

NextGen® Connect

User Guide

for version 3.8

June 18, 2019

Contents

1. Introduction	6
2. Getting Started	8
2.1 System Requirements	9
2.2 Download and Installation	11
2.3 The Mirth Connect Server Manager	17
2.4 Launching the Mirth Connect Administrator	20
2.5 The Web Dashboard	24
2.6 Changing the Database Type	25
2.7 Using Java 9 or greater	29
3. The Fundamentals of NextGen Connect	30
3.1 About Channels and Connectors	31
3.2 About Message Data	35
3.3 The Message Processing Lifecycle	37
3.4 About Data Types	40
3.5 About Filters	42
3.6 About Transformers	44
4. The Administrator Launcher	46
4.1 Introduction / Installation	47
4.2 The Launcher Interface	51
5. Mirth Connect Administrator	53
5.1 Monitoring Views	58
5.1.1 Dashboard View	59
5.1.1.1 Dashboard Table	61
5.1.1.2 Filtering By Channel Name or Tag	65
5.1.1.3 Server Log	70
5.1.1.4 Connection Log	72
5.1.1.5 Global Maps	74
5.1.1.6 Dashboard Tasks	76
5.1.2 Message Browser View	80
5.1.2.1 Metadata Table	83
5.1.2.2 Message Content Tab	87
5.1.2.3 Mappings Tab	90
5.1.2.4 Errors Tab	92
5.1.2.5 Attachments Tab	94
5.1.2.6 Searching Messages	99
5.1.2.7 Message Browser Tasks	103
5.1.3 Alerts View	110
5.1.3.1 Alerts Table	111
5.1.3.2 Alerts Tasks	112
5.1.4 Events View	113
5.1.4.1 Events Table	115
5.1.4.2 Event Attributes Table	117
5.1.4.3 Searching Events	118
5.1.4.4 Event Tasks	120
5.2 Management Views	121
5.2.1 Channels View	122
5.2.1.1 Channel Table	124
5.2.1.2 Channel Tasks	128
5.2.1.3 Group Tasks	131
5.2.2 Users View	133
5.2.2.1 Users Table	134
5.2.2.2 Users Tasks	135
5.2.3 Settings View	137
5.2.3.1 Server Settings Tab	139
5.2.3.2 Administrator Settings Tab	143
5.2.3.3 Tags Settings Tab	146
5.2.3.4 Configuration Map Settings Tab	149

5.2.3.5 Database Tasks Settings Tab	151
5.2.3.6 Resources Settings Tab	153
5.2.3.7 Data Pruner Settings Tab	156
5.2.3.8 Settings Tasks	159
5.2.4 Extensions View	160
5.2.4.1 Installed Connectors Table	162
5.2.4.2 Installed Plugins Table	163
5.2.4.3 Installing a New Extension	164
5.2.4.4 Extension Tasks	165
5.3 Editing Views	167
5.3.1 Edit Channel View	168
5.3.1.1 Summary Tab	170
5.3.1.2 Source Tab	197
5.3.1.3 Destinations Tab	214
5.3.1.4 Scripts Tab	222
5.3.1.5 Edit Channel Tasks	223
5.3.2 Edit Filter / Transformer Views	224
5.3.2.1 Message Templates Tab	226
5.3.2.2 Message Trees Tab	229
5.3.2.3 Reference Tab	234
5.3.2.4 Creating New Rules / Steps	238
5.3.2.5 Rule / Step Table	240
5.3.2.6 Filter Rule Properties	241
5.3.2.7 Transformer Step Properties	246
5.3.2.8 Response Transformers	255
5.3.2.9 Working With Iterators	257
5.3.2.10 Viewing Generated Script	262
5.3.2.11 Filter Tasks	263
5.3.2.12 Transformer Tasks	265
5.3.3 Edit Global Scripts View	267
5.3.4 Edit Code Templates View	269
5.3.4.1 Code Template Library Table	271
5.3.4.2 Edit Library Panel	272
5.3.4.3 Edit Code Template Panel	273
5.3.4.4 Code Template Tasks	277
5.3.5 Edit Alert View	280
5.3.5.1 Alert Error Types and Regex	282
5.3.5.2 Alert Enabled Channels	283
5.3.5.3 Alert Actions	284
5.3.5.4 Edit Alert Tasks	285
5.4 Other Tasks	286
5.4.1 Notifications	288
6. Data Types	289
6.1 Delimited Text Data Type	290
6.2 DICOM Data Type	292
6.3 EDI / X12 Data Type	293
6.4 HL7 v2.x Data Type	294
6.5 HL7 v3.x Data Type	297
6.6 JSON Data Type	298
6.7 NCPDP Data Type	299
6.8 Raw Data Type	300
6.9 XML Data Type	301
6.10 Batch Processing	302
6.11 JavaScript Batch Script	303
7. Source Connectors	304
7.1 Channel Reader	305
7.2 DICOM Listener	306
7.3 Database Reader	310
7.4 File Reader	314
7.5 HTTP Listener	321
7.6 JMS Listener	324

7.7 JavaScript Reader	327
7.8 TCP Listener	328
7.8.1 Basic TCP Transmission Mode	331
7.8.2 MLLP Transmission Mode	333
7.9 Web Service Listener	335
8. Destination Connectors	336
8.1 Channel Writer	337
8.2 DICOM Sender	339
8.3 Database Writer	342
8.4 Document Writer	344
8.5 File Writer	346
8.6 HTTP Sender	350
8.7 JMS Sender	354
8.8 JavaScript Writer	357
8.9 SMTP Sender	359
8.10 TCP Sender	363
8.11 Web Service Sender	367
9. NextGen Connect Integration Engine and JavaScript	370
9.1 About JavaScript	371
9.2 Using JavaScript in Mirth Connect	378
9.3 Using the JavaScript Editor	384
9.4 Variable Maps	388
9.4.1 The Variable Map Lookup Sequence	392
9.5 Attachment JavaScript Functions	393
9.6 The User API (Javadoc)	396
10. Velocity Variable Replacement	398
11. NextGen Connect Command Line Interface	400
12. NextGen Connect REST API	401
13. Installation Directory	404
13.1 Application Data Directory	405
13.2 Configuration Directory	407
13.2.1 The dbdrivers.xml File	408
13.2.2 The log4j.properties File	409
13.2.3 The log4j-cli.properties File	411
13.2.4 The mirth.properties File	412
13.2.5 The mirth-cli-config.properties File	420
13.3 Other Files and Folders	421
14. FAQ	422
15. Channel Development Best Practices and Tips	428
16. Troubleshooting	435
17. Commercial Support / Extensions	442
17.1 Advanced Alerting	443
17.2 Advanced Clustering	444
17.3 ASTM E1381 Transmission Mode	445
17.4 ASTM E1394 Data Type	446
17.5 Channel History	447
17.6 Email Reader	448
17.7 FHIR Connector	449
17.8 Interoperability Connector Suite	451
17.9 LDAP Authorization	453
17.10 Message Generator	454
17.11 Multi-Factor Authentication	455
17.12 Serial Connector	456
17.13 SSL Manager	457
17.14 User Authorization	458
17.15 NextGen Results CDR Connector	459
18. Training	460

User Guide

Introduction

The NextGen Connected Health Mission

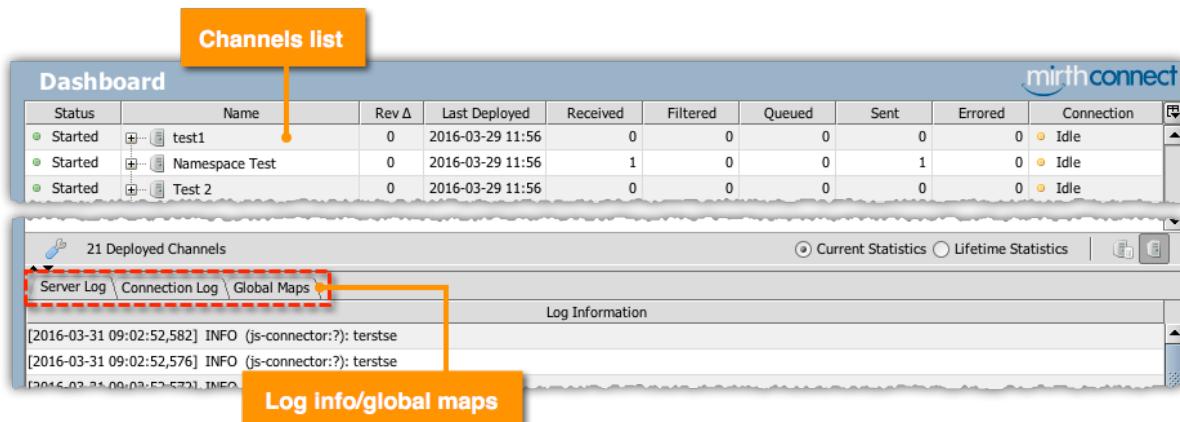
NextGen® Connected Health helps many of the nation's largest, most respected healthcare entities streamline their care-management processes to satisfy the demands of a regulatory, competitive healthcare industry. With NextGen Connected Health, NextGen Healthcare's goal is to provide the healthcare community with a secure, efficient, cost-effective means of sharing health information. The natural product of this aim is a family of applications – which includes NextGen® Connect Integration Engine (formerly Mirth Connect) – flexible enough to manage patient information, from small practices to large HIEs, so our clients and users can work confidently and effectively within the healthcare-delivery system.

About NextGen Connect Integration Engine

Like an interpreter who translates foreign languages into the one you understand, NextGen Connect Integration Engine translates message standards into the one your system understands. Whenever a "foreign" system sends you a message, NextGen Connect Integration Engine's integration capabilities expedite the following:

- Filtering – NextGen Connect Integration Engine reads message parameters and passes the message to or stops it on its way to the transformation stage.
- Transformation – NextGen Connect Integration Engine converts the incoming message standard to another standard (e.g., HL7 to XML).
- Extraction – NextGen Connect Integration Engine can "pull" data from and "push" data to a database.
- Routing – NextGen Connect Integration Engine makes sure messages arrive at their assigned destinations.

Users manage and develop channels (message pathways) using the interface known as the *Administrator*.



The Healthcare Interoperability Challenge / Solution

Most often, patient data is exchanged via computer systems (e.g., a doctor's office sends patient records to a hospital, a clinic sends a prescription request to a pharmacy). Such communication is not foolproof. Data can be delayed or lost, and privacy is not always assured, making the transaction less efficient and reliable than it could be. Contributing factors include:

- protocol conflicts between sites
- mismatched versions of record-keeping software
- costly software licensing
- HIPAA privacy and security

- incompatible data due to varied software and communication methods
- lack of control and flexibility related to software use.

In the following diagram, you can see NextGen Connect Integration Engine's flexibility. In it, a lab's data system sends an HL7 message to NextGen Connect Integration Engine via a Minimal Lower Layer Protocol (MLLP). NextGen Connect Integration Engine inserts patient data into an Electronic Health Record (EHR) database, creates a PDF (portable document format) file, and sends an email message with the PDF file attached.



The next diagram shows how NextGen Connect Integration Engine reads patient data from a hospital's Electronic Medical Record (EMR) system. With elements mapped in its own channel, NextGen Connect Integration Engine generates an HL7 message and sends it to a client for outpatient care. Multiple configurations are available depending on how the channel is constructed.



Using Open Source Software

NextGen Connect Integration Engine is licensed under [Mozilla Public License 1.1](#). Because it is open source, customers benefit from a vast array of contributions and testing from scores of healthcare professionals that comprise a vibrant public community. Many issues are resolved quickly, and community input is adapted to make Mirth Solutions more helpful and user friendly. If you are hesitant about using open-source software, be assured that NextGen Healthcare fully backs its entire open source suite with:

- support services to match every need level and budget
- professional services to complete your integration project quickly and correctly
- hosting services that offer you HIPAA (Health Insurance Portability and Accountability Act)-grade security
- an array of physical and virtual appliances to save you time and resources so you can deploy on a standardized, reliable platform.

Getting Started

A stand-alone instance of NextGen Connect Integration Engine comes with an installer for Windows, Linux, and Mac OS X / macOS.



This section mainly pertains to the standalone version of NextGen Connect Integration Engine. If you have purchased a NextGen Application Management Console, you may disregard the download/install procedures.

This section is separated into the following topics:

- [System Requirements](#)
- [Download and Installation](#)
- [The Mirth Connect Server Manager](#)
- [Launching the Mirth Connect Administrator](#)
- [The Web Dashboard](#)
- [Changing the Database Type](#)
- [Using Java 9 or greater](#)

System Requirements

The NextGen Connect Integration Engine Server is a fully standalone application that does not require any sort of application server. You do **not** need to install any sort of container service like Tomcat, Glassfish, etc.

Java Requirements

The NextGen Connect Integration Engine Server, Command Line Interface, and Administrator are cross-platform applications that only require a JRE/JDK (Java installation). As of version 3.7, [OpenJDK](#) is supported in addition to the official [Oracle JRE/JDK](#). OpenJDK has many different distributions from various organizations, but the one we recommend using is the official [Oracle OpenJDK](#) distribution.

Supported Java versions for each NextGen Connect Integration Engine version appear in this table:

	3.0.2 - 3.1.x	3.2.x - 3.4.x	3.5.x	3.6.x	3.7.x+
Java 6	✓	✗	✗	✗	✗
Java 7	✓	✓	✗	✗	✗
Java 8	✓	✓	✓	✓	✓
Java 9	✗	✗	✗	✓	✓
Java 10	✗	✗	✗	✓	✓
Java 11+	✗	✗	✗	✗	✓



When using Java 9 or greater, there are some additional JVM options that need to be set. See [Using Java 9 or greater](#) for more info.

Database Requirements

The NextGen Connect Integration Engine Server requires a database for its configuration and message store. For quick deployment, development, and testing, NextGen Connect Integration Engine already includes an embedded database (Apache Derby). For production, the latest version of NextGen Connect Integration Engine supports the following databases:

- PostgreSQL 8.3+
- MySQL 5.6+
- Oracle 10gR2+
- SQL Server 2005+

Note that the above database requirements only apply to what is used for the configuration and message store of the NextGen Connect Integration Engine Server, and have no impact on which databases NextGen Connect Integration Engine can interface with.

Download and Installation

The NextGen Connect Integration Engine installer (for Windows, Linux, and Mac OS X / macOS) automatically upgrades the previous version of the software installed on your system.

1. Click this link: <https://www.mirth.com/>.
2. On the Mirth Open Source page, click **DOWNLOAD NOW**.
3. On the **Downloads** page, click the appropriate installer link for your system.

Mirth Connect 3.5.0

Important Information: [What's New](#) | [Release Notes](#) | [Upgrade Guide](#)

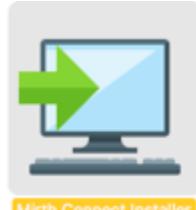
Platform	System Requirements	Version	Downloads
Windows	XP, Vista, 7, Server	3.5.0.8232	32-bit Installer 32-bit Zip 64-bit Installer 64-bit Zip
Linux	2.4 kernel or newer	3.5.0.8232	RPM Tar.gz Installer
Mac OS X	10.X	3.5.0.8232	Installer Tar.gz

Mirth Connect 3.5.0 Command Line Interface

Platform	System Requirements	Version	Downloads
Windows	XP, Vista, 7, Server	3.5.0.8232	32-bit Zip 64-bit Zip
Linux / Mac OS X	2.4+ kernel / 10.X	3.5.0.8232	Tar.gz

[Archived Downloads](#)

4. When the download is complete, double-click the download file, then double-click the **Mirth Connect Installer** icon.



5. On the **Mirth Connect Setup Wizard**, click **Next >**.



Yes, update the existing installation is the default selection. If, you'd like to install NextGen Connect Integration Engine elsewhere on your system, select **No, install into a different directory**, and install NextGen Connect Integration Engine in your desired location.



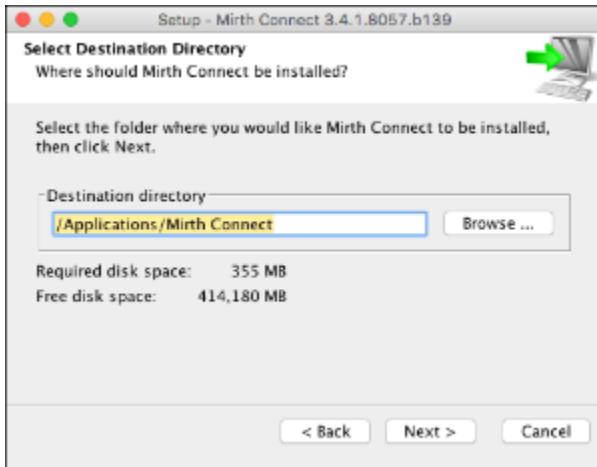
6. In the **License Agreement** window, read the license agreement, select **I accept the agreement**, then click **Next >**.



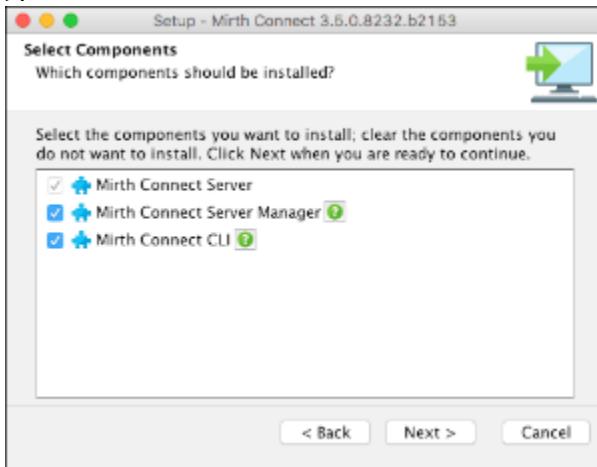
7. In the **Destination directory** section, click **Browse...**, find and select the folder in which you want to install NextGen Connect Integration Engine, then click **Next >**.



- If you selected **Yes...** in step 5 (you're updating a previously installed version of NextGen Connect Integration Engine), continue to step 8.
- If you selected **No...** in step 5 (you're installing NextGen Connect Integration Engine for the first time), complete step 7.

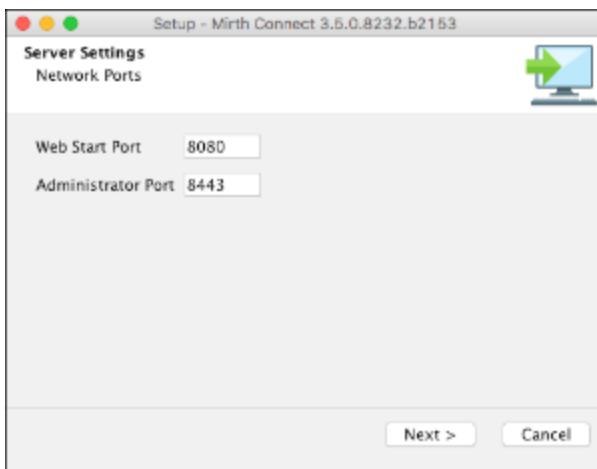


8. On the **Select Components** dialog, select the Mirth Connect components you want to install, then click **Next >**.

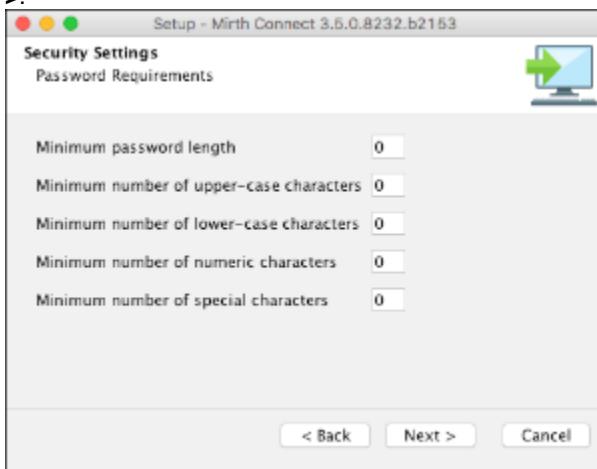


⚠ The **Mirth Connect Server** component is greyed out because it is not an option, but you can select /deselect **Mirth Connect Server Manager** or **Mirth Connect CLI**.

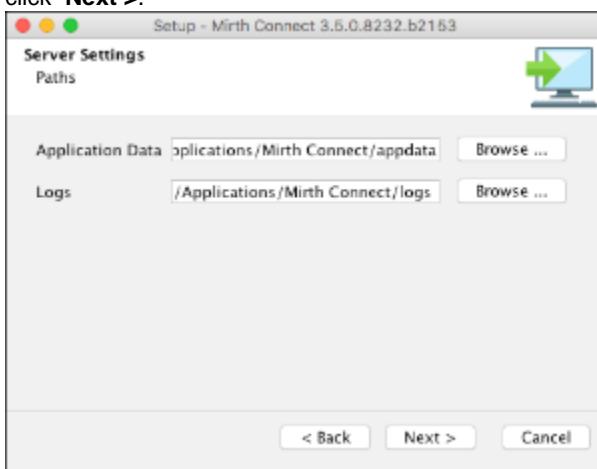
9. On the **Server Settings – Network Ports** dialog, enter the web start and administrator port values as needed, then click **Next >**.



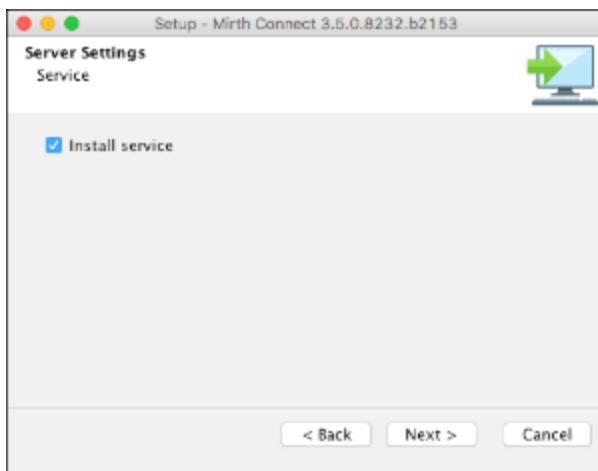
10. On the **Security Settings – Password Requirements** dialog, set your password parameters, then click **Next >**.



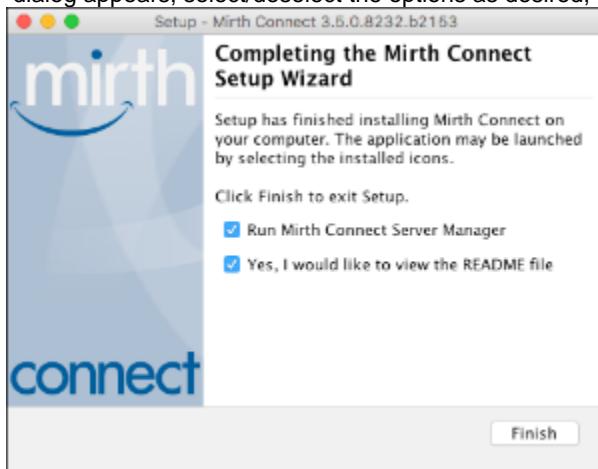
11. On the **Server Settings – Paths** dialog, set the server-settings paths for the Application Data and Logs, then click **Next >**.



12. On the **Server Settings – Service** dialog, click **Next >**. NextGen Connect Integration Engine begins its installation.

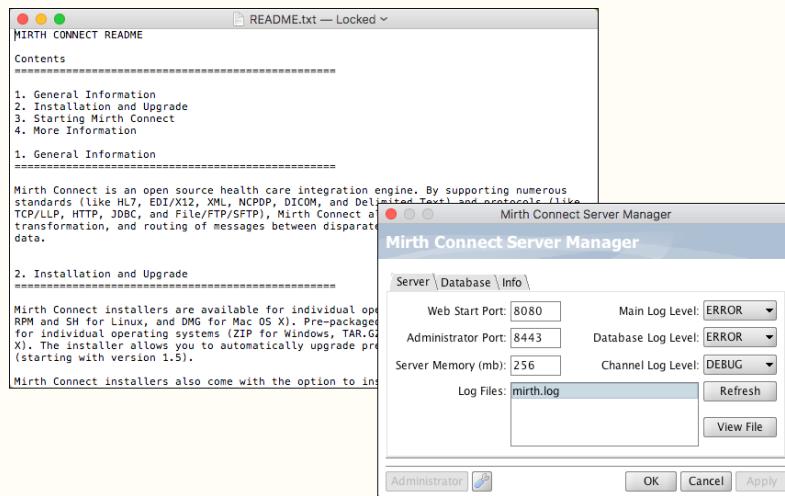


13. Once NextGen Connect Integration Engine installs, the **Completing the Mirth Connect Setup Wizard** dialog appears; select/deselect the options as desired, then click **Finish**.





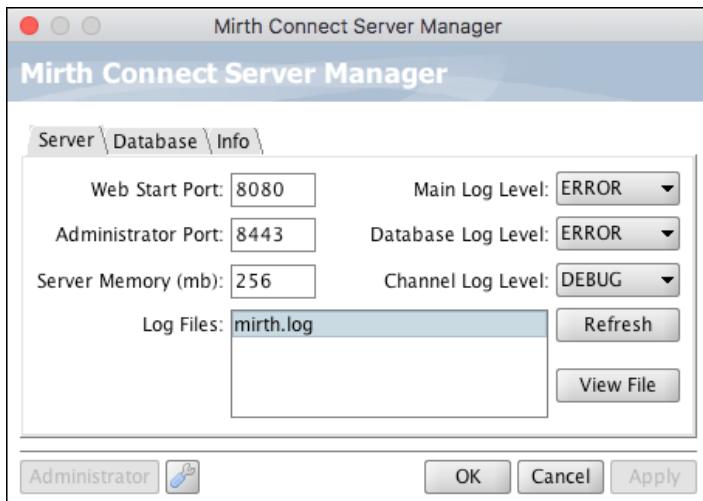
Depending on the options you chose in the **Setup Wizard**, the Mirth Connect Server Manager or the **README** file – or both – appear.



The Mirth Connect Server Manager

Windows and Mac OS X versions install a **Mirth Connect Server Manager**, an application that resides in the system tray (Windows) or in the **Applications > Mirth Connect** folder (Mac). The **Mirth Connect Server Manager** dialog consists of **Server**, **Database**, and **Info** pages.

Server Page



The Server page allows you to enter details about your ports, server memory, log levels, and view log files. The following fields are available on the Server page.

Function	Description
Web Start Port	Accesses the launch page for the Mirth Connect Administrator. Default: 8080
Administrator Port	This port is used by the Mirth Connect Administrator to communicate with the NextGen Connect Integration Engine Server. Default: 8443
Server Memory (mb)	The server's maximum available memory (Java max heap size). By default this is 256 MB, but for large production instances you will typically want to increase this value.
Main Log Level	These fields allow you to select the applicable log level from the drop-down menus. Available options include:
Database Log Level	<ul style="list-style-type: none"> • <i>ERROR</i> • <i>WARN</i> • <i>INFO</i> • <i>DEBUG</i> • <i>TRACE</i>
Channel Log Level	Depending on the log level, messages of the selected level or lower will pass into that level's log when the system logs a certain-level message.
Refresh	Select this button to update the most recent list of log files identified in the Log Files area.

View File	Select this button to display a selected log file. This field is grayed out if a file is not selected.
Administrator	(PC only) Opens the Mirth Connect login page (inactive on Macs; see Launching the Mirth Connect Administrator)
OK	Saves your changes and exits the Mirth Connect Server Manager.
Cancel	Exits the Mirth Connect Server Manager without saving your changes.
Apply	Applies changes to the field and drop-down settings but does not exit the Mirth Connect Server Manager. This button is grayed out unless changes have been made to the page settings.

Database Page



The Database page allows you to manage NextGen Connect Integration Engine's internal database. The following fields are available on the Server page.

Function	Description
Type	Select the database on which NextGen Connect Integration Engine will store data.
URL	The Java Database Connectivity (JDBC) URL string associated with selected type.
Username/Password	The user's unique personal identifier/access code

! NextGen Connect Integration Engine's default database, **Apache Derby**, is included only to help you set up quickly. Because it is not a production-level database, Mirth Corporation strongly recommends that you **not** use Derby for production.

Info Page



The **Info** page shows the NextGen Connect Integration Engine server and Java versions as well as the server ID and a link to the Mirth Corporation website.

Launching the Mirth Connect Administrator

The Mirth Connect Administrator is used to develop and manage all channels and can be run anywhere in relation to the NextGen Connect Integration Engine server. The Mirth Connect Administrator consists of menus and view panels that change depending on the task (e.g., Dashboard, Channel, Edit Channel, Settings).



You no longer need to use Java Web Start! We've released a new Administrator Launcher that you can use instead: [The Administrator Launcher](#)

Install the Administrator Launcher

1. Open a web browser and navigate to your launch page. This is usually <http://localhost:8080> or <https://localhost:8443>, though you may need to edit the IP or port depending on your settings.
2. Click on the cog icon next to the blue "Download Administrator Launcher" button.
3. Choose the operating system for your local machine that you want to install the Administrator Launcher on.

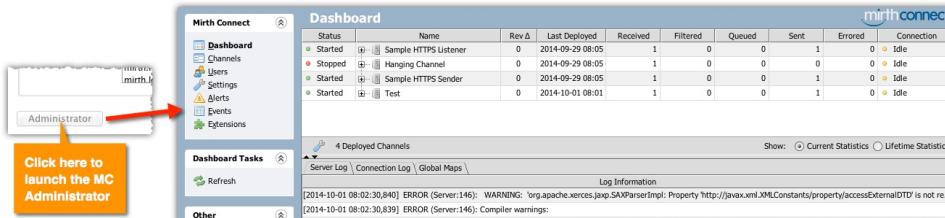


4. Click the blue "Download Administrator Launcher" button to download the installer file. See here for instructions on installing: [Administrator Launcher Installation](#)

Launch the Administrator

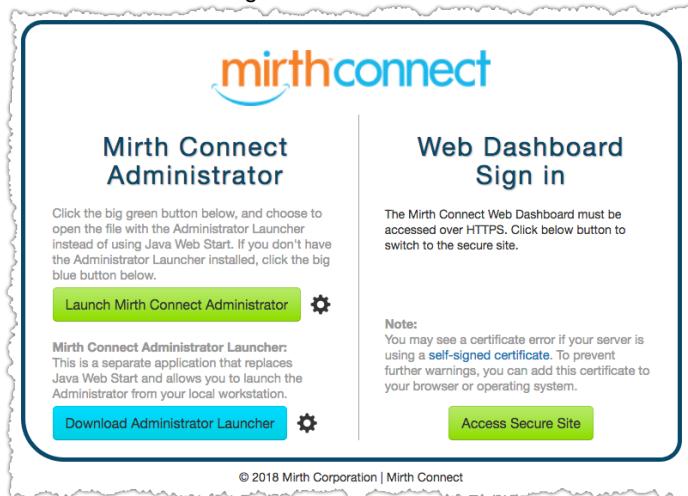
Use the following steps to log in to the Mirth Connect Administrator.

- 1. Using the Server Manager:** On Windows/Linux, you can launch the Mirth Connect Administrator using the **Administrator** button in the bottom-left corner of the Mirth Connect Server Manager – Server page.



- 2. Using the Launch web page:**

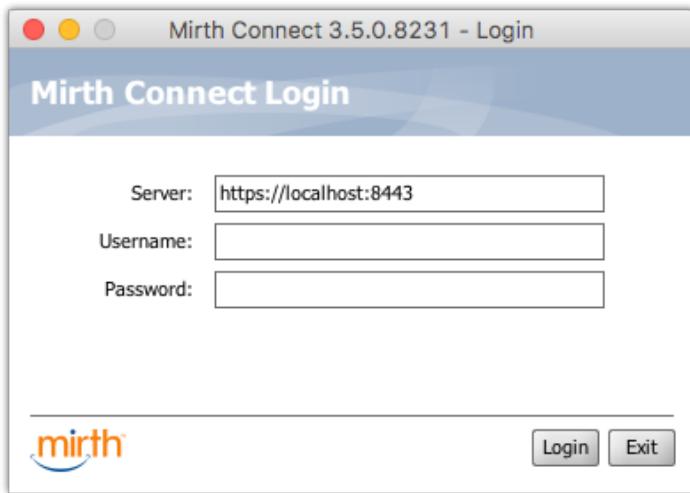
- Open a web browser and navigate to your launch page. This is usually <http://localhost:8080> or <https://localhost:8443>, though you may need to edit the IP or port depending on your settings.
- Click the blue "Download Administrator Launcher" button to download the installer file. See here for instructions on installing: [Administrator Launcher Installation](#)



- The **Access Secure Site** button takes you to the **Web Dashboard Sign In** page. For more information, go here: [The Web Dashboard](#)

Login Dialog

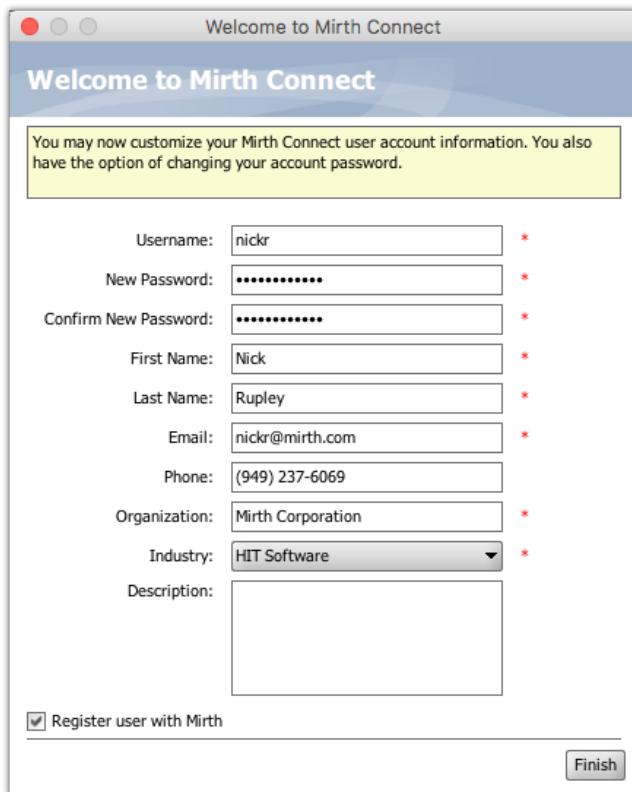
When you first launch the Administrator view, you'll be taken to the login dialog:



The Server URL should be pre-filled with the correct IP address and port, but you can change them here if needed. If this is your first time logging into NextGen Connect Integration Engine, the default username is "**admin**" and the default password is "**admin**". Click **Login** to continue.

First Login Dialog

If this is your first time logging into NextGen Connect Integration Engine, you'll be presented with a registration dialog to set your username / password and other details.



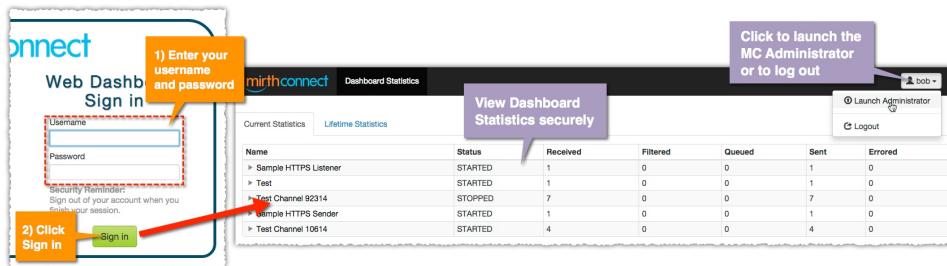
After logging in, go here to learn more about using the Administrator: [Mirth Connect Administrator](#)

The Web Dashboard

The Web Dashboard is a quick and easy way to view channel / connector statistics across your server without needing to launch the Administrator. It runs in a web browser, and is also supported on mobile devices.

To login to the Web Dashboard:

- In the address field of your browser, type `localhost`: followed by the Web Start Port number. (See the **Server Manager – Server** page; default: `8080`)
- Click the **Access Secure Site** button for **Web Dashboard Sign in**, the page changes, prompting you to enter a **Username** and **Password**.



Once you're logged in, you'll see all your deployed channels in a table. To view connector-level statistics, expand the arrow next to your channel name. You can also switch between current and lifetime statistics.

Name	Status	Received	Filtered	Queued	Sent	Errored
Test HTTP Listener	Started	32	0	0	32	0
Test	Started	4	0	3	0	0
Mirth Results Sender PROD	Started	33	0	0	33	0
LLP Inbound 22341	Started	1	0	0	0	1
Source	Started	1	0	0	0	0
Destination 1	Started	1	0	0	0	1
Create and Write PDF	Started	24	0	0	24	0

From the Web Dashboard you can launch the Administrator interface by clicking on the arrow next to the user avatar in the top-right, and selecting **Launch Administrator**. You can also logout of the Web Dashboard from this dropdown menu.

