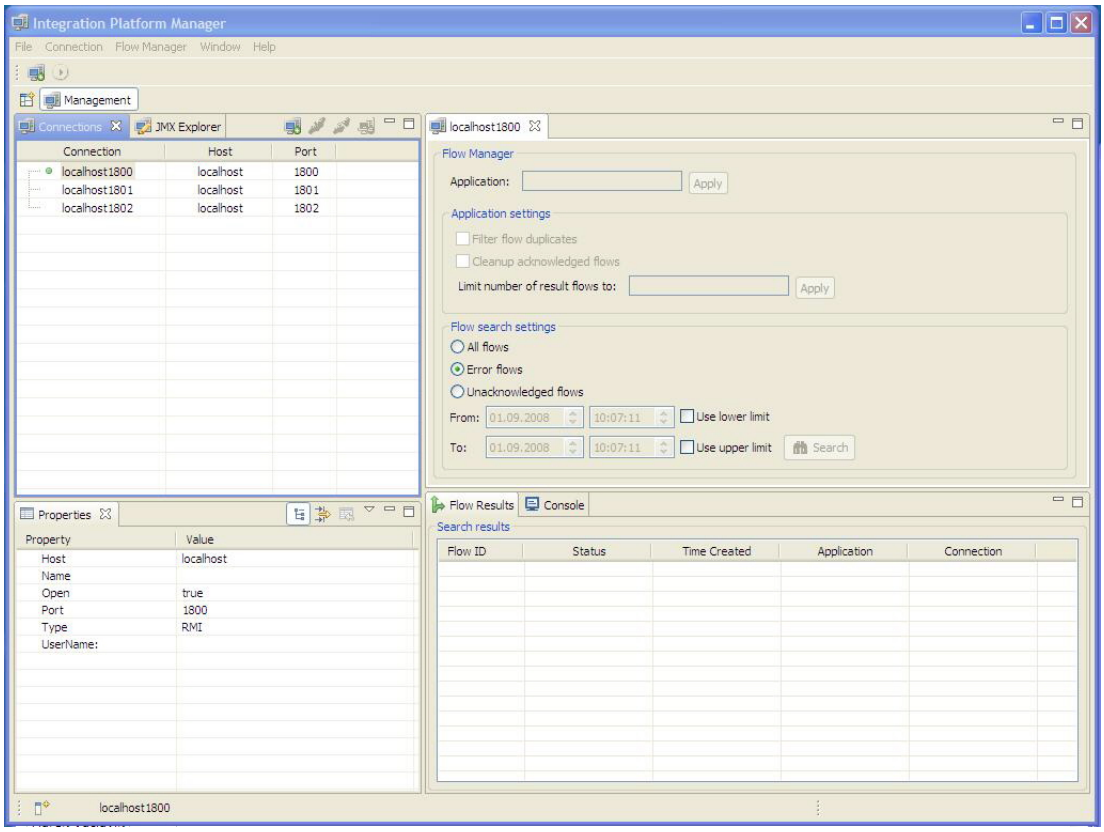


Figure 15: User interface of the Integration Platform Manager

14.4 Integration Platform Manager Perspectives

Integration Platform Manager uses two perspectives, Management and JMX.

NOTE

The most recently used perspective and the arrangement of the views will be saved by the Integration Platform Manager and loaded automatically when you start the application the next time.

14.4.1 Open Management perspective

The *Management* perspective provides a representation of all Integration Platform Manager elements. This perspective also enables the use of Flow Manager.

1. Click  (Open Perspective)

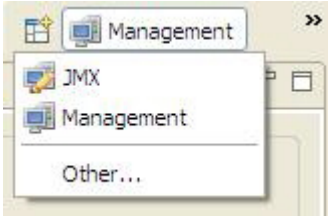


Figure 16: Selecting a perspective

2. From the list, select **Management**.

   ⇨ The perspective and all its views and editors opens.

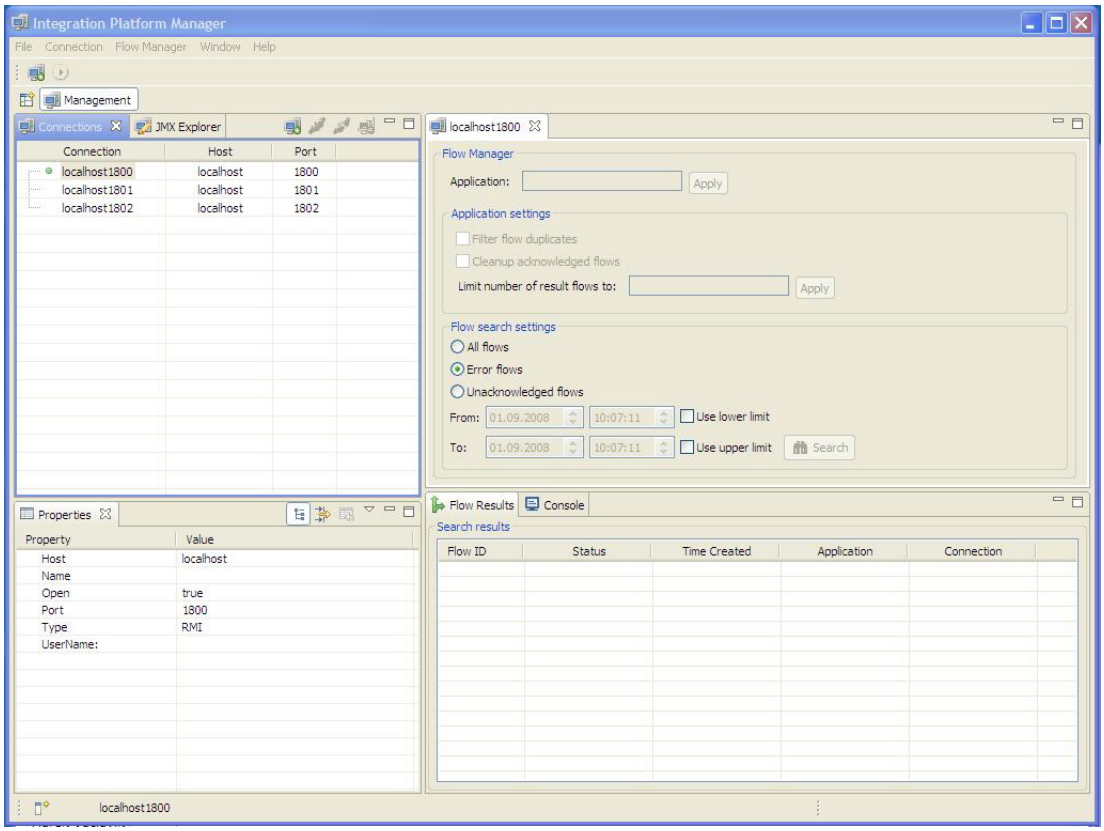


Figure 17: Management perspective

14.4.2 Open JMX perspective

Use the JMX perspective to administer JMX over the Integration Platform.

1. Click  (Open perspective)
2. From the list, select JMX.

   ⇨ The perspective opens.

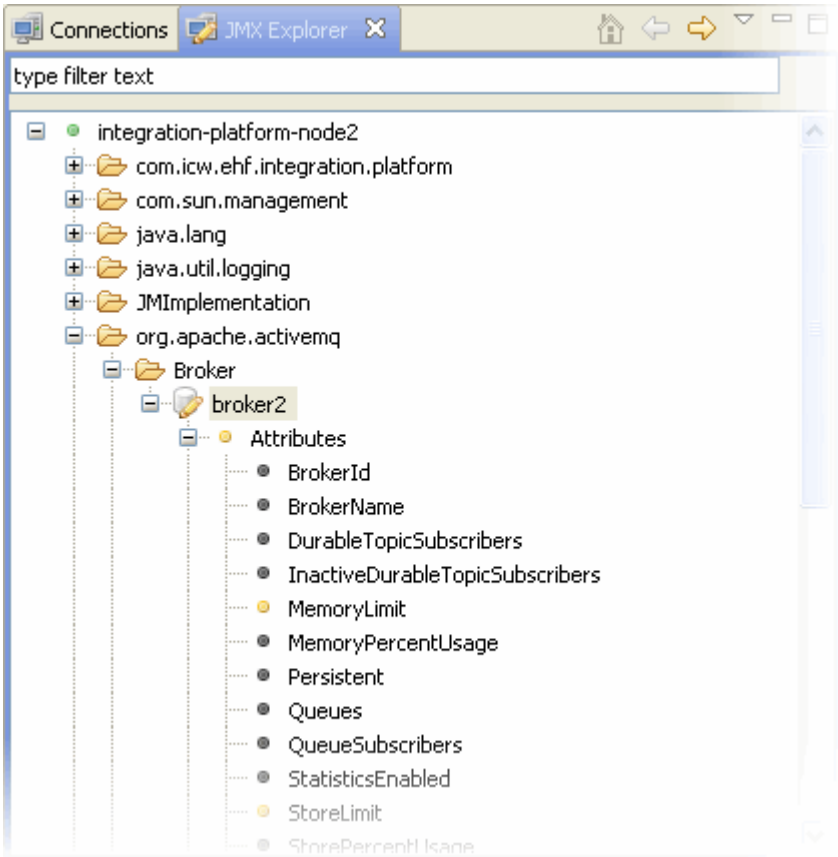


Figure 18: JMX perspective

14.4.3 Customizing and resetting perspectives

You can make additional adjustments to the perspectives, or move, hide and unhide window areas. The Integration Platform Manager saves the most recently used perspective and the arrangement of the views. For more information on Eclipse-based user interfaces, see Eclipse User Interface Guidelines (Version 2.1) with 3.x updates: http://wiki.eclipse.org/index.php/User_Interface_Guidelines.

If you want to restore the default settings for the perspective:

1. Right-click the perspective you want.
2. In the context menu, select Reset.
   ⇨ The perspective is reset.

14.5 Connections

Each Connection in the user interface represents a link to a JVM that is running on an Integration Platform. Generally, multiple applications or scenarios may be run on the Integration Platform, each of them associated with a connection of its own.

NOTE

Connections are not created as editable objects and cannot be modified retroactively.

There are two main states that Connections can have:

- **Open**: In an open connection, components of the Integration Platform Manager can access resources of the connection (MBeans, Flows, and so on).
- **Closed**: In a closed connection, the resources are not available to the Integration Platform Manager.

NOTE

The state of a connection can be changed via the user interface, see section Close connection [page 129].

14.5.1 Update connection parameters

1. In the Connections perspective, delete the obsolete Connection, see section Delete connection [page 129].
2. Create a new connection with the new connection parameters, see section Setting up a connection [page 125].

14.5.2 Setting up a connection

You can create a Connection in the *Connections* view.

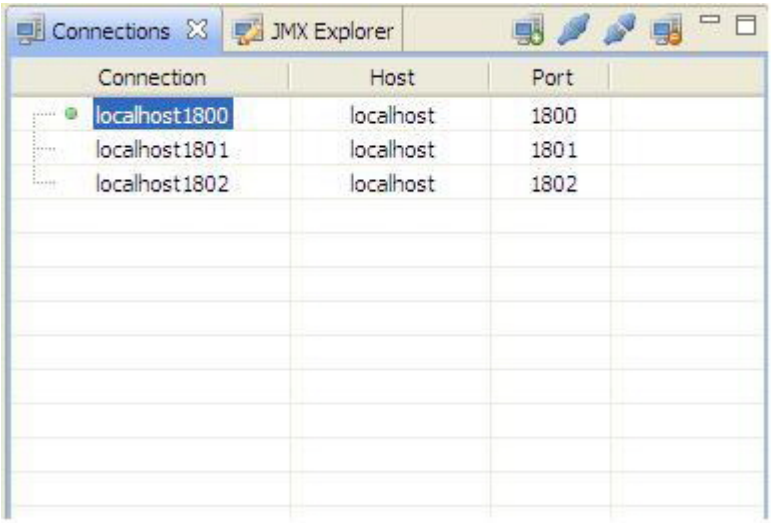


Figure 19: Connections view

1. In the menu bar, click Connection > New > **Connection**.

   ⇨ The *New Connection* dialog opens.

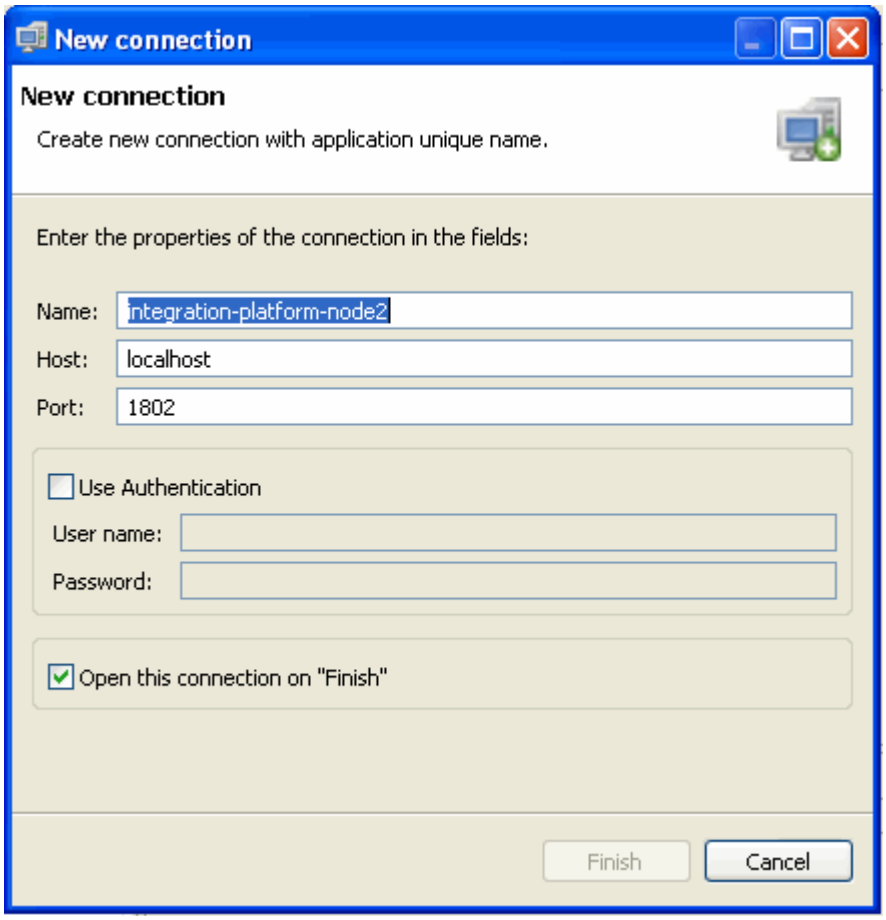


Figure 20: New connection dialog

2. In the dialog, define the following properties:

Property	Description	Dependencies to MSB core application
`Last name`	Enter a unique name for the connection. The name must be unique or else an error message will be returned.	
`Host`	The address of the computer that is running the Integration Platform Manager JVM.	Server, on which MSB is running.
`Port`	The JMX port for this Connection.	Parameter `activemq.management.connector.port`

Property	Description	Dependencies to MSB core application
`User name`	Enter a username if a user will be authenticating the connection.	
`Password`	Enter a password if a user will be authenticating the connection. The password will remain permanently linked to this connection. It will not be asked for again when the connection is re-established later. The password is stored locally on the computer as a Base64-encoded character string.	