Changing the Database Type

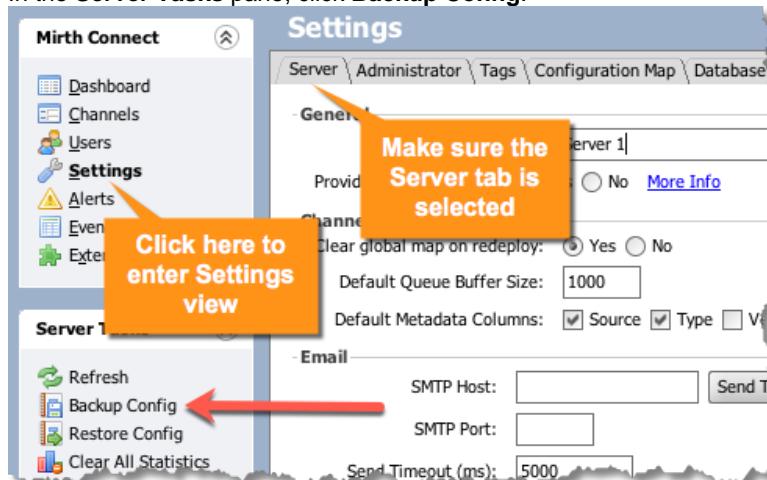
By default, NextGen Connect Integration Engine is configured to connect to an embedded Apache Derby database for quick deployment, development, and testing. When using NextGen Connect Integration Engine in production environments, we recommend changing the underlying database to one of the supported servers:

- PostgreSQL 8.3+
- MySQL 5.6+
- Oracle 10gR2+
- SQL Server 2005+

Backup Current Server Configuration

A Server Configuration file is a snapshot of your current server settings, including all channels, alerts, scripts, and other properties. This configuration may be backed up from one NextGen Connect Integration Engine server and restored into a different server. If you've just installed NextGen Connect Integration Engine and you don't yet have any channels or other configurations, you can skip this section. To backup your server configuration:

1. Log into the **Mirth Connect Administrator** (instructions [here](#)).
2. In the **Mirth Connect** pane, click **Settings**.
3. Select the **Server** tab.
4. In the **Server Tasks** pane, click **Backup Config**.



5. Enter a location to save the server configuration XML file, then click **Save**.
6. Review the Information message, then click **OK**.



The Server Configuration file does not include Users, Events, or Message / Attachment data. To export events, go [here](#). To export message / attachment data, go [here](#).

Change Database Settings

Using the Mirth Connect Server Manager (Windows / Linux):

1. Double-clicking the **Mirth** icon in your system tray to open the Mirth Connect Server Manager.
2. Select the **Database** tab.



3. In the **Type** field, select the database server you want to connect to from the pull-down menu. The JDBC URL field will be auto-populated with a sample URL.
4. In the **URL** field, change the IP, port, and database name as needed.
5. Enter the **Username** and **Password**, if needed.



6. Click **Apply**.
7. Click **OK** to exit the Server Manager.

Editing the Properties File Directly (Windows / Linux / Mac):

1. Navigate to the directory where you installed NextGen Connect Integration Engine.
2. Open the **mirth.properties** file inside the "conf" folder.

```

62
63 # options: derby, mysql, postgres, oracle, sqlserver
64 database = postgres
65
66 # examples:
67 # Derby      jdbc:derby:${dir.appdata}/mirthdb;create=true
68 # PostgreSQL  jdbc:postgresql://localhost:5432/mirthdb
69 # MySQL      jdbc:mysql://localhost:3306/mirthdb
70 # Oracle     jdbc:oracle:thin:@localhost:1521:DB
71 # SQLServer   jdbc:jtds:sqlserver://localhost:1433/mirthdb
72 database.url = jdbc:postgresql://localhost:5432/mirthdb
73
74 # if using a custom driver, specify it here
75 #database.driver =
76
77 # maximum number of connections allowed for the connection pool
78 database.max-connections = 20
79
80 # database credentials
81 database.username = user
82 database.password = pass
83

```

3. Edit the following properties as needed:
 - "database": The type of database server to connect to (derby, mysql, postgres, oracle, sqlserver)
 - "database.url": The JDBC URL connection string
 - "database.driver": If you need to use a custom JDBC Driver class, enter that fully-qualified class name here.
 - "database.username": Username for the database connection.
 - "database.password": Password for the database connection.
4. Save the file.

Restart the Mirth Connect Server

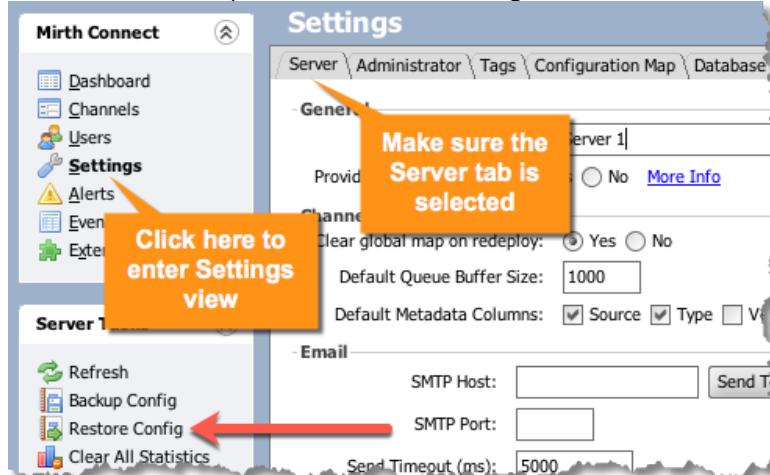
1. If you are on a Windows or Linux operating system and you have NextGen Connect Integration Engine installed as a service/daemon, open the Server Manager (instructions above).



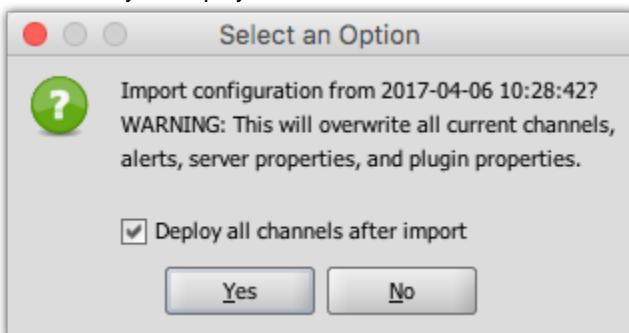
2. Click **Restart** and verify the server has started up successfully.

Restore Server Configuration

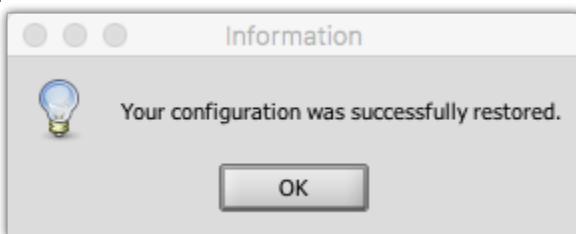
1. Launch the **Mirth Connect Administrator**(instructions [here](#)).
2. In the **Mirth Connect** pane, click **Settings**.
3. Select the **Server** tab if not already selected.
4. In the **Server Tasks** pane, click **Restore Config**.



5. Select the server configuration XML file you exported earlier, then click **Open**.
6. You'll be prompted to confirm your action. You can also choose whether or not you want channels to automatically be deployed after the restore finishes.



7. Click **Yes** to begin the restore process.
8. Once the restore process is complete, a dialog appears indicating whether the restore was successfully performed.



9. Click **OK** to close the information message.

Using Java 9 or greater

As documented in the [System Requirements](#), NextGen Connect Integration Engine version 3.7 supports Java 8 at minimum. In order to use Java 9 or greater, you'll first need to perform an extra manual step.

In the [Installation Directory](#), there should be a **docs** folder. Inside, there is a file called **mcservice-java9+.vmoptions**. This contains some extra JVM options that allow NextGen Connect Integration Engine to run with Java 9 or greater without errors or warnings.

Copy the contents into either the **mcserver.vmoptions** or **mcservice.vmoptions** file, depending on whether your deployment uses mcserver or mcservice. Then restart NextGen Connect Integration Engine.

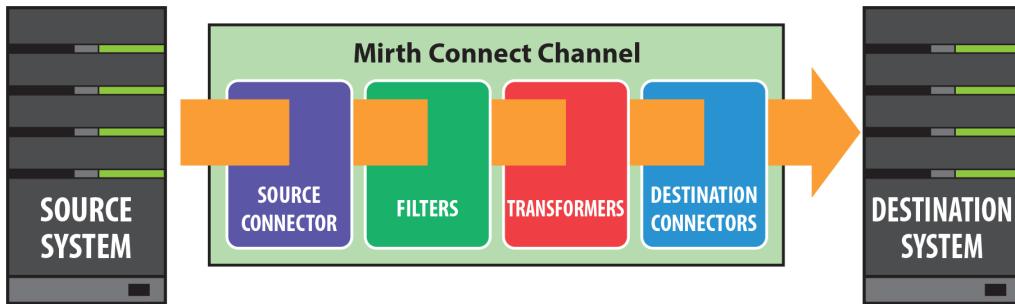
The Fundamentals of NextGen Connect

This section is separated into the following topics:

- [About Channels and Connectors](#)
- [About Message Data](#)
- [The Message Processing Lifecycle](#)
- [About Data Types](#)
- [About Filters](#)
- [About Transformers](#)

About Channels and Connectors

As explained in the [Introduction](#) section, NextGen Connect Integration Engine is an integration engine that can receive data from a variety of sources and take powerful actions on that data, including sending the data out to multiple external systems. It can also transform data from one format to another, or extract pieces of the data that you can act on or send outbound. The interfaces you configure that perform these jobs are called **channels**.



A NextGen Connect Integration Engine channel consists of multiple **connectors**. A connector is a piece of a channel that does the job of getting data into NextGen Connect Integration Engine (a **source connector**), or sending data out to an external system (a **destination connector**). Every channel has exactly one source connector, and at least one destination connector. Because a channel may have multiple destination connectors, it can be thought of as a process that obtains data from a particular source, and sends it out to many different places. For example you may receive data over HTTP, then write the data out to a file somewhere, *and also* insert pieces of the data into your custom database.

Channel Components

General Channel Properties

General Channel properties are configured on the [Summary Tab](#) within the [Edit Channel View](#) and include:

- Unique ID, name, and description.
- Links to [Code Template Libraries](#): These let the channel know which custom functions are available in specific JavaScript contexts.
- Links to [Library Resources](#): These let the channel know which custom Java classes are available in specific connectors and/or JavaScript contexts.
- [Deploy / Start Dependencies](#): These determine which channels are *dependent* on this one, and also which channels are *dependencies* for this one. This way you can have some channels deploy and start before others.
- [Attachment Handler](#): This allows you to extract pieces of any incoming message and store them separately. As a message processes through a channel, multiple copies of it will be held in memory at once (for the raw / transformed / encoded versions of a message). Attachments are stored only once, so by using them you can greatly reduce your channels' memory footprint.
- [Message Storage Settings](#): These determine how much message data to store / retain and whether to encrypt content. These settings affect the performance of the channel and also determine whether you can enable persistent queuing on your connectors.
- [Message Pruning Settings](#): These determine how long to keep message data around before automatically removing it with the [Data Pruner](#). You can also decide to archive data out to a file somewhere before pruning it.
- [Custom Metadata Columns](#): These allow you to extract pieces of data from your message and store them in dedicated columns in the internal NextGen Connect Integration Engine database. You can then view and search on them in the [Message Browser](#).

Source Connector

Every channel has exactly one source connector gets data into NextGen Connect Integration Engine from an external system. The source connector is configured on the [Source Tab](#) within the [Edit Channel View](#). In addition to the standard [Connector Components](#), source connectors include:

- A **Source Queue** that can be enabled or disabled. When enabled, the channel acts as a store-and-forward service that can receive messages and send acknowledgements immediately to the originating system, without having to wait for the message to process through the entire channel.
- A **Batch Processor** that can be enabled or disabled. When enabled, the channel will take any incoming data and split it into multiple messages that each proceed discretely through the channel. More information here: [Batch Processing](#)
- A **Response Selector** that determines what response to send back to the originating system, if applicable. You can choose to auto-generate a response based on the **inbound data type** of the source transformer. You can also return the response from a specific destination, or a completely custom response.
- A **Max Processing Threads** option. By default this is set to 1, meaning that only one message can be processed through a channel at any given time (does not include asynchronous processes like the destination queue). Increasing this setting can greatly improve channel performance / throughput, at the cost of message order preservation.

Destination Connectors

Every channel has at least one destination connector that sends data out to an external system. Destination connectors are configured on the [Destinations Tab](#) within the [Edit Channel View](#). In addition to the standard [Connector Components](#), destination connectors include:

- An **Enabled** flag that determines whether the destination is currently being used. A channel must have at least one destination enabled at any given time.
- A **Wait for previous destination** setting that determines what **chain** a destination connector belongs to. (For additional information, see [Destination Chains](#).)
- A **Response Transformer**. This is like a regular transformer, except it has its own **response inbound data type** and **response outbound data type**, and does the job of modifying the response that an external system returned to a destination connector. It also allows you to decide when to queue / force-error a message. (For additional information, see [Response Transformers](#).)

Channel Scripts

Channel Scripts can be configured on the [Scripts Tab](#) within the [Edit Channel View](#). There are four special scripts associated with a channel:

- **Deploy Script**: This runs once right before a channel is deployed.
- **Preprocessor Script**: This runs once for every message, after the source connector sends a message to the channel and after the [attachment handler](#) has optionally extracted data, but before the message has reached the source filter/transformer. The job of the preprocessor is to modify the incoming message.
- **Postprocessor Script**: This runs once for every message, after the source connector and all destinations have completed (excluding asynchronous processes like the destination queue), but before the source connector sends a response back to the originating system. The postprocessor script has access to responses from all executed destination, and can return custom response that the source connector can use.
- **Undeploy Script**: This runs once right before a channel is undeployed.

Connector Components

General Connector Properties

Every connector has a name and a "metadata ID." For a source connector, the name is always "Source" and the metadata ID is always zero (0). For a destination connector, the name is configurable and the metadata ID is some value greater than zero. The first destination connector in your channel starts at metadata ID one (1), the next one will be two, and so on. Even if you rename a destination connector, the metadata ID will remain the same.

Connector-Specific Properties

Every connector has its own custom set of properties. The properties you configure for a [TCP Listener](#) will be different from a [Database Writer](#) and so on. Here is a list of source and destination connectors supported by NextGen Connect Integration Engine:

- [Source Connectors](#)
 - Channel Reader
 - DICOM Listener
 - Database Reader
 - File Reader
 - HTTP Listener
 - JMS Listener
 - JavaScript Reader
 - TCP Listener
 - Web Service Listener
- [Destination Connectors](#)
 - Channel Writer
 - DICOM Sender
 - Database Writer
 - Document Writer
 - File Writer
 - HTTP Sender
 - JMS Sender
 - JavaScript Writer
 - SMTP Sender
 - TCP Sender
 - Web Service Sender

Additional connectors are made available as [commercial extensions](#):

- [Email Reader](#)
- [Serial Connector](#)
- [NextGen Results CDR Connector](#)

Filter

A filter is the piece of a connector that decides whether a message should proceed to the next step or not. It is configured on the [Edit Filter View](#) within either the [Source Tab](#) or [Destinations Tab](#) within the [Edit Channel View](#). For additional information, see [About Filters](#).

Transformer

A transformer is the piece of a connector that modifies a message, converts a message from one format to another, and extracts pieces of the message for later use. It is configured on the [Edit Transformer View](#) within either the [Source Tab](#) or [Destinations Tab](#) within the [Edit Channel View](#). For additional information, see [About Transformers](#).

All transformers have an **inbound data type** and an **outbound data type** used to determine how data is parsed and converted. For additional information on data types, see [About Data Types](#)

About Message Data

In Mirth Connect, a **message** refers to a single overall dispatch of data through the source connector and destination connectors within a channel. Messages are further separated into **connector messages** which is the piece of the message that flows through a single connector. For example if your channel has two destinations, then a single message as it processes through your channel will have three connector messages associated with it: One for the source connector, and two more for the two destination connectors. These connector messages correspond to the rows of data you see in the [Metadata Table](#) within the [Message Browser](#).

Note that messages don't always correspond 1-to-1 with a file that you read in, or a particular stream of data. As explained in the [About Channels and Connectors](#) section, the source connector may contain a batch processor which takes a raw inbound stream of data and splits it into multiple messages. So if you have a single file containing 100 HL7 v2.x messages, your channel could read that in and process 1 message or 100 messages, depending on how the source connector is configured.

Message Metadata

Metadata typically refers to important information about the message, but not the actual message content. Most important is the **message ID**. For any given channel, every message has a unique integer ID associated with it. This ID is used to organize data in the message browser and join connector messages together.

Each connector has its own ID, referred to as the **connector metadata ID**. The source connector always has a metadata ID of zero (0). Destination connector metadata IDs start at one (1) and increments for each new destination you add to the channel.

Each connector message has a **status**, which tells the channel the current processing state. For example, the status could be *RECEIVED*, which means the raw data has been committed to the database, but the message is in the middle of being processed. It could be *QUEUED*, which may mean it's sitting in a queue waiting to be processed, or it has been attempted to be processed one or more times but hasn't yet been successful.

Metadata also includes important timestamps which you can use to analyze and diagnose issues. Every connector message stores a **received date**, which is the time at which its data was committed to the database. For destination connectors, the **send date** and **response date** let you know the time a message was dispatched outbound, and the time a response was received from the external system. The differences between these timestamps can give insights into your channel performance.

This also includes **custom metadata columns**. These are configurable from the [Summary Tab](#) within the [Edit Channel View](#), and allow you to create your own columns that show up in the [Metadata Table](#) and are searchable in the [Message Browser](#).

All of this information and more is visible in the [Metadata Table](#) within the [Message Browser](#).

Message Content

Message content is the actual data that gets processed. As a message flows through your channel, different versions of the data are stored for each connector, depending on the modifications your channel needs to make.

Source Connector

- **Raw** – The state of the message as it enters the connector.
- **Processed Raw** – The state of the message after passing through the preprocessor script.
- **Transformed** – The serialized internal representation of the message, which exists only if a connector has a filter or transformer configured.
- **Encoded** – The state of the message as it exits the transformer (includes changes made to the transformed data).
- **Response** – The message sent back to the originating system (at the very end, after all destinations finish).

Destination Connector

- **Raw** – The state of the message as it enters the connector. For a destination connector, this is the same as the source encoded data.
- **Transformed** – The serialized internal representation of the message, which exists only if a connector has a filter or transformer configured.
- **Encoded** – The state of the message as it exits the transformer (includes changes made to the transformed data).
- **Sent** – The message/connector properties used by the destination connector to send messages to the outbound system.
- **Response** – The message received from the outbound system after the destination sends the message.
- **Response Transformed** – The serialized internal representation of the response, which exists only if a destination connector has a response transformer configured.
- **Processed Response** – The state of the response as it exits the response transformer (includes changes made to the transformed data).

These pieces of content are specific to each individual connector message. So a source connector will have Raw / Transformed / Encoded data, and each destination connector will have its own Raw / Transformed / Encoded data.



When a message flows from the source connector to the destination connectors, the Encoded Data from the source becomes the Raw Data for each destination. **However**, Raw Data is not "daisy-chained" from destination to destination. If you have three destinations, the Raw Data for each and every destination will be identical to the Encoded Data from the source connector message.

More additional information on the various content types, see [Message Content Types, Variable Maps, Error Content Types](#).

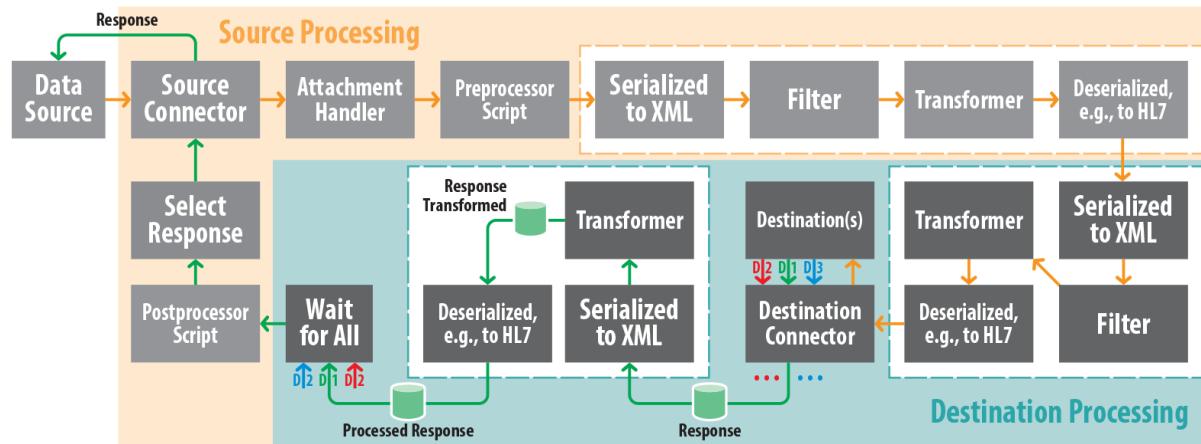
Message Attachments

An attachment is a piece of data extracted from the raw incoming message and stored separately. Attachments are not associated with connector messages, but instead with the overall message. The extraction happens at the very beginning of the message lifecycle, even before the preprocessor script runs. When a destination connector dispatches data outbound, any attachments associated with the message will be automatically re-inserted into the outgoing data. In this way, attachment data is only stored once, and multiple copies of it for each connector and for each content type (e.g. Raw / Transformed / Encoded) will not be stored. Using attachments can greatly improve the memory footprint of your channels.

More additional information on attachments and how they are extracted, see [Attachment Handlers](#).

The Message Processing Lifecycle

As explained in the [About Message Data](#) section, each source and destination connector has various versions of the message data as it flows through the channel. The *raw* inbound message enters NextGen Connect Integration Engine, is received by the source connector, is filtered and *transformed*, then *encoded* and sent to the destination connector. From the source connector, the raw inbound message can be passed through multiple destination connectors where it can again be influenced by filters and transformers before it is processed, encoded, and sent on.



Source Processing Steps

1. A message or stream of data is received by the source connector.
2. The batch processor decides whether to split the message into multiple messages. If so, the below steps are repeated for each message returned by the batch processor.
3. The attachment handler extracts and/or stores any attachment data.
4. The post-attachment-handler content (after any attachments have been extracted) is stored as *raw* data.
5. The content runs through the preprocessor script and is stored as *processed raw* data.
6. The content is serialized (converted) to the internal representation of the **inbound data type** (e.g. XML).
7. The content runs through the filter, and the message is either accepted or filtered. If the message gets filtered, flow stops here and jumps down to the **Final Processing Steps**.
8. The content runs through the transformer, where it may be modified.
9. The post-transformer content is stored as the *transformed* data.
10. The content is deserialized (converted) from the **outbound data type's** internal representation (e.g. XML) into the actual outbound format (HL7, EDI, etc.).
11. The content is stored as *encoded* data.

⚠️ If no filter or transformer are configured, there will not be any transformed or encoded data. In that case, the content used from here on will just be the raw data (or processed raw data if a preprocessor modified it).

12. The resulting content is passed on to the first destination connector of each [destination chain](#).

Destination Processing Steps



These steps are repeated for each destination connector that a message flows through.

1. The encoded content from the source connector is used by each destination connector as its *raw data*.
2. The content is serialized (converted) to the internal representation of the **inbound data type** (e.g. XML).
3. The content runs through the filter, and the message is either accepted or filtered. If the message gets filtered, flow stops for the current destination here. If there are additional destinations in the current **chain**, these steps are repeated for the next destination. Otherwise, assuming all other destination chains have finished, flow jumps down to the [Final Processing Steps](#).
4. The content runs through the transformer, where it may be modified.
5. The post-transformer content is stored as the *transformed data*.
6. The content is deserialized (converted) from the **outbound data type's** internal representation (e.g. XML) into the actual outbound format (HL7, EDI, etc.).
7. The content is stored as *encoded data*.



If no filter or transformer are configured, there will not be any transformed or encoded data. In that case, the content used from here on will just be the raw data.

8. The destination connector builds a message from all available previous content, stores it as *sent data*, and sends it to the outbound system.
9. A response is received by the destination connector and stored as the *response data*.
10. If a response transformer is configured, the response content is serialized (converted) to the internal representation of the **response inbound data type** (e.g. XML).
11. The response content runs through the response transformer, where it may be modified.
12. The post-response-transformer content is stored as the *response transformed data*.
13. The response content is deserialized (converted) from the **response outbound data type's** internal representation (e.g. XML) into the actual outbound format (HL7, EDI, etc.).
14. The response content is stored as *processed response data*.

Final Processing Steps

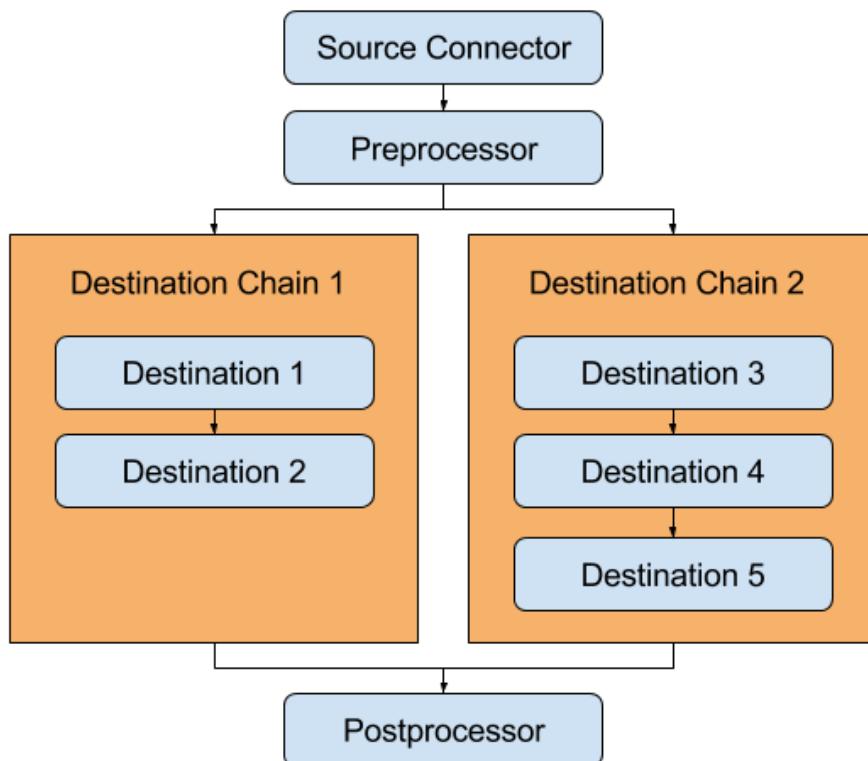


This point is reached when all destination chains have finished processing the message. If the message was filtered or errored out on the source connector, then flow will immediately jump down here.

1. The postprocessor script is executed. It can return a response that the source connector may use.
2. The source connector decides what response to send back to the originating system, if any. This may be an auto-generated value, the response payload from a destination connector dispatch, a response returned from the postprocessor script, or a completely custom value residing in the response map.
3. The selected response is stored as the source connector's *response data* and is sent back to the originating system, if needed.

Destination Chains

A channel's destinations are grouped into one or more **destination chains**. Each destination chain processes simultaneously with respect to each other, however in any particular chain, a message will flow through each destination *in order*. It looks something like this:



In the above example, there are 5 total destinations. However **Destination 3** does not wait on **Destination 2**, so it marks the beginning of a new chain. If each destination takes 1 second to process, then the overall time it takes the message to process through the channel will *not* be 5 seconds, but rather 3 seconds. When flowing through the destination connectors, a message will take only as long as the longest destination chain.

About Data Types

A **data type** tells a filter / transformer how to parse a certain format. The **inbound** data type *serializes* the incoming **raw data** into its internal representation. Then the filter / transformer executes, possibly modifying this internal representation, or even completely overwriting it with a different internal representation. Finally, the **outbound** data type takes this **transformed data** and *deserializes* it into the proper outbound format (the **encoded data**).

For example, the [HL7 v2.x data type](#) serializes ER7 data like this:

```
MSH|^~\&|ADT1|SHM|SHMADT|SHM|200812091126|SECURITY|ADT^A01^ADT_A01|MSG000  
EVN|A01|200812091126||  
PID|1|1001|1001^5^M11^ADT1^MR^SHM~123456789^^^USSSA^SS||OHALLAHAN^COLLEEN  
NK1|1|OHALLAHAN^BRITTANY^M|SIS^SISTER|||N^NEXT-OF-KIN  
PV1|1|I|2000^2012^01||||001122^ZOIBERG^JOHN^|||SUR||||1|A0|
```

Into this:

```
<HL7Message>  
  <MSH>  
    <MSH. 1>|</MSH. 1>  
    <MSH. 2>^~\&|</MSH. 2>  
    <MSH. 3>  
      <MSH. 3. 1>ADT1</MSH. 3. 1>  
    </MSH. 3>  
    <MSH. 4>  
      <MSH. 4. 1>SHM</MSH. 4. 1>  
    </MSH. 4>  
    <MSH. 5>  
      <MSH. 5. 1>SHMADT</MSH. 5. 1>  
    </MSH. 5>  
    <MSH. 6>  
      <MSH. 6. 1>SHM</MSH. 6. 1>  
    </MSH. 6>  
    <MSH. 7>  
      <MSH. 7. 1>200812091126</MSH. 7. 1>  
    </MSH. 7>  
    <MSH. 8>  
      <MSH. 8. 1>SECURITY</MSH. 8. 1>  
    </MSH. 8>  
    <MSH. 9>  
      <MSH. 9. 1>ADT</MSH. 9. 1>  
      <MSH. 9. 2>A01</MSH. 9. 2>  
      <MSH. 9. 3>ADT_A01</MSH. 9. 3>  
    </MSH. 9>  
    <MSH. 10>  
      <MSH. 10. 1>MSG00001</MSH. 10. 1>  
    </MSH. 10>  
    <MSH. 11>  
      <MSH. 11. 1>P</MSH. 11. 1>  
    </MSH. 11>  
    <MSH. 12>  
      <MSH. 12. 1>2.5</MSH. 12. 1>  
    </MSH. 12>  
    <MSH. 13/>
```

In this case, the internal representation is XML. That's because for HL7 v2.x, the message object you use or manipulate in a filter / transformer is an E4X XML object. However not all data types use XML for their internal representation. The JSON data type uses JSON as you might expect, and the object you use in a filter / transformer is just a regular JavaScript Object. The Raw data type does no serialization or deserialization, so its internal representation is identical to the inbound message, and the object used in the filter / transformer is just a Java String.

For additional information, see [Data Types](#). NextGen Connect Integration Engine supports the following data types:

- [Delimited Text Data Type](#)
- [DICOM Data Type](#)
- [EDI / X12 Data Type](#)
- [HL7 v2.x Data Type](#)
- [HL7 v3.x Data Type](#)
- [JSON Data Type](#)
- [NCPDP Data Type](#)
- [Raw Data Type](#)
- [XML Data Type](#)
- [Batch Processing](#)
- [JavaScript Batch Script](#)

The following additional data type is made available as a commercial extension: [ASTM E1394 Data Type](#)

About Filters

The filter is the piece of a connector that decides whether a message should proceed to the next step or not. It is configured on the [Edit Filter View](#) within either the [Source Tab](#) or [Destinations Tab](#) within the [Edit Channel View](#).

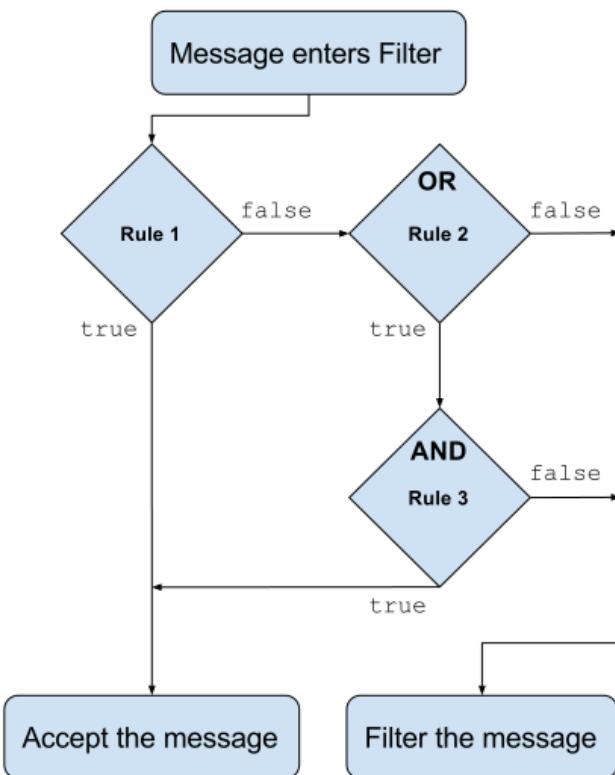
A filter returns **true** or **false**. When the filter returns true, the message is said to have been *accepted*. When the filter returns false, the message is said to have been *filtered*.

- If the [source connector](#) filters out a message, it will not flow through the source transformer, and will not be processed by any of the destination connectors.
- If a [destination connector](#) filters out a message, it will not flow through the destination transformer, and will not be dispatched outbound by that destination connector. However *other* destinations may still process the message.

A filter is comprised of multiple **rules**. Each rule is joined together with an **operator**, which can be *AND* or *OR*. For example a filter may look like this:

- Accept the message if: Rule 1 returns true *OR* Rule 2 returns true *AND* Rule 3 returns true

The standard order-of-operations means that *AND* takes logical precedence over *OR*, like this:



The "msg" Object

In order to decide whether a message needs to be filtered or not, you will typically need to test pieces of the incoming message. As mentioned in the [About Data Types](#) section, when the message enters a filter / transformer, it gets serialized into an internal representation. This is the variable **msg**, which may be an E4X XML Object, a JavaScript Object, or a Java String, depending on the data type implementation.

Filter Rule Types

NextGen Connect Integration Engine supports the following filter rule types:

- Rule Builder Filter Rule
- JavaScript Filter Rule
- External Script Filter Rule
- Iterator Filter Rule

About Transformers

A transformer is the piece of a connector that modifies a message, converts a message from one format to another, and extracts pieces of the message for later use. It is configured on the [Edit Transformer View](#) within either the [Source Tab](#) or [Destinations Tab](#) within the [Edit Channel View](#).

A transformer has an **inbound data type**, and an **outbound data type**. These may be the same (e.g. HL7 v2.x to HL7 v2.x), or they could be different (e.g. HL7 v2.x to JSON). For additional information on data types, see [About Data Types](#).

A transformer is also comprised of multiple **steps**. Each transformer step modifies the message, extracts a piece of the message, or performs some other general task.

The "msg" Object

This is the same as in a filter. For more information, see [About Filters](#).

The "tmp" Object

This is similar to **msg**, except that it is the internal representation of the **outbound template** configured in your [transformer settings](#). It will only be available in the transformer when you have an outbound template configured.

The **tmp** variable is used when you want to convert a message from one format to a completely different format (e.g. HL7 v2.x to JSON). Or, it can be used to selectively include pieces of the incoming message and map them into the outbound message.

Response Transformers

The Response Transformer is a special type of transformer specific to destination connectors. It works the same as a regular transformer, except that the data being transformed is not the message flowing through the channel, but instead the response payload that the destination connector received from the external system (if applicable). For additional information, see [Response Transformers](#)

A destination response is comprised not only of the response data, but also the **status** (e.g. SENT, ERROR), **status message**, and **error message**. Response transformers can be used to modify these latter pieces as well. For example if a message gets set to ERROR by the destination connector, in the response transformer you can choose to override that and set the status to SENT instead based on some custom logic.



Response transformers will only execute if there is an actual response payload to transform. For example if you are using an [HTTP Sender](#) destination and it fails to connect to the remote server, then obviously there is no response payload. The one exception to this rule is if the response inbound data type is set to [Raw](#). In that case, because the Raw data type does not need to perform any serialization, the response transformer will always execute even if there is no response payload.

Transformer Step Types

NextGen Connect Integration Engine supports the following transformer step types:

- [Mapper Transformer Step](#)
- [Message Builder Transformer Step](#)
- [JavaScript Transformer Step](#)
- [External Script Transformer Step](#)
- [XSLT Transformer Step](#)
- [Destination Set Filter Transformer Step](#)
- [Iterator Transformer Step](#)

The Administrator Launcher

Introduction / Installation

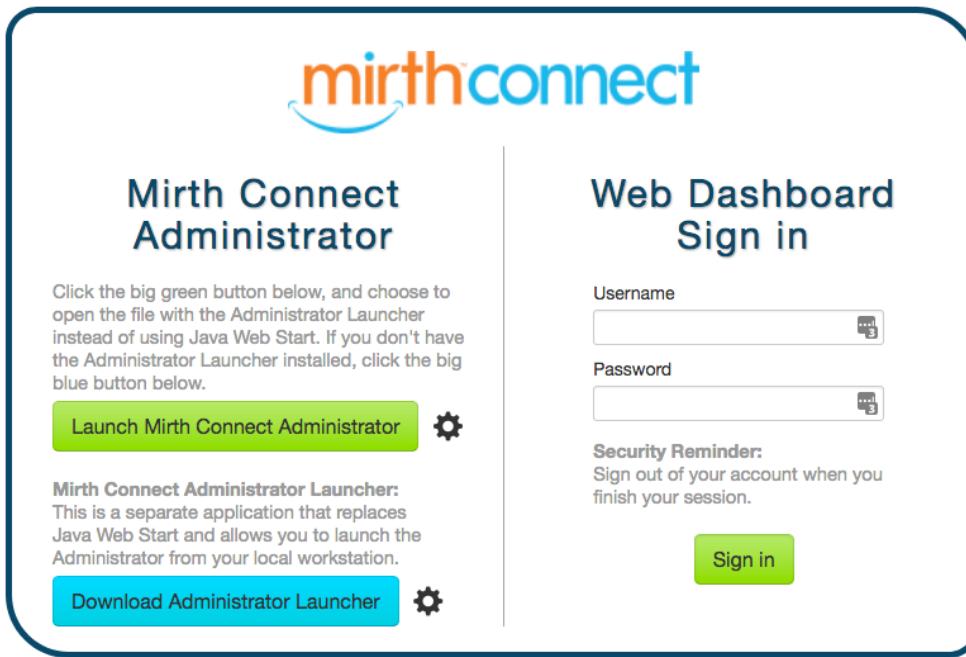
The Administrator is the graphical user interface (GUI) that is used to login to a NextGen Connect server and create /modify/deploy channels. In previous versions this was launched on your local desktop with Java Web Start. However in Java 11 and later, Java Web Start was completely removed from Java.

The Administrator Launcher is a replacement for Java Web Start, and can be used to launch the Administrator on your local desktop. It can be used regardless of which version of NextGen Connect you're using or which version of Java it's running on.

Download

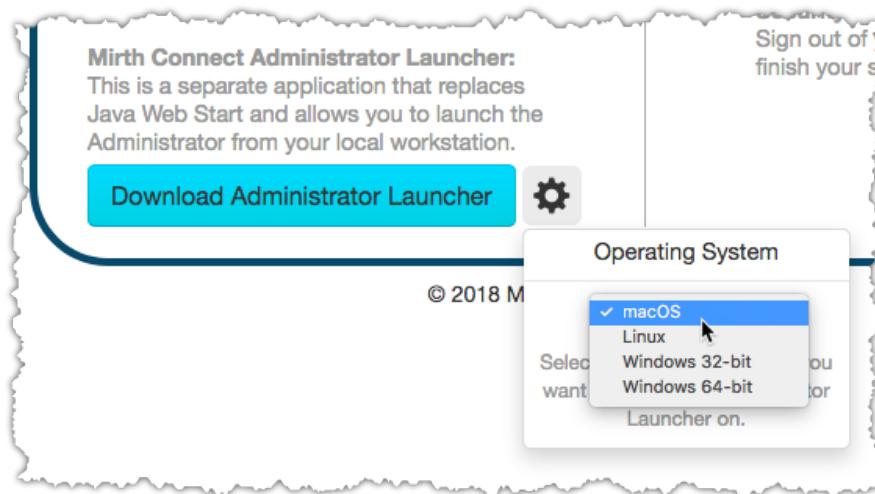
The Administrator Launcher comes bundled with your Connect installation. When you go through the graphical installer for Connect itself, you are given the option to also install the Administrator Launcher at the same time.

However you may also want to install the Administrator Launcher on your *local*/workstation. In a web browser, navigate to your main 8080/8443 launch page. For example, <http://localhost:8080>. Note that your IP/DNS or port may be different depending on your mirth.properties configuration.



© 2018 Mirth Corporation | Mirth Connect

Click on the cog icon next to **Download Administrator Launcher** to select the operating system you're using on your local system:



Finally, click the blue **Download Administrator Launcher** button to download the installer.

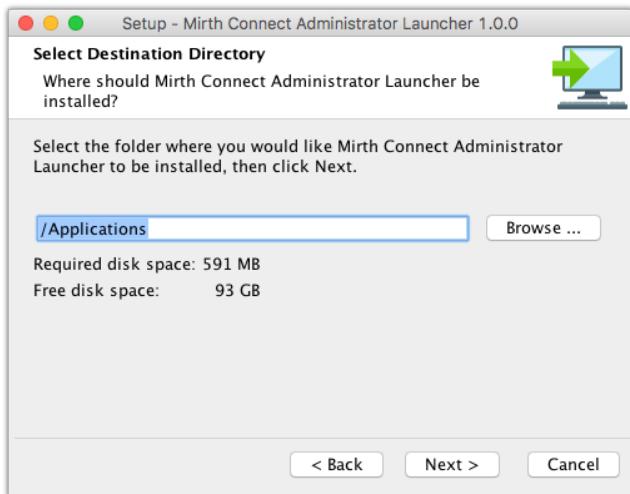
i The installers for the Administrator Launcher are also available separately on our public downloads page: <https://www.mirth.com/Downloads>

i When downloading the Administrator Launcher, the page first looks to see if you have an installer file in public_html/installers. If none exists the button will simply point to the public download link. If you're on a network with no internet access, you can still allow downloading the Administrator Launcher by placing the appropriate installer files in public_html/installers.

Installation



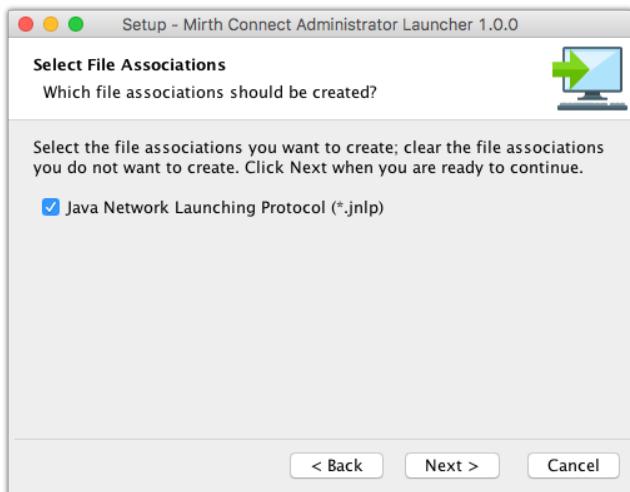
Follow the prompt and click **Next** to start. You'll be asked to view and accept an end-user license agreement, and then you'll be asked where you want to install the Administrator Launcher to:



Next, you'll be asked whether you want to associate JNLP files with the Administrator Launcher. In previous versions Java Web Start would have been used, but since that has now been removed from Java, it's recommended to check this box.



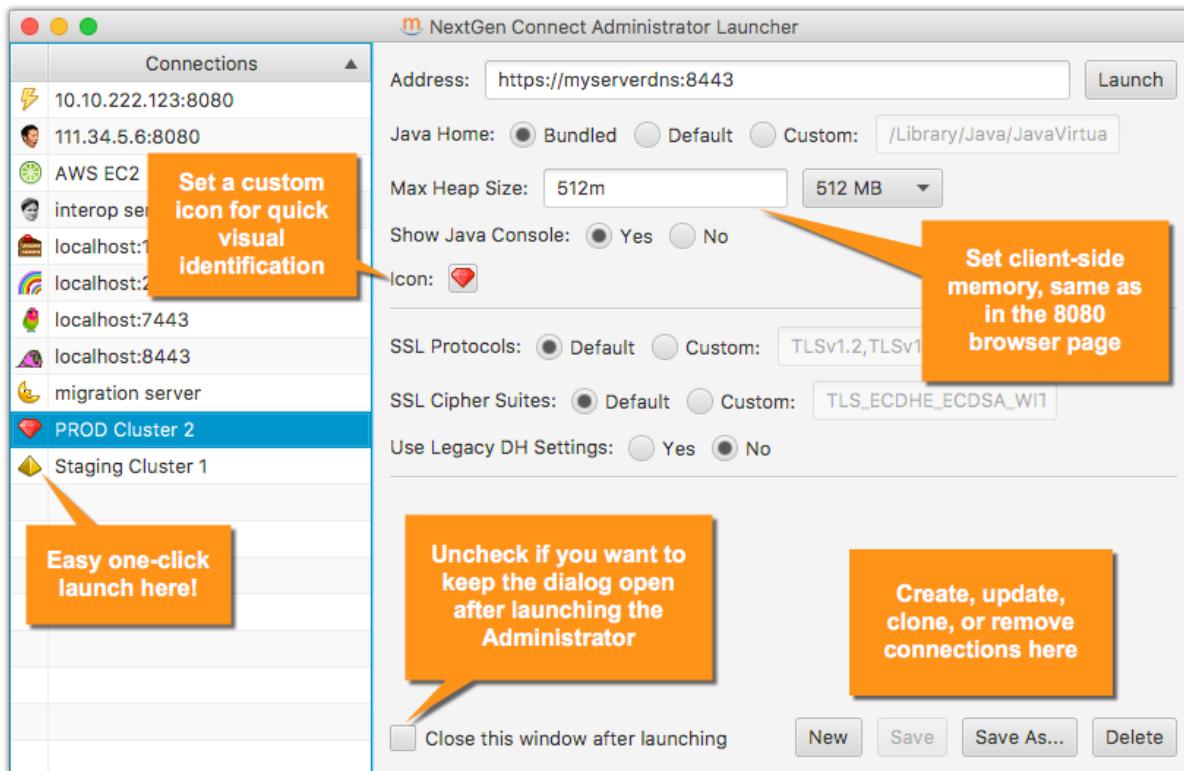
Note: This will update your operating system file preferences, but you may still want to update your web browser file associations to open JNLP files with the Administrator Launcher.



After clicking **Next**, the installation process will run, and you can choose to start the Administrator Launcher immediately after exiting the installer.



The Launcher Interface



On the left-hand side you see all your currently saved connections. You can click on the name to select the connection, and double-click to edit its name. You can also launch the Administrator immediately by clicking the icon next to the connection name.

On the right-hand side you see the current connection settings:

Field	Default Value	Description
Address		The address of your Connect server. This may be either the plain HTTP (8080) or HTTPS (8443) address.
Java Home	Bundled	The Administrator Launcher comes bundled with JRE 8. By default the Administrator will launch using this bundled JRE. You can also choose to launch with the system default JRE, or a specific Java installation.
Max Heap Size	512m	The maximum amount of memory to allocate for the Administrator Java virtual machine.
Show Java Console	No	If enabled, a separate console dialog will pop-up along with the Administrator. You can use this console to see exception stacktraces and other system information.
Icon		You can use the icon to help you quickly identify different connections in the left-hand table.

SSL Protocols	Default	The SSL protocols to use when connecting to the server.
SSL Cipher Suites	Default	The SSL cipher suites to use when connecting to the server.
Use Legacy DH Settings	No	Enables legacy Diffie-Hellman settings that allow smaller key sizes. This may be needed when connecting to older 2.x versions of Connect.

At the bottom there are buttons you can use to create a **New** connection, **Save** the currently selected connection, save the current connection settings as a new connection (**Save As...**), or delete the currently selected connection.

Mirth Connect Administrator

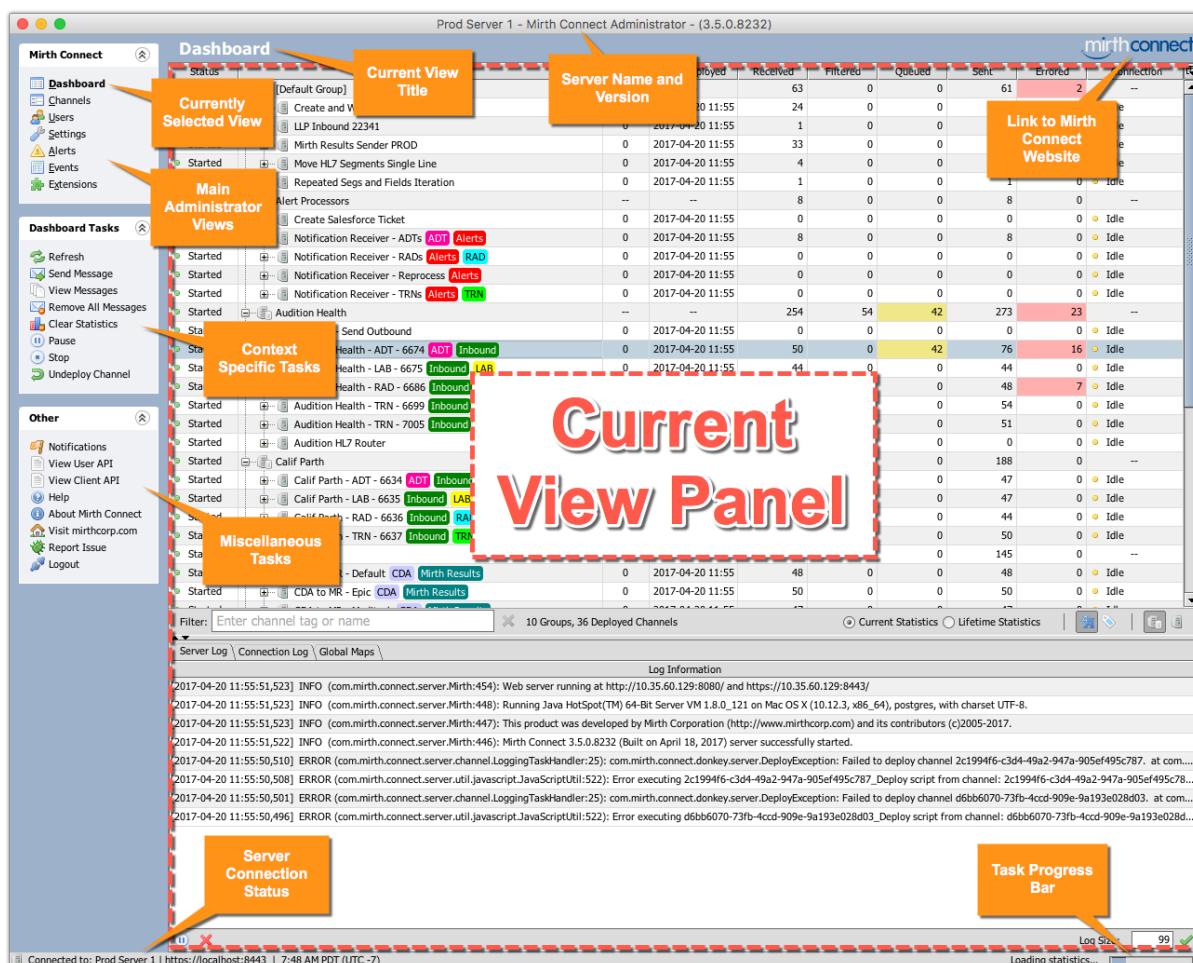
The Mirth Connect Administrator is the graphical user interface (GUI) that is used to login to a NextGen Connect Integration Engine server and create/modify/deploy channels. It is launched via the [Administrator Launcher](#) and can be accessed from the main launch page in a browser. For additional information on launching the Administrator, see [Launching the Mirth Connect Administrator](#).

This section is separated into the following topics:

- [Administrator Layout](#)
- [Working With Tables](#)
- [Monitoring Views](#)
- [Management Views](#)
- [Editing Views](#)
- [Other Tasks](#)

Administrator Layout

Main view selections and context-specific tasks are located on the left-hand side of the window. On the right side is the currently selected view. After you login to the Administrator, you'll first be taken to the [Dashboard View](#).

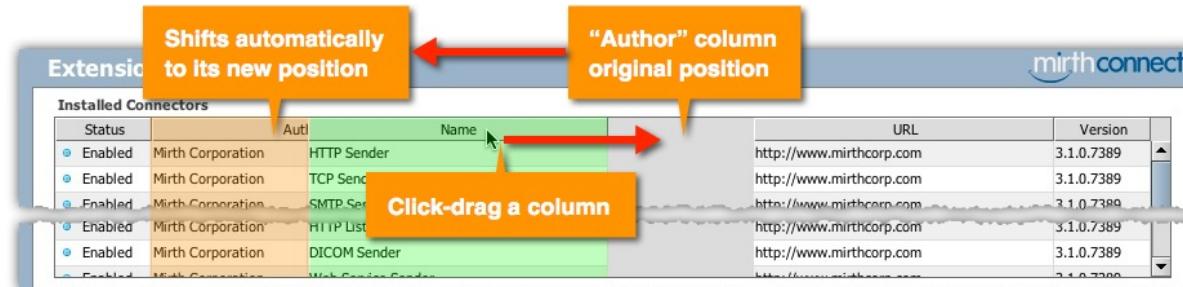


Working With Tables

Many of the tables throughout the Mirth Connect Administrator share the same general control scheme. Common operations include:

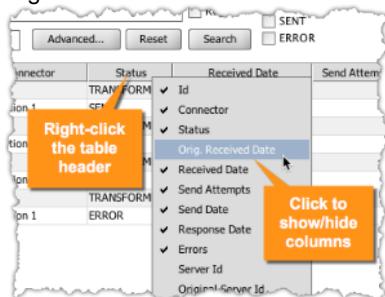
Arranging Columns

- Click and hold the header of the column you'd like to move.
- Drag the column to its new location. As you drag a column, the next columns automatically slide in the opposite direction, to make room for the column you are moving.

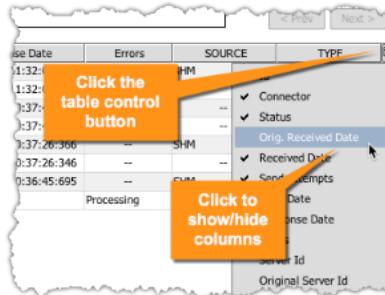


Showing / Hiding Columns

- Right-click the table header:



- OR, click the table control button in the top-right portion of the table:



Sorting Tables By Column

Not all tables support sorting, but for the ones that do:

- Click one of the table headers to sort ascendingly.

- Click the same table header a second time to sort descendingly:

Received	Filtered	Queued	Sent	Errored	C
266	0	0	266	0	
63	0	0	61	2	
38	0	0	35	9	
36	0	0	32	0	
28	0	0	28	0	
26	0	0	26	0	
1	0	0	9	0	Id
1	0	0	6	0	Id

Expanding / Collapsing Rows

- Click the plus/minus icons:

- OR, right-click the table header:

[Default Group]	Name	Id
Alert Processors	8-4969-ace4-3cb2da79b	775
Notification Receiver - ADT	5608	661
Notification Receiver - RAD	ec1	ec2
Notification Receiver - TRN	-40c4-a66-d2d8f546f	-43ef-9546-c12edc1d
Audition Health	z-4cc8-bd15-c7fd2154	z-4eb2-9f6f-3338c0011
Audition Health - ADT - 6	e-41be-ef4-744bd2a189	a-4cd7-9fe8-600423077
Audition Health - LAB - 6	ad7b5446-6064-4eeb-8214-929e6509	
Audition Health - RAD - 6	ad7b5446-6064-4eeb-8214-929e6509	
Audition Health - TRN - 6	ad7b5446-6064-4eeb-8214-929e6509	
Calif Parth	ad7b5446-6064-4eeb-8214-929e6509	
Calif Parth - ADT - 6634	ad7b5446-6064-4eeb-8214-929e6509	

Multi-selecting Rows

Not all tables support multi-row selection, but for the ones that do:

- Click one of the rows in the table.
- To select contiguous rows, hold down the **Shift** key and select a different row.
 - While keeping the **Shift** held down, press the **Up/Down** arrow keys to select additional rows.

- To select discontiguous rows, hold down the **Ctrl** (or **&**) key and select a different row.

Notification Receiver - TRNs	Alerts	dafcfda
Audition Health		0c1e0c9
Audition Health - ADT - 6674	ADT	In 582ca89
Audition Health - LAB - 6675	Inbound	83834b
Audition Health - RAD - 6686	Inbound	36a1caf
Audition Health - TRN - 6699	Inbound	d4d0388
Audition Health - TRN - 7005	Inbound	dd124a9
Calif Parth		ad7b54
Calif		a57b2c1
Calif	AE	58db4at
Calif	RA	c28a520
Calif	TR	9d621b
Mirth Resu		87c8970
CDA to MR - Default	CDA	Mirth Resu 513454

Searching Table Content

You can quickly scroll the table to a particular row by searching on a particular string.

- Press **Ctrl+F** (or **+F**) to open the Find Dialog.

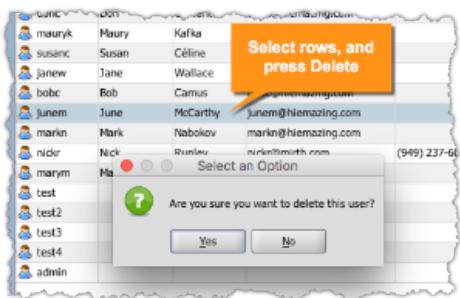
Status	Received Date	Send Attempts	Send Date	Response Date	Errors	SOURCE	
TRANSFORMED	2017-04-10 08:24:28:061	1	--	2017-04-10 08:24:28:070	--	SHM	
SENT	2017-04-10 08:24:28:064	1	2017-04-10 08:24:28:066	2017-04-10 08:24:28:066	--	SHM	
TRANSFORMED	2017-04-10 08:24:24:160	1	--	2017-04-10 08:24:24:168	--	SHM	
SENT	2017-04-10 08:24:24:162	1	2017-04-10 08:24:24:165	2017-04-10 08:24:24:165	--	SHM	
TRANSFORMED	2017-04-10 08:24:24:14:	1	--	2017-04-10 08:24:14:250	--	Hospital B	
SENT	2017-04-10 08:24:14:2	1	--	2017-04-10 08:24:14:247	--	Hospital B	
TRANSFORMED	2017-04-10 08:24:12:2	1	--	2017-04-10 08:24:12:248	--	Sacred Heart	
SENT	2017-04-10 08:24:12:2	1	--	2017-04-10 08:24:12:244	--	Sacred Heart	
TRANSFORMED	2017-04-10 08:24:10:2	1	--	2017-04-10 08:24:10:776	--	LAMH	
SENT	2017-04-10 08:24:10:2	1	--	0:773	2017-04-10 08:24:10:773	--	LAMH

- Click the **Find** button. If a matching row is found, it will be selected and scrolled to.

Status	Received Date	Send Attempts	Send Date	Response Date	Errors	SOURCE	
TRANSFORMED	2017-04-10 08:24:28:061	1	--	2017-04-10 08:24:28:070	--	SHM	
SENT	2017-04-10 08:24:28:064	1	2017-04-10 08:24:28:066	2017-04-10 08:24:28:066	--	SHM	
TRANSFORMED	2017-04-10 08:24:24:160	1	--	2017-04-10 08:24:24:168	--	SHM	
SENT	2017-04-10 08:24:24:162	1	2017-04-10 08:24:24:165	2017-04-10 08:24:24:165	--	SHM	
TRANSFORMED	2017-04-10 08:24:24:14:	1	--	2017-04-10 08:24:14:250	--	Hospital B	
SENT	2017-04-10 08:24:14:2	1	--	2017-04-10 08:24:14:247	--	Hospital B	
TRANSFORMED	2017-04-10 08:24:12:2	1	--	2017-04-10 08:24:12:248	--	Sacred Heart	
SENT	2017-04-10 08:24:12:2	1	--	2017-04-10 08:24:12:244	--	Sacred Heart	
TRANSFORMED	2017-04-10 08:24:10:2	1	--	2017-04-10 08:24:10:776	--	LAMH	
SENT	2017-04-10 08:24:10:2	1	--	0:773	2017-04-10 08:24:10:773	--	LAMH

Deleting Rows

Not all tables support row deletion. But for the many that do, you can simply select the row(s) you wish to delete, and press the **Delete** button:



Monitoring Views

This section is separated into the following topics:

- Dashboard View
- Message Browser View
- Alerts View
- Events View

Dashboard View

This is the default view you will see after logging into the Mirth Connect Administrator. The dashboard gives you an overall view of all your deployed channels, and also allows you to start/stop specific channels and connectors.

Status	Name	Rev Δ	Last Deployed	Received	Filtered	Queued	Sent	Errorred	Connection
Started	[Default Group]	--	--	63	0	0	61	2	--
Started	Create and Write PDF	0	2017-04-12 07:22	24	0	0	24	0	Idle
Started	LLP Inbound 22341	0	2017-04-12 07:22	1	0	0	0	1	Idle
Started	Mirth Results Sender PROD	0	2017-04-12 07:22	33	0	0	33	0	Idle
Started	Move HL7 Segments Single Line	0	2017-04-12 07:22	4	0	0	3	1	Idle
Started	Repeated Segs and Fields Iteration	0	2017-04-12 07:22	1	0	0	1	0	Idle
Started	Alert Processors	--	--	8	0	0	8	0	--
Started	Create Salesforce Ticket	0	2017-04-12 07:22	0	0	0	0	0	Idle
Started	Notification Receiver - ADTS	ADTS	Alerts	8	0	0	8	0	Idle
Started	Notification Receiver - RAD	RAD	RAD	0	2017-04-12 07:22	0	0	0	Idle
Started	Notification Receiver - Reprocess	Alerts	Alerts	0	2017-04-12 07:22	0	0	0	Idle
Started	Notification Receiver - TRNS	TRNS	TRNS	0	2017-04-12 07:22	0	0	0	Idle
Started	Audition Health	--	--	253	53	42	272	23	--
Started	Audition Health - ADT - 6674	ADT	Inbound	50	0	42	76	16	Idle
Started	Audition Health - LAB - 6675	Inbound	LAB	0	2017-04-12 07:22	44	0	44	Idle
Started	Audition Health - RAD - 6686	Inbound	RAD	0	2017-04-12 07:22	55	0	48	7 Idle
Started	Audition Health - TRN - 6699	Inbound	TRN	0	2017-04-12 07:22	53	53	0	Idle
Started	Audition Health - TRN - 7005	Inbound	TRN	0	2017-04-12 07:22	51	0	51	Idle
Started	Calif Parth	--	--	188	0	0	188	0	--
Started	Calif Parth - ADT - 6634	ADT	Inbound	47	0	47	0	0	Idle
Started	Calif Parth - LAB - 6635	Inbound	LAB	0	2017-04-12 07:22	47	0	47	Idle
Started	Calif Parth - RAD - 6636	Inbound	RAD	0	2017-04-12 07:22	44	0	44	Idle
Started	Calif Parth - TRN - 6637	Inbound	TRN	0	2017-04-12 07:22	50	0	50	Idle
Started	Mirth Results	--	--	145	0	0	145	0	--
Started	CDA to MR - Default	CDA	Mirth Results	0	2017-04-12 07:22	48	0	48	Idle
Started	CDA to MR - Epic	CDA	Mirth Results	0	2017-04-12 07:22	50	0	50	Idle
Started	CDA to MR - Meditech	CDA	Mirth Results	0	2017-04-12 07:22	47	0	47	Idle
Stopped	Routers	--	--	1,883	0	0	1,883	0	--

Filter: Enter channel tag or name 9 Groups, 36 Deployed Channels Current Statistics Lifetime Statistics Log Information

Log Information

```
[2017-04-12 07:23:00,210] INFO (com.mirth.connect.server.Mirth:454): Web server running at http://10.35.60.129:8080 and https://10.35.60.129:8443
[2017-04-12 07:23:00,209] INFO (com.mirth.connect.server.Mirth:448): Running Java HotSpot(TM) 64-Bit Server VM 1.8.0_121 on Mac OS X (10.12.3, x86_64), postgres, with charset UTF-8.
[2017-04-12 07:23:00,209] INFO (com.mirth.connect.server.Mirth:447): This product was developed by Mirth Corporation (http://www.mirthcorp.com) and its contributors (c)2005-2017.
[2017-04-12 07:23:00,207] INFO (com.mirth.connect.server.Mirth:446): Mirth Connect 3.5.0.8231 (Built on April 7, 2017) server successfully started.
```

Connected to: Prod Server 1 | https://localhost:8443 | 8:53 AM PDT (UTC-7) Log Size: 99 ✓

Navigation

To reach the Dashboard View from a different location, click on the **Dashboard** link in the task panel in the upper-left:



This section is separated into the following topics:

- Dashboard Table
- Filtering By Channel Name or Tag
- Server Log
- Connection Log
- Global Maps
- Dashboard Tasks

Dashboard Table

This is the main section of the dashboard that shows the current status and statistics for your deployed channels. For general information about working with tables in NextGen Connect Integration Engine, see [Working With Tables](#).

The screenshot shows a table with the following columns: Status, Name, Rev Δ, Last Deployed, Received, Filtered, Queued, Sent, Errored, and Connection. The table lists 33 deployed channels across 9 groups. Annotations provide details on various UI elements:

- Click on headers to sort:** Points to the column headers.
- Table Controls:** Points to the top right corner of the table area.
- Expand / collapse channel groups:** Points to the group expand/collapse icons.
- Filter by channel name / tags:** Points to the filter input field and the currently selected tags.
- Currently filtered channels / groups:** Points to the list of currently filtered items.
- Toggles how channel tags are shown in the table:** Points to the 'Current Statistics' radio button.
- Toggles group-level view:** Points to the 'Lifetime Statistics' radio button.

At the bottom left, there is a filter input field: "Filter: Enter channel tag or name" and a status indicator: "9 Groups, 33 Deployed Channels". At the bottom right, there are radio buttons for "Current Statistics" and "Lifetime Statistics", along with other table control icons.

Dashboard Table Columns

Column	Description
Status	<p>The status of the deployed channel. Common statuses include:</p> <ul style="list-style-type: none"> • Deploying: The channel is in the middle of being deployed. If the channel appears to be stuck in this state, it may mean a deploy script is taking a long time. • Undeploying: The channel is in the middle of being undeployed. If the channel appears to be stuck in this state, it may mean a shutdown script is taking a long time. • Starting: The channel is in the middle of starting up. If the channel appears to be stuck in this state, it may be in the middle of recovering unfinished messages. • Started: The channel is currently started. If the node color is orange instead of green, it means that not all connectors underneath the channel are started. • Pausing: The channel is in the middle of pausing. The source connector is in the middle of stopping, and will wait until all currently processing messages have finished before doing so. • Paused: The channel is currently paused. The source connector is currently stopped, which means the channel will no longer receive messages from its configured source, but messages queued up on destinations may still flow outbound. You can still manually send or reprocess messages while a channel is paused. • Stopping: The channel is in the middle of stopping. All currently processing messages will be finished first before the channel stops. If the channel appears to be stuck in this state, it may mean a message is taking a long time to finish processing. • Stopped: The channel is currently stopped. No messages will be received or sent out from the channel. <p>When the row is a Group rather than a Channel, the group status will be combined from all the channel statuses. For example if some channels are Started and some are Starting, the overall group status will be Starting. In other cases where channels have differing statuses you will see the Mixed group status.</p>
Name	The name of the connector, channel, or group. This column also shows any tags associated with a channel.
Rev	<p>The number of times the channel was saved since it was last deployed. This value will be highlighted if it is greater than 0, or if any code templates linked to the channel have changed since the channel was last deployed.</p> <p>Rev = Channel Revision - Deployed Revision</p>
Last Deployed	The time this channel was last deployed. This value will be highlighted if it is within the last two minutes.
Received	The number of messages received and accepted by the channel's source connector.
Filtered	The number of messages filtered out by the channel's source connector or any of its destination connectors.
Queued	The number of messages currently queued by the channel's source connector or any of its destination connectors. This value will be highlighted if it is greater than 0.
Sent	The number of messages that have been sent by all destination connectors in the channel.
Errored	The number of messages that errored somewhere in the channel. This value will be highlighted if it is greater than 0.
Connection	<p>The current activity of the channel / connector. Not all connector types implement this column, so the default value you'll see is Idle. Common connection statuses include:</p> <ul style="list-style-type: none"> • Idle: The source connector is not currently in the process of receiving a message.

- **Reading:** The source connector is currently reading a message into the channel. Generally used by polling source connectors.
- **Writing:** The destination connector is currently dispatching a message outbound.
- **Polling:** The source connector is currently polling for messages to read in.
- **Receiving:** The source connector is currently receiving a message from an external system.
- **Sending:** The destination connector is currently dispatching a message outbound.
- **Waiting For Response:** The destination connector has sent the message outbound and is waiting on a response from the remote system.
- **Connected:** The source connector currently has one or more clients connected to it. This generally will also include a number indicating how many clients are currently connected.



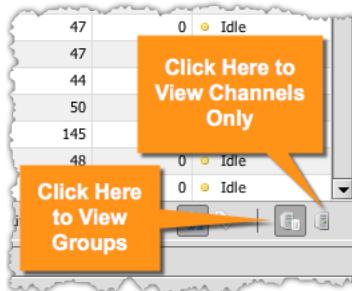
If the connection status is highlighted in red, it typically means that the source connector has reached its configured maximum number of allowed clients. Currently connected clients will be allowed to send messages but no new clients will be able to connect.

Viewing Messages For a Channel

To view messages for a particular channel, you can select the channel and then click the **View Messages** task, as described in [Dashboard Tasks](#). An alternative method is to simply double-click the channel from the dashboard table.

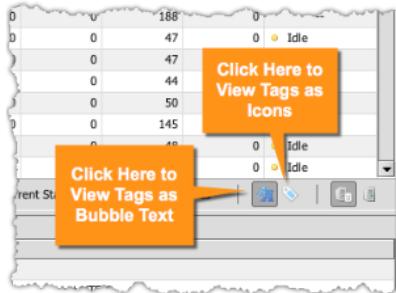
Show or Hide Channel Groups

By default the channels will be organized under top-level group nodes in the tree. All channels that are not part of a group will be organized underneath [Default Group]. If you only want to see the channels without group organization, click on the control icons at the bottom-right of the table:



Change How Tags Are Displayed

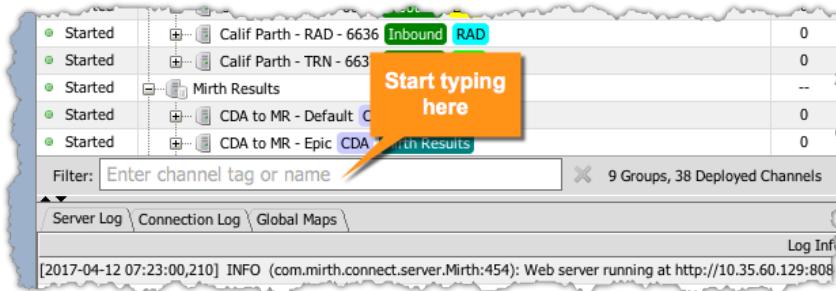
By default tags will be displayed as bubble-text next to the channel names in the table. To change this to a small icon instead, click on the control icons at the bottom-right of the table:



To not show tags in the table at all, simply click the currently selected icon again to deselect it.

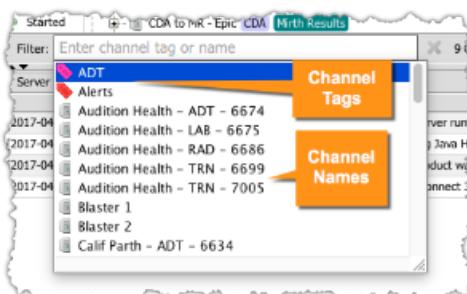
Filtering By Channel Name or Tag

The Filter field, located at the bottom-left of the table, allows you to quickly search for channels, only showing channels that match the name / tag entered in the Filter field.



View All Available Tags / Names

To view all tags / channel names in a list, select the Filter field so it has focus, and then press the Down arrow key:

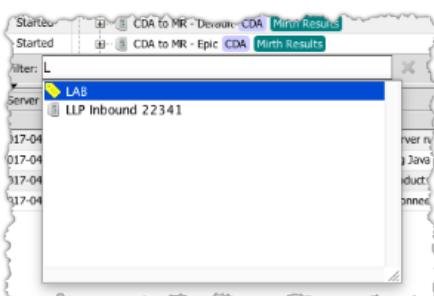


Note that channel tags will show up in the list with the tag icon:

Channel names will show up in the list with the channel icon:

Auto-Complete Tags / Names

Start typing into the field, and all tags / channel names that partially match will automatically show up in the drop-down list:



Filter By Channel Tags

To apply a channel tag filter:

- Select a channel tag from the drop-down list, then hit the **Enter** key,
- **OR** double-click a channel tag from the drop-down list,
- **OR** type a tag into the text field, then hit the **Enter** key.

The tag will show up in the filter field, and channels in the table will be filtered down to only those that have the matching tag:

Status	Name	Rev Δ	Last Deployed	Received	Filtered
Started	Alert Processors	--	--	8	0
Started	Notification Receiver - ADTs	0	2017-04-12 07:22	8	0
Started	Audition Health	--	--	50	0
Started	Audition Health - ADT - 6674	0	2017-04-12 07:22	50	0
Started	Calif Parth	--	--	47	0
Started	Calif Parth - ADT - 6634	0	2017-04-12 07:22	47	0
Started	SamWorther Tech	--	--	46	0
Started	SamWorther Tech - ADT - 7712	0	2017-04-12 07:22	46	0

When a filter is present, the status label next to the Filter field will display the total/visible/filtered counts for both channel groups and channels.

You can filter by multiple tags. Simply select the field and follow the instructions above to add another tag. When multiple tags are present in the filter, the resulting channels in the table will be those that match **all** filtered tags.

Status	Name	Rev Δ	Last Deployed	Received
Started	Audition Health	--	--	50
Started	Audition Health - ADT - 6674	0	2017-04-12 07:22	50
Started	Calif Parth	--	--	47
Started	Calif Parth - ADT - 6634	0	2017-04-12 07:22	47
Started	SamWorther Tech	--	--	46
Started	SamWorther Tech - ADT - 7712	0	2017-04-12 07:22	46

Filter By Channel

To filter down to a specific channel:

- Select a channel name from the drop-down list, then hit the **Enter** key,
- **OR** double-click a channel name from the drop-down list,
- **OR** type a channel name into the field, then hit the **Enter** key.

Filter By Partial Channel Name

To filter down to all channels that partially match a certain string:

- Type your search string into the Filter field.
- Hit the **Enter** key.

Status	Name	Rev Δ	Last Deployed	Received
Started	Audition Health	--	--	50
Started	Audition Health - ADT - 6674	0	2017-04-12 07:22	50

If the string you wish to filter by already matches the beginning of a channel name, then that channel will automatically be selected instead. If you don't want that to happen:

- Type your search string into the Filter field.
- Press the **Esc** key to close the drop-down list.
- Press the **Enter** key.

The screenshot shows the Mirth Connect Administrator interface. The left sidebar has a 'Dashboard' icon selected. The main area displays a table titled 'Dashboard' with columns for 'Status', 'Name', 'Rev Δ', and 'Last'. A filter bar at the top says 'Filter: Audition x Enter channel tag or name' with a red 'X' button. Below the table is a log viewer with tabs for 'Server Log', 'Connection Log', and 'Global Maps'. The log shows a single entry: '[2017-04-12 07:23:00,210] INFO (com.mirth.connect.server.Mirth:454): Web server running at http://10.35.60.129:8080/ and https://10.35.60.129:8443/'.