Table 52: Properties of a connection

3. Activate the control box **Open this connection on "Finish"**, if you want the connection to open immediately once it has been set up.
4. Save your changes by clicking **Finish**.
   ⇨ The connection is now set up.

NOTE

Before you can use a connection as an entity for the Flow Manager editor or JMX Explorer you must first open it.
After closing and opening the application again the most recently connections to be set up remain available. In this case close the connection manually, see section Delete connection [page 129].

14.5.3 Properties of a connection

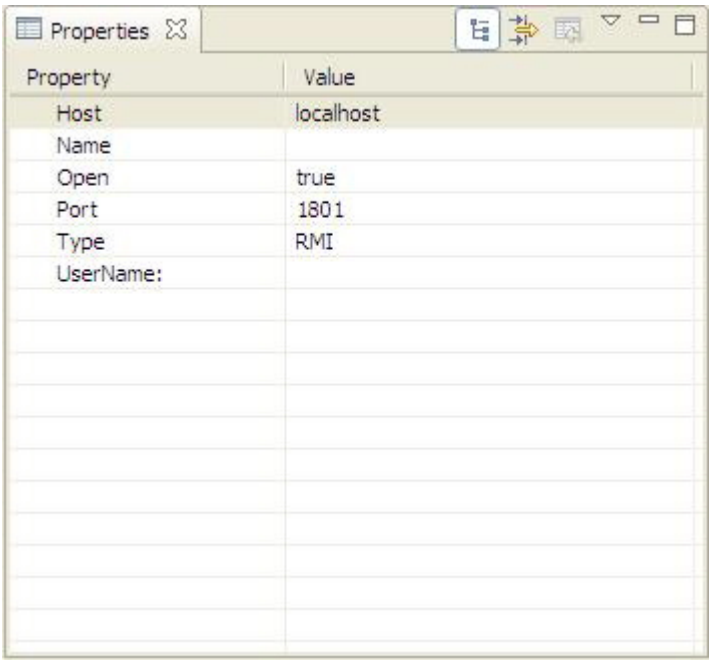


Figure 21: Properties connection view

You can view the properties of a connection in the *Properties* window area.

- **Host**: Address of the computer that is running the Integration Platform Manager JVM.
- **Name**: Name of the connection. The name in the Integration Platform Manager is unique.
- **Open**: Connection status:
  - **Open**: value is set to `true`.
  - **Closed**: value is set to `false`.
- **Port**: JMX port for this Connection. For more information about specific attributes can be found in section Monitor MSB [page 63].
- **Type**: Type is always *RMI*, because only this connection protocol is supported.
- **UserName**: User name for authentication. (optional)

14.5.4 Open connection

The services of the Integration Platform Manager (Flow Manager and JMX Explorer) that access the server via the connection are only available if the connection is open.

1. Go to the *Connections* view.
2. Mark a connection.
3. In the context menu, select **Open**.

⇨ The connection is now open and can be used as an entity for the Flow Manager editor or JMX Explorer, see section Flow Manager Editor [page 130].

14.5.5 Close connection

The editors of the Integration Platform Manager (Flow Manager and JMX Explorer) that access the server via the connection are no longer available if the connection is closed.

1. Go to the *Connections* view.
2. Select the connection in the *Connections* view.
3. In the context menu, select **Close**.

14.5.6 Delete connection

After closing and opening the application again, the created Connections will become available.

1. Go to the *Connections* view.
2. Select the connection in the *Connections* view.
3. In the context menu, select **Delete.**

15 Flow Manager in IPM

Flow Manager is a server-based service for monitoring and managing the message transfer on the Integration Platform.

For more information see Flow Manager [page 114].

15.1 Flow Manager Editor

The Flow Manager editor is used to access Flow Manager from the Integration Platform Manager screen. You use Integration Platform Manager to perform administrative tasks, search for message flows, or to replay flows. Do this by calling up the Flow Manager editor. A Flow Manager editor is always associated with exactly one connection.

15.1.1 Open Flow Manager Editor

1. Set up a connection, seethe section Setting up a connection [page 125].
2. Open the connection.
3. From the context menu, select **Open Flow Manager**.
   ⇨ The Flow Manager Editor will open.

15.1.2 Flow Manager component administration

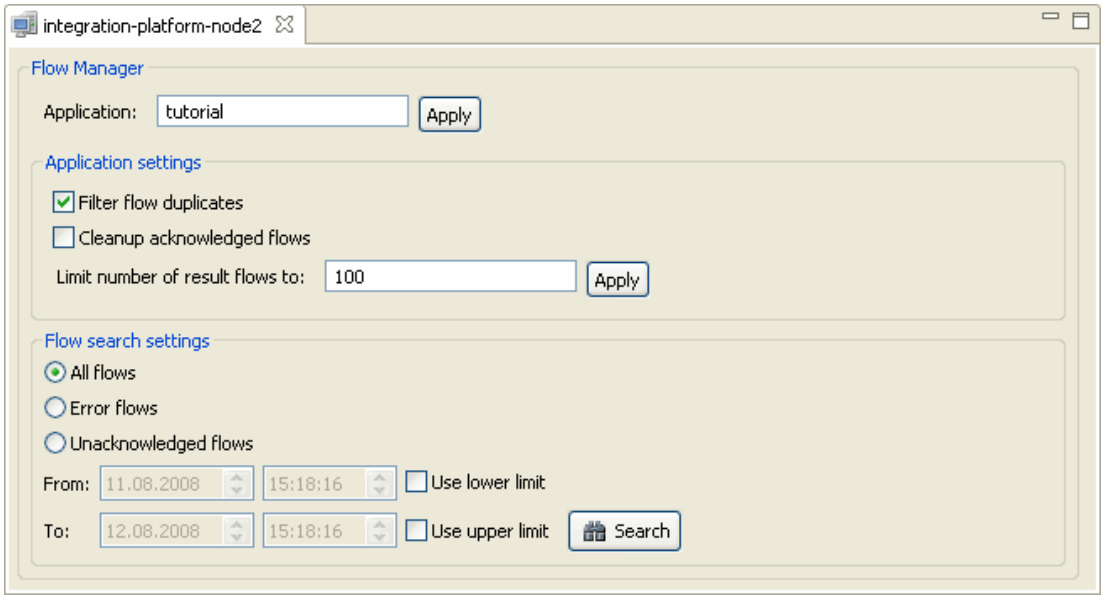


Figure 22: Flow Manager Editor

NOTE

For closed connections, **Apply** and **Search** will be grayed out.

In the Flow Manager Editor, you can set the following overall preferences.

- **Application**: The central setting in the Flow Manager Editor is the application.

  Enter the name of the application you want to monitor here. The Flow Manager Editor will only show those flows that pertain this specific application.