Status	Name	Rev Δ	Last
Started	Audition Health	--	
Started	Audition Health - ADT - 6674	ADT Inbound	0 2017
Started	Audition Health - LAB - 6675	Inbound LAB	0 2017
Started	Audition Health - RAD - 6686	Inbound RAD	0 2017
Started	Audition Health - TRN - 6699	Inbound TRN	0 2017-0
Started	Audition Health - TRN - 7005	Inbound TRN	0 2017-

Filter By Multiple Criteria

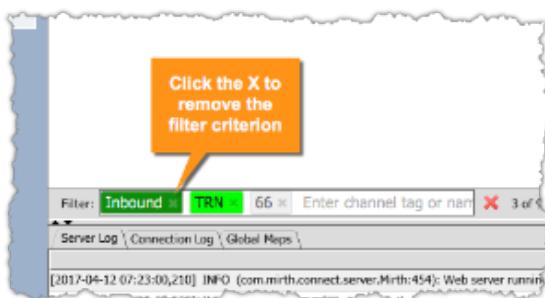
You can filter by both channel tags and channel names at the same time. Follow the instructions above to add both tags and search strings to the field:

This screenshot shows the same interface as the previous one, but with a more complex filter applied. The 'Filter' bar now contains three items: 'Inbound x', 'TRN x', and '66 x'. The table shows a subset of the previous data, specifically rows where the channel name contains 'Inbound' and the tag contains 'TRN'. The log viewer shows the same initial entry as before.

Status	Name	Rev Δ	Last
Started	Audition Health	--	
Started	Audition Health - TRN - 6699	Inbound TRN	0 2017-0
Started	Calif Parth	--	
Started	Calif Parth - TRN - 6637	Inbound TRN	0 2017-0
Started	SamWorther Tech	--	
Started	SamWorther Tech - TRN - 6699	Inbound TRN	0 2017-0

Clear Filter Criteria

To clear specific tags / channel name searches from the filter criteria, click the X button next to the box inside the field:

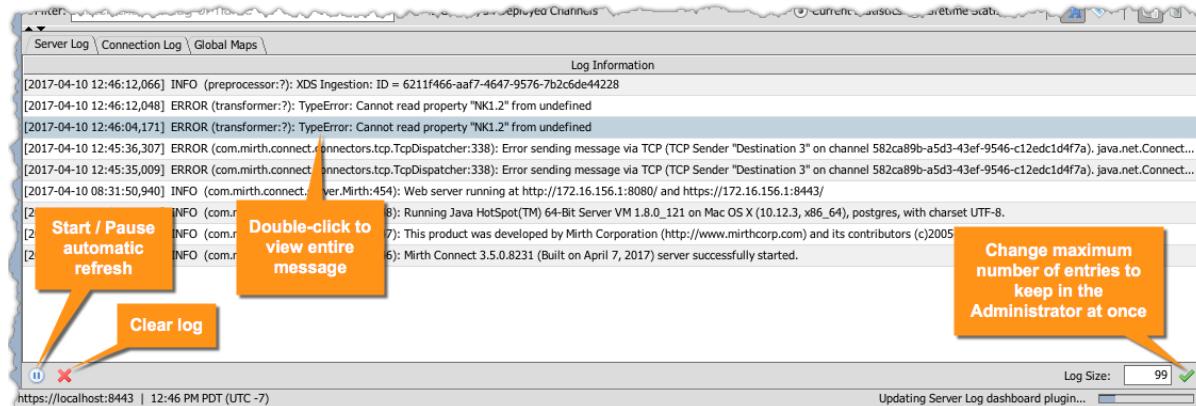


To clear all filter criteria, click the red X button next to the Filter field:



Server Log

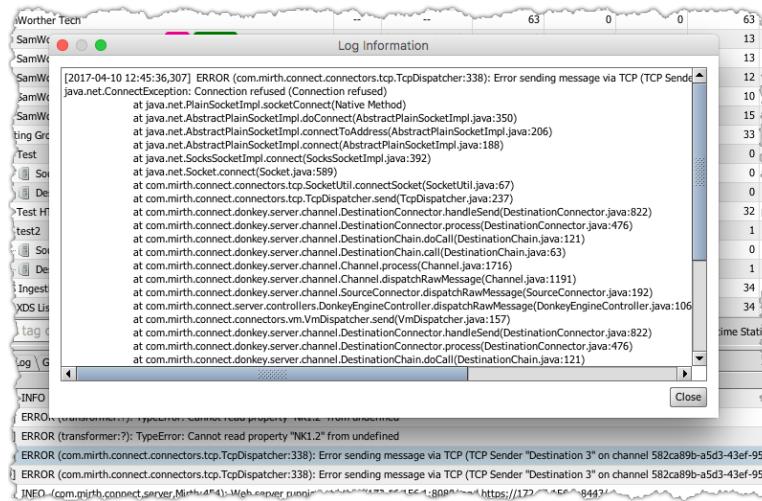
The Server Log tab in the Dashboard is a place to view the latest entries that have been written to the server logs. These entries mirror what gets written to the logs folder in the installation directory. For additional information, see [Installation Directory](#).



The entries shown in the Server Log tab follow a standard format:

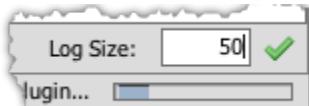
- The current timestamp. Example: **[2017-04-10 08:31:50,938]**
- The severity level of the log entry: **OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL**
- The category/class name and line number, if available. Example: **(com.mirth.connect.server.Mirth:446):**
- The actual log message.

For long messages, the value shown in the table will be truncated. To show the entire message for these entries, simply double-click on the row:



By default a maximum of 99 entries will be shown at once in the Server Log tab. Newly received entries will cause the oldest entries to automatically be removed. If you want to reduce the number of log entries that may be shown at one time:

- Edit the **Log Size** field in the bottom-right.
- Click the check icon:



To clear the logs shown in your Administrator session, click the **X** button in the bottom-left.

To start/pause the logs from being updated in your Administrator session, click the start/pause button in the bottom-left. While the Server Log tab is paused, new entries will not be pulled from the server. This allows you to analyze a particular log entry without fear of it being evicted from your session due to the max log size.

Connection Log

The **Connection Log** is a detailed view of events that occur as messages flow through your channels. When a message is received or sent out, various events get logged here. It is an ephemeral log, so it is kept in memory only and not written to a file anywhere.

Status	Name	Rev Δ	Last Deployed	Received	Filtered	Queued	Sent	Errored	Connection
Started	[Default Group]	--	--	63	0	0	61	2	--
Started	Create and Write PDF	0	2017-04-11 10:25	24	0	0	24	0	Idle
Started	LLP Inbound 22341	0	2017-04-11 10:25	1	0	0	0	1	Idle
Started	Mirth Results Sender PROD	0	2017-04-11 10:25	33	0	0	33	0	Idle
Started	Move HL7 Segments Single Line	0	2017-04-11 10:25	4	0	0	3	1	Idle
Started	Repeated Segs and Fields Iteration	0	2017-04-11 10:25	1	0	0	1	0	Idle
Started	Alert Processors	--	--	8	0	0	8	0	--
Started	Create Salesforce Ticket	0	2017-04-11 10:25	0	0	0	0	0	Idle
Started	Notification Receiver - ADTs	ADT	Alerts	0	2017-04-11 10:25	8	0	8	Idle
Started	Notification Receiver - RADs	RAD	Alerts	0	2017-04-11 10:25	0	0	0	Idle
Started	Notification Receiver - Reprocess	Alerts	RAD	0	2017-04-11 10:25	0	0	0	Idle
Started	Notification Receiver - TRNs	TRN	Alerts	0	2017-04-11 10:25	0	0	0	Idle
Started	Audition Health	--	--	253	53	42	272	23	--
Started	Audition Health - ADT - 6674	ADT	Inbound	0	2017-04-11 10:25	50	0	42	Idle
Started	Source	--	--	50	0	0	0	8	Idle
Started	Destination 1	--	--	42	0	42	0	0	Idle
Started	Destination 2	--	--	42	0	0	42	0	Idle
Started	Destination 3	--	--	42	0	0	34	8	Idle
Started	Audition Health - LAB - 6675	Inbound	LAB	0	2017-04-11 10:25	44	0	0	Idle
Started	Audition Health - RAD - 6686	Inbound	RAD	0	2017-04-11 10:25	55	0	0	Idle
Started	Audition Health - TRN - 6699	Inbound	TRN	0	2017-04-11 10:25	53	53	0	Idle
Started	Audition Health - TRN - 7005	Inbound	TRN	0	2017-04-11 10:25	51	0	0	Idle
Started	Calif Parth	--	--	188	0	0	188	0	--
Started	Calif Parth - ADT - 6634	ADT	Inbound	0	2017-04-11 10:25	47	0	0	Idle
Started	Calif Parth - LAB - 6635	Inbound	LAB	0	2017-04-11 10:25	47	0	0	Idle
Started	Calif Parth - RAD - 6636	Inbound	RAD	0	2017-04-11 10:25	44	0	0	Idle
Started	Calif Parth - TRN - 6637	Inbound	TRN	0	2017-04-11 10:25	50	0	0	Idle
Started	Mirth Results	--	--	145	0	0	145	0	--

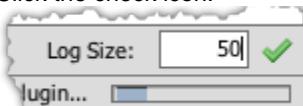
Timestamp	Channel	Connector Info	Event	Info
2017-04-11 10:42:24.127	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Idle	0.0.0.0.0.0.52467 -> 127.0.0.1:10124
2017-04-11 10:42:24.127	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Disconnected	0.0.0.0.0.0.52467 -> 127.0.0.1:10124
2017-04-11 10:42:24.127	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Sending	127.0.0.1:52467 -> 127.0.0.1:10124
2017-04-11 10:42:24.127	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Connected	127.0.0.1:52467 -> 127.0.0.1:10124
2017-04-11 10:42:24.126	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Connecting	Trying to connect on 127.0.0.1:10124...
2017-04-11 10:42:16.262	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Idle	0.0.0.0.0.0.52458 -> 127.0.0.1:10124
2017-04-11 10:42:16.262	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Disconnected	0.0.0.0.0.0.52458 -> 127.0.0.1:10124
2017-04-11 10:42:16.262	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Sending	127.0.0.1:52458 -> 127.0.0.1:10124
2017-04-11 10:42:16.262	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Connected	127.0.0.1:52458 -> 127.0.0.1:10124
2017-04-11 10:42:16.261	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Connecting	Trying to connect on 127.0.0.1:10124...
2017-04-11 10:42:14.765	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Idle	0.0.0.0.0.0.52444 -> 127.0.0.1:10124
2017-04-11 10:42:14.765	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Disconnected	0.0.0.0.0.0.52444 -> 127.0.0.1:10124
2017-04-11 10:42:14.765	Audition Health - ADT - 6674	Destination: TCP Sender - Destination 3	Sending	127.0.0.1:52444 -> 127.0.0.1:10124

The message shown in the **Info** column depends on the type of channel / connector used. For example, for a TCP Sender destination this will typically show the connected socket information:

- <local IP> : <local port> —> <remote IP> : <remote port>

By default a maximum of 250 entries will be shown at once in the **Connection Log** tab. Newly received entries will cause the oldest entries to automatically be removed. If you want to change the number of log entries that may be shown at once (up to a maximum of 999):

- Edit the **Log Size** field in the bottom-right.
- Click the check icon:



To clear the logs shown in your Administrator session, click the **X** button in the bottom-left.

To start/pause the logs from being updated in your Administrator session, click the start/pause button in the bottom-left. While the **Connection Log** tab is paused, new entries will not be pulled from the server. This allows you to analyze a particular log entry without fear of it being evicted from your session due to the max log size.

Global Maps

The **Global Maps** tab allows you to view the current state of the Global Map and the Global Channel Maps specific to particular channels. For additional information on these maps, see [Variable Maps](#).

The screenshot shows the Mirth Connect Administrator interface with the 'Global Maps' tab selected. The main area displays a table of global map entries. The columns are: Channel, Key, and Value. The table includes entries for Audition Health and Calif Parth, with their respective map details shown below the table.

Channel	Key	Value
Audition Health - ADT - 6674	adtPort1	6674
Audition Health - ADT - 6674	codeMap	{120047=ST, 120021=TX}
<Global Map>	portMap	{Parth=7003, Audition=7002}

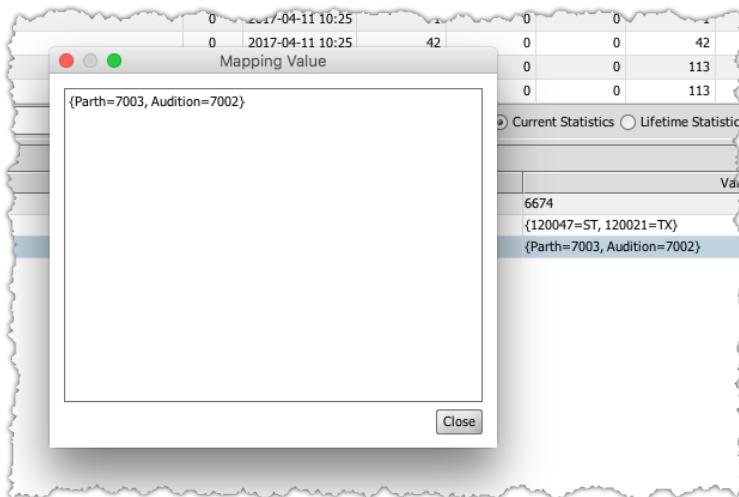
The Global Maps table shows all current global map entries, and all current global channel map entries for the channels that are currently selected in the dashboard table. You can multi-select channels to view global channel map entries across multiple channels at once.

Global Maps Table Columns

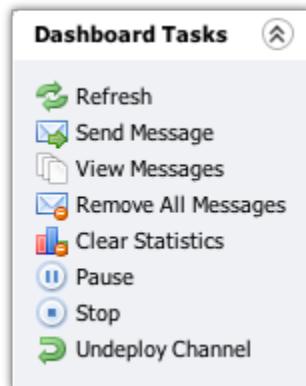
Column	Description
Channel	This will either be the channel name, or "<Global Map>" if the entry refers to the global map rather than a global channel map.
Key	The string used to uniquely identify the entry within the given map.
Value	The string representation of the current value residing in the map entry. Note that although the Global

Maps table will show the string representation, the actual object resides in memory on the server side. If the object has no string representation you may see an entry like "java.lang.Object@123abc" instead.

For values with long string representations, the value shown in the table will be truncated. To show the entire message for these entries, simply double-click on the value cell:

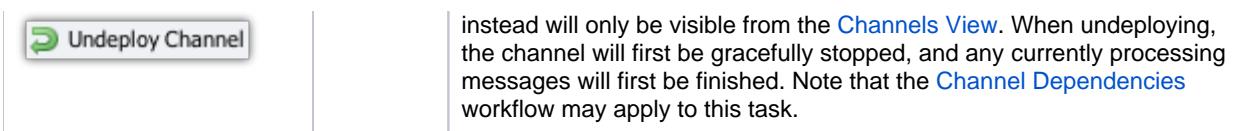


Dashboard Tasks



The following context-specific tasks are available from the Dashboard View:

Task Icon	Task Name	Description
	Refresh	Updates the dashboard table and the currently selected dashboard tab. Note that the dashboard automatically refreshes at an interval defined in the Administrator Settings .
	Send Message	Sends a message to the selected channel.
	View Messages	View messages for the selected channel. Enters the Message Browser View .
	Remove All Messages	Removes all messages and attachments stored for the selected channel(s). This action cannot be undone.
	Clear Statistics	Resets the current statistics to zero for the selected channel(s) / connector(s). This option is only available when Current Statistics is selected in the dashboard table.
	Start	Starts or resumes the selected channel(s) / connector(s). Note that the Channel Dependencies workflow may apply to this task.
	Pause	Pauses the selected channel(s). This is equivalent to stopping the source connector. Note that the Channel Dependencies workflow may apply to this task.
	Stop	Stops the selected channel(s) / connector(s). Any currently processing messages will first be finished. Note that the Channel Dependencies workflow may apply to this task.
	Halt	Attempts to stop the selected channel(s) <i>immediately</i> . Any messages currently processing through the channel will be left incomplete. The channel will attempt to recover unfinished messages when it is started back up.
	Undeploy Channel	Removes the channel from the list of deployed channels. Once undeployed, the channel will no longer appear on the dashboard, and



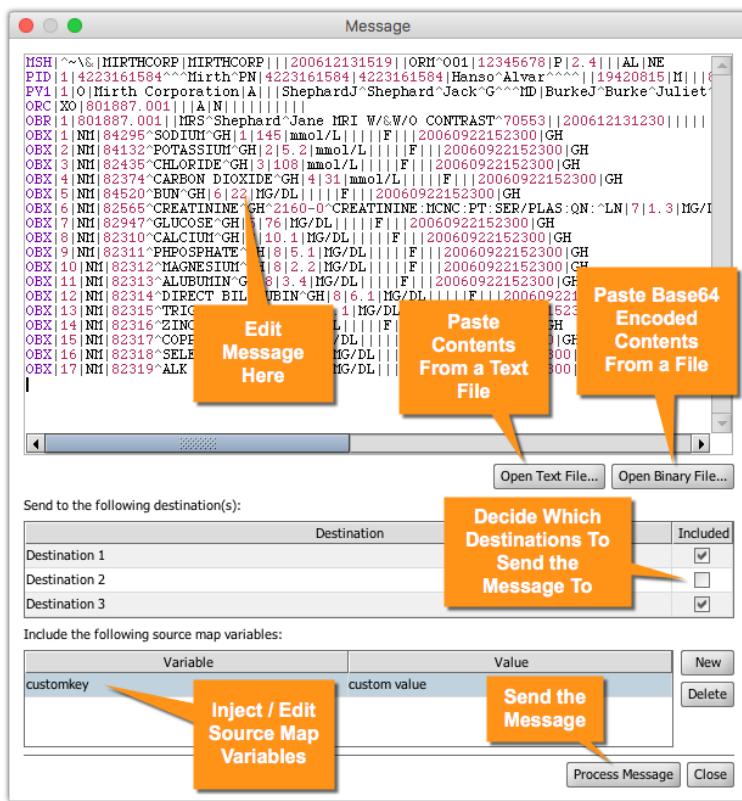
Send Message

When you click the **Send Message** task, a dialog will pop up allowing you to select the message to send. You can type in a custom message, or select a file from disk to process. When reading a file, you can choose to open it as a "Text" file, which will use the default Java charset encoding. Or, you can open it as a "Binary" file, which will paste in the Base64-encoded contents of the actual file bytes.

When sending a message to a channel, you can use the destinations table to select in advance which destinations you want to process the message through.

If your channel logic is dependent on specific source map variables, you can also inject values for those in this dialog.

Once you're ready to send the message, click on the **Process Message** button.



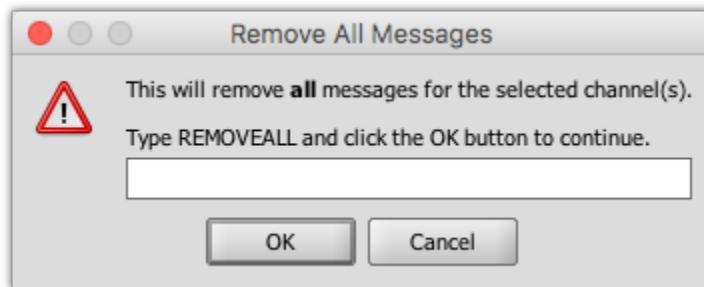
Remove All Messages

When you click the **Remove All Messages** task, a confirmation dialog pops up. Note that this action removes **all** messages and attachments for all currently selected channels.



- **Include selected channels that are not stopped (channels will be temporarily stopped while messages are being removed)**
 - You may only Remove All Messages for channels that are currently stopped. If you select a channel that is not stopped and remove all messages without checking this check box, nothing will happen. When this option is checked, any channels not currently stopped will be stopped prior to removing the messages. After the removal finishes, the channels will automatically be started back up.
- **Clear statistics for affected channel(s)**
 - If checked, all channels that have messages removed will also have their current statistics reset to zero. This option is checked by default.

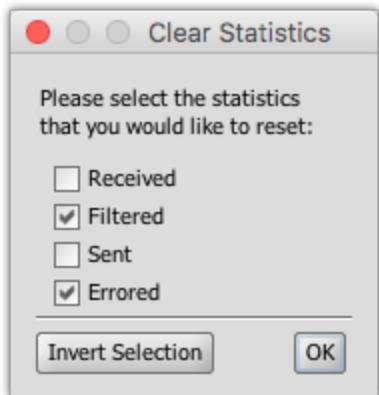
After clicking **Yes** to continue, you'll see the following additional dialog:



This is an additional layer of security to make sure you don't remove message data accidentally. To proceed you must type in "REMOVEALL" in all capital letters and click **OK**. If this additional layer of security is annoying, you can disable it in the [Administrator Settings Tab](#).

Clear Statistics

When you click the **Clear Statistics** task, a confirmation dialog pops up. Select the statistics you wish to reset to zero. You can quickly select all values by clicking the **Invert Selection** button. When you're ready, click the **OK** button to proceed.



Note that this only resets current statistics. To reset lifetime statistics, go to the [Server Settings Tab](#).

Message Browser View

The **Message Browser** allows you to view and search historical and actively processing messages for your channels. It also has options for importing, exporting, removing, and reprocessing messages.

Prod Server 1 - Mirth Connect Administrator ~ (3.5.0.8231)

Channel Messages - Audition Health - TRN - 6699

Mirth Connect

- [Dashboard](#)
- [Channels](#)
- [Users](#)
- [Settings](#)
- [Alerts](#)
- [Events](#)
- [Extensions](#)

Message Tasks

- [Refresh](#)
- [Send Message](#)
- [Import Messages](#)
- [Export Results](#)
- [Remove All Messages](#)
- [Remove Results](#)
- [Remove Message](#)
- [Reprocess Results](#)
- [Reprocess Message](#)

Other

- [Notifications](#)
- [View User API](#)
- [View Client API](#)
- [Help](#)
- [About Mirth Connect](#)
- [Visit mirthcorp.com](#)
- [Report Issue](#)
- [Logout](#)

Start Time:

All Day
 RECEIVED
 TRANSFORMED
 Current Search

End Time:

 FILTERED
 QUEUED
 Max Message Id: 53

Text Search:

 Regex
 SENT
 Connectors: (any)

Page Size:

[Advanced...](#)
[Reset](#)
[Search](#)

Results 1 - 20 of 53
[Count](#)

Page
of 3
[Go](#)

[< Prev](#)
[Next >](#)

Id	Connector	Status	Received Date	Response Date	Errors	SOURCE	TYPE
53	Source	TRANSFORMED	2017-04-11 10:42:24:260	2017-04-11 10:42:24:278	--	Epic	MDM-T02
	With Observations	SENT	2017-04-11 10:42:24:262	2017-04-11 10:42:24:269	--	Epic	MDM-T02
52	Source	TRANSFORMED	2017-04-11 10:42:20:287	2017-04-11 10:42:20:316	--	Epic	MDM-T02
	With Observations	FILTERED	2017-04-11 10:42:20:269	--	--	FACILITY A	MDM-T01
51	Source	TRANSFORMED	2017-04-11 10:42:18:394	2017-04-11 10:42:18:417	--	FACILITY A	MDM-T01
	With Observations	FILTERED	2017-04-11 10:42:18:396	--	--	FACILITY A	MDM-T01
	Without Observations	SENT	2017-04-11 10:42:18:403	2017-04-11 10:42:18:412	--	FACILITY A	MDM-T01
50	Source	TRANSFORMED	2017-04-11 10:42:16:722	2017-04-11 10:42:16:765	--	FACILITY A	MDM-T01
	With Observations	FILTERED	2017-04-11 10:42:16:743	--	--	FACILITY A	MDM-T01
	Without Observations	SENT	2017-04-11 10:42:16:751	2017-04-11 10:42:16:759	--	FACILITY A	MDM-T01
49	Source	TRANSFORMED	2017-04-11 10:42:15:185	2017-04-11 10:42:15:267	--	Epic	MDM-T02
	With Observations	SENT	2017-04-11 10:42:15:214	2017-04-11 10:42:15:232	--	Epic	MDM-T02

Messages \ Mappings \

Raw Encoded

```
HSRH|20160510074633|DEP|T02|12345|D|2_3
PID|1|1123|HATH|EP|1|T0001|Hath-J-DR|1123|12345|00104|IN|[AfrAm]506_S_HAMILTON_AVE``^MADISON_WI``53505^US``^DN|DN|(608)123-9998|(608)123-5679||S||18273652
PV1|||^CARE HEALTH SYSTEMS~^~|||1123^HATHREWS-JAMES-A~^~|||^12345|||PA|
TXA|CN|20160510071633|1123^HATHREWS-JAMES-A~^~|||^12345|||PA|
OBX|1|TX|||Clinical summary: Based on the information provided, the patient likely has viral sinusitis commonly called a head cold.
OBX|2|TX|||Diagnosis: Viral Sinusitis
OBX|3|TX|||Prescription: Zithromax 300 mg
OBX|4|TX|||Prescription: benzonatate (Tessalon Perles) 100mg oral tablet 30 tablets, 5 days supply. Take one to two tablets by mouth three times a day a
```

Format XML Messages

Navigation

From the Dashboard

Click on the **Dashboard** link in the **Mirth Connect** task panel in the upper-left:



Then select the channel you want to view messages for, and click the **View Messages** task in the left task list. For additional information, see [Dashboard Tasks](#):

1) Select a channel

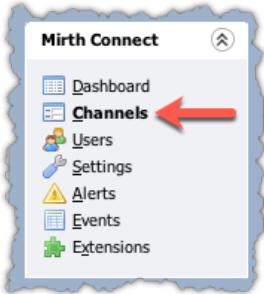
2) Click View Messages

Status	Name	Rev Δ	Last Deployed	Received	Filtered	Queued	Sent	Errored	Connection
Started	TLS TCP Sender 7771	0	2016-03-29 11:56	2	0	0	1	1	Idle
Started	ORU Inbound 6678	0	2016-03-29 11:56	0	0	0	0	0	Idle
Started	MRM Inbound 7050	0	2016-03-29 11:56	0	0	0	0	0	Idle
Stopped	Tcp Sender 6661	0	2016-03-29 11:56	0	0	0	0	0	Idle
Started	test3	0	2016-03-29 11:56	3	0	0	3	0	Idle
Started	TLS TCP Listener 7771	0	2016-03-29 11:56	1	0	0	1	0	Idle
Started	test2	1	2016-03-29 11:56	0	0	0	0	0	Idle
Started	MDM Inbound 6679	0	2016-03-29 11:56	0	0	0	0	0	Idle

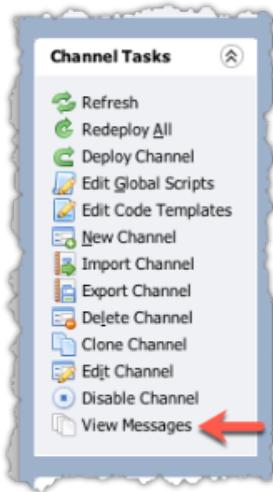
OR, double-click the channel in the Dashboard Table.

From the Channels View

Click on the **Channels** link in the Mirth Connect task panel in the upper-left:



Then select the channel you want to view messages for, and click the **View Messages** task in the left task list . For additional information, see [Channel Tasks](#):



This section is separated into the following topics:

- [Metadata Table](#)
- [Message Content Tab](#)
- [Mappings Tab](#)
- [Errors Tab](#)
- [Attachments Tab](#)
- [Searching Messages](#)
- [Message Browser Tasks](#)

Metadata Table

This table shows metadata (e.g., ID, status, timestamps) about the messages that match your current search window. The number of messages shown in the table at once is determined by the Page Size option (additional information about the page size option, see [Searching Messages](#)). When you first enter the Message Browser, by default the latest 20 messages (by ID) will appear.

ID	Connector	Status	Received Date	Response Date	Errors	SOURCE	TYPE
53	Source	TRANSFORMED	2017-04-11 10:42:24:260	2017-04-11 10:42:24:278	--	Epic	MDM-T02
	With Observations	SENT	2017-04-11 10:42:24:262	2017-04-11 10:42:24:269	--	Epic	MDM-T02
52	Source	TRANSFORMED	2017-04-11 10:42:20:287	2017-04-11 10:42:20:316	--	FACILITY A	MDM-T01
	With Observations	FILTERED	2017-04-11 10:42:20:290	--	--	FACILITY A	MDM-T01
51	Source	TRANSFORMED	2017-04-11 10:42:18:394	2017-04-11 10:42:18:417	--	FACILITY A	MDM-T01
	With Observations	FILTERED	2017-04-11 10:42:18:396	--	--	FACILITY A	MDM-T01
50	Source	TRANSFORMED	2017-04-11 10:42:16:722	2017-04-11 10:42:16:765	--	FACILITY A	MDM-T01
	With Observations	FILTERED	2017-04-11 10:42:16:743	--	--	FACILITY A	MDM-T01
49	Source	TRANSFORMED	2017-04-11 10:42:15:185	2017-04-11 10:42:15:267	--	FACILITY A	MDM-T01
	With Observations	SENT	2017-04-11 10:42:15:214	2017-04-11 10:42:15:232	--	Epic	MDM-T02

Messages are organized by ID, and multiple rows for each **connector message** appears in the table, depending on your [search criteria](#).

Adding a Column to the Metadata Table

By default eight default columns are shown: *Id*, *Connector*, *Status*, *Received date*, *Response Date*, *Errors*, *Source*, and *Type*. You can add columns to and remove columns from the list one at a time using the Column Options icon /menu in the list's column header.



All Messages Lists for all channels have the same column configuration; that is, any columns added to /removed from the Messages List of one channel are added to/removed from the Messages List of all channels. You can add as many available columns as you like, though not all will be visible onscreen at once. If all columns cannot be viewed onscreen, a horizontal scroll bar appears below the Messages List. Scroll left or right to view the desired columns/data.

Status	Send Date	Received Date	Response Date	Errors	Original Id	Import Id	SOURCE	TYPE
SFORMED	--	2016-02-26 18:52:12:032	--	--	--	--	Sacred Heart	ADT-A08
	2016-02-26 18:52:12:037	2016-02-26 18:52:12:034	2016-02-26 18:52:12:037	--	--	--	Sacred Heart	ADT-A08
SFORMED	--	2016-02-26 18:52:12:020	--	--	--	--	Sacred Heart	ADT-A08
	2016-02-26 18:52:12:026	2016-02-26 18:52:12:023	2016-02-26 18:52:12:026	--	--	--	Sacred Heart	ADT-A08
SFORMED	--	2016-02-26 18:52:12:007	--	--	--	--	Sacred Heart	ADT-A08
	2016-02-26 18:52:12:016	2016-02-26 18:52:12:012	2016-02-26 18:52:12:016	--	--	--	Sacred Heart	ADT-A08
SFORMED	--	2016-02-26 18:38:58:123					Sacred Heart	ADT-A08

Scroll right to view other columns

Removing a column does not remove its data; the data reappears when you add the column back to the Messages List.

Use the following steps to add a column to the Message List.

- At the far-right end of the Message Lists' column header, click the **Column Options** icon

2. On the Column Options menu, select or deselect the column(s) you would like to add or remove from the column header.

The screenshot shows a table with columns: Id, Connector, Status, Received Date, Response Date, Errors, SOURCE, and TYPE. A purple callout points to the 'Connector' column header with the text 'Eight default columns'. An orange callout points to the 'TYPE' column header with the text 'Click to open the Column Options menu, then select/deselect options as desired'. A red box highlights the 'TYPE' column header. A dropdown menu is open, listing columns with checkboxes: Id (checked), Connector (checked), Status (checked), Orig. Received Date (selected), Received Date (checked), Send Attempts (unchecked), Send Date (unchecked), Response Date (checked), and Errors (checked). The 'Connector' option is also listed at the bottom.



The following list-related tasks are also available on the Column Options menu:

- Collapse All*—Shows only the Source Connector row of each message in the Messages List.
- Expand All*—Shows the Source Connector row *and* Destination row(s) of each message in the Messages List (as in the previous graphic).
- Restore Default*—Reconfigures the Messages List to contain the eight default columns (see the introductory paragraph in this section).

You can pause the pointer on a column header to reveal a tool-tip with a description of that column's function.

The screenshot shows the same table as above. A yellow callout with a dashed border points to the 'Connector' column header with the text 'Hover over a column header...'. Another yellow callout with a dashed border points to the 'Connector' column header with the text '...to view a description of the column's function'. A mouse cursor is hovering over the 'Connector' column header, and a tooltip box appears below it containing the text: 'The historic name of the connector at the time the message was processed.' A red dashed box highlights the 'Connector' column header.

For general information about working with tables in NextGen Connect Integration Engine, go here: [Working With Tables](#)

Metadata Table Columns

Column	Description
Id	The unique ID that identifies the message within the current channel. Note that a single message can have multiple connector messages for each connector (source / destinations) in the channel. Each connector message row is organized by this column in the table.
Connector	The name of the connector. (e.g. "Source", "Destination 1")
Status	The current status of the connector message. Valid values include: <u>Source connector statuses:</u> <ul style="list-style-type: none"> Received: If the source queue is enabled, this means the message has been committed to the database and added to the source queue, but has not yet been processed. If the source queue is disabled, this means the message has not yet finished processing through the source connector. Filtered: The message has been rejected by the source filter, and will not flow any further through the channel.

- **Transformed:** The message has passed the source filter/transformer, and the source encoded data has been dispatched to any destinations.
- **Error:** An error occurred while processing the message through the source connector. This typically means that the preprocessor or source filter/transformer script failed.

Destination connector statuses:

- **Received:** The inbound data for the destination connector has been committed to the database, but the destination has not yet finished processing the message.
- **Filtered:** The message has been rejected by the destination filter, and will not be dispatched by this destination. Other destinations may still dispatch this message.
- **Sent:** The message has been successfully dispatched / written out by the destination connector.
- **Queued:** The message either has not been attempted to be dispatched yet, or it has failed to dispatch and is waiting in the queue to be attempted again.
- **Error:** An error occurred while processing the message through the destination connector. This could mean that an error occurred in the destination filter/transformer, that the destination failed to dispatch the message outbound, or that the destination was able to dispatch the message but the outbound system returned a failure response (such as an HTTP 500). If the error occurred in the destination filter/transformer, then the message will not process through the rest of the connectors in the current destination chain. If the error occurred in the dispatcher, the message may still be processed through subsequent destinations in the current chain.
- **Pending:** The destination was able to dispatch / write the message outbound, but has not yet finished processing the message through the response transformer.

Orig. Received Date	The date and time the original message was received. This value is not updated if the message gets overwritten by a reprocess operation.
Received Date	The date and time the message was received by the source/destination connector. This value may update if the message gets overwritten by a reprocess operation.
Send Attempts	<ul style="list-style-type: none"> • Source Connector: The number of times the connector attempted to send the response back to the point of origin. • Destination Connector: The number of times the connector attempted to send the message outbound.
Send Date	<ul style="list-style-type: none"> • Source Connector: N/A • Destination Connector: The date and time immediately before the most recent send attempt.
Response Date	<ul style="list-style-type: none"> • Source Connector: The date and time immediately before the connector attempted to send the response back to the point of origin. • Destination Connector: The date and time immediately after the connector received and stored the response from the outbound system.
Errors	<p>Indicates whether an error exists for this message. It is possible for a message to have errors associated with it even if the message status is not ERROR. Possible values include:</p> <ul style="list-style-type: none"> • Processing: An error occurred during a user-script, such as the preprocessor or the filter / transformer. • Response: An error occurred generating the response, or sending the response from the source connector to its point of origin. • Postprocessor: An error occurred during the postprocessor script. • Multiple: Two or more of the above errors occurred.

Server Id	The ID of the server that processed the message through the connector.
Original Server Id	The ID of the server that originally processed the message, in the case that the current message has been imported or taken over by another server.
Original Id	The original ID of the reprocessed message. This value only exists for reprocessed messages.
Import Id	The original ID of an imported message. This value only exists for imported messages.
Import Channel Id	The original channel ID of an imported message. This value only exists for messages imported from a different channel.

Custom Metadata Columns

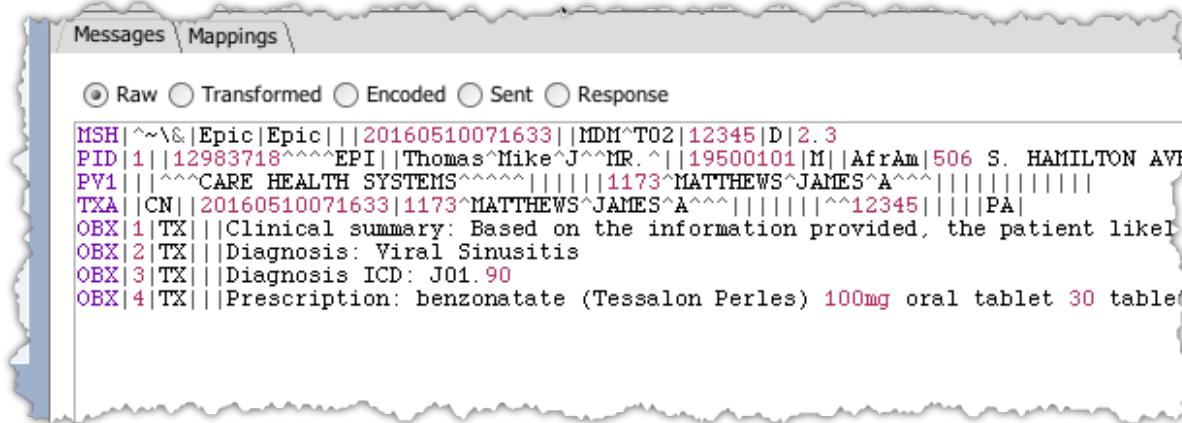
Custom Metadata Columns for the current channel will also show up in the metadata table. These columns will vary depending on the channel. By default the **Source** and **Type** columns are added for all new channels, though this can be changed in the [Server Settings](#).