  Click **Apply** to save your changes.
- **Filter Flow duplicates**: If this function is active, prevents messages from being sent to a specific endpoint for a second time during a replay, see section Replay flows [page 134].

You have the following options:

- Activate this check box, if successful message transfers should not be repeated in case of a replay.
- Deactivate the check box if each message should always be sent to each end point in case of a replay.

Click **Apply** to save your changes.

NOTE

The setting defined in the option only applies to the **Application** in the dialog.

- **Cleanup Acknowledged Flows**: Active cleanup means that for flow objects that were processed successfully the stored information will be partially deleted. This way, the stored incoming message will be deleted For flows that have reached the overall status *acknowledged*, the saved inbound message will be deleted. This flow can no longer be replayed.

NOTE

Cleanup is used to reduce the amount of storage memory needed.

The general information about a flow remains stored minus the message content and can be viewed at a later time in the flow history.

You have the following options:

- Activate this option if you need the replay function often or for test purposes during the pilot phase.
- Deactivate this option if lowering storage space is a priority for you.

Click **Apply** to save your changes.

NOTE

The setting defined in the option only applies to the Application in the dialog.
If the cleanup function is not activated, much storage space can be in used in a short amount of time because of data that was not deleted.

- **Limit number of result flows to**: An entry in this field limits the size of the result delivered by a Flow Manager search. The default maximum number of returned Flows is 100. When a search finds more than 100 Flows in the database only the 100 most recent ones will be shown.

Click **Apply** to save your changes.

15.1.3 Search flows

You can define your own filters for flow searches in the *Flow search settings* window of the Flow Manager Editor. This way, only flows will be displayed that meet the search criteria set there.

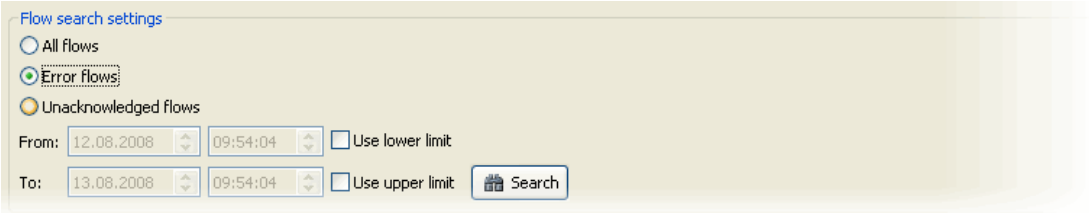


Figure 23: Filter for flow search criteria

1. As primary search criteria, define which flow categories you want to search for:
   - All Flows
   - Error Flows
   - Unacknowledged Flows
     These states are explained in detail in section States of a flow [page 115].
2. You can limit the search time. The time refers to the point in time at which the flow was received by the **Application**.
3. Do this by selecting the check box **Lower Time Limit** to specify the lower time limit and the **Upper Time Limit** check box to enter the upper time limit.

4. Select a date from the list fields that are now active.
5. Click **Search** or, alternatively, from the menu bar click **Flow Manager | Search Flows**

⇨ The results of your search will be displayed in the Flow Results view, see section Flow results view [page 133].

15.1.3.1 Flow results view

After the search is complete, the results are displayed according to the criteria in the *Flow Results* view.

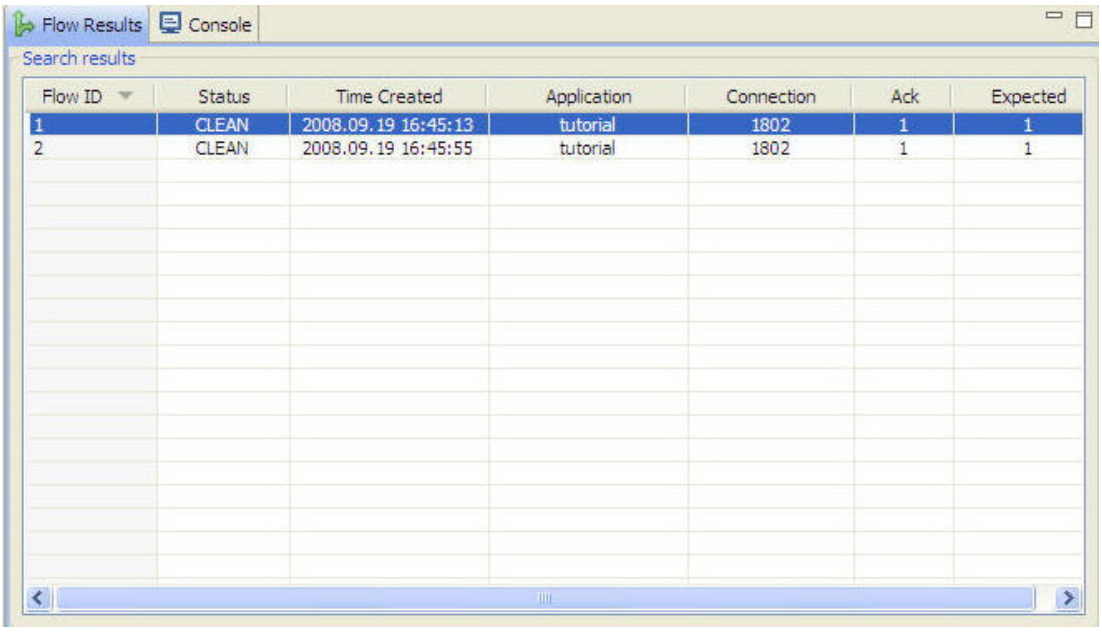


Figure 24: Flow Results view

You can sort this view according to the following criteria by clicking the respective cell in the headline of the table.

- **Flow ID**: Unique identifier, with which a flow can be identified. IDs are issued by the system based on creation time.
- **Status**: State of the flow. Possible states are described in detail in section States of a flow [page 115].
- **Time Created**: Time at which a message was received by the application.
- **Application**: Value of the Application field that you issue in the Flow Manager Editor, see section Flow Manager [page 114].

- **Connection**: Value of the Connection field that you issue when creating a new connection, see section Setting up a connection [page 125].
- **ACK**: Number of times a message was registered as *Acknowledged*. With each new registration the ACK property of the flow will increase by 1.
- **Expected**: Expected number of registered messages as Acknowledged.

NOTE

The *Flow Results* view has a direct dependency on the Flow Management Editor. If you click on the *Flow Results* view, the respective Flow Management Editor will be activated, the connections of which can be mapped to flows. To run a search, the parameters are defined in the respective Flow Manager Editor.

15.1.3.2 Replay flows

On the way to an end point, sections (flow parts) are defined. Sometimes flows do not reach their final end point, for example when an end point of a flow part is not available. Initiate the flow again when all endpoints are again available.

You can replay flows context-sensitively via the Integration Platform Manager.

1. Select one or more flows from the Flow Results view, see section Flow Results view [page 133].
2. In the context menu, select **Replay Selected Flows**.

15.1.3.3 Properties view (Flows)

In the Properties view, you can see the properties of a flow. The content is displayed depending on the selected objects.

In the *Flow Results* view, select a flow.

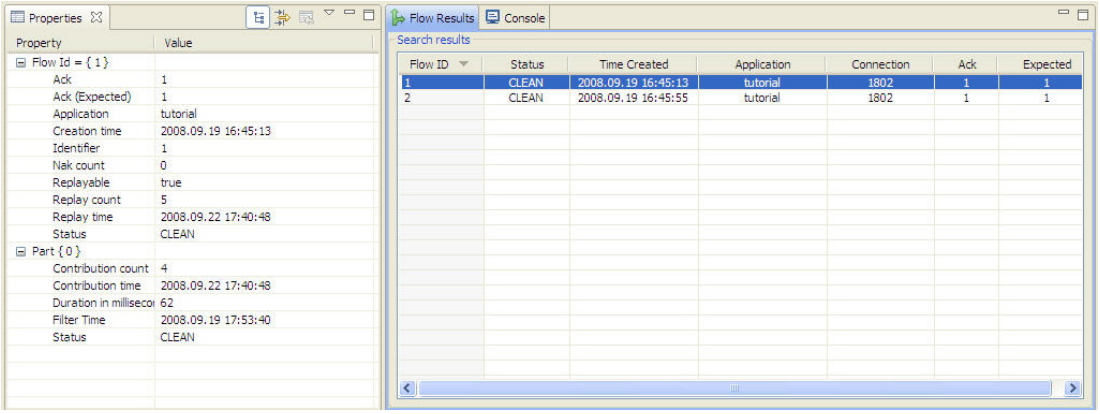


Figure 25: Properties view (Flows)

The properties of flow parts are displayed in the *Properties* view under the **Part** element. Individual parts can be expanded and collapsed via the tree structure control elements.

For more information see section States of a flow [page 115].

15.2 JMX in IPM

Java applications can be managed with JMX technology using MBeans. With JMX, MBeans can be edited at runtime. You will usually use JConsole to do this. The Integration Platform Manager offers an alternative way to access the MBeans mechanism.

15.2.1 Viewing connection(s) in the JMX Explorer view

The JMX resources of a connection are displayed in the JMX Explorer view.

1. Create a connection as described in section Setting up a connection [page 125].
2. Open the connection.
3. Select the connection in the *Connections* view.
4. From the context menu, select **Show in > JMX Explorer**.

⇨ The JMX Explorer view will open.

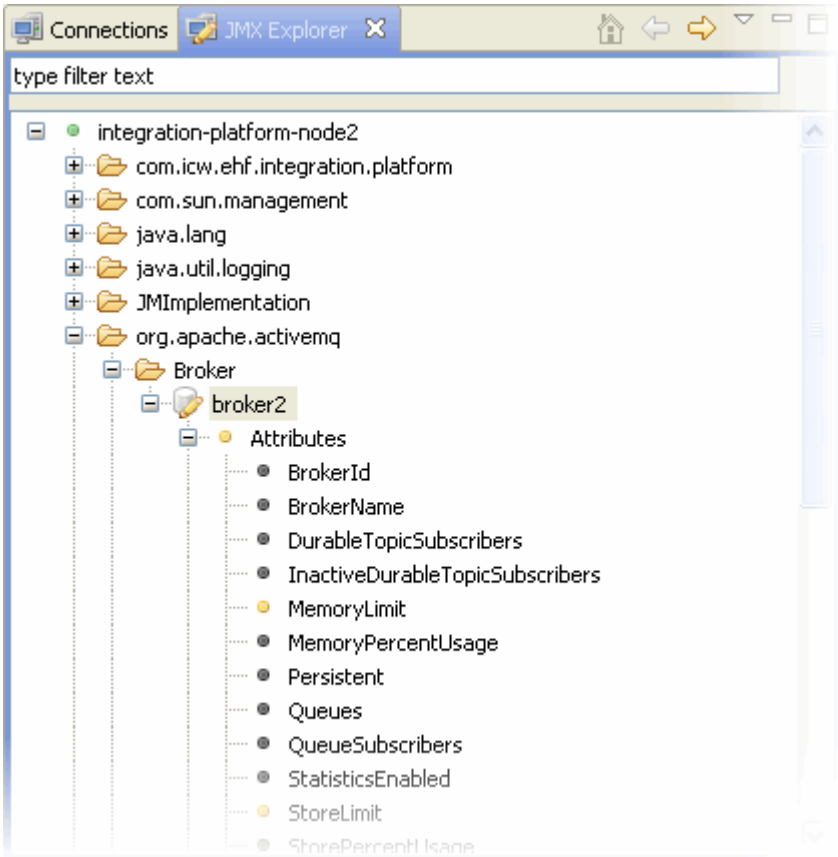


Figure 26: JMX Explorer view

- If the connection is closed, only the *JMX Explorer* view will display only one node having the same name as the connection. Child elements, if any, will not be displayed.
- If a connection is *open* and selected in the *Connections* view, a tree structure with MBean nodes of this connection will be displayed in the JMX Explorer. In this case, the connection is represented by a green icon. If the connection is closed again, the tree structure will no longer be available.
- If a connection was deleted, the node with the name of the connection, along with any child elements will be deleted from the *JMX Explorer* view.
- When you add a new connection it will not automatically appear in the *JMX Explorer* view. You initiate this connection in JMX Explorer over the *Connections* view.

15.2.2 JMX Explorer view

15.2.2.1 MBeans

The tree structure in the JMX Explorer depicts the MBean resources on the server. Fundamentally, these MBeans comprise operations and their associated attributes. The tree structure of the control elements enables access to these elements.

Attributes and operations, when present, are also represented by different colored icons. The colors indicate the way they can influence the MBean.

Icon	Elements
Amber	Editable attributes
Gray	Non-editable attributes
Red	Operations

Table 53: MBean icons

15.2.2.2 Filtering MBeans

1. In the JMX Explorer view, enter a filter criterion in the text field.

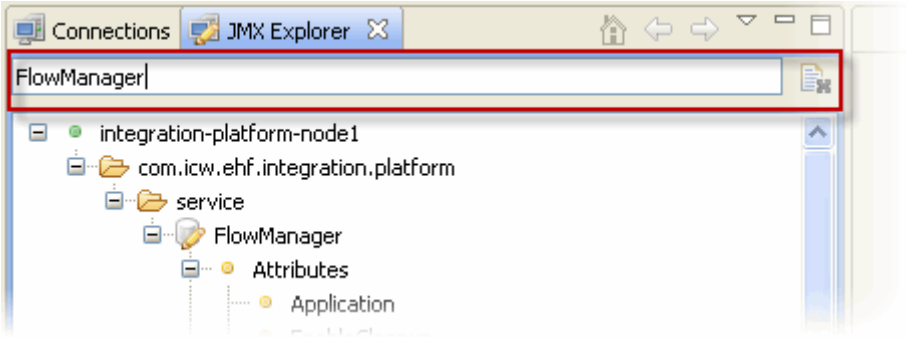


Figure 27: Filtering MBeans in the JMX Explorer view

⇨ Only nodes including associated parent and child nodes that match the entered string will be displayed.

15.2.3 Properties view

The Properties view displays properties of the elements selected from the tree structure. Which properties are shown depends on the selected element:

- If MBean operation and attribute elements are selected, the Properties view will show any child elements for the respective node.
- If MBean attributes or operations are selected, their properties will be displayed. Operations cannot be performed from this view.
- If an MBean attribute is selected, its value can be edited from the Properties view as long as the the attribute is an editable one ( ) and its type is `primitive` (`java.lang.String, int, double` etc...).

15.2.3.1 Editing attributes

1. In the *Properties* view, click on the Value **column** of the respective attribute.
2. Click **Apply** to save your changes.
3. Enter a value.

   ⇨ The changes are transmitted to the server.