Column	Description
Source	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Sending Facility value in MSH.4.1.
Type	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Type and Trigger values in MSH.9.1 and MSH.9.2.
Version	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Version value in MSH.12.1. Note that this custom metadata column is not added to new channels by default, though this may be changed in the Server Settings Tab .

Note that the above values are only the **defaults**, and that users can change those values inside of a transformer.

Message Content Tab

The **Messages** tab in the Message Browser shows the actual content of the message at various states as it processed through the selected connector message.



Viewing Message Content

- Select a connector message in the [Metadata Table](#).
- Select the **Messages** tab (located below the Metadata Table).
- Select the radio button corresponding with the content type you want to view.

The screenshot shows the 'Channel Messages' screen in Mirth Connect. The left sidebar includes 'Dashboard', 'Channels', 'Users', 'Settings', 'Alerts', 'Events', and 'Extensions'. Under 'Message Tasks', there are options like 'Refresh', 'Send Message', 'Import Messages', 'Export Results', 'Remove All Messages', 'Remove Results', 'Remove Message', 'Reprocess Results', and 'Reprocess Message'. The main area shows a table of messages with columns: Id, Connector, Status, Received Date, Response Date, and Errors. One message is highlighted with a yellow arrow and labeled '1) Select a connector message'. Another message is highlighted with a yellow arrow and labeled '2) Click the Messages tab'. The bottom section shows the message content with radio buttons for 'Raw', 'Transformed', 'Encoded', 'Sent', and 'Response'. A yellow arrow points to the 'Raw' button and is labeled '3) Select the type of content to view'.

ID	Connector	Status	Received Date	Response Date	Errors
54	Source	TRANSFORMED	2017-04-12 13:10:17:962	2017-04-12 13:10:18:335	--
	With Observations	SENT	2017-04-12 13:10:18:006	2017-04-12 13:10:18:307	--
	Without Observations	FILTERED	2017-04-12 13:10:18:307	--	--
53	Source	TRANSFORMED	2017-04-11 10:42:20:290	2017-04-11 10:42:20:297	24:278
	With Observations	SENT	2017-04-11 10:42:20:297	2017-04-11 10:42:20:306	24:269
	Without Observations	FILTERED	2017-04-11 10:42:20:297	2017-04-11 10:42:20:306	--
52	Source	TRANSFORMED	2017-04-11 10:42:20:287	2017-04-11 10:42:20:316	--
	With Observations	FILTERED	2017-04-11 10:42:20:290	--	--
	Without Observations	SENT	2017-04-11 10:42:20:297	2017-04-11 10:42:20:306	--
51	Source	TRANSFORMED	2017-04-11 10:42:18:394	2017-04-11 10:42:18:417	--
	With Observations	FILTERED	2017-04-11 10:42:18:396	--	--
	Without Observations	SENT	2017-04-11 10:42:18:403	2017-04-11 10:42:18:412	--
50	Source	TRANSFORMED	2017-04-11 10:42:16:722	2017-04-11 10:42:16:765	--
	With Observations	SENT	2017-04-11 10:42:16:743	--	--

Current Search
Max Message Id: 54
Date Range: (any) to (any)
Statuses: (any)
Connectors: (any)

Messages | **Mappings**

Raw | Transformed | Encoded | Sent | Response

```

MSH|^~\&|Epic|Epic|||20160510071633||MDM^T02|12345|D|2.3
PID|1||12983718^~^EPI||Thomas^Mike^J^MR.|||19500101|M||AfrAm|506 S. HAMILTON AVE^~^MADISON^WI^53505^
PV1|||CARE HEALTH SYSTEMS^~^|||1173^MATTHEWS^JAMES^A^~^|||12345||PA|
TXA||CN||20160510071633|1173^MATTHEWS^JAMES^A^~^|||12345||PA|
OBX|1|TX||Clinical summary: Based on the information provided, the patient likely has viral sinusitis
OBX|2|TX||Diagnosis: Viral Sinusitis
OBX|3|TX||Diagnosis ICD: J01.90
OBX|4|TX||Prescription: benzonatate (Tessalon Perles) 100mg oral tablet 30 tablets, 5 days supply

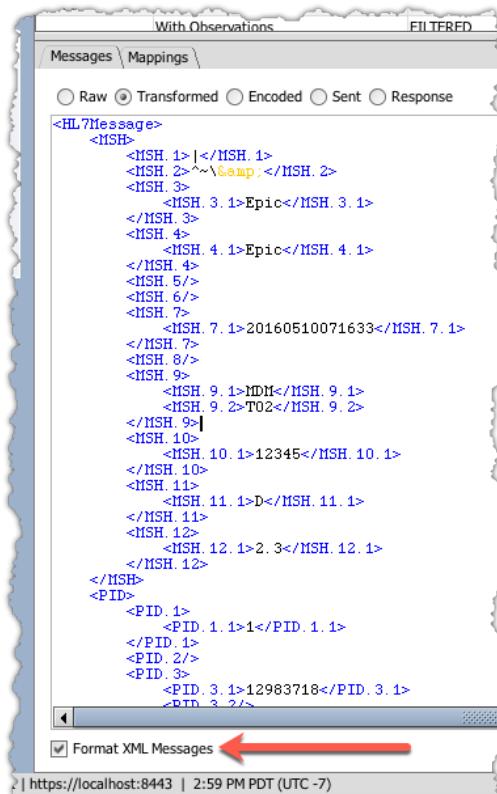
```

Message Content Types

Content Type	Description
Raw	The inbound message as received and stored by channel, after the attachment handler has extracted any attachments but before the preprocessor script has modified the message.
Processed Raw	The altered inbound message after the preprocessor script has executed.
Transformed	When a message enters a transformer, the raw (or processed raw) data is serialized into an internal representation, which may be XML, JSON, or Raw depending on the inbound data type. The transformer then has a chance to alter this serialized data. This content is the internal representation of the message after the transformer has executed.
Encoded	After a message leaves the transformer, the Transformed Data (internal representation of the message) is deserialized into the outbound data type. This is referred to as the Encoded Data. The encoded data for a source connector is equivalent to the raw data for a destination connector.
Sent	A snapshot of the destination connector properties immediately before the destination connector attempts to dispatch the message.
Response	The Response object returned by the destination connector after a dispatch has been attempted. This will include the response status, the status message, and the actual response payload (if present).
Response Transformed	This is the same as the Transformed content, except that it is for the response data. This will be the internal representation of the response content, serialized into the response inbound data type.
Processed Response	This is the same as the Encoded content, except that it is for the response data. This will be the internal serialized representation of the response content, deserialized into the response outbound data type.

Formatting Messages

The Format Messages check box allows you to pretty-print XML and JSON messages with whitespace and indentation so that it's easier to read. Note that this does not alter the actual message content, only how it is displayed in the Administrator.



The screenshot shows the NextGen Connect application interface. At the top, there's a navigation bar with tabs like 'With Observations' and 'FILTERED'. Below the navigation bar, there's a toolbar with buttons for 'Raw', 'Transformed' (which is selected), 'Encoded', 'Sent', and 'Response'. The main content area displays an HL7 message in XML format. The XML code is as follows:

```
<HL7Message>
<MSH>
<MSH.1>|</MSH.1>
<MSH.2>^~\&lt;&gt;</MSH.2>
<MSH.3>
<MSH.3.1>Epic</MSH.3.1>
</MSH.3>
<MSH.4>
<MSH.4.1>Epic</MSH.4.1>
</MSH.4>
<MSH.5>
<MSH.6>
<MSH.7>
<MSH.7.1>20160510071633</MSH.7.1>
</MSH.7>
<MSH.8>
<MSH.9>
<MSH.9.1>MDM</MSH.9.1>
<MSH.9.2>T02</MSH.9.2>
</MSH.9>
<MSH.10>
<MSH.10.1>12345</MSH.10.1>
</MSH.10>
<MSH.11>
<MSH.11.1>D</MSH.11.1>
</MSH.11>
<MSH.12>
<MSH.12.1>2.3</MSH.12.1>
</MSH.12>
</MSH>
<PID>
<PID.1>
<PID.1.1>1</PID.1.1>
</PID.1>
<PID.2>
<PID.3>
<PID.3.1>12983718</PID.3.1>
</PID.3>

```

At the bottom left of the message content area, there is a checkbox labeled 'Format XML Messages' with a checked status. A red arrow points to this checkbox. The bottom status bar shows the URL 'https://localhost:8443' and the time '2:59 PM PDT (UTC -7)'.

Mappings Tab

The **Mappings** tab shows all entries stored in [variable maps](#) for the selected connector message.

Scope	Variable	Value
Source	sourceChannelId	21019e3d-dc22-4dac-83b6-2c24e5601d21
Source	sourceMessageId	1
Source	replaced	false
Source	reprocessed	true
Source	destinationSet	[1, 2]
Connector	mirth_source	Epic
Connector	mirth_version	2.3
Connector	mirth_type	MDM-T02
Response	d1	SENT: Message routed successfully to channel id: none

Viewing Mappings

- Select a connector message in the [Metadata Table](#).
- Select the **Mappings** tab (located below the Metadata Table).
- Double-click entries in the table to view the values in a separate dialog.

Prod Server 1 - Mirth Connect Administrator - (3.5.0.8231)

Channel Messages - Audition Health - TRN - 6699

Start Time: [] 08:49 AM ▾ All Day RECEIVED
End Time: [] 08:49 AM ▾ TRANSFORMED
Text Search: [] Regex FILTERED
Page Size: 20 QUEUED Advanced... Reset Search SENT ERROR

ID	Connector	Status	Received Date	Response Date	Errors	Source
54	Source	TRANSFORMED	2017-04-12 13:10:17:962	2017-04-12 13:10:18:335	--	Epic
	With Observations	SENT	2017-04-12 13:10:18:006	2017-04-12 13:10:18:307	--	Epic
	Without Observations	FILTERED	2017-04-12 13:10:18:307	--	--	Epic
53	Source	TRANSFORMED	2017-04-12 13:10:18:307	2017-04-12 13:10:18:307	278	Epic
	With Observations	SENT	2017-04-12 13:10:18:307	2017-04-12 13:10:18:307	269	Epic
	Without Observations	FILTERED	2017-04-12 13:10:18:307	--	--	Epic
52	Source	TRANSFORMED	2017-04-11 10:42:20:287	2017-04-11 10:42:20:316	--	FACILITY A
	With Observations	FILTERED	2017-04-11 10:42:20:290	--	--	FACILITY A
	Without Observations	SENT	2017-04-11 10:42:20:297	2017-04-11 10:42:20:306	--	FACILITY A
51	Source	TRANSFORMED	2017-04-11 10:42:18:394	2017-04-11 10:42:18:417	--	FACILITY A
	With Observations	SENT	2017-04-11 10:42:18:396	--	--	FACILITY A
	Without Observations	FILTERED	2017-04-11 10:42:18:403	2017-04-11 10:42:18:412	--	FACILITY A
50	Source	TRANSFORMED	2017-04-11 10:42:16:722	2017-04-11 10:42:16:765	--	FACILITY A
	With Observations	FILTERED	2017-04-11 10:42:16:743	--	--	FACILITY A

Messages \ Mappings

Scope	Variable	Value
Source	sourceChannelId	21019e3d-dc22-4dac-83b6-2c24e5601d21
Source	sourceMessageId	1
Source	replaced	false
Source	reprocessed	true
Source	destinationSet	[1, 2]
Connector	mirth_source	Epic
Connector	mirth_version	2.3
Connector	mirth_type	MDM-T02
Response	d1	SENT: Message routed successfully to channel id: none

Mappings Table Columns

Column	Description
Scope	The type of map the entry is stored in. For additional information, see Variable Maps .

Variable	The unique key identifying the entry within the given map.
Value	The actual value of the entry.

Errors Tab

If any error content exists for a connector message, a separate Errors tab will be present in the message browser when you select that row in the Metadata Table. You can tell beforehand whether a connector message has error content by looking at the Errors column in the metadata table. For additional information, see [Metadata Table](#).

The screenshot shows the Mirth Connect interface with the 'Messages' tab selected. In the main pane, a table displays connector messages with their status, timestamp, and destination. One message in the 'Source' row is highlighted in blue and has an orange arrow pointing to it from the left margin, indicating it is selected. Below the table, a detailed error log is displayed in a text area. The log starts with 'Transformer error' and includes several lines of Java stack trace, ending with 'at java.lang.Thread.run(Thread.java:745)'. At the bottom of the interface, a status bar indicates 'Connected to: Prod Server 1 | https://localhost:8443 | 4:00 PM PDT (UTC -7)'.

Viewing Message Errors

- Select a connector message in the [Metadata Table](#).
- Select the Errors tab (located below the Metadata Table). Note that this tab is only available if the connector message actually has error content.
- Select the radio button corresponding to the content type you want to view.

This screenshot is similar to the one above, showing the Mirth Connect interface with the 'Messages' tab selected. A connector message in the 'Source' row is selected, indicated by an orange arrow. Three callouts provide instructions:

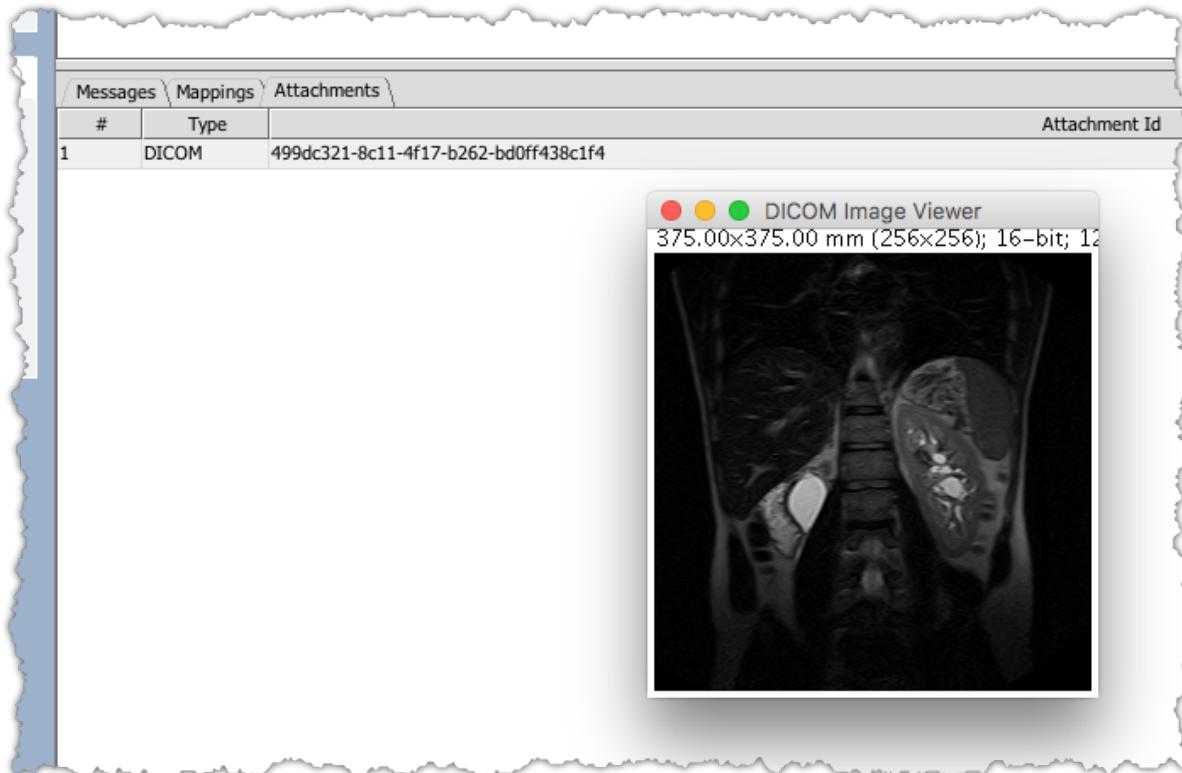
- 1) Select a connector message**: An orange box with an orange arrow pointing to the selected 'Source' message in the table.
- 2) Click the Errors tab**: An orange box with an orange arrow pointing to the 'Errors' tab located below the table.
- 3) Select the type of content to view**: An orange box with an orange arrow pointing to the radio button selection area, which includes options like 'Processing Error' and a detailed error log.

Error Content Types

Content Type	Description
Processing Error	An error occurred during a user-script, such as the preprocessor or the filter / transformer.
Response Error	An error occurred generating the response, or sending the response from the source connector to its point of origin.
Postprocessor Error	An error occurred during the postprocessor script .

Attachments Tab

If a message has attachments associated with it, a separate Attachments tab will be visible when selecting any of the connector message rows within the message in the Metadata Table.



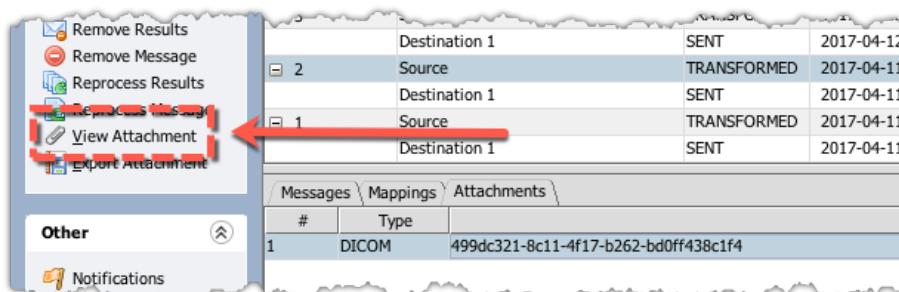
Attachment Table Columns

Column	Description
#	The sequence numbers associated with the attachment(s). This usually starts at 1 and increase for every attachment in the table. In some cases for DICOM messages you may see a single row that has a range of numbers in this column, indicating that multiple attachments have been stored for a single DICOM message.
Type	The type of attachment stored. This will either be a MIME type like "text/plain" or "application/pdf", or "DICOM" in the case of DICOM image attachments.
Attachment Id	The unique ID of the attachment. If multiple attachments have been combined into a single entry you'll see a comma-separated list of IDs.

Viewing Attachments

To view attachment content:

- Double-click the row in the Attachments table,
- OR, select the row in the Attachments table, then click the **View Attachment** task to the left:



A confirmation dialog pop up, asking you what type of viewer to use:



There are four types of attachment viewers: Text, Image, DICOM, and PDF. The dialog will try to pre-select the viewer for you based on the Type of the attachment. If you need to override that selection and choose a different viewer, you can do that by using the drop-down menu.

If you want to bypass this dialog and have the Message Browser always automatically open the viewer based on the Type of the attachment, you can check the "Always choose" check box. This setting can be reset later in the [Administrator Settings Tab](#).

Once you've chosen a viewer, click the **OK** button to proceed.

Text Attachment Viewer

Use this for text/* type attachments. A dialog pops up with the textual representation of the data:



The data is assumed to be base Base64 encoded, so by default the Decode Base64 Data check box is checked. To view the raw data, uncheck that check box:

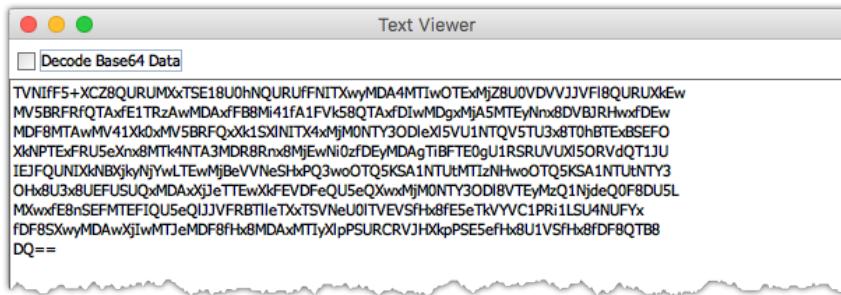


Image Attachment Viewer

Use this for image/* type attachments. A dialog pops up and render the image for you:



If the image cannot be rendered for any reason, an error dialog appears instead of the image:



DICOM Attachment Viewer

Use this for DICOM attachments. A dialog pops up to show the image data for the DICOM message as well as some metadata about it:

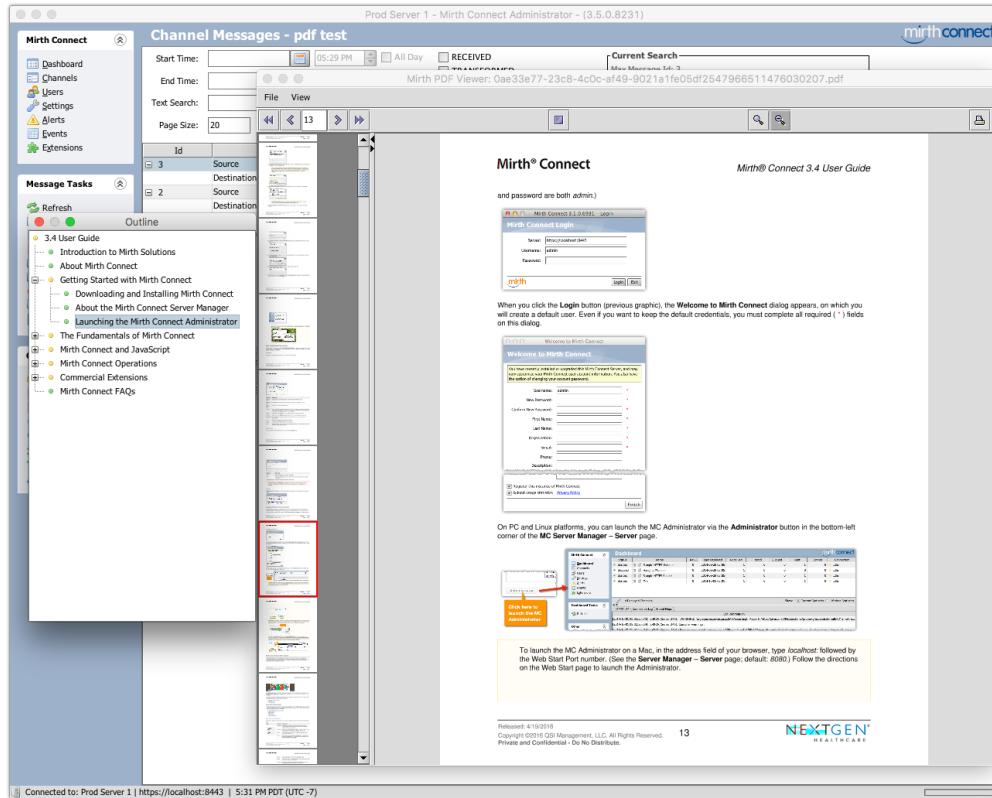


If the attachment has multiple image slices, use the horizontal scroll bar at the bottom to switch between them:



PDF Attachment Viewer

Use this for application/pdf type attachments. A PDF viewer pops up, allowing you to switch between pages, print the document, and more. If the PDF has a table of contents included, that is shown in a separate dialog to the side, allowing you to browse the outline and skip directly to a particular section:



Searching Messages

The **Message Browser** supports a wide range of searches, from a general search using a few filters, to finely grained searches using many filters and including the **Advanced...** dialog. The Advanced dialog allows you to enter a number of ID, content, and metadata filters to fine-tune your searches.

This screenshot shows the 'Channel Messages' search interface. It includes fields for Start Time, End Time, Text Search, and Page Size. A 'Common search options' section contains checkboxes for RECEIVED, TRANSFORMED, FILTERED, QUEUED, SENT, and ERROR. A 'Current Search' box displays search criteria: Max Message Id: 54, Date Range: (any) to (any), Statuses: (any), and Connectors: (any). Below the search bar is a table of message logs with columns for Trd, Connector, Status, Response Date, Error, and Type. Annotations explain various features:

- Max number of results to display in the metadata table at once:** Points to the Page Size dropdown.
- Additional Search Options:** Points to the 'Advanced...' button.
- Perform the Search:** Points to the 'Search' button.
- Reset all search criteria back to defaults:** Points to the 'Reset' button.
- Common search options:** Points to the checkboxes in the 'Common search options' section.
- Current Search:** Points to the 'Current Search' box.
- Messages shown in the metadata table follow these criteria:** Points to the search criteria in the 'Current Search' box.
- Pagination Options:** Points to the 'Page' dropdown and 'Count' button.

Message Search Options

This diagram shows the 'Channel Messages - A' search interface with numbered callouts:

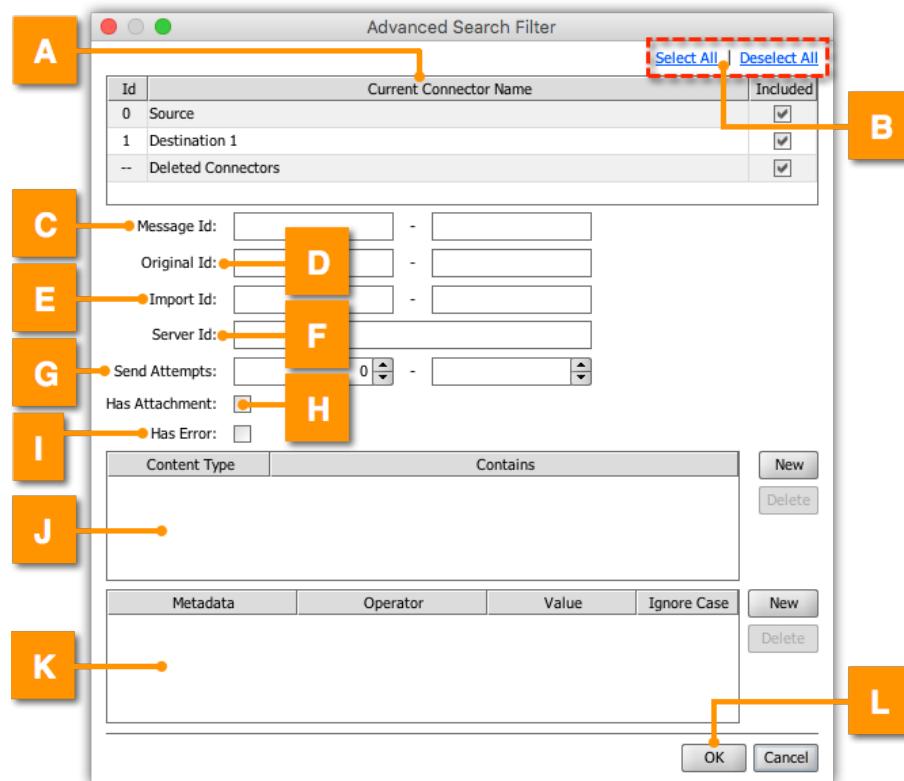
- A:** Start Time and End Time fields.
- B:** Text Search field and Regex checkbox.
- C:** Page Size dropdown.
- D:** Advanced... button.
- E:** Reset button.
- F:** Search button.
- G:** Common search options checkboxes (RECEIVED, TRANSFORMED, FILTERED, QUEUED, SENT, ERROR).
- H:** Current Search box.
- I:** Results 1-1 of 1 and Count button.
- J:** Page dropdown and Go button.
- K:** < Prev button.
- L:** Next > button.

Item	Name	Description
A	Start /End Time	Use the Calendar icons to populate these fields with the start/end dates of the desired messages; use the Hour fields to restrict the search to a range of hours, or check the All Day box to include messages that started/ended at all hours of the day.
B	Regex	Search all textual message content that matches the regular expression pattern given in the Text Search field. Regex matching could be a very costly operation and should be used with caution, especially with a large amount of messages. Any message content that was encrypted by this channel will not be searchable. The Regex option is only supported on PostgreSQL, Oracle, and MySQL databases.
C	Page Size	The maximum amount of messages that will appear in the Metadata Table at once. The default value is 20, but you can change that in the Administrator Settings Tab .
D	Text Search	Enter text (e.g., patient name, IP address) to include message content, metadata, and connector names containing the entered text. When you enter text in this field and click the Search button, the Select an Option dialog appears, informing you that <i>Text searching may take a long time, depending on the amount of messages being searched</i> , then asks

		<i>Are you sure you want to proceed?</i> Click the Yes or No button as desired. You can disable / enable the search confirmation dialog in the Administrator Settings Tab .
E	Advanced...	Reveals the Advanced Search Filter dialog consisting of more specific search-filter options. If any advanced search options have been set, this button will appear in bold text .
F	Reset	Removes all filters from the previous search and resets all search options to their defaults.
G	Search	Performs the message search with the currently set search options.
H	Statuses	Check these boxes to search for messages with <i>RECEIVED</i> , <i>TRANSFORMED</i> , <i>FILTERED</i> , <i>QUEUED</i> , <i>SENT</i> , and/or <i>ERROR</i> statuses.
I	Current Search (Filters)	The filters you selected for your search. The Current Search box displays various data relative to the most recent search. Note that when you perform a search, the server returns the Max Message ID that matched your search criteria, and that is shown as an additional criterion in this box. This is so that your search remains consistent and does not include new messages since your last search when you perform one of the Refresh / Export / Reprocess / Remove tasks. For additional information on tasks, see Message Browser Tasks .
J	Count	Performs a COUNT query on the server, and returns how many of the channel's messages match the search criteria. This number may be greater than the total number of messages currently shown in the Metadata Table because of the Page Size option.
K	Page	(Multi-page lists) Activates when you click the Count button Enter a value (coinciding with the page range) in this field, and click the Go button to display the desired page.
L	< Prev / Next >	(Multi-page lists) Click the < Prev button to go back one page or the Next > button to advance one page.

Advanced Search Filter

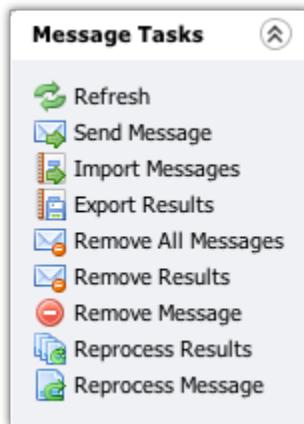
Click the "**Advanced...**" button in the message browser search pane to display the Advanced Search Filter. (See Option E in the previous section.)



Item	Name	Description
A	Connectors Table	Use this table to include only certain connectors in the search criteria. The Id column refers to the metadata ID of the connector, where the Source connector is always 0, and destination connectors will have metadata IDs greater than zero. The Current Connector Name column shows the current name of each connector. Even if a destination gets renamed, the metadata ID will remain static, and will be used for the search. The Deleted Connectors row refers to historical connector messages associated with destinations that have since been deleted.
B	Select /Deselect All	Click to select/deselect all connectors in the Connectors Table .
C	Message Id	The minimum and maximum message IDs to include in the search. This could refer to the current ID (Message Id), the original ID of a reprocessed message (Original Id), or the original ID of an imported message (Import Id)
D	Original Id	
E	Import Id	For example with Message Id , enter 1 in the first field and 10 in the second field to find messages 1-10. Enter 1 in both the first and second field to narrow the search down to only message 1.
F	Server Id	The ID of the server that processed the message.
G	Send Attempts	Enter min (first field) / max (second field) values to find messages with the number of send attempts in that range.
H	Has Attachment	If checked, the search only includes messages with one or more attachments.

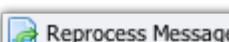
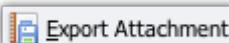
I	Has Error	If checked, the search only includes messages with one or more errors .
J	Content Search Table	<p>Click the box's New button to add a search filter, then click the row in the Content Type column, and select a menu option.</p> <p>Double-click the row in the Contains column, and enter your search string. When multiple rows are present in this table, only connector messages that match all entries will be included in the search.</p> <p>The available content types include message content, variable map content, and error content.</p>
K	Metadata	<p>Click the box's New button to add a search filter, then click the row in the Metadata column, and select a custom-metadata menu option.</p> <p>Click the row in the Operator column, and select a search operator (=, <i>CONTAINS</i>, <i>STARTS/ENDS WITH</i>), which determines the metadata to search for.</p> <p>Double-click the row in the Value column and enter your search string.</p> <p>Ignore Case: If checked, the search is not case-sensitive.</p> <p>When multiple rows are present in this table, only connector messages that match all entries will be included in the search.</p>
L	OK / Cancel	Click the OK button to save or the Cancel button to discard your changes.

Message Browser Tasks



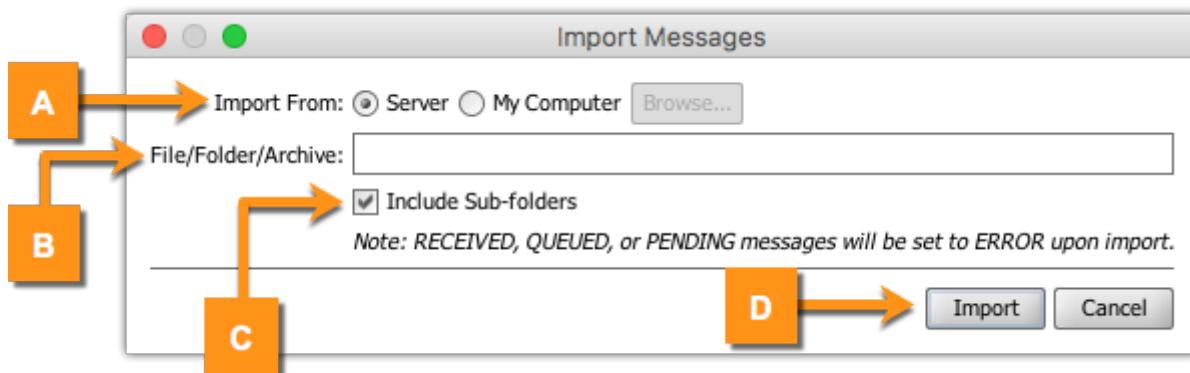
The following context-specific tasks are available from the Message Browser View:

Task Icon	Task Name	Description
	Refresh	<p>Updates messages within the current search page. Note that this is different from clicking the Search button. The Search operation performs a completely new search without a Max Message Id and resets the count and pagination options.</p> <p>The Refresh task re-performs the search using the existing Max Message Id, so that no new messages since the last Search will be included. Also, it preserves the current page being viewed instead of reverting back to the first page.</p>
	Send Message	Sends a message to the current channel. For additional information, see Send Message .
	Import Messages	Imports messages from a file. Note that imported messages are only directly stored in the channel's message data, and not actually processed through the channel.
	Export Results	Exports all messages that match the current search criteria. Note that this will include messages on all pages, not just the messages in the current page. To see how many messages will be exported, click the Count button.
	Remove All Messages	Removes all messages and attachments stored for the current channel.
	Remove Results	Removes all messages that match the current search criteria. Note that this will include messages on all pages, not just the messages in the current page. To see how many messages will be removed, click the Count button.
	Remove Message	Removes a single connector message. Note that removing a Source connector message will also remove all of its associated destination connector messages.

 Reprocess Results	Reprocess Results	Reprocesses all messages that match the current search criteria. Note that this will include messages on all pages, not just the messages in the current page. To see how many messages will be reprocessed, click the Count button.
 Reprocess Message	Reprocess Message	Reprocesses a single message. You can choose to copy and send the message as a completely new message, or to overwrite the current message.
 View Attachment	View Attachment	Opens up a viewer dialog to display the attachment in the Administrator. This task is only visible when an attachment is selected in the Attachments Tab . For additional information, see Viewing Attachments .
 Export Attachment	Export Attachment	Exports the attachment data to disk. This task is only visible when an attachment is selected in the Attachments Tab .

Import Messages

Note that only messages that have been exported with the "XML serialized message" option are able to be imported.



Item	Name	Description
A	Import From	If Server is chosen, messages will be imported from the server filesystem. If My Computer is chosen, messages from the machine running the Administrator will be imported into the server. Click the Browse button to select a file or folder from your local computer.
B	File /Folder /Archive	Select a specific message XML file, a folder containing message XML files, or a zip/tar.gz/tar.bz2 compressed archive containing message XML files.
C	Include Sub-folders	If checked, all sub-directories within the selected folder will be searched recursively for messages to import.
D	Import	Click this to perform the import operation.



The message appears on the list with its contents in the **Messages** window at the bottom of the page.

Newly imported message

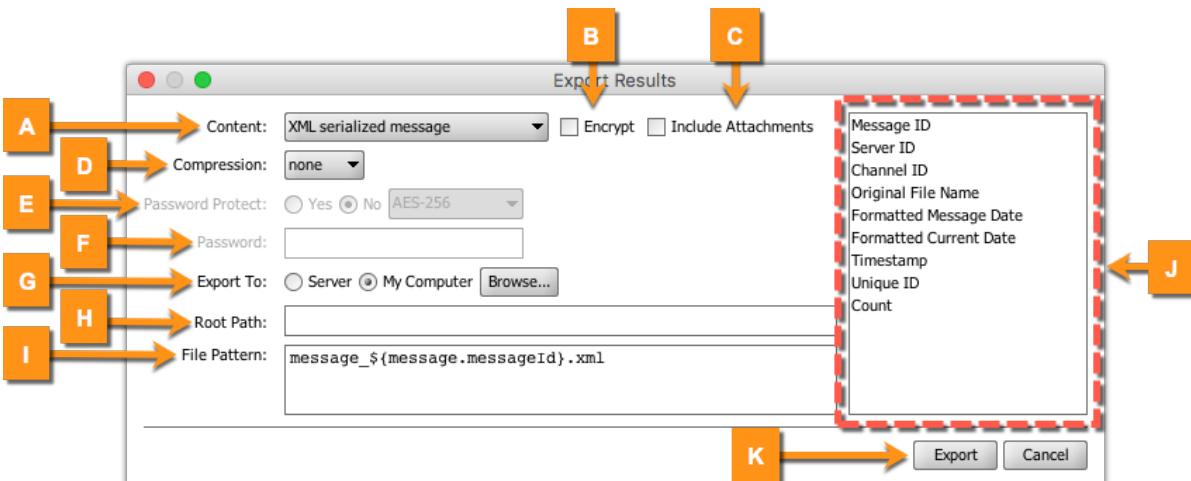
A unique identifier for each imported message appears in the **Import Channel Id** column. This alphanumeric code lets you readily identify messages that have been imported into the channel.

- If you import a message into the channel from which it originated, an Import Channel Id is not assigned, and the row in that column will show a dash (—), by which you can identify the message as original to the selected channel.

The **Import Channel Id** column (and others) can be chosen via the Column Options icon at the rightmost side of the Message Lists' column header. For additional information, see [Showing / Hiding Columns](#).

Export Results

When you click the Export Results task, an options dialog pops up. Note that **all** messages matching the current search criteria will be exported, not just the ones shown on the current page.

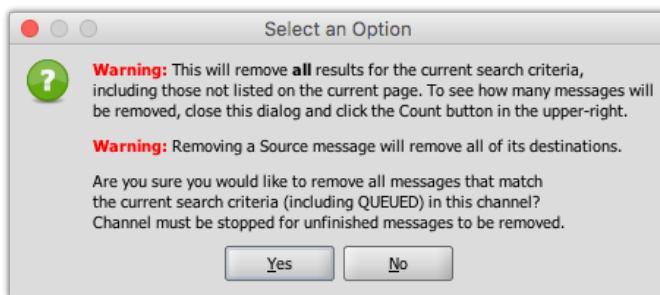


Item	Name	Description
A	Content	The type of content to export from the messages. Note that XML serialized message is the only type that can be re-imported, and it includes all content across the message in a single XML file.

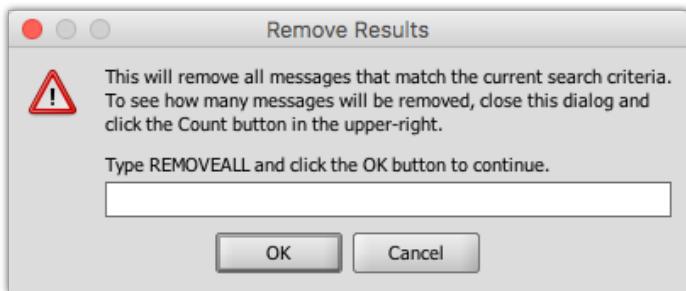
B	Encrypt	If checked, the exported message content will be encrypted with the server's encryption key.
C	Include Attachments	If checked and the content type is set to XML serialized message , the exported file will contain all attachments associated with the message.
D	Compression	When compression is enabled, the files/folders created according to the File Pattern will be put into a compressed file in the Root Path. The following compression types are supported: ZIP, tar.gz, tar.bz2
E	Password Protect	Only available if ZIP compression is enabled. If Yes is chosen, the resulting ZIP file will be protected with the given password. You can also choose what encryption algorithm to use when password-protecting files. The following algorithms are supported: Standard, AES-128, AES-256
F	Password	If password protection is enabled, this password will be used.
G	Export To	Export messages directly on the server running Mirth Connect, or pull messages from the server and store them locally on the machine from which you are running the Administrator.
H	Root Path	The root path to store the exported files/folder or compressed file. If messages are being exported directly to the server, relative paths will be resolved against the Mirth Connect Server home directory.
I	File Pattern	The file/folder pattern in which to write the exported message files. The file pattern may use variables from the list on the right side of the dialog.
J	Variables	Contains several common variables to aid in populating the File Pattern. For additional information, see Velocity Variable Replacement .
K	Export	Click Export to perform the export operation.

Remove Results

When you click the Remove Results task, a confirmation dialog pops up. Note that **all** messages and associated attachments matching the current search criteria will be removed, not just the ones shown on the current page. Also, removing a Source connector message will also automatically remove all of its associated destination connectors as well.



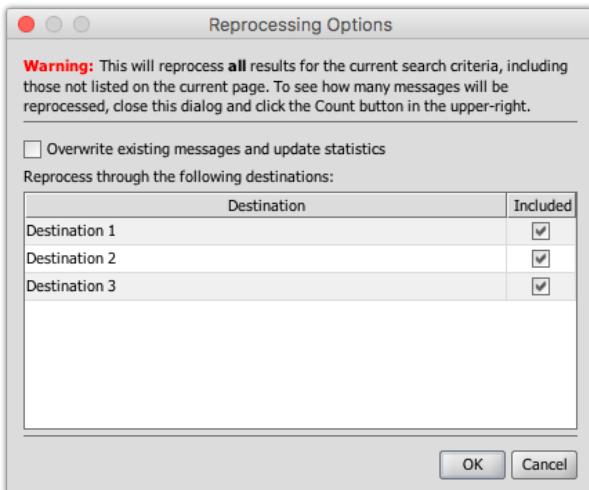
After clicking **Yes** to continue, you'll see the following additional dialog:



This is an additional layer of security to make sure you don't remove message data accidentally. To proceed you must type in "**REMOVEALL**" in all capital letters and click **OK**. Note, if you find this additional layer of security tedious, it can be disabled in the [Administrator Settings Tab](#).

Reprocess Results

When you click the Reprocess Results task, a confirmation dialog pops up. Note that **all** messages and associated attachments matching the current search criteria will be reprocessed, not just the ones shown on the current page.



Other than the additional warning, the options here are the same as in the [Reprocess Message](#) task.

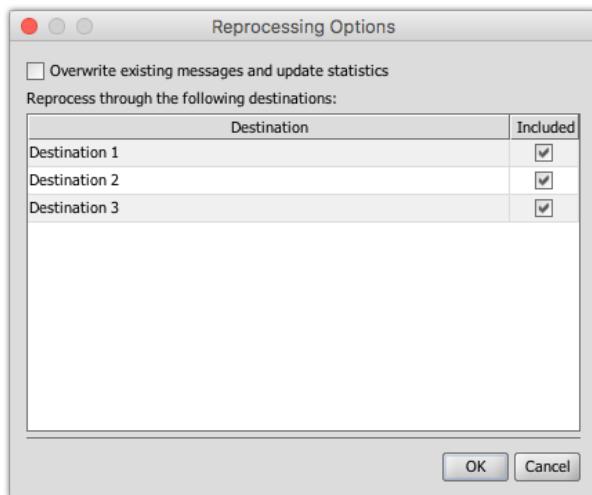
After clicking **Yes** to continue, you'll see the following additional dialog:



This is an additional layer of security to make sure you don't reprocess a large amount of messages accidentally. To proceed you must type in "**REPROCESSALL**" in all capital letters and click **OK**. Note, if you find this additional layer of security tedious, it can be disabled in the [Administrator Settings Tab](#).

Reprocess Message

When you click the Reprocess Message task, a confirmation dialog pops up. You are given the option to overwrite the existing message or process the data through the channel as a completely new message. You can also choose to exclude certain destinations from being executed during the reprocess operation. Note that the message will always flow through the Source connector first, regardless of which destinations you exclude.

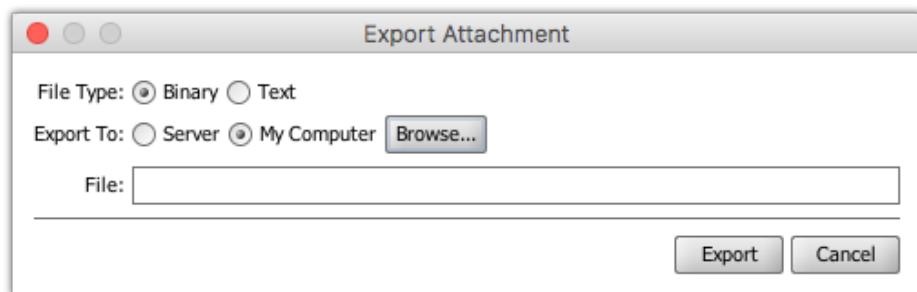


When a message is reprocessed, two new [Source Map](#) variables will be injected into the message:

- **reprocessed**: Boolean indicating that the current message is the result of a reprocess operation. When present, the value of this variable will always be **true**.
- **replaced**: Boolean indicating that the current message has been overwritten as a result of a reprocess operation.

Export Attachment

When you click the Export Attachment task, a confirmation dialog pops up:



Item	Description
File Type	Choose Binary to write the raw bytes of the attachment out to file. Choose Text to write the Base64 encoded representation of the attachment out.
Export To	Choose Server to export the attachment directly to the machine running the Mirth Connect Server. Choose My Computer to pull the attachment down to the client side and write it out to a location on the machine running the Administrator.

File

The location where the attachment will be saved.

Alerts View

An **alert** is a process that listens for certain types of events and triggers based on configurable settings. From these triggers you can take various actions, like dispatching an e-mail to a user or specific address, or sending a message to a channel. NextGen Connect Integration Engine comes with a built-in, Error-based, alerting system that listens for error events from selected channels. The [Advanced Alerting](#) extension adds on this by also including powerful metric-based alerts, escalation levels, scheduling, notification throttling, and other advanced features.

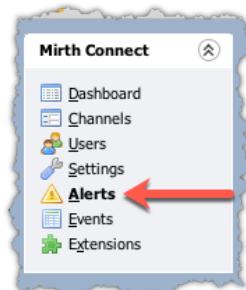
The Alerts View is like a dashboard for your currently configured alerts. You can view which ones are enabled and how many times an alert has triggered since it was last enabled.

Status	
Enabled	ADT Demographics Errors
Enabled	Consent Handler Alert
Enabled	ORU Lab Results Errors
Disabled	Test Alert 1
Disabled	Test Alert 2

	Id	Alerted
1dc41386-7ccf-4f5f-949a-7ac58192b90e	5	
2e82f27c-da3e-434f-909b-acfd0143b0ae	0	
5e52e340-b49b-4dd6-bb90-19625579f960	0	
a5bfdc3e-2f4b-4266-b52d-e0cdf2ff445e	--	
c2c9b11f-e9ad-4d64-b50a-bf0090e5cccd0	--	

Navigation

Click the **Alerts** link in the **Mirth Connect** task panel at the upper-left:



This section is separated into the following topics:

- [Alerts Table](#)
- [Alerts Tasks](#)

Alerts Table

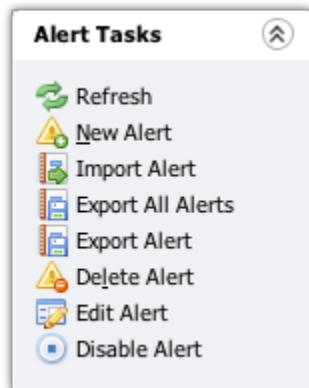
This is the main section of the Alerts View that shows the current status and action counters for all your configured alerts. For general information about working with tables in NextGen Connect Integration Engine, see, [Working With Tables](#).

Status	Name	Id	Alerted
Enabled	ADT Demographics Errors	1dc41386-7ccf-4f5f-949a-7ac58192b90e	0
Enabled	Consent Handler Alert	2e82f27c-d33e-434f-909b-acfd0143b0ae	0
Enabled	ORU Lab Results Errors	5e52e340-b49b-4dd6-bb90-19c525579f960	0
Disabled	Test Alert 1	a5bfdc3e-2f4b-4266-b52d-e0cdf2ff445e	--
Disabled	Test Alert 2	c2c9b11f-e9ad-4d64-b50a-bf0090e5ccd0	--

Alerts Table Columns

Column	Description
Status	Indicates whether the alert is enabled or disabled.
Name	The name of the alert.
Id	The unique ID for the alert.
Alerted	For enabled alerts, the number of times the alert has taken action (e.g. sent an e-mail) since it was last enabled.

Alerts Tasks



The following context-specific tasks are available from the Alerts View:

Task Icon	Task Name	Description
Refresh	Refresh	Updates the Alerts Table. Note that the Alerts view automatically refreshes at an interval defined in the Administrator Settings .
New Alert	New Alert	Creates a new alert and enters the Edit Alert View .
Import Alert	Import Alert	Imports an alert from an XML file on disk.
Export All Alerts	Export All Alerts	Exports all alerts to separate XML files on disk.
Export Alert	Export Alert	Exports a single alert to an XML file on disk.
Delete Alert	Delete Alert	Removes the alert from the server.
Edit Alert	Edit Alert	Enters the Edit Alert View for the selected alert.
Enable Alert	Enable Alert	Activates the alert so that it will begin receiving and acting upon events.
Disable Alert	Disable Alert	Deactivates the alert so that it will stop receiving and acting upon events.

Events View

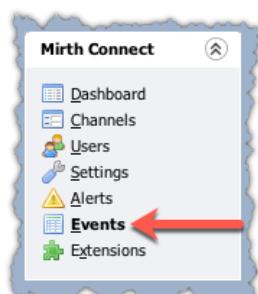
The **Event Browser** allows you to view and search all user / system events that have occurred on your server. This includes any user actions like logging in and modifying / deploying channels. It also include system events like startup / shutdown, and the [Data Pruner](#).

Level	Date & Time	Name	Server ID	User	Outcome	IP Address
i	2017-04-19 12:10:01-582	Get libraries invoked through Directory ...	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-19 12:10:01-490	Get resources	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-19 12:09:56-115	Login	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-19 12:01:43-780	Error executing d6bb6070-73fb-4cc4-90...	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 12:01:43-708	Error executing 2c1994f6-c3d4-49a2-94...	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 12:01:38-064	Server startup	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 07:38:17-834	Server shutdown	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 07:38:10-111	Logout	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-19 00:00:00-082	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 00:00:00-032	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-19 00:00:00-020	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 16:04:06-822	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 15:59:56-952	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 15:58:50-239	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 15:58:50-207	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 15:58:50-190	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 15:58:50-175	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 15:58:50-172	Start pruner invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:42:42-714	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:41:02-894	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:33:49-634	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:33:49-404	Set plugin properties invoked through D...	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:56-559	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:47-264	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 14:32:46-997	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
i	2017-04-18 14:32:46-894	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:46-857	Start pruner invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:45-397	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:41-452	Update channel	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
i	2017-04-18 14:32:34-255	Update channel	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1

Name	Value
Messages Pruned	62
Content Rows Pruned	620
Channel ID	cece0124-2b8c-415e-ab8d-3a5e9b0feecd
Channel Name	SamWorther Tech - TRN - 7715
Time Elapsed	0 minutes, 0 seconds

Navigation

Click on the **Events** link in the **Mirth Connect** task panel in the upper-left:



This section is separated into the following topics:

- [Events Table](#)
- [Event Attributes Table](#)
- [Searching Events](#)
- [Event Tasks](#)

Events Table

This table shows metadata (ID, timestamp, operation name) about each event that has been stored by the server. The Page Size option dictates the number of events shown in the table simultaneously (for additional information, see [Searching Events](#)). When you first enter the Event Browser, by default the latest 100 events appear. For general information about working with tables in NextGen Connect Integration Engine, see [Working With Tables](#).

Level	Date & Time	Name	Server ID	User	Outcome	IP Address
INFO	2017-04-19 12:10:01:582	Get libraries invoked through Directory ...	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-19 12:10:01:490	Get resources	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-19 12:09:56:115	Login	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
ERROR	2017-04-19 12:01:43:780	Error executing d6bb6070-73fb-4cc0-90... ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)		✓	
ERROR	2017-04-19 12:01:43:708	Error executing 2c1994f6-c3d4-49a2-94... ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)		✓	
INFO	2017-04-19 12:01:38:060	Server startup	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-19 07:38:17:838	Server shutdown	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-19 07:38:10:143	Logout	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-19 07:38:10:111	Set user preferences	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-19 00:00:00:082	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-19 00:00:00:032	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-19 00:00:00:020	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 16:04:06:822	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 15:59:56:952	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 15:58:50:230	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 15:58:50:207	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 15:58:50:190	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 15:58:50:175	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 15:58:50:172	Start pruner invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:42:42:714	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:41:02:892	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:33:49:635	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:33:49:404	Set plugin properties invoked through D...	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:56:550	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:47:262	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 14:32:46:997	Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	0 (System)	✓	
INFO	2017-04-18 14:32:46:891	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:46:857	Start pruner invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:45:397	Get status invoked through Data Pruner	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:41:452	Update channel	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1
INFO	2017-04-18 14:32:34:255	Update channel	ceae9089-431e-4156-8b25-baa77594e7c4	1 (admin)	✓	127.0.0.1

Metadata Table Columns

Column	Description
ID	The unique ID of the event in the database. This column is hidden by default. For additional information, see Showing / Hiding Columns .
Level	An icon indicating the event's severity level (Information , Warning , or Error).
Date & Time	The date and time the event was logged.
Name	The name of the logged operation, or a description of the warning / error.
Server ID	The unique ID of the server from which the event was logged.
User	If the event originated from a user action, this will be both the user ID and user name. If no user is associated with the event, it will show 0 (System) .
Outcome	For user requests and server tasks, this icon indicates whether the action was successful or not. For example a failed user login attempt will show a failure icon.
IP	For user requests, the IP address of the originating client.

Address

Event Attributes Table

The attributes table at the bottom of the Event Browser shows more detailed information about a particular event. This will typically include the channel ID / name, if the event is a channel operation like start/stop/deploy. If the event is an error, the attributes table will typically show any exception associated with the event.

Name	Value
Messages Pruned	62
Content Rows Pruned	620
Channel ID	cecee124-2b8c-415e-ab8d-3a5e9b8feecd
Channel Name	SamWorther Tech - TRN - 7715
Time Elapsed	0 minutes, 0 seconds

If an attribute value is too long and is truncated in the table, you can double-click the table row to display the value in a separate dialog:

The screenshot shows the Event Browser interface. At the top, there's a list of log entries. In the center, a modal dialog titled "Mapping Value" displays a Java stack trace. At the bottom, there's a table titled "Exception". An orange callout box points to the first row of the "Exception" table, which contains the text "com.mirth.connect.server.MirthJavascriptTransformerException: CHANNEL: Blaster 2 SOURCE CODE: 51: fu...".

Name	Value
Exception	com.mirth.connect.server.MirthJavascriptTransformerException: CHANNEL: Blaster 2 SOURCE CODE: 51: fu...

Double-click here to view entire value

Searching Events

The **Event Browser** search capabilities are very similar to the Message Browser. For information on many of these shared components, see [Searching Messages](#).

The following search options are unique to the Event Browser:

The screenshot shows the 'Events' search interface. At the top, there are fields for 'Start Time' and 'End Time' (both set to 12:10 PM), a 'Name' search field (labeled A), and a 'Page Size' dropdown set to 100. To the right of these are three checkboxes for 'INFORMATION', 'WARNING', and 'ERROR' levels, which are highlighted with a red dashed box and labeled B. Below the search controls is a table of event logs. The first two rows are visible, showing timestamp, name, and server ID.

Level	Date & Time	Name	Server ID
Info	2017-04-19 12:10:01:582	Get libraries invoked through Directory ...	ceae9089-431e-4156-8b25-baa77594e7c4
Info	2017-04-19 12:10:01:490	Get resources	ceae9089-431e-4156-8b25-baa77594e7c4

Item	Name	Description
A	Name	The operation name or a short description of the event's warning / error. All events that contain this string somewhere in the name (case-insensitive) will be returned.
B	Level	Check these boxes to search for events with the <i>INFORMATION</i> , <i>WARNING</i> , and/or <i>ERROR</i> levels.

Advanced Search Filter

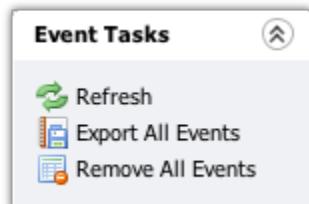
Click on the "Advanced..." button in the event browser search pane to open the Advanced Search Filter:

The dialog box has several search fields: 'User' (set to ALL), 'Outcome' (set to ALL), 'IP Address' (empty), and 'Server ID' (empty). Arrows labeled C, D, E, and F point from the table below to these respective fields. At the bottom are 'OK' and 'Cancel' buttons.

Item	Name	Description
C	User	The user associated with the event. The System user is reserved for events where no user is associated, such as startup / shutdown or the Data Pruner.
D	Outcome	For user requests and server tasks, Outcome indicates whether the action was successful or not.
E	IP	For user requests, the IP address of the originating client.

	Address	
F	Server ID	The unique ID of the server from which the event was logged.

Event Tasks



The following context-specific tasks are available from the Events View:

Task Icon	Task Name	Description
	Refresh	<p>Updates events within the current search page. Note that this is different from clicking the Search button. The Search operation performs a completely new search without a Max Event Id and resets the count and pagination options.</p> <p>The Refresh task re-performs the search using the existing Max Event Id, so that no new events since the last Search will be included. Also, it preserves the current page being viewed instead of reverting back to the first page.</p>
	Export All Events	Exports all events to a file in the Application Data Directory .
	Remove All Events	Deletes all events from the database. You will be prompted to optionally export all events first.

Export All Events

When you click this task, a confirmation dialog appears:



Click **Yes** to continue. A comma-separated value (CSV) file will be written into an "exports" folder inside your [Application Data Directory](#).

Management Views

This section is separated into the following topics:

- [Channels View](#)
- [Users View](#)
- [Settings View](#)
- [Extensions View](#)

Channels View

The Channels view is the main management screen for all channels configured on your NextGen Connect Integration Engine server. From this view you can create, delete, import / export, clone, enable / disable, and deploy channels. It also serves as a management interface for channel groups.

Status	Data Type	Name	Id	Local Id	Description	Rev Δ	Last Deployed	Last Modified
Enabled	HL7 v2.x	[Default Group] Create and Write PDF	9256b9c1-f558-432a-874e-f002201b79e8	5	Channels not part of...	--	2017-04-12 07:22	2017-04-05 17:17
Enabled	HL7 v2.x	LLP Inbound 22341	77aa6487-7810-45b3-9b10-105e72cf57b	3		0	2017-04-12 07:22	2017-04-05 17:17
Enabled	HL7 v2.x	Mirth Results Sender PROD	8d04a98-3057-49e0-a529-765d9582759	4		0	2017-04-12 07:22	2017-04-05 17:17
Enabled	HL7 v2.x	Move HL7 Segments Single Line	48f1163a-59ed-41d4-83de-65ad9cbba0d	6		0	2017-04-12 07:22	2017-04-06 11:51
Enabled	HL7 v2.x	Repeated Segs and Fields Iteration	77fe70b2-4302-4537-9530-938e106674	7		0	2017-04-12 07:22	2017-04-06 10:44
Enabled	HL7 v2.x	test	140eb063-01db-45ec-80e8-40e9f09e64c8	42		0	2017-04-13 12:26	2017-04-13 12:26
Enabled	HL7 v2.x	test2	25a442a3-a004-4d1f-a7b8-6d066bed7127	38		0	2017-04-12 13:23	2017-04-12 13:23
Enabled	Raw	Alert Processors	70715179-7848-4969-acce-3cb2da79b0c7	--	Channels that receive...	--	--	2017-04-10 08:15
Enabled	Raw	Create Salesforce Ticket	7591b3e8-fcce-41c1-a117-79177f5b51	12		0	2017-04-12 07:22	2017-03-20 10:49
Enabled	Raw	Notification Receiver - ADT	1a459116-c7a1-47fc-900d-925305688682	10	Demographics alerts...	0	2017-04-12 07:22	2017-03-20 10:49
Enabled	Raw	Notification Receiver - RAD	e3a42187-2ded-4caa-8a03-12d57661f14	23		0	2017-04-12 07:22	2017-03-20 10:49
Enabled	HL7 v2.x	Notification Receiver - Reprocess	d0fe3f76-e2e7-4b8d-8c33-19fd06ed12b8	30	Reprocess messages...	0	2017-04-12 07:22	2017-03-20 10:49
Enabled	Raw	Notification Receiver - TRN	daefcfdf-8ef1-4be0-80f4-93f2fedec252	13		0	2017-04-12 07:22	2017-03-20 10:49
Enabled	Raw	Audition Health	0c1e09b-037f-40c4-a662-d24f8546fb37	--	--	--	--	2017-04-10 08:15
Enabled	HL7 v2.x	Audition Health - ADT	582ca9b-5a5d-43ef-9546-c12edcd1d47a	8	Source ADT from Au...	0	2017-04-12 07:22	2017-04-11 11:06
Enabled	HL7 v2.x	Audition Health - LAB	83834bf5-473e-4cc8-bd15-c76d21547eab	14	Source LAB from Aud...	0	2017-04-12 07:22	2017-03-20 10:49
Enabled	HL7 v2.x	Audition Health - RAD	36a1ca3f-5d8a-4eb2-9f66-3338c0012b0	27	Source RAD from Au...	0	2017-04-12 07:22	2017-03-20 10:49
Enabled	HL7 v2.x	Audition Health - TRN	44d0388b-343e-41be-b4f4-7442b2a199d6	15	Source TRN from Au...	0	2017-04-12 07:22	2017-03-20 10:49
Enabled	HL7 v2.x	Audition Health - TRN	dd124a9e-313a-4cd7-9f8e-6004230771a5	26	Source TRN from Au...	2	2017-04-12 07:22	2017-04-13 14:57
Enabled	Raw	Calif Partn	ad7b5446-6064-4eeb-8214-9296e5080d00	--	--	--	--	2017-04-10 08:15
Enabled	HL7 v2.x	Calif Partn - ADT	575b2c1d-c8ec-4466-b666-a18a190e315	16		0	2017-04-12 07:22	2017-03-20 10:49
Enabled	HL7 v2.x	Calif Partn - LAB	58ab06f6-38b7-48a4-a18c-1a73e3f55404	20		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	HL7 v2.x	Calif Partn - RAD	c28502a-bf24-45c2-ab13-4cd491163e47	29		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	HL7 v2.x	Calif Partn - TRN	96d6210f-436a-4165-a27f-5a4e9ff939d5	21		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	Raw	Mirth Results	87c89700-4b76-4cd2-a580-9d1d4ec9beac	--	--	--	--	2017-04-10 08:15
Enabled	HL7 v3.x	CDA to MR - Default	513454f2-4109-4f68-8749-75e2914ed4c4	25		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	HL7 v3.x	CDA to MR - Epr	172c9f2e-47a5-4eb8-9b78-b5733faea9	18		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	HL7 v3.x	CDA to MR - Meditech	955951b-3f0f-481d-8013-9e92f651d32	9		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	Raw	Routers	5ab38ab-0880-4151-99ac-23bd162bc2f	--	--	--	--	2017-04-10 08:24
Enabled	HL7 v2.x	Blaster 1	2c1994f6-c304-4bec-942a-905f4f95787	31		0	2017-04-12 07:22	2017-04-10 08:23
Enabled	HL7 v2.x	Blaster 2	d6bb6b70-73fb-4cc9-909e-9a193e028d03	33		0	2017-04-12 07:22	2017-04-10 08:23
Enabled	XML	Cluster Router 1 - 6661	21019e3d-dc22-4dc2-83b6-2c4e5601d21	32		0	2017-04-12 07:22	2017-03-20 10:50
Enabled	Raw	SamWorther Tech	c3ba6f61-9b96-4e94-a007-28fbca7458e	--	--	--	--	2017-04-10 08:15
Enabled	HL7 v2.x	SamWorther Tech - ADT	751871a-b014-4d93-b46e-b8f982d439	11		0	2017-04-12 07:22	2017-03-20 10:51
Enabled	HL7 v2.x	SamWorther Tech - LAB	30844994-abf4-4cc2-a322-ee1abd010861	22		0	2017-04-12 07:22	2017-03-20 10:51
Enabled	HL7 v2.x	SamWorther Tech - RAD	5fda8eab-abb7-49fc-bad4-4361c57945	28		0	2017-04-12 07:22	2017-03-20 10:51
Enabled	HL7 v2.x	SamWorther Tech - TRN	2f211686-9ede-47a6-9817-5626c711a1d	17		0	2017-04-12 07:22	2017-03-20 10:51
Enabled	HL7 v2.x	SamWorther Tech - TRN	cece124-2b8c-415e-ab8d-3a5e9b8feecd	24		0	2017-04-12 07:22	2017-03-20 10:51
Enabled	Raw	Testing Group	d568a1e7-7002-4676-a8e8-15689739326	--	--	--	--	2017-04-13 12:26
Enabled	HL7 v2.x	halftest	11b07389-ab46-4dc5-a4b8-b73d0f35c32	37		0	2017-04-12 07:22	2017-04-11 11:55
Enabled	HL7 v2.x	image test	876059b1-c831-4eb7-8ea8-3601bc56ef40	40		0	2017-04-12 17:17	2017-04-12 17:17
Enabled	HL7 v2.x	pdf test	a6f3c085-21cd-465b-beb5-5a5e548222	41		0	2017-04-12 17:28	2017-04-12 17:28
Enabled	DICOM	Test DICOM Listener	b75af7cb-a55d-4e23-a03c-9795604b03	34		0	2017-04-12 07:22	2017-04-11 09:17
Enabled	DICOM	Test DICOM Sender	f7d24402-9ce4-4a9b-b063-202683931386	35		0	2017-04-12 07:22	2017-04-11 09:16

Navigation

Click the Channels link in the Mirth Connect task pane at the top-left:



This section is separated into the following topics:

- [Channel Table](#)
- [Channel Tasks](#)
- [Group Tasks](#)

Channel Table

This is the main section of the Channels view that shows the status and metadata for your currently configured channels. For general information about working with tables in NextGen Connect Integration Engine, see [Working With Tables](#).

The screenshot shows a table of channels with the following columns: Status, Data Type, Name, Id, Local Id, Description, Rev Δ, Last Deployed, and Last Modified. The table is sorted by Name.

Annotations on the interface:

- Click on headers to sort**: Points to the column headers for Name, Id, Local Id, Description, Rev Δ, Last Deployed, and Last Modified.
- Expand / collapse channel groups**: Points to a group header for 'Calif Parth' which contains multiple sub-channel entries.
- Filter by channel name / tags**: Points to a search bar at the bottom left labeled 'Filter: Enter channel tag or name'.
- Currently filtered channels / groups**: Points to the search results summary '9 Groups, 39 Channels, 39 Enabled'.
- Table Controls**: Points to the 'Table Controls' button in the top right corner.
- Toggle how channel tags are shown in the table**: Points to the 'Toggles group-level view' button in the bottom right corner.
- Toggles group-level view**: Points to the 'Toggles group-level view' button in the bottom right corner.

Channel Table Columns

Column	Description
Status	<p>Indicates whether the channel is enabled or disabled. A channel may only be deployed once it is enabled. When the NextGen Connect Integration Engine server starts up, all enabled channels are automatically deployed.</p> <p>For channel groups, this column is Enabled if all channels underneath the group are enabled, Disabled if all channels underneath the group are disabled, and Mixed if some channels are enabled and others disabled.</p>

Data Type	The inbound data type of the channel's source connector.
Name	The name of the channel / group. This column also shows any tags associated with a channel. A special [Default Group] group is present in the table and may not be removed. When viewing the channel table in group-level mode, all channels that are <i>not</i> part of a group will be organized into this group.
Id	The unique ID of the channel / group.
Local Id	The unique numeric ID of a channel used to identify the set of tables associated with the channel's messages, attachments, and statistics.
Description	The description for the channel / group.
Rev	The number of times the channel was saved since it was last deployed. This value will be highlighted if it is greater than 0, or if any code templates linked to the channel have changed since the channel was last deployed. Rev = Channel Revision - Deployed Revision
Last Deployed	The time this channel was last deployed. This value will be highlighted if it is within the last two minutes.
Last Modified	The time this channel was last modified.

Show or Hide Channel Groups

For information on how to show / hide channel groups, see [Show or Hide Channel Groups](#).

Change How Tags Are Displayed

For information on how to change how tags are displayed in the channel table, see [Change How Tags Are Displayed](#).

Filtering By Channel Name or Tag

For information on how to filter the channel table down to specific channels based on names or tags, see [Filtering By Channel Name or Tag](#).

Dragging and Dropping

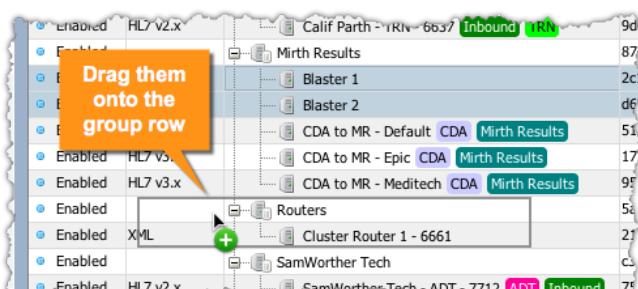
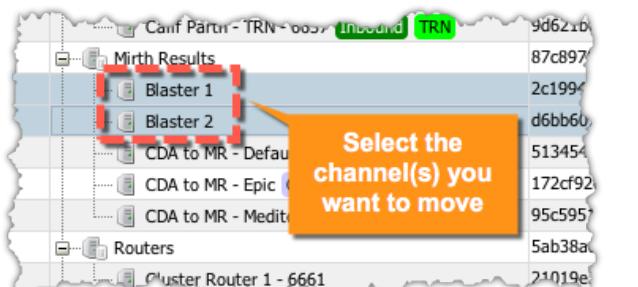
Get the Channel Name / ID

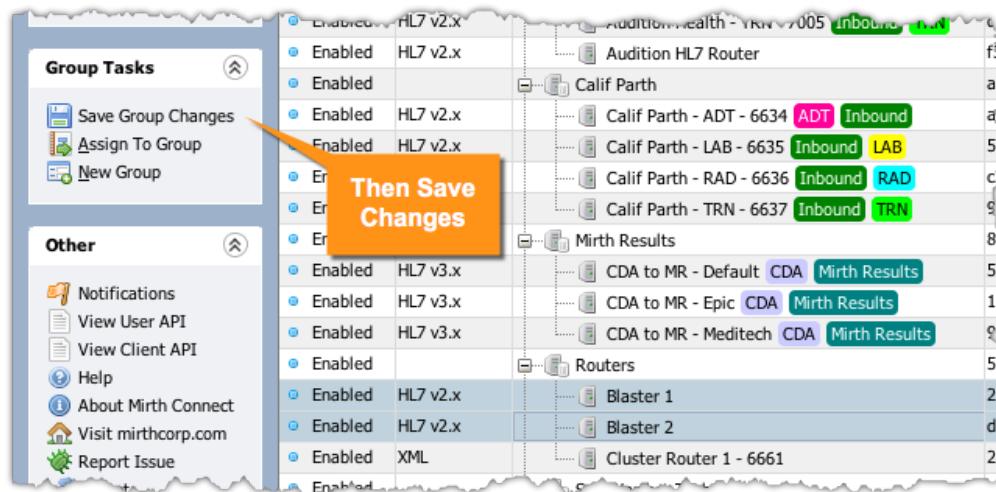
You can quickly copy channel names / IDs by selecting rows in the channel table and dragging them into the text editor of your choice:



Assign Channels to a Group

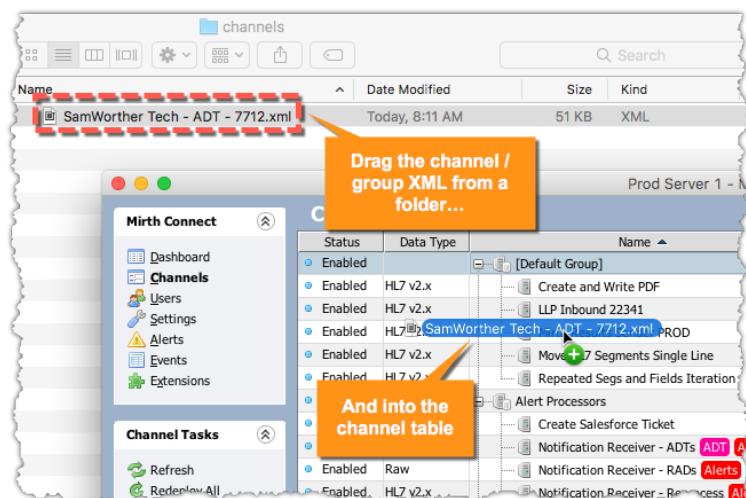
To move channels to a specific group, select the channel rows and drag them onto the group row:





Import Channels / Groups Using Drag-and-Drop

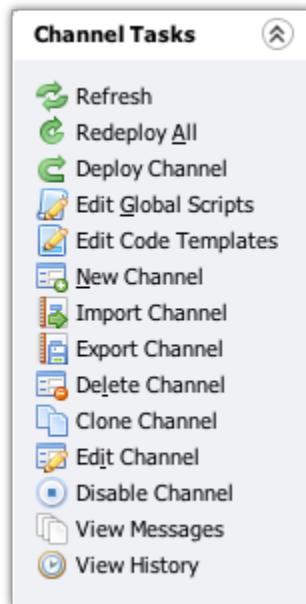
Importing channels / groups may be done using the corresponding **Import** task (for additional information see [Channel Tasks](#) and [Group Tasks](#)), or by simply dragging the XML files you wish to import from a folder into the channel table directly:



For channel files, the [Import Channel](#) workflow applies. For group files, the [Import Group](#) workflow applies.

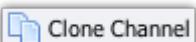
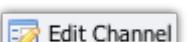
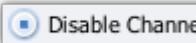
Unlike when using the task operations, importing using drag-and-drop also allows you to import multiple channels or multiple groups simultaneously, by multi-selecting XML files from a folder and dragging them into the channel table.