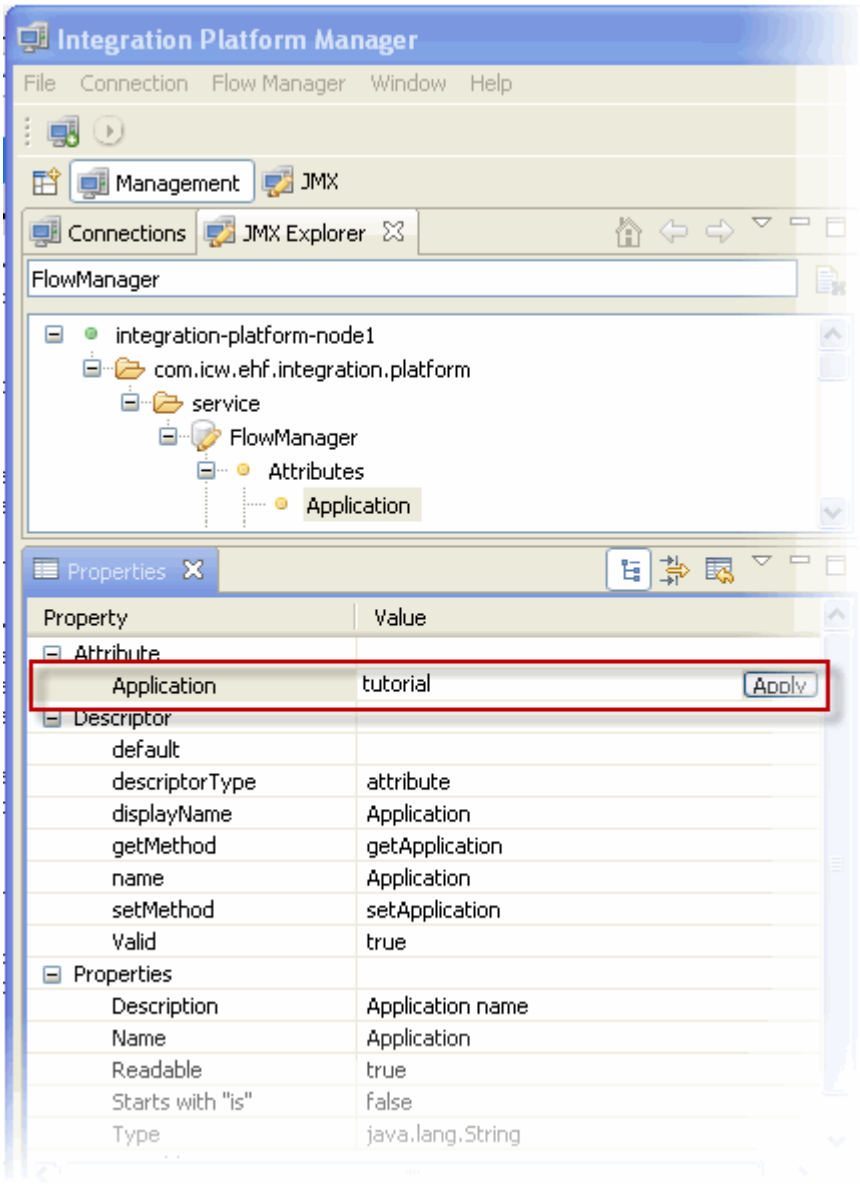


Figure 28: Editing attributes in the Properties view

NOTE

Attributes can be edited, but operations cannot. You perform operation from the JMX Editor, see section JMX Editor [page 139].

15.2.4 JMX Editor

The JMX Editor enables you to administer MBeans within the Java process of the Integration Platform. Use the JMX Editor to monitor or change flows or Java-specific parameters that are independent of the Integration Platform.

15.2.4.1 Opening JMX Editor

You have a number of options to open the JMX Editor by double-clicking:

- An MBean **Attributes** node
- Subordinate MBean Attributes node
- MBean **Operations** node
- Subordinate MBean Operations node

Although the JMX Editor is opened by double-clicking MBean attribute or operation nodes it is actually mapped to a connection. If a second request to open a JMX Editor is sent from the connection, the first JMX Editor will be closed and another instance opened (depending on the selected object).

The JMX Editor will be closed if all associated connections are deleted.

The JMX Editor will be disabled if the status of all associated connections is set to close.

15.2.4.2 Changing MBean attributes

Aside from the *Properties* view, you can also use the JMX Editor to assign attributes to MBeans:

1. Set up a connection as described in section Setting up a connection [page 125] if necessary.
2. Open the **connection** in the JMX Editor, see section Viewing connections [page 135].
3. Go to the MBean element.
4. Open the **JMX Editor** by double-clicking an **attributes** node or one of its child nodes.
5. Click any field in the JMX Editor.
6. Change the value.
7. Click **Apply** to save your changes.
   ⇨ The output will be in the respective *Console* view, see section Console view [page 143].

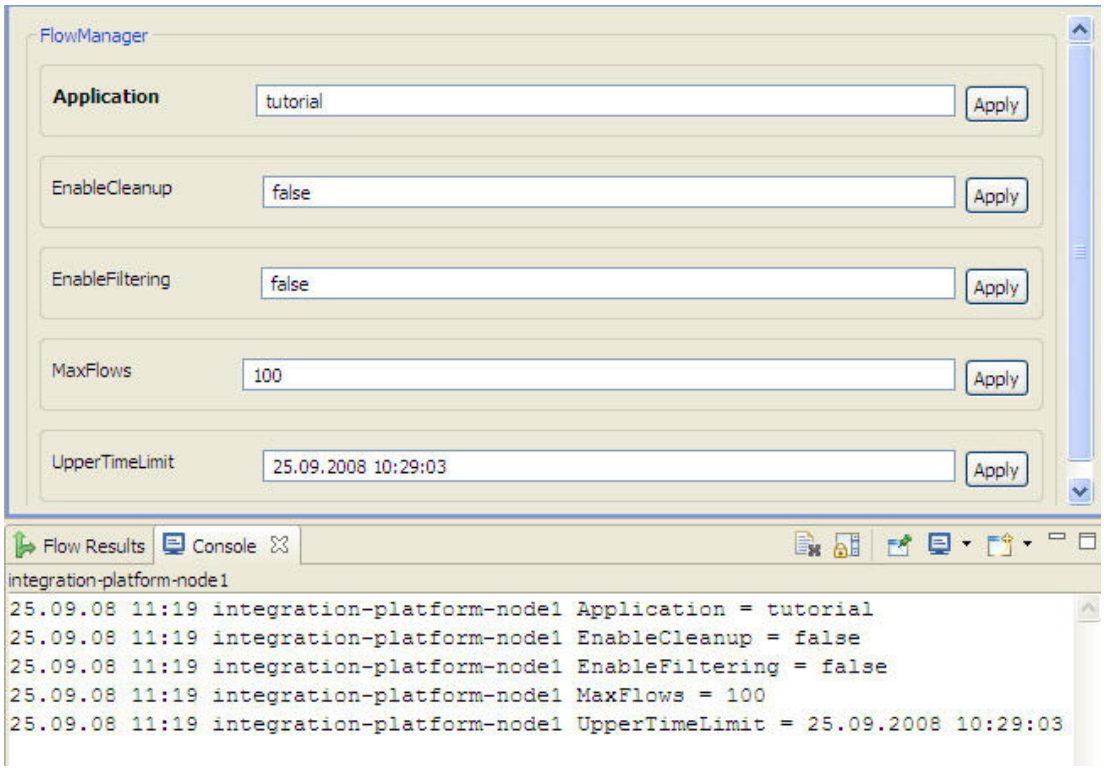


Figure 29: Changing attribute values (MBeans)

15.2.4.3 Display of attribute values

When independent, Java-specific properties are being monitored, some MBean nodes will contain attribute lists. An example for a list of this kind can be seen in Figure Attribute lists for MBean nodes.

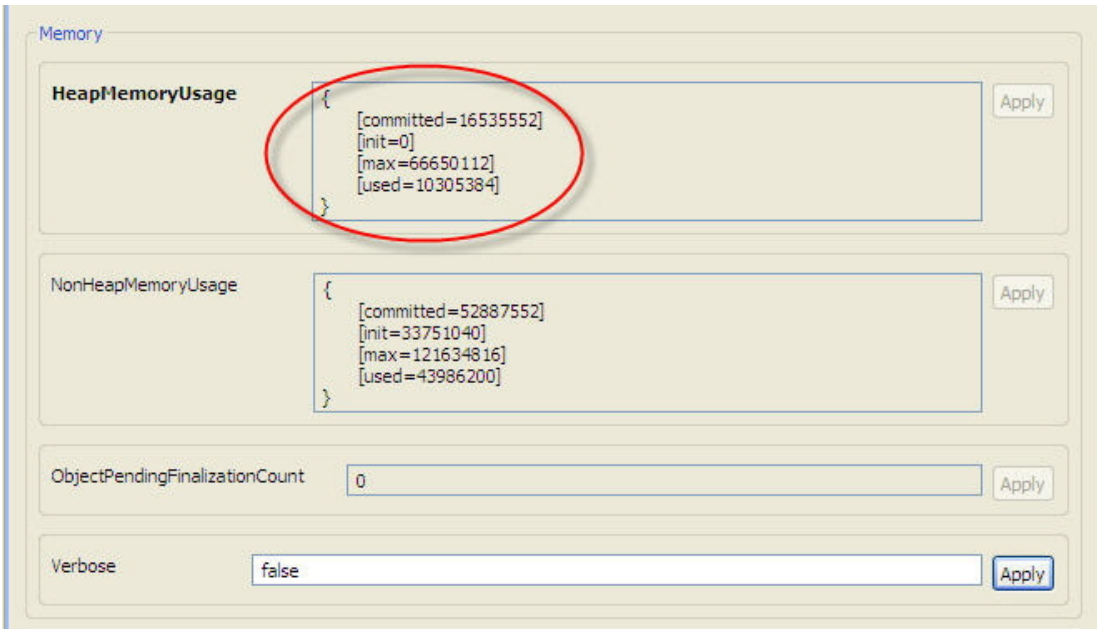


Figure 30: Attribute lists for MBean nodes

The list elements are enclosed by braces:

```
{
    [list element]
    [list element]
}
```

A single element and its element value are enclosed by square brackets.

Sometimes elements can be interlaced:

```
{
    [element name = element value]
    [element name = element value]
}


{
    [element name = {[element name = element value]
                     ...
                     [element name = element value]
                     }
}
```

Attribute lists are predefined and can be neither created nor edited in the JMX Editor.

15.2.4.4 Performing operations (MBeans)

1. Set up a connection as described in section Setting up a connection [page 125] if necessary.
2. Open the **connection** in the JMX Editor, see section Viewing connection(s) in the JMX Explorer view [page 135].
3. Go to the MBean element.

4. Open the JMX Editor by double-clicking the **operations** node or one of its child nodes.
5. Enter a parameter for the operation.
6. Run the operation by clicking the respective button.
   ⇨ The output will be in the respective *Console* view, see section Console view [page 143].

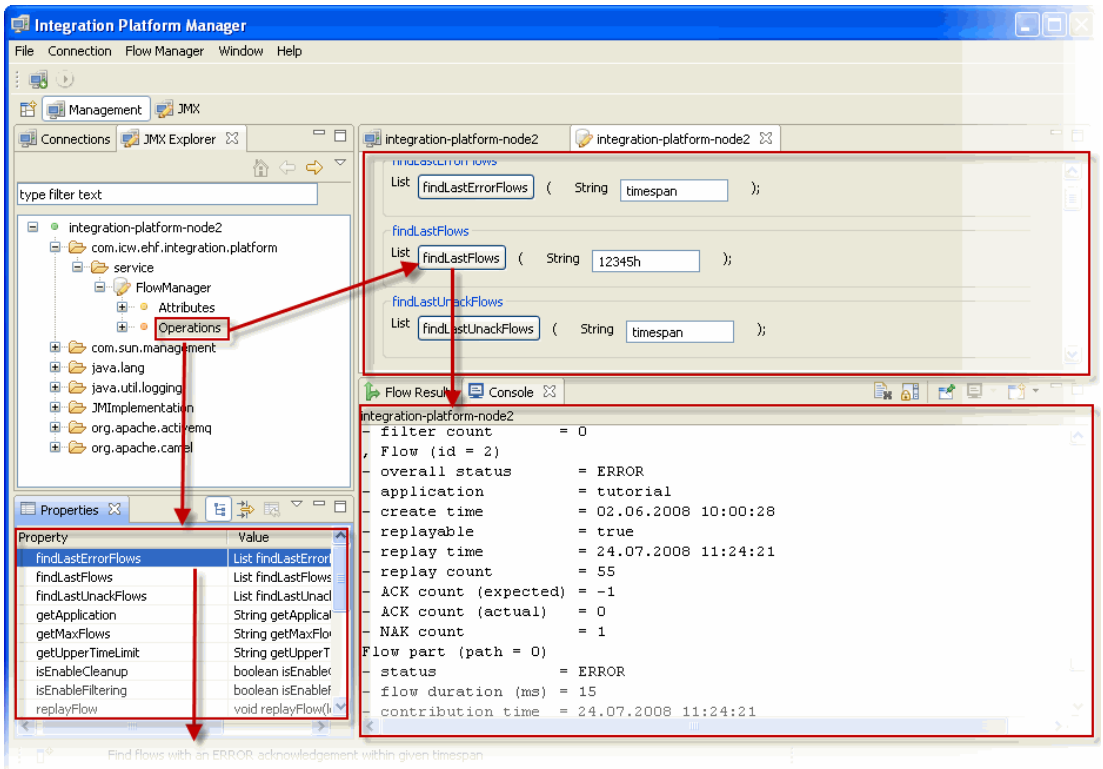


Figure 31: Running an operation and result in the Console view

15.2.4.5 Console view

After running an operation, see [page 142], or changing the value of an attribute, see [page 140], the corresponding Console view will be displayed where the corresponding context-dependent output can be viewed. The focus hereby is on currently associated connection with JMX Editor.

Alternatively, you can control which connection-specific Console view you will see from the Console view:

1. In the Console view, click Display Selected Console  .
2. Select the connection you want from the list.
   ⇨ The console will be displayed according to the selected connection.