Channel Tasks



The following context-specific tasks for channels are available from the Channels View:

Task Icon	Task Name	Description
	Refresh	Updates the channel table.
	Redeploy All	Undeploys all channels, then deploys all currently enabled channels.
	Deploy Channel	Deploys the currently selected channel(s). If a group is selected, deploys all channels underneath that group. Note that the Channel Dependencies workflow may apply to this task.
	Edit Global Scripts	Enters the Edit Global Scripts View .
	Edit Code Templates	Enters the Edit Code Templates View .
	New Channel	Creates a new channel and enters the Edit Channel View . The channel is not yet saved .
	Import Channel	Imports a channel from an XML file and enters the Edit Channel View . The channel is not yet saved .
	Export Channel	Exports the selected channel(s) to their own respective XML files.
	Delete	Removes the selected channel(s) from the server. All message /

 Delete Channel	Channel	attachment data will also be deleted.
 Clone Channel	Clone Channel	Copies the channel and enters the Edit Channel View . The channel is not yet saved . Note that you will be prompted to give the channel a new unique name first.
 Edit Channel	Edit Channel	Enters the Edit Channel View for the selected channel. This can also be accessed by double-clicking the channel row.
 Enable Channel	Enable Channel	Mark this channel as ready to be deployed. Enabled channels will automatically be deployed when the server starts up or when the Redeploy All task is performed.
 Disable Channel	Disable Channel	Mark this channel as not ready to be deployed. The channel may not be deployed until it is re-enabled. However, if the channel was <i>already</i> deployed, then disabling it will not automatically undeploy it.
 View Messages	View Messages	Enters the Message Browser View for the selected channel.

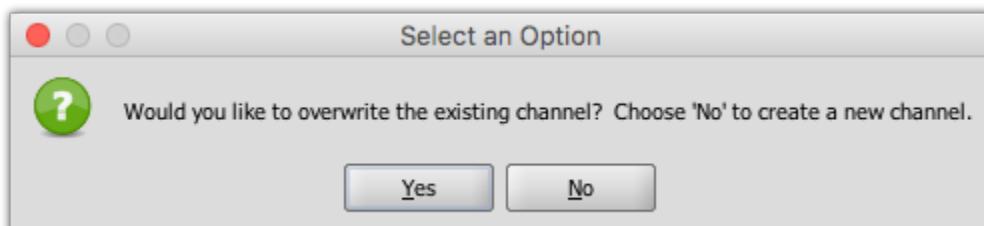
Import Channel

When importing channels, you have the option to include all [code template libraries](#) associated with the channel as well. A special dialog pops up for this purpose if any code template libraries were included: [Importing Code Templates / Libraries](#)

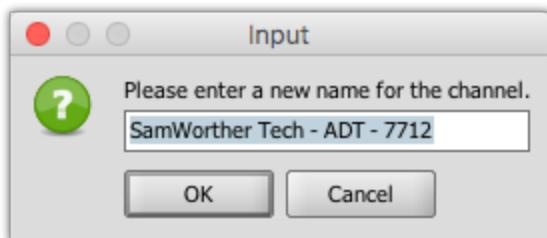
If the channel you're importing has the same name as an existing channel on the server, a warning dialog will be shown:



You will then be asked whether you want the imported channel to overwrite the existing channel with the same name:

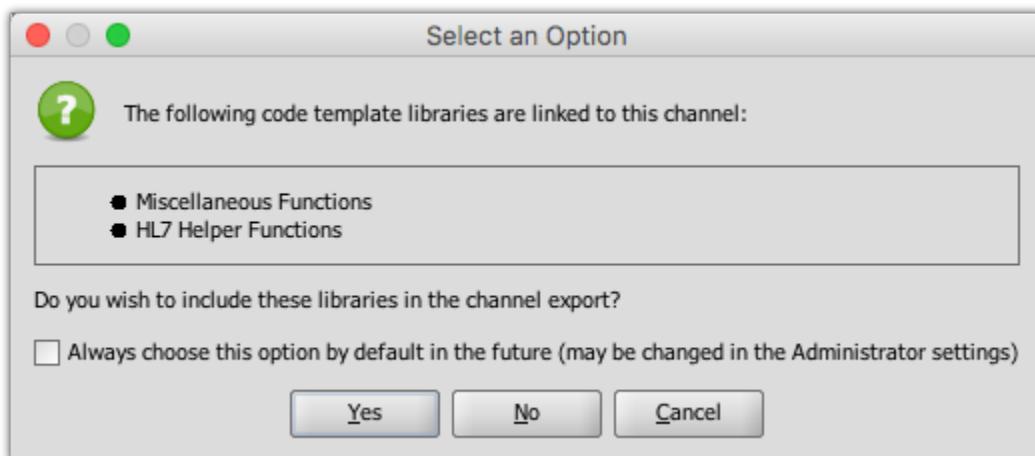


If **No** is chosen, you will then be prompted to enter a new unique name for the imported channel:



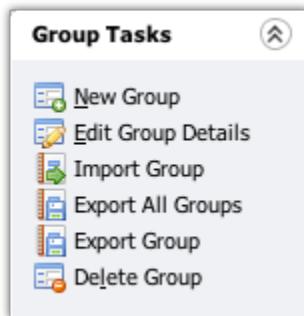
Export Channel

When exporting channels, you have the option to include all [code template libraries](#) associated with the channel as well:



If **Yes** is chosen, the code template libraries shown in the dialog will be included and embedded into the channel export XML. If you want to automatically make the same choice (Yes or No) for all future exports, you can check the "**Always choose**" check box. You can change this again later in the [Administrator Settings Tab](#).

Group Tasks

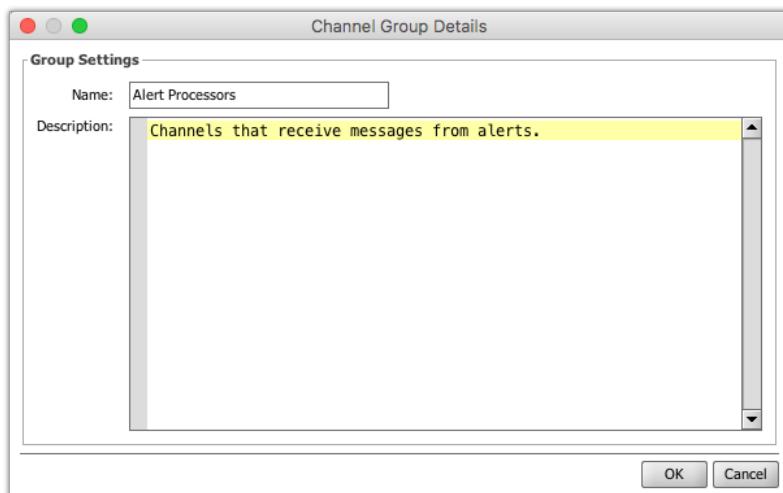


The following context-specific tasks for channel groups are available from the Channels View:

Task Icon	Task Name	Description
	Save Group Changes	Saves all changes made to channel groups and refreshes the Channels view.
	Assign To Group	Moves the selected channel(s) into a specific group.
	New Group	Creates a new channel group and adds it to the table. The user is prompted for the group name and description as described in the Edit Group Details task.
	Edit Group Details	Allows the user to edit the group name and description. This can also be accessed by double-clicking the group row.
	Import Group	Imports a channel group from an XML file.
	Export All Groups	Exports all channel groups in the table to their respective XML files. The Export Group workflow applies here.
	Export Group	Exports the selected channel group(s) to their respective XML files.
	Delete Group	Removes the channel group from the table. All channels that were within this group are automatically moved to the [Default Group] group.

Edit Group Details

A dialog is shown allowing you to edit the group name and description:

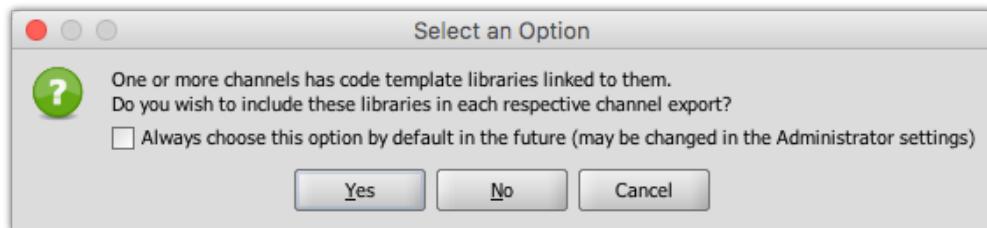


Import Group

The import group workflow follows the [Import Channel](#) workflow for each channel in the group.

Export Group

Similar to [exporting channels](#), when exporting channel groups you have the option to include all associated [code template libraries](#):



If **Yes** is chosen, the code template libraries shown in the dialog will be included and embedded into the channel group export XML. Whether you choose Yes or No, if you want to automatically make the same choice for all future exports, you can check the "**Always choose**" check box. That setting may be changed later in the [Administrator Settings Tab](#).

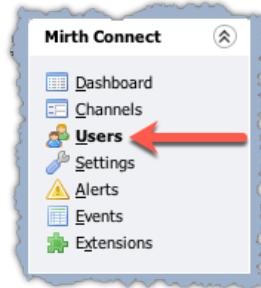
Users View

The Users view is the main management screen for all users that have access to login to Mirth Connect, whether it's via the [Administrator](#), the [Command Line Interface](#), or the [REST API](#). In this view you can add / remove / edit users. If you're looking for information on how to restrict access to specific operations to specific user roles, see [User Authorization](#).

Username	First Name	Last Name	Email	Phone Number	Organization	Industry	Last Login	Description
user0	First	Last0				HIE	2017-04-14 16:36:14	
recovery							2017-04-14 16:38:43	
donc	Don	Cervantes	donc@hiemazing.com			HIE		
mauryk	Maury	Kafka	mauryk@hiemazing.com			HIE		
susanc	Susan	Céline	susanc@hiemazing.com			HIE		
janeW	Jane	Wallace	janeW@hiemazing.com			HIE		
bobc	Bob	Camus	bobc@hiemazing.com			HIE		
junem	June	McCarthy	junem@hiemazing.com			HIE		
markn	Mark	Nabokov	markn@hiemazing.com			HIE		
nickr	Nick	Rupley	nickr@mirth.com	(949) 237-6069	Mirth Corporation	HIT Software	2017-04-10 09:17:21	
marym	Mary	Melville	marym@hiemazing.com			HIE	2017-04-10 09:33:35	
test								
test2								
test3								
admin							2017-04-17 07:42:19	

Navigation

Click the Users link in the Mirth Connect task pane at the top-left:



This section is separated into the following topics:

- [Users Table](#)
- [Users Tasks](#)

Users Table

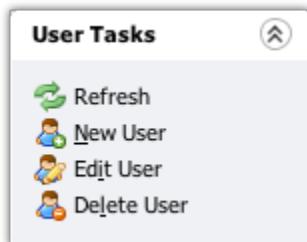
This is the main section of the Users view that shows details about all the users configured on your server. For general information about working with tables in Mirth Connect, see [Working With Tables](#).

Username	First Name	Last Name	Email	Phone Number	Organization	Industry	Last Login	Description
user0	First	Last0					2017-04-14 16:36:14	
recovery							2017-04-14 16:38:43	
dunc	Don	Cervantes	dunc@hiemazing.com			HIE		
mauryk	Maury	Kafka	mauryk@hiemazing.com			HIE		
susanc	Susan	Céline	susanc@hiemazing.com			HIE		
janew	Jane	Wallace	janew@hiemazing.com			HIE		
bobc	Bob	Camus	bobc@hiemazing.com			HIE		
junem	June	McCarthy	junem@hiemazing.com			HIE		
markn	Mark	Nabokov	markn@hiemazing.com			HIE		
nickr	Nick	Rupley	nickr@mirth.com	(949) 237-6069	Mirth Corporation	HIT Software	2017-04-10 09:17:21	
marym	Mary	Melville	marym@hiemazing.com			HIE	2017-04-10 09:33:35	
test								
test2								
test3								
admin							2017-04-17 07:42:19	

Users Table Columns

Column	Description
Username	The unique name the user will use to login with.
First Name	The user's first (given) name.
Last Name	The user's last (family) name.
Email	The e-mail address of the user. When sending an alert to a user, this e-mail address will be used.
Phone Number	The user's phone number.
Organization	The user's organization.
Industry	The user's work field / industry (e.g. HIE, Hospital, Lab).
Last Login	The date and time of the user's last successful login.
Description	A description or additional details for the user.

Users Tasks



The following context-specific tasks are available from the Users View:

Task Icon	Task Name	Description
Refresh	Refresh	Updates the users table.
New User	New User	Creates a new user and adds it to the table.
Edit User	Edit User	Edits the details of an existing user.
Delete User	Delete User	Removes the user from the server.

Creating / Editing Users

When creating a new user or editing an existing user, the following dialog appears:

The screenshot shows a 'User' application window titled 'New User'. It contains a form with the following fields:

Username:	donc	*
New Password:	*****	*
Confirm New Password:	*****	*
First Name:	Don	
Last Name:	Cervantes	
Email:	donc@hiemazing.com	
Phone:	(123) 456-7890	
Organization:	Mirth	
Industry:	HIE	
Description:	Tier 1 Support	

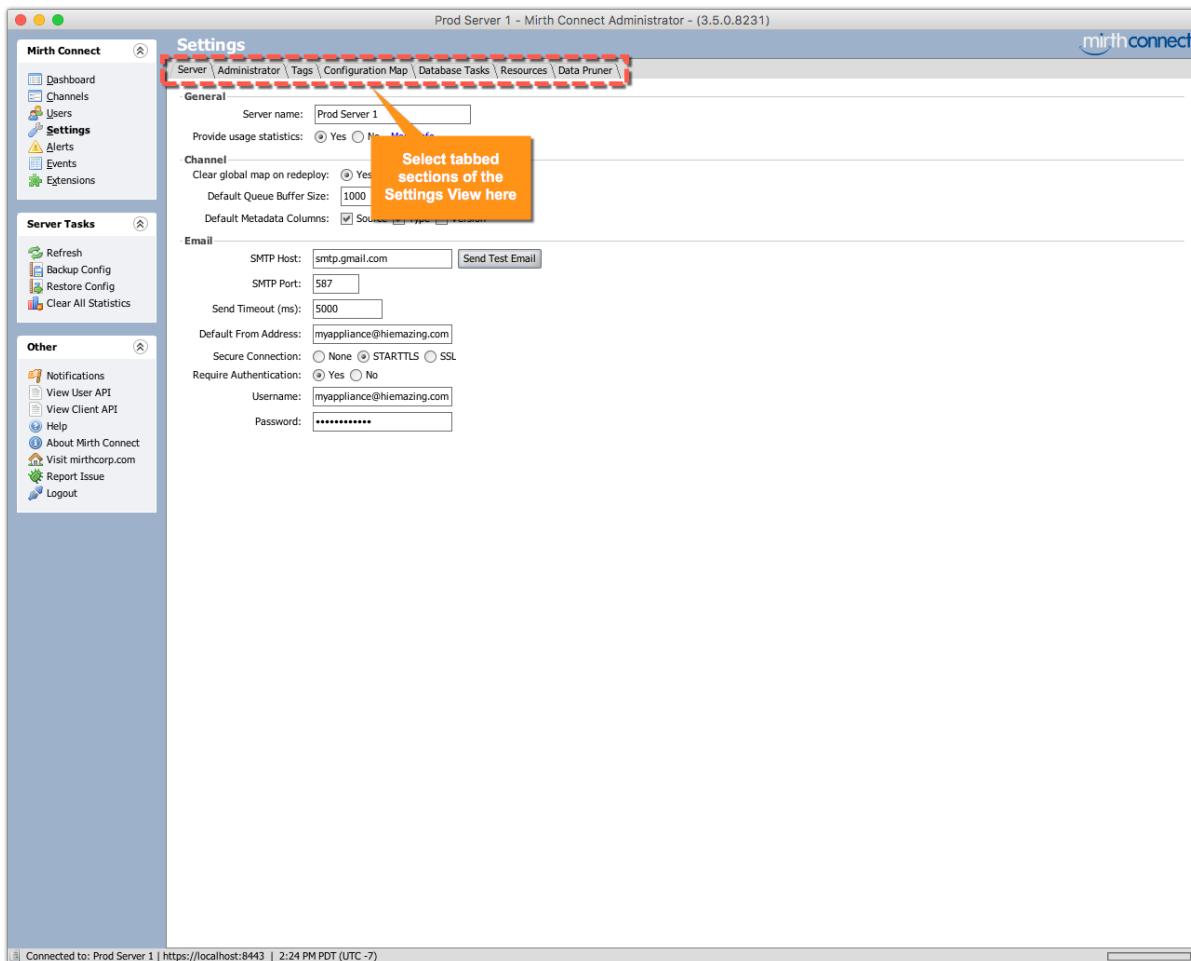
At the bottom right of the dialog are two buttons: 'Cancel' and 'Finish'.

In general the fields shown here are the same as the columns displayed in the [Users Table](#).

If a new user is being created, then the New Password and Confirm New Password fields is required as well as the username. The value in both password fields must match exactly.

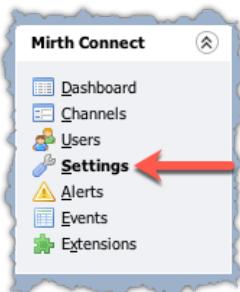
Settings View

This view encompasses a wide variety of management settings / tasks for your NextGen Connect Integration Engine server. Some of the settings are channel-specific (e.g. channel tags), and others are more general (e.g. default e-mail settings).



Navigation

Click the **Settings** link in the Mirth Connect task pane at the top-left:

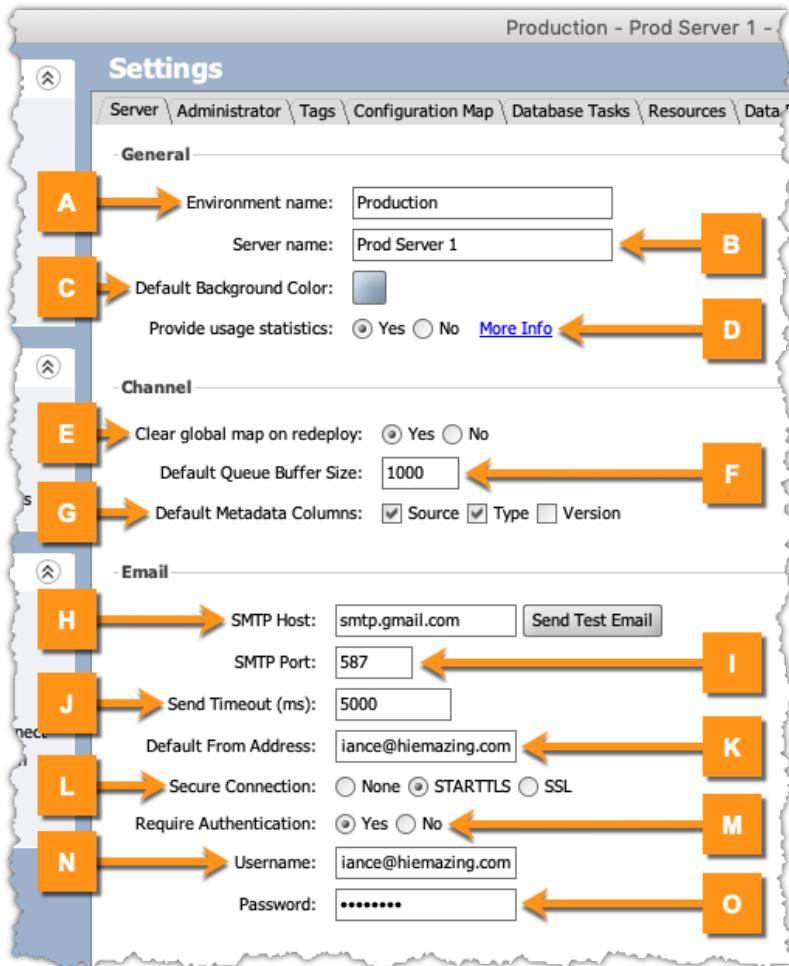


This section is separated into the following topics:

- [Server Settings Tab](#)
- [Administrator Settings Tab](#)
- [Tags Settings Tab](#)
- [Configuration Map Settings Tab](#)
- [Database Tasks Settings Tab](#)
- [Resources Settings Tab](#)
- [Data Pruner Settings Tab](#)
- [Settings Tasks](#)

Server Settings Tab

This tab shows general global settings that pertain to the server or default channel properties.



General Settings

Item	Name	Description
A	Environment name	The name of this Connect environment. This is tied to the database, so there is only one environment per database / cluster.
B	Server name	The server name which will appear in the Administrator title, taskbar/dock and desktop shortcut. This setting applies for all users on this server.
C	Default Background Color	The default Administrator GUI background color this server should use. Users can override this with their own custom background color in the Administrator Settings Tab .
D	Provide usage statistics	Toggles sending usage statistics to Mirth. These statistics do not contain any PHI or channel/script implementations. They help Mirth determine which connectors or areas of NextGen Connect Integration Engine are most widely used.

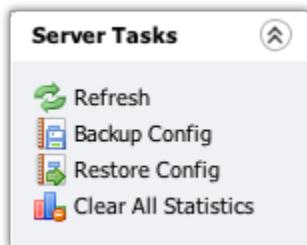
Channel Settings

Item	Name	Description
E	Clear global map on redeploy	Toggles clearing the global map when redeploying all channels. Default = Yes.
F	Default Queue Buffer Size	The default source/destination queue buffer size to use for new channels. Default = 1000 (messages).
G	Default Metadata Columns	Source, Type, Version : these check boxes determine which custom metadata columns will be added by default when a user creates a new channel. The user can choose to remove the column on the channel's Summary tab. For additional information, see Custom Metadata Columns .

Email Settings

Item	Name	Description
H	SMTP Host	SMTP host used for global e-mail settings (e.g. alerts). The Send Test Email button next to this field uses the currently configured settings to send a simple testing e-mail to the Default From Address.
I	SMTP Port	SMTP port used for global e-mail settings (e.g. alerts).
J	Send Timeout (ms)	SMTP socket connection timeout in milliseconds. Default = 5000 (ms) or 5 secs.
K	Default From Address	Default address to use in the "From" field for e-mails. Sending a test e-mail will also dispatch to this address.
L	Secure Connection	Determines whether to use implicit TLS (SSL), explicit TLS (STARTTLS), or no encryption at all for dispatching e-mails.
M	Require Authentication	Toggles authentication for sending e-mails.
N	Username	The username to authenticate as.
O	Password	The password to authenticate with.

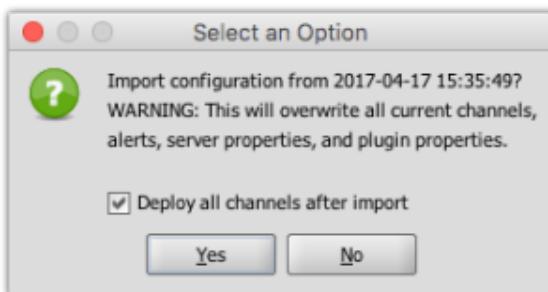
Tasks



Task Icon	Task Name	Description
	Backup Config	<p>Exports a snapshot of your current server settings, including all channels, alerts, scripts, and other properties. This configuration may be backed up from one NextGen Connect Integration Engine server and restored into a different server.</p> <div style="border: 1px solid red; padding: 5px;"> <p>! The Server Configuration file does not include Users, Events, or Message / Attachment data. To export events, see Event Tasks - Export All Events. To export message / attachment data, see Message Browser Tasks - Export Results.</p> </div>
	Restore Config	<p>Overwrites all server settings, including all channels, alerts, scripts, and other properties, with the ones in a given XML file.</p> <p>Note that if a channel currently exists on your server and the same channel is present in the server configuration file, the message / attachment data for that channel will not be modified.</p>
	Clear All Statistics	Resets the current and lifetime message statistics for all channels on your server.

Restore Config

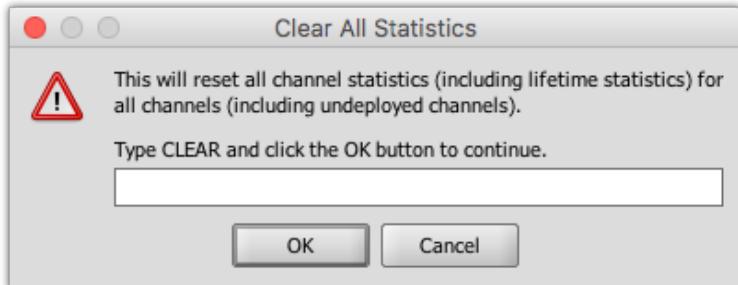
After selecting a server configuration XML file to restore, you'll be presented with the following dialog:



Click **Yes** to proceed, and optionally choose whether you want all resulting enabled channels to automatically be deployed.

Clear All Statistics

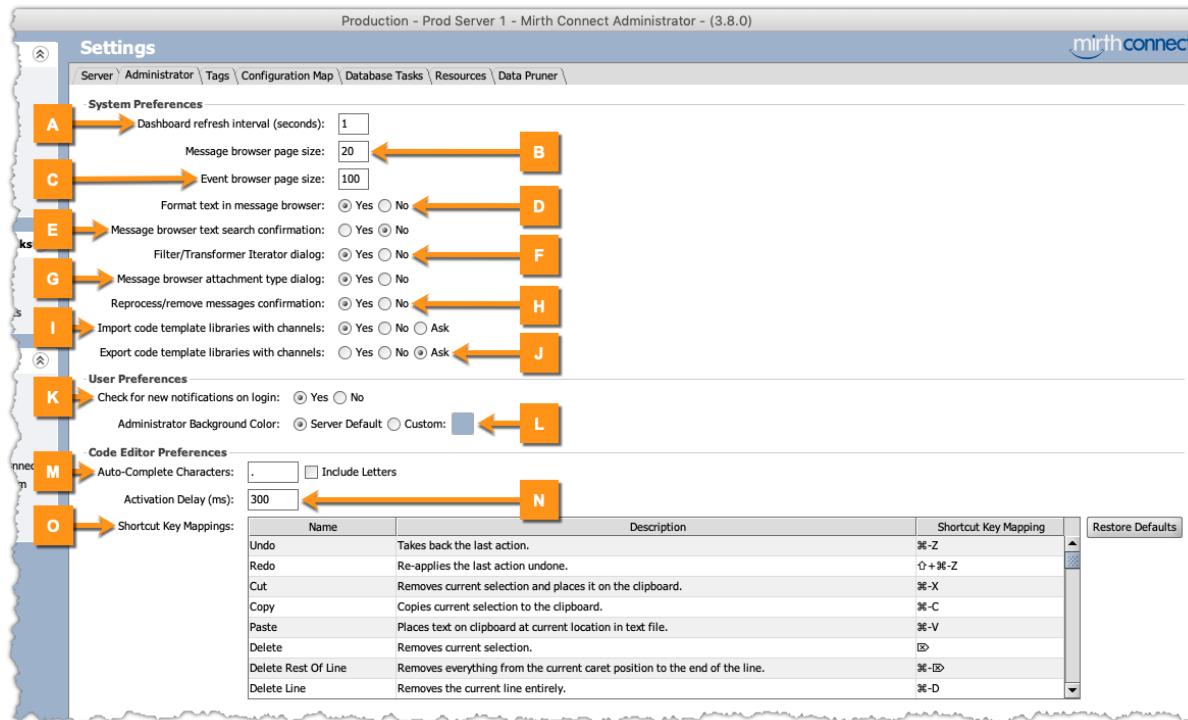
After clicking this task a confirmation dialog will be shown:



As a safeguard you must type "**CLEAR**" in all capital letters to proceed with the operation.

Administrator Settings Tab

This tab shows settings specific to the client machine running the Mirth Connect Administrator. The settings in this tab apply to any and all Administrator instances you login to.



System Preferences

Item	Name	Description
A	Dashboard refresh interval	Interval in seconds at which to refresh the Dashboard and Alerts views. Reduce this for faster updates, and increase it for servers with more channels or when you have high latency between the client and server.
B	Message browser page size	Sets the default page size for the message browser.
C	Event browser page size	Sets the default page size for the event browser.
D	Format text in message browser	Pretty print XML/JSON messages in the message browser by default.
E	Message browser text search confirmation	Show a confirmation dialog in the message browser when attempting a text search, warning users that the query may take a long time depending on the amount of message being searched.
F	Filter /Transformer	Show a confirmation dialog in the filter/transformer views when dragging and dropping elements from the message tree, asking users whether to use an Iterator.

	Iterator dialog	
G	Message browser attachment type dialog	Show a selection dialog in the message browser when viewing attachments to allow the user to select a specific attachment viewer. If No is selected, the attachment viewer will be automatically chosen from the MIME type.
H	Reprocess /remove messages confirmation	Show a confirmation dialog in the message browser when reprocessing or removing multiple messages that forces the user to type in "REPROCESSALL" or "REMOVEALL" first before proceeding.
I	Import code template libraries with channels	When attempting to import channels that have code template libraries linked to them, select Yes to always include them, No to never include them, or Ask to prompt the user each time.
J	Export code template libraries with channels	When attempting to export channels that have code template libraries linked to them, select Yes to always include them, No to never include them, or Ask to prompt the user each time.

User Preferences

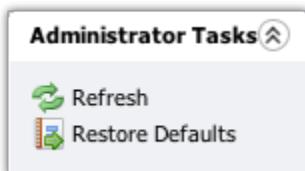
Settings in this section are specific to the currently logged in user on the current server.

Item	Name	Description
K	Check for new notifications on login	Checks for notifications from NextGen (e.g., announcements, available updates) relevant to this version of NextGen Connect Integration Engine whenever the user logs in.
L	Administrator Background Color	If Server Default is selected, the background color chosen on the Server Settings Tab will be used. If Custom is selected, the following color will be used instead.

Code Editor Preferences

Item	Name	Description
M	Auto-Complete Characters	The auto-completion popup will be triggered after any of these characters are typed. If the Include Letters check box is checked, auto-completion will also be triggered after any letter (a-z) is typed.
N	Activation Delay (ms)	The amount of time to wait after typing an activation character before opening the auto-completion popup menu.
O	Shortcut Key Mappings	This table shows common code editor actions and their corresponding shortcut key sequence. To change the key mapping for any action, double-click the Shortcut Key Mapping column, press Escape to clear the current value, enter the new key sequence, and press the Enter key. The Restore Defaults button will restore all shortcut key mappings back to the default settings.

Tasks



Task Icon	Task Name	Description
Restore Defaults	Restore Defaults	Restores all locally-stored Administrator settings to their defaults. This includes everything in the System Preferences and Code Editor Preferences , as well as other hidden settings like dashboard table column/sort order.

Tags Settings Tab

This tab provides a general management view for all channel tags configured on your server. You can edit tag names /colors, and easily include multiple tags across multiple channels with just a few clicks.

Name	Color	Channel Count
ADT		4
Alerts		4
CDA		3
Inbound		15
LAB		3
Mirth Results		3
RAD		4
TRN		6
XDS		1

Channels

Channel selections will be applied to the currently selected tags.

Filter: Select All | Deselect All

Name

- Audition Health - ADT - 6674
- Audition Health - LAB - 6675
- Audition Health - RAD - 6686
- Audition Health - TRN - 6699
- Audition HL7 Router
- Audition - Send Outbound
- Blaster 1
- Blaster 2
- Calif Parth - ADT - 6634
- Calif Parth - LAB - 6635
- Calif Parth - RAD - 6636
- Calif Parth - TRN - 6637
- CDA to MR - Default
- CDA to MR - Epic
- CDA to MR - Meditech
- Cluster Router 1 - 6661
- Create and Write PDF
- Create Salesforce Ticket
- haltest

Connected to: Prod Server 1 | https://localhost:8443 | 11:59 AM PDT (UTC -7)

Tags Table

This is the table located in the top half of the Tags tab. The table has the following columns:

Column	Description
Name	The name of the tag. Double-click this column to edit the name.
Color	The color of the tag, shown when tags are rendered inside the Dashboard / Channels views. Click the colored box in the middle of the column to change the tag color.
Channel Count	The number of channels that currently include this tag.

For general information on working with tables, see [Working With Tables](#).

Adding a Tag

To add a new tag, click the **Add** button at the top-right of the tags table:



A new row will be added to the table. Double-click the **Name** column to change the name, and click the colored box in the middle of the Color column to change the color. Then use the Channels table down below to select which channels should include the new tag.

Removing a Tag

To remove tags, select the rows you wish to delete and then click the **Delete** button at the top-right of the tags table:



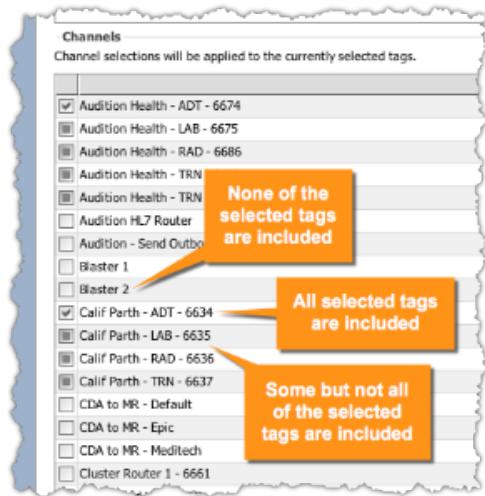
Channels Table

This table shows a list of all channels currently configured on your server. Click one of the column headers at the top of the table to sort by selection status or channel name. Click the check boxes to include the selected tags on specific channels.

You can use the **Select All / Deselect All** links to quickly include or exclude the selected tags on all currently filtered channels.

Indeterminate Check Boxes

If a check box in the table shows a grey box, it means that you have multiple tags selected in the tags table above, and the channel in question is included on **only some** of the selected tags.



If you click on the check box while it's in this state it will change to the checked state, meaning that all of the selected tags are now included on the channel.

Configuration Map Settings Tab

The configuration map is one of the available [Variable Maps](#) that can be used from within channel properties or scripts. It is unique in the fact that it is stored in the [Application Data Directory](#) as a flat file. It is intentionally **not** included in server config exports. This way you can use variables from this map in your channel properties, and the same channel can be used across multiple NextGen Connect Integration Engine servers without having to edit the channel for each server.

By default the values in the table are not immediately shown. Click the **Show values** button to reveal all of them on screen.

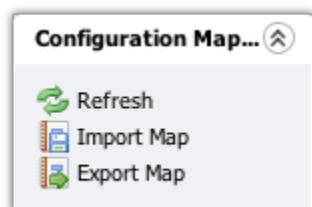
This tab allows you to edit the entries in the configuration map for the current server. Double-click on any of the cells in the table to edit the values. Click the **Add/Remove** buttons at the top-right to add/remove entries from the table.

Key	Value	Comment
emailDomain	nextgen.com	Default e-mail address suffix to use.
clientMap	{ "ports": [7791, 7792, 7795], "clients": { "Hospital A": { ... } } }	Map of ports and other client information to use in channels.
defaultADTPort	6667	
resourceDirectory	/folders/resources	

Table Columns

Column	Description
Key	The unique key the identifies this entry within the configuration map.
Value	The actual value of this entry.
Comment	For documentation purposes only.

Tasks



Task Icon	Task Name	Description
	Import Map	Overwrites all entries in the table with the ones in the given properties file.
	Export Map	Exports all entries in the table to a properties file.



Database Tasks Settings Tab

Occasionally in new versions of NextGen Connect Integration Engine certain things change in the underlying database, but depending on the implications those changes are not always *automatically* performed when you upgrade. Or, old tables are left behind and no longer used, but those tables are not automatically deleted in case you want to keep them or back them up first.

This tab shows you all cleanup or optimization tasks for the internal NextGen Connect Integration Engine database. If no tasks show up in the table, then that means your database is up to date.

Status	Name	Description	Start Time
Idle	Remove Old Pre-3.3 Code Template Table	Remove the OLD_CODE_TEMPLATE table which was renamed as part of the upgrade to 3.3.	--
Idle	Remove Old 2.x Attachment Table	Remove the OLD_ATTACHMENT table which was renamed as part of the upgrade from 2.x to 3.x.	--
Idle	Remove Old 2.x Message Table	Remove the OLD_MESSAGE table which was renamed as part of the upgrade from 2.x to 3.x.	--
Idle	Add Metadata Index	Add index (ID, STATUS, SERVER_ID) on the message metadata table to improve queue performance.	--

Name	Id
CDA to MR - Meditech	95c5951b-3f0f-481d-8013-b9e2fc61db32
CDA to MR - Epic	172cf92e-47a5-4a8d-9bf8-70573faeca9
CDA to MR - Default	513454f2-4109-4f68-8747-95e2914ed4cb

Database Tasks Table Columns

Column	Description
Status	<ul style="list-style-type: none"> Idle: The task is available to run. Running: The task is in the middle of running.
Name	The name of the task.
Description	A short description of the task.

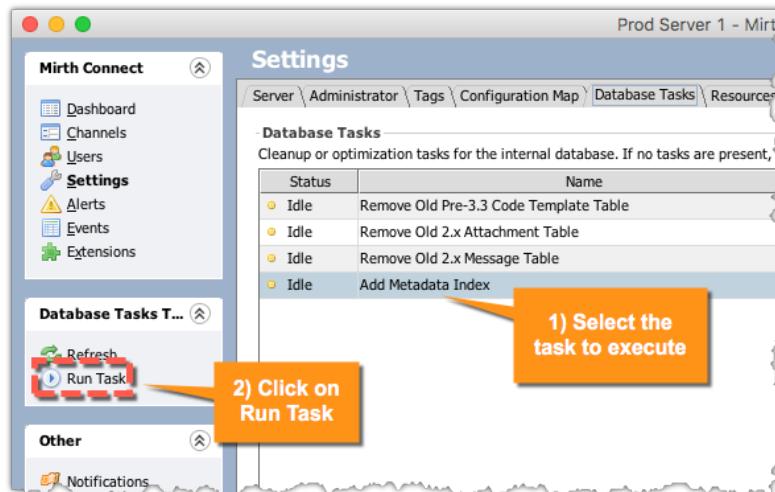
Start Time	If the task is currently running, this will show the time it started.
------------	---

Affected Channels Table

This table only applies to database tasks that are performed on specific channel tables. It will show all channels that will be affected by the selected task if it gets executed. Note that some tasks require that channels be Stopped first before the task will actually be performed.