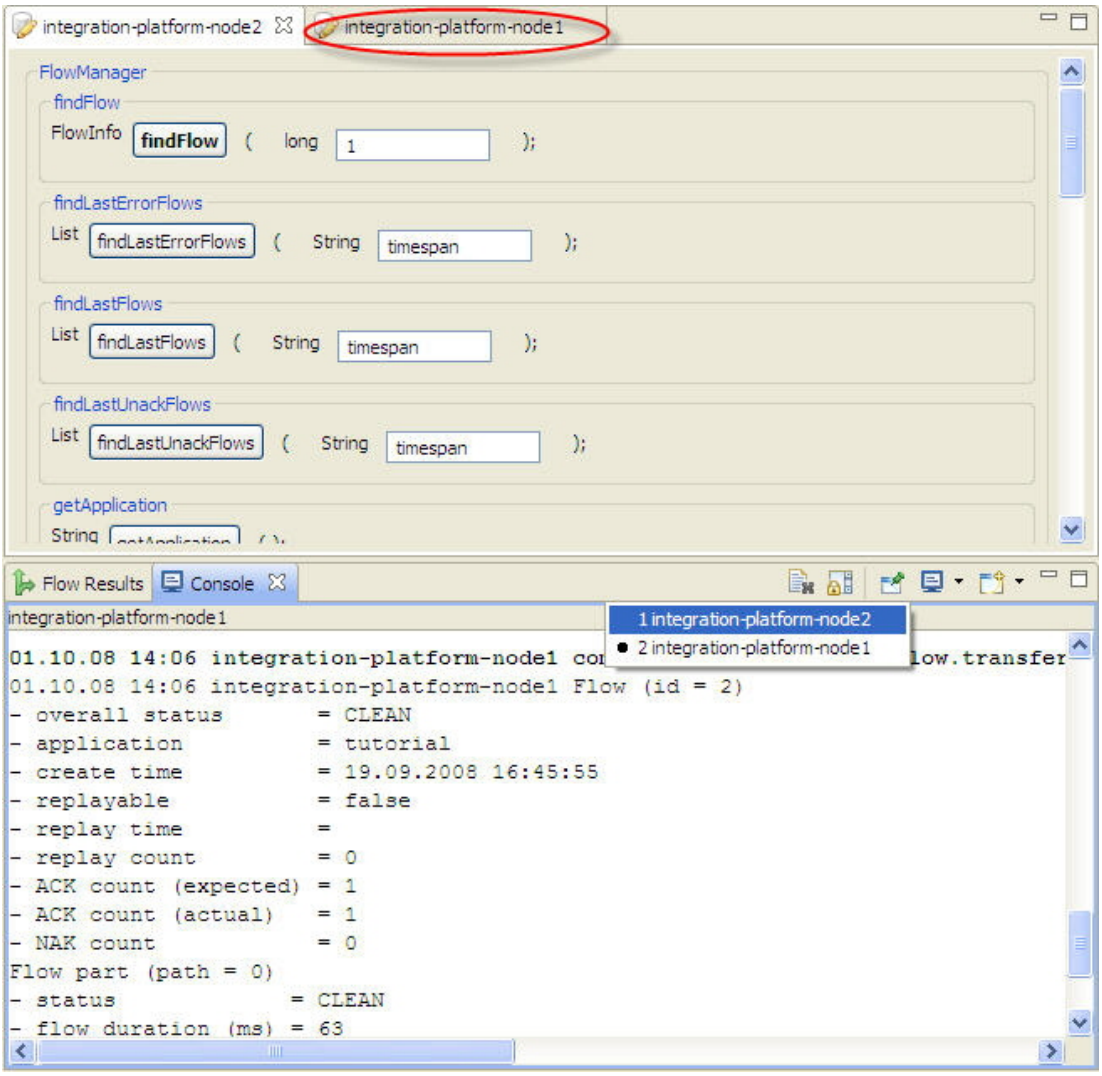


Figure 32: Console view of a corresponding connection

16 Glossary

Term	Description
.NET	The Microsoft .NET Framework is a software technology available with several Microsoft Windows operating systems
ACK	Acknowledgement, or confirmation message
ActiveMQ	Open Source Implementation of JMS
ADT	Admission, Discharge and Transfer
API	Set of function calls an operating system (or other software generally) provides for use by application programmers Application Programming Interface: ● Set of function calls an operating system (or other software generally) provides for use by application programmers ● The defined specification for such a set of calls
ASN.1	Abstract Syntax Notation One
CA	Certificate Authority
Deployment	Distribution, installation and configuration of software on target systems
Endpoint	Component of an integration middleware (enterprise service bus) directed at external systems.
File (binding)poller endpoint	Endpoint that periodically checks a directory for new files. The files are read and their contents summarized in the form of a message.
Groovy	Object-oriented, dynamic programming language for the Java platform as an alternative to the Java programming language.
HL7 2	Health Level 7, Version 2 (also “Version 2.x”) Protocol for message exchange from Health Level Seven, Inc.
HL7 3	Health Level 7, Version 3 New messaging standard from Health Level Seven, Inc. A group of messaging specifications. It contains a generic model for healthcare objects (RIM) and a modeling methodology for new health care domains
HTTP	Hypertext Transfer Protocol The Hypertext Transfer Protocol (HTTP) is a protocol for the transfer of data over a network. It's primary use is for loading websites and other data from the World Wide Web (www) in a Web browser.
HTTPS	Hypertext Transfer Protocol Secure The Hypertext Transfer Protocol over SSL is a URI schema that defines an additional layer between HTTP and TCP. It serves to encrypt and authenticate communication between Web server and browser.

Term	Description
IHE Actor	Information systems or applications that produce, manage or act on information are represented as functional units called IHE Actors. Each actor supports a specific set of IHE transactions. A given information system may support one or more IHE actors.
IN1, IN2, IN3	HL7v2 segments that contain patient insurance information
jconsole	JMX client delivered together with JDK distributions. Enables monitoring and administration of target applications at runtime over JMX (MBeans).
JDK	Java Development Kit
JMS	Java Message Service
JMX	Java Management Extensions
JVM	Java Virtual Machine
keystore	A repository for X.509 certificates used to authenticate SSL connections.
KILL signal	An instrument of an operating system to terminate a running process.
HIS	Hospital Information System A hospital information system (HIS) is the sum of all information processing units used to process medical and administrative data in a hospital. It includes computer programs, personnel and non IT-based information systems. The term is often restricted to mean the computer-based components of the HIS. Sometimes an additional distinction is made between the HIS as central system and specialized systems, such as radiological information systems (RIS), laboratory information systems (LIS), etc.
IPF	Open eHealth Integration Platform. Extension to Apache Camel that adds a lot of eHealth-specific integration functionality.
MLLP	Minimal Lower Layer Protocol
MSB	Medical Service Bus
MSH-3	Field “Sending Application” in a HL7 v2 Message
MSH-4	Field “Sending Facility” in a HL7 v2 Message
NAK	Negative Acknowledgement
OID	Object Identifier Object Identifier: A hierarchical ID that uniquely identifies a device type or a source system.
PRPA	Patient management interaction (message) under HL7v3.
pxsa	Standard adapter - MSB component
pixpdq	PIX/PDQ adapter - MSB component
RMI	Remote Method Invocation

Term	Description
root user	Superuser/Administrator in UNIX-like operating systems
SOAP	Simple Object Access Protocol SOAP is a protocol for sharing XML-based files between systems and for performing remote procedure calls
TCP	Transmission Control Protocol, a network protocol used in the Internet
truststore	A repository of X.509 certificates used for authentication in secure connections (SSL).
XCPD	Cross-community Patient Discovery, an IHE profile
XML	Extensible Markup Language Hierarchical modeling standard for semi-structured data defined by the World Wide Web Consortium (W3C).