Running a Database Task

To run a database task, simply click the task row in the table above, then click on the **Run Task** link to the left:



Depending on the task, a confirmation dialog may be shown:



Resources Settings Tab

Resources are shared services that can be used in specific channels / connectors or in other places throughout the NextGen Connect Integration Engine server. They may include custom Java libraries to use within scripts, or services to handle outbound connections.

Name	Type	Global Scripts
[Default Resource]	Directory	<input checked="" type="checkbox"/>
ActiveMQ 5.9.1	Directory	<input type="checkbox"/>
SaxonHE9 5.1.7	Directory	<input type="checkbox"/>
Apache POI	Directory	<input type="checkbox"/>

Directory Settings

Directory: `/folders/apache-activemq-5.9.1/lib` Include All Subdirectories

Description:

```
=====
Welcome to Apache ActiveMQ
=====
Apache ActiveMQ is a high performance Apache 2.0 licensed
Message Broker and JMS 1.1 implementation.

Getting Started
=====
To help you get started, try the following links:
Getting Started
http://activemq.apache.org/version-5-getting-started.html

Building
http://activemq.apache.org/version-5-getting-started.html#GettingStarted-WindowsSourceInstallation
http://activemq.apache.org/version-5-getting-started.html#GettingStarted-UnixSourceInstallation

Examples
http://activemq.apache.org/examples.html
```

Loaded Libraries:

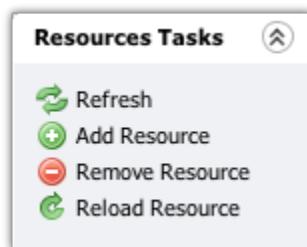
Library
activemq-broker-5.9.1.jar
activemq-client-5.9.1.jar
activemq-console-5.9.1.jar
activemq-jaas-5.9.1.jar
activemq-kahadb-store-5.9.1.jar
activemq-openwire-legacy-5.9.1.jar
activemq-protobuf-1.1.jar
activemq-rar.txt
activemq-spring-5.9.1.jar
activemq-web-5.9.1.jar
camel/activemq-camel-5.9.1.jar
camel/camel-core-2.12.3.jar
camel/camel-jms-2.12.3.jar
camel/camel-spring-2.12.3.jar
extra/mqtt-client-1.8.jar
geronimo-jee-management_1.1_spec-1.0.1.jar

Connected to: Prod Server 1 | https://localhost:8443 | 1:39 PM PDT (UTC -7)

Resources Table Columns

Column	Description
Name	The name of the resource. Double-click this column to edit the name. Note that the [Default Resource] cannot be renamed or removed.
Type	The type of resource / service. The [Default Resource] type cannot be changed.
Global Scripts	If checked, any libraries included with the resource will automatically be made available for use in the Global Scripts .

Tasks

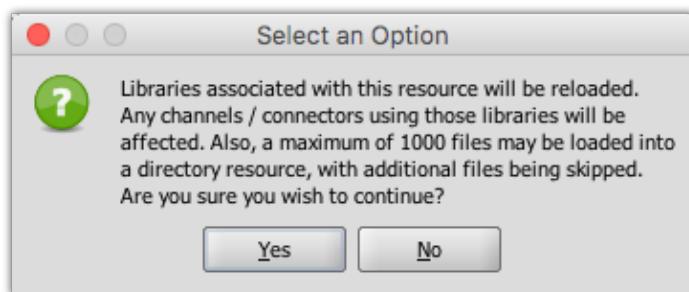


Task Icon	Task Name	Description
	Add Resource	Adds a new resource to the table.
	Remove Resource	Removed the selected resource from the table.
	Reload Resource	Reloads the selected resource on the server.

Reload Resource

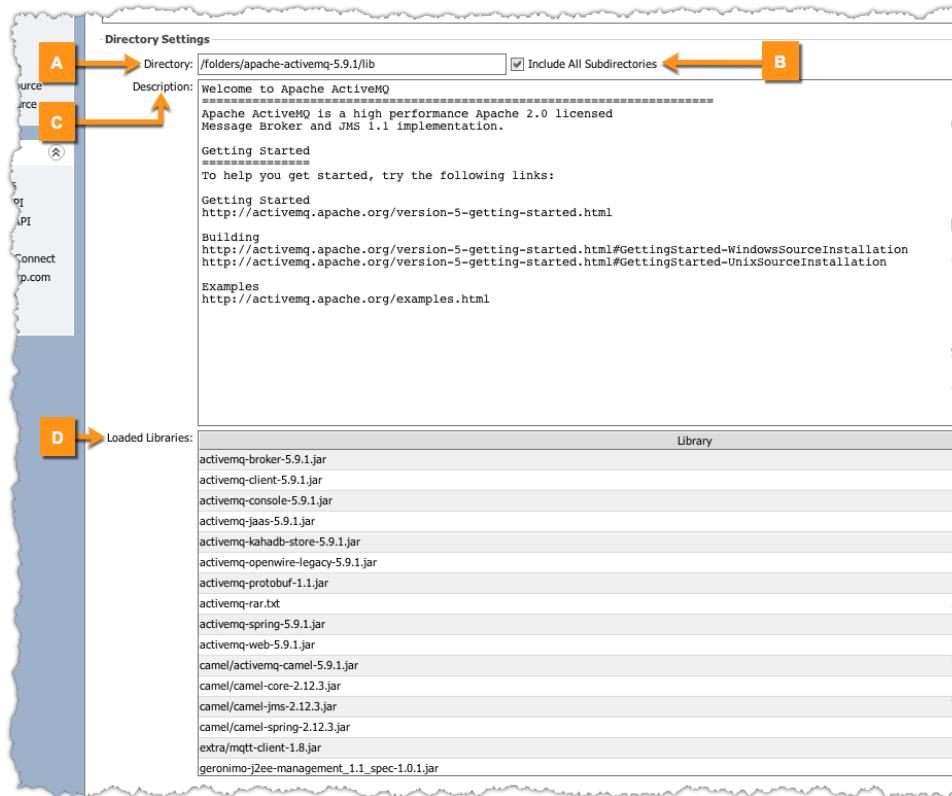
The actual implementation of this task depends on the type of resource. For **Directory** resources the directory will be re-scanned for files, and all channels / connectors using the resource will have their respective classloaders destroyed and recreated with the new list of libraries.

When clicking this task a confirmation dialog will be shown:



Directory Resource

The Directory resource allows you to easily load Java libraries (or other files) so that they can be used in a channel or connector.



Item	Name	Description
A	Directory	The path to the directory to load files from. If a relative path is used, the path will be relative to the Mirth Connect installation directory.
B	Include All Subdirectories	Select Yes to traverse directories recursively and search for files in each one.
C	Description	A description for the resource.
D	Loaded Libraries	All files currently loaded and ready for use by channels / connectors. If you've saved the resource but don't see any libraries in this table yet, try using the Refresh task to the left.

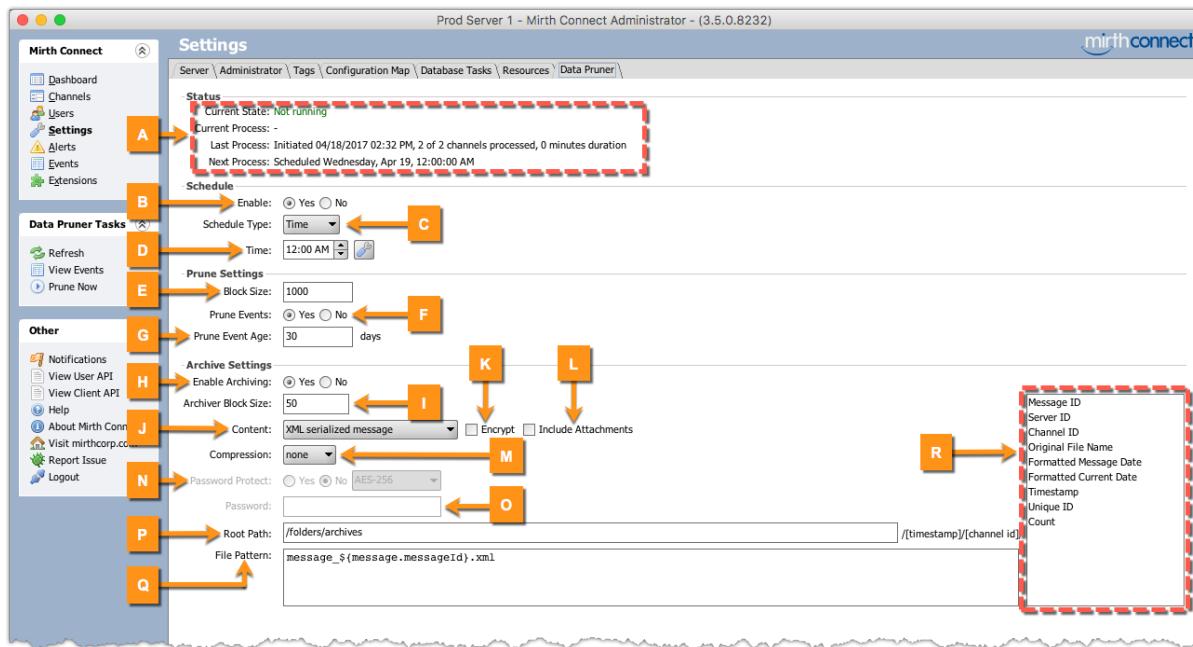
Using Resources in Channels / Connectors

Once you've added a resource, the next step is to include it on a channel or connector. For more information on how to do that, see [Library Resources](#).

Data Pruner Settings Tab

This tab allows you to configure the global data pruner / archiver settings. In addition to seeing the current status of the pruner, you can set a specific schedule for it, and decide whether message data should be archived out to disk before being pruned.

Note that message / attachment data for channels will not be pruned unless the corresponding pruning settings on the channel have been enabled. For additional information, see [Message Pruning Settings](#).



Status

Click the **Refresh** task to update the status.

Item	Name	Description
A	Current State	Indicates whether the Data Pruner is currently running or not.
A	Current Process	Displays the start time of the currently executing pruning job, how many channels have been processed, and how long the job has been running.
A	Last Process	Displays the start time of the last executed pruning job, how many channels were processed, and how long the job took to run.
A	Next Process	Shows the date and time of the next scheduled pruning job.

Schedule

Item	Name	Description
B	Enable	Determines whether the Data Pruner will run on a recurring schedule. If this is disabled, the

		Data Pruner may will be executed manually.
C	Schedule Type	The type of schedule to use for the pruner. Possible types include Interval , Time , and Cron . For additional information, see Polling Settings .
D	Time	When the Time schedule type is used, this will be the time of day the Data Pruner will execute. If a different schedule type is used, this field may be different. For additional information, see Polling Settings .

Prune Settings

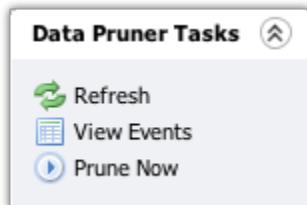
Item	Name	Description
E	Block Size	The number of messages that will be pruned at a time. This value must be between 50 and 10,000. The recommended value for most servers is 1,000. Larger values will use more memory but typically will cause the pruner to perform more quickly.
F	Prune Events	If enabled, event records older than the Prune Event Age will be pruned.
G	Prune Event Age	Events older than this number of days will be pruned if Prune Events is enabled.

Archive Settings

Item	Name	Description
H	Enable Archiving	Determines whether message / attachment data will be saved to disk first before being pruned.
I	Archiver Block Size	The number of messages that will be cached by the archiver. Increase this value to improve performance. Decrease this value to reduce memory usage. This value must be between 1 and 1,000. The recommended value for most servers is 50.
J	Content	The type of content to export from the messages. Note that XML serialized message is the only type that can be re-imported, and it includes all content across the message in a single XML file.
K	Encrypt	If checked, the exported message content will be encrypted with the server's encryption key.
L	Include Attachments	If checked and the content type is set to XML serialized message , the exported file will contain all attachments associated with the message.
M	Compression	When compression is enabled, the files/folders created according to the File Pattern will be put into a compressed file in the Root Path. The following compression types are supported: ZIP, tar.gz, tar.bz2
N	Password Protect	Only available if ZIP compression is enabled. If Yes is chosen, the resulting ZIP file will be protected with the given password below. You can also choose what encryption algorithm to use when password-protecting files. The following algorithms are supported: Standard, AES-128, AES-256
O	Password	If password protection is enabled above, this password will be used.
P	Root Path	The root path to store the exported files/folder or compressed file. Relative paths will be resolved against the NextGen Connect Integration Engine Server home directory.

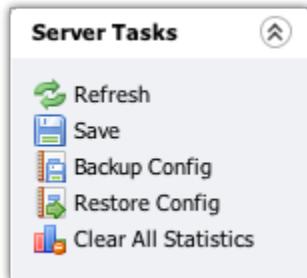
Q	File Pattern	The file/folder pattern in which to write the exported message files. Variables from the list to the right may be used in the pattern.
R	Variables	Contains several common variables you can use to populate the File Pattern with. For additional information, see Velocity Variable Replacement .

Tasks



Task Icon	Task Name	Description
	View Events	View events specific to the Data Pruner. This leaves the Settings View and enters the Events View .
	Prune Now	Manually starts a pruning / archiving job.
	Stop Pruner	Stops the currently running pruning job prematurely.

Settings Tasks



The following context-specific tasks are available from the Settings View and are common across most of the tabs:

Task Icon	Task Name	Description
Refresh	Refresh	Updates the currently selected settings tab.
Save	Save	Saves all configured settings on the currently selected settings tab.

Extensions View

The Extensions view allows you to manage all plugins and connectors installed on your NextGen Connect Integration Engine server. This is where you would go to install any [Commercial Support / Extensions](#). From this view you can install / uninstall / enable / disable extensions, and view properties such as the build number or description.

Status	Name	Author	URL	Version
Enabled	Channel Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Channel Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Database Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Database Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	DICOM Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	DICOM Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Document Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	File Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	File Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	HTTP Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	HTTP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JavaScript Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JavaScript Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JMS Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JMS Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	SMTP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	TCP Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	TCP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Web Service Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Web Service Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232

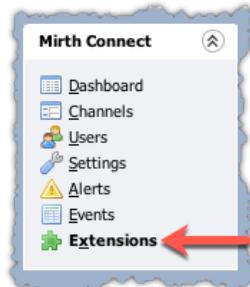
Status	Name	Author	URL	Version
Enabled	Dashboard Connector Status Monitor	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Data Pruner	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Delimited Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Destination Set Filter Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	DICOM Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	DICOM Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Directory Resource Plugin	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	EDI Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	External Script Filter Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	External Script Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Global Map Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	HL7v2 Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	HL7v3 Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	HTTP Authentication Settings	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Image Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JavaScript Filter Rule	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JavaScript Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	JSON Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Mapper Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232
Enabled	Message Builder Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232

Install Extension from File System
File:

Connected to: Prod Server 1 | https://localhost:8443 | 2:07 PM PDT (UTC -7)

Navigation

Click the **Extensions** link in the NextGen Connect Integration Engine task pane at the top-left:



This section is separated into the following topics:

- [Installed Connectors Table](#)
- [Installed Plugins Table](#)
- [Installing a New Extension](#)
- [Extension Tasks](#)

Installed Connectors Table

The Installed Connectors table at the top of the Extensions view shows all source or destination connectors that are installed for your NextGen Connect Integration Engine server. A connector is a type of extension, but are separated here into their own table for simplicity. Double-click a row in the table to view properties for an installed connector. For general information about working with tables in NextGen Connect Integration Engine, see [Working With Tables](#).

Installed Connectors					
Status	Name ▾	Author	URL	Version	
Enabled	Channel Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Channel Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Database Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Database Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	DICOM Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	DICOM Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Document Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	File Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	File Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	HTTP Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	HTTP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JavaScript Reader	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JavaScript Writer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JMS Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JMS Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	SMTP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	TCP Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	TCP Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Web Service Listener	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Web Service Sender	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	

Installed Connectors Table Columns

Column	Description
Status	Indicates whether the connector is enabled or disabled.
Name	The name of the connector.
Author	The development author of the connector. This will be Mirth Corporation for all official extensions.
URL	A link to the author website.
Version	The version of the connector installed. This will usually be the same as the NextGen Connect Integration Engine version, but there may also be a build number associated with the extension, indicating more granular versioning.

Installed Plugins Table

The Installed Plugins table at the top of the Extensions view shows all plugins that are installed for your NextGen Connect Integration Engine server. Plugins are extensions, but separated from connectors for simplicity. Double-click a row in the table to view properties for an installed plugin. For general information about working with tables in NextGen Connect Integration Engine, see [Working With Tables](#).

Installed Plugins					
Status	Name	Author	URL	Version	
Enabled	Dashboard Connector Status Monitor	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Data Pruner	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Delimited Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Destination Set Filter Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	DICOM Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	DICOM Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Directory Resource Plugin	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	EDI Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	External Script Filter Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	External Script Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Global Map Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	HL7v2 Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	HL7v3 Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	HTTP Authentication Settings	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Image Viewer	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JavaScript Filter Rule	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JavaScript Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	JSON Data Type	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Mapper Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	
Enabled	Message Builder Transformer Step	Mirth Corporation	http://www.mirthcorp.com	3.5.0.8232	

Installed Plugins Table Columns

Column	Description
Status	Indicates whether the plugin is enabled or disabled.
Name	The name of the plugin.
Author	The development author of the plugin. This will be Mirth Corporation for all official extensions.
URL	A link to the author website.
Version	The version of the plugin installed. This will usually be the same as the NextGen Connect Integration Engine version, but there may also be a build number associated with the extension, indicating more granular versioning.

Installing a New Extension

Extensions are packaged into ZIP files. To install an extension in NextGen Connect Integration Engine:

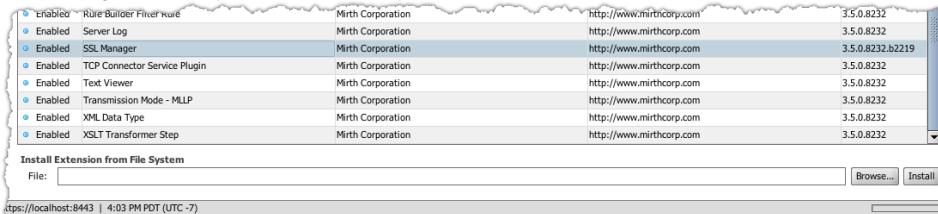
1. Navigate to the installation section at the bottom of the Extensions view.
2. Click the "Browse..." button and select the ZIP file of the extension to install.



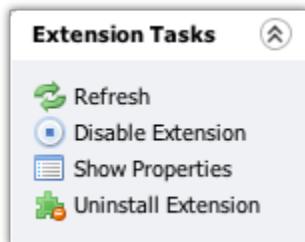
3. Click the **Install** button. If the extension ZIP was valid, you'll see a notification telling you to restart the server:



4. Restart the NextGen Connect Integration Engine server and [launch the Mirth Connect Administrator](#) again.
5. You'll see the new extension listed in either the [Installed Connectors Table](#) or [Installed Plugins Table](#), depending on what type of extension you installed.



Extension Tasks

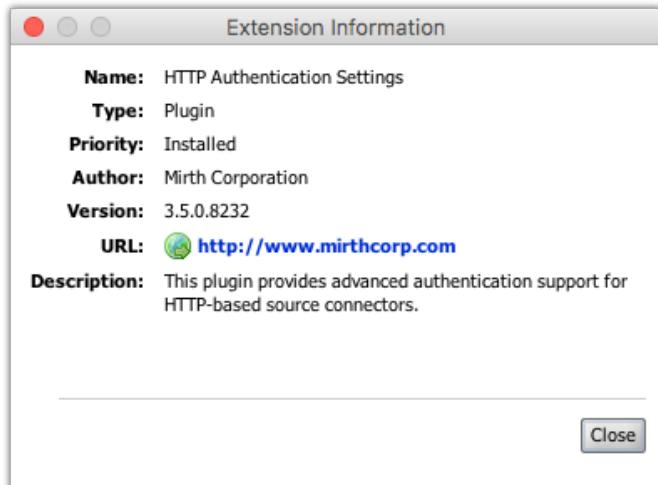


The following context-specific tasks are available from the Extensions View:

Task Icon	Task Name	Description
	Refresh	Updates the installed connectors and installed plugins tables.
	Enable Extension	Enables the extension so that it can be used in your NextGen Connect Integration Engine installation. This requires a restart of the server.
	Disable Extension	Disables the extension so that it can no longer be used in your NextGen Connect Integration Engine installation. This requires a restart of the server. <div style="border: 2px solid red; padding: 5px; margin-top: 10px;"> ! Channels / alerts / resources that depend on an extension that you disable may stop functioning correctly. For example, channels may show up in the Channel Table as "invalid" if a connector or data type it was using has been disabled. </div>
	Show Properties	Displays properties for the currently selected connector or plugin.
	Uninstall Extension	Uninstalls the extension completely from your NextGen Connect Integration Engine instance. This requires a restart of the server. <div style="border: 2px solid red; padding: 5px; margin-top: 10px;"> ! Channels / alerts / resources that depend on an extension that you uninstall may stop functioning correctly. For example, channels may show up in the Channel Table as "invalid" if a connector or data type it was using has been uninstalled. </div>

Show Properties

Clicking on the Show Properties task will show a dialog with more information on the extension:



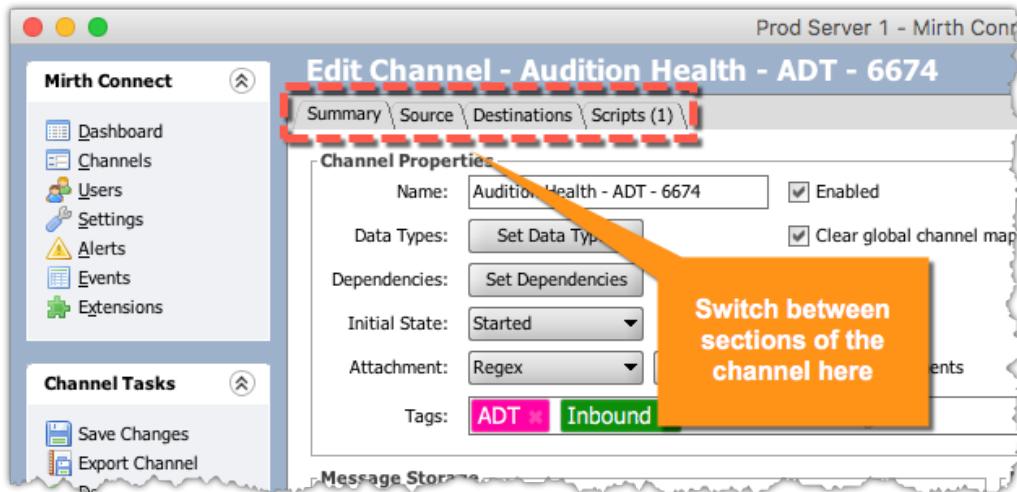
Editing Views

This section is separated into the following topics:

- [Edit Channel View](#)
- [Edit Filter / Transformer Views](#)
- [Edit Global Scripts View](#)
- [Edit Code Templates View](#)
- [Edit Alert View](#)

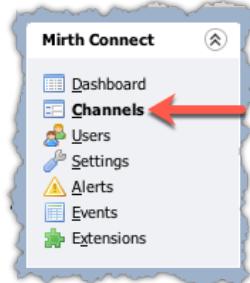
Edit Channel View

The Edit Channel view is where channels are configured. General settings and other properties not specific to a connector are configured on the [Summary Tab](#). Connector-specific settings are configured on the [Source](#) and [Destinations](#) tabs. Channel-level scripts not specific to connectors are configured on the [Scripts Tab](#).

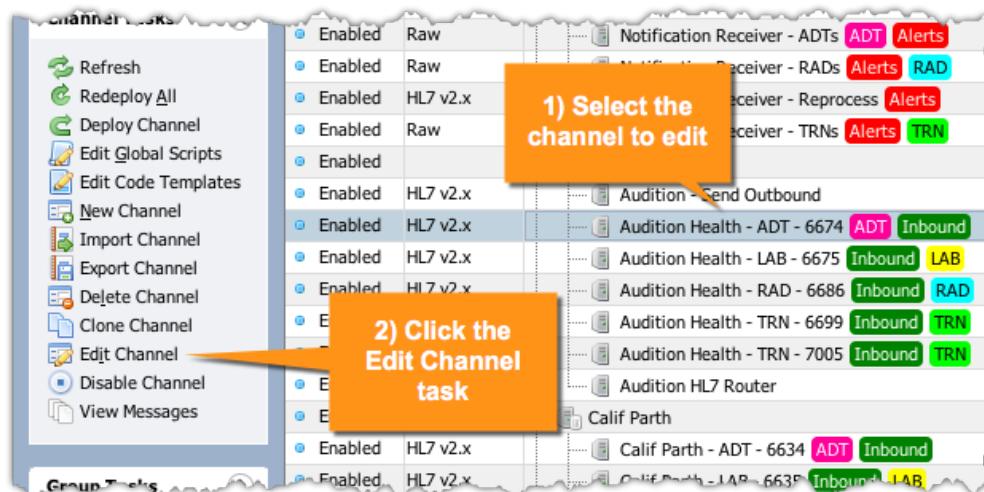


Navigation

Click the **Channels** link in the Mirth Connect task pane at the top-left to enter the [Channels View](#):



In the [Channel Table](#), select the channel you wish to edit, and click the **Edit Channel** task to the left:



OR, just double-click the channel row in the table.

This section is separated into the following topics:

- Summary Tab
- Source Tab
- Destinations Tab
- Scripts Tab
- Edit Channel Tasks

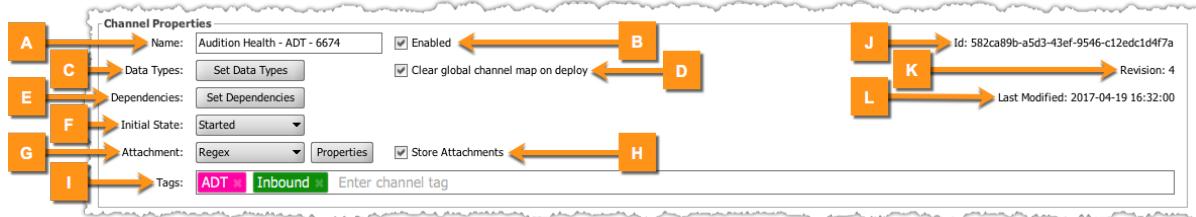
Summary Tab

The **Summary** tab is what you first see when entering the [Edit Channel View](#). This is where general settings not specifically tied to connectors are configured, like the channel name, initial state, storage settings, and description.

Configuration of channel properties in the Summary Tab is separated into the following sections:

- [Channel Properties](#)
- [Message Storage Settings](#)
- [Message Pruning Settings](#)
- [Custom Metadata Columns](#)
- [Channel Description](#)

Channel Properties

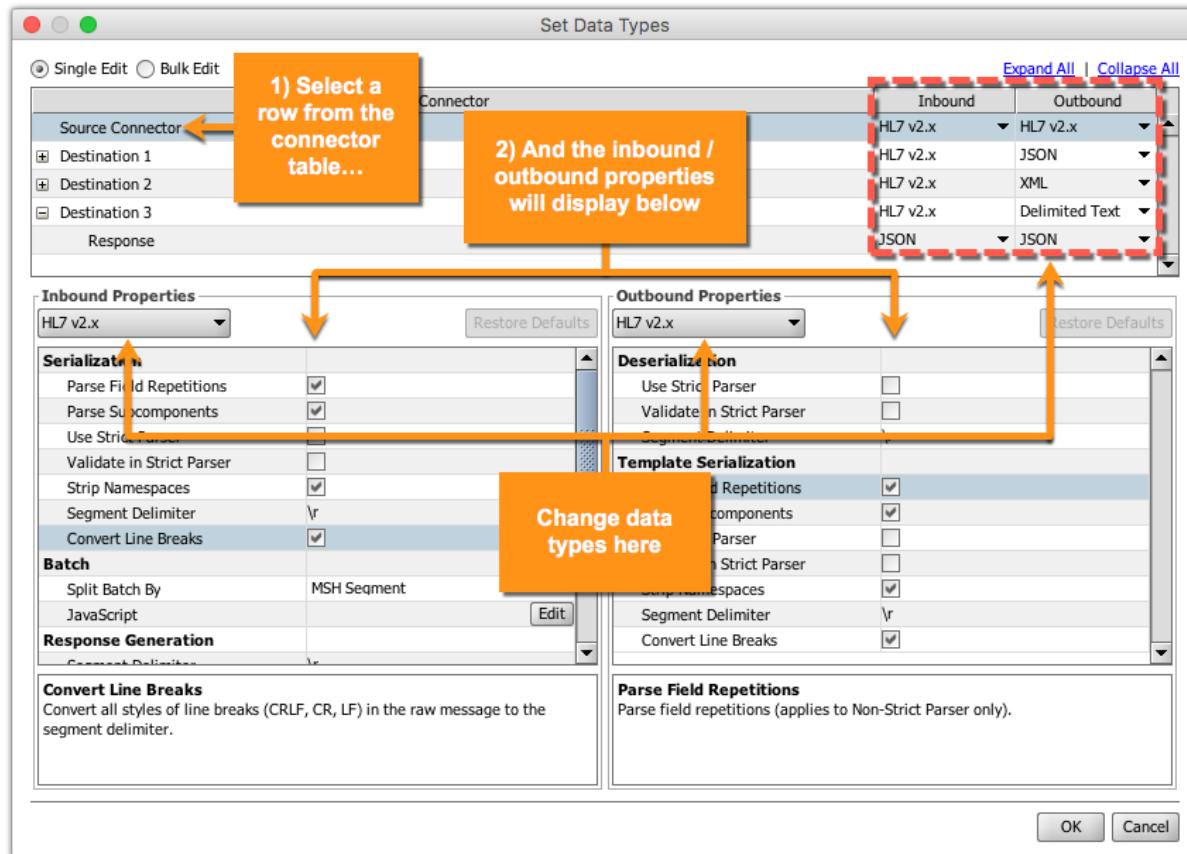


Item	Name	Description
A	Name	The name of the channel. Only alphanumeric (a-z / 0-9) characters, spaces, hyphens, and underscores are allowed.
B	Enabled	Indicates whether the channel is ready to be deployed. All enabled channels will automatically be deployed with the NextGen Connect Integration Engine server starts up. If there is an error in your channel configuration, this will automatically be unchecked.
C	Set Data Types	Configure data types across your entire channel, from the source connector to each of your destination connectors. Allows easy bulk editing of data type properties. For additional information, see Set Data Types Dialog .
D	Clear global channel map on deploy	If checked, the global channel map for this channel will be cleared out whenever the channel is redeployed.
E	Set Dependencies	Configures Code Template Libraries , Library Resources , and Deploy / Start Dependencies for this channel. For more information, see Set Dependencies Dialog .
F	Initial State	Determines what state the channel should be in after it gets deployed. <ul style="list-style-type: none"> Started: The entire channel (source and destination connectors) will be started. Paused: Only the destination connectors will be started. The channel will not automatically receive new messages, but existing queued messages will still dispatch outbound. Stopped: The channel will be stopped. No messages will be received or dispatched outbound.
G	Attachment Handler	Determines how the channel will extract and store attachments. Each attachment handler does this in a different way. Select the handler to use from the drop-down menu, and click the Properties button to edit the configuration for that handler. For additional information, see Attachment Handlers .
H	Store Attachments	If checked, any attachments that are extracted will be stored in the database and available for reattachment. If unchecked, attachments may be extracted from the incoming message but are not stored.
I	Tags	The list of channel tags included on this channel. Type into this field to auto-complete currently existing tags. Or, type a new tag name and hit Enter to create a whole new tag. More information here: Filtering By Channel Name or Tag <p>When new tags are created here, the color is chosen at random. This may be changed later in the tag management view: Tags Settings Tab</p>
J	Id	The ID of the channel. This field can be selected for copy/paste purposes.

K	Revision	The current revision of the channel. Every time the channel is saved, the revision will automatically increment if anything actually changed. This does not apply to certain metadata items like the Enabled flag, the Last Modified date, pruning settings, and tags.
L	Last Modified	The date and time the channel was last successfully saved.

Set Data Types Dialog

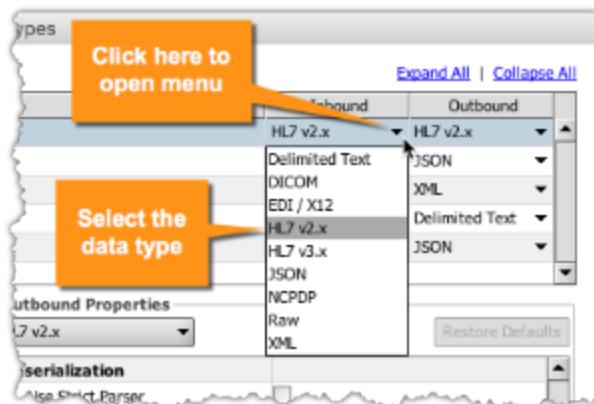
This is the main management dialog for all data types configured across your entire channel. You can see at a glance and easily configure what data format you're expecting to receive, what formats you're converting it into for each outbound destination, and what response formats you are receiving from external systems. You can also configure various properties for each data type, which determines how your data can be used inside of filters / transformers and elsewhere. For additional information on data types, see [About Data Types](#) and [Data Types](#).



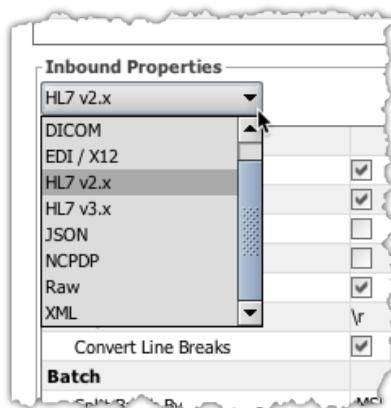
Selecting Data Types

The table at the top of the dialog shows all currently configured connectors for your channel. Since response transformers have their own inbound / outbound data types, for destination connectors you can expand the node in the tree and view the Response row underneath a particular destination.

By default for a new channel, all data types are set to [HL7 v2.x](#). They can be changed in two ways. First, the connector table has Inbound and Outbound columns with drop-down menus. Click the arrow next inside of the cell and make a selection:



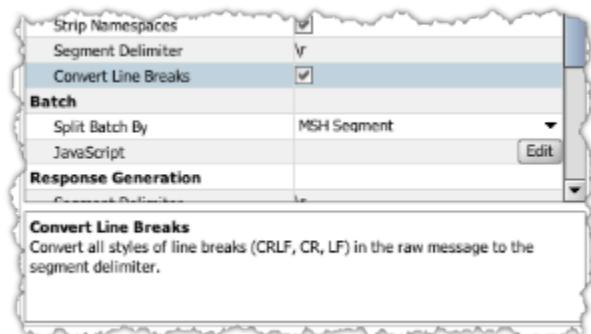
Or, use the drop-down menu in the **Inbound Properties** or **Outbound Properties** sections:



- i** Since a message flows from the source connector directly to each destination connector, every destination inbound data type is always equal to the source outbound data type. Therefore, you may notice that the inbound data type is not editable for destination rows. In order to change the destination inbound data type, change the source outbound data type.

Changing Data Type Properties

After selecting a connector / response row in the table above, the **Inbound Properties** and **Outbound Properties** sections below will show all currently configured properties for the respective data types. Click on the rows in the properties table to view descriptions of categories or specific properties:



Depending on the connector row chosen above, the data type itself, and whether the data type being used as inbound or outbound, the categories of properties shown in the table will vary. For additional information on data type properties and options for specific data types, see [Data Types](#).

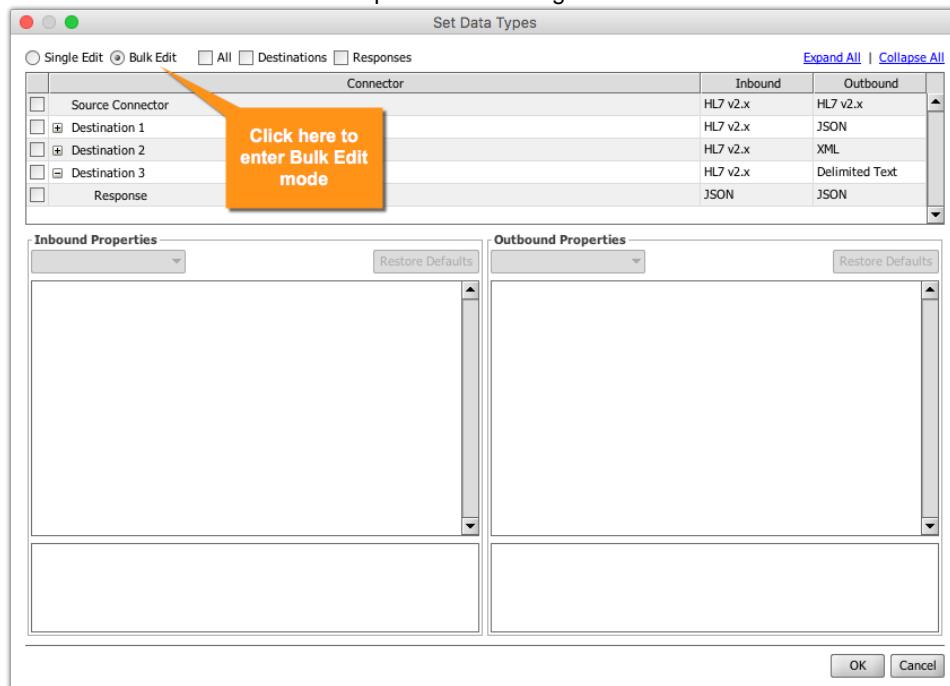
Any properties you've changed or that are not equal to the default value will be shown in **bold**. To revert any changes you make back to the defaults for the selected data type, click on the **Restore Defaults** button above the properties table:



Bulk Edit Mode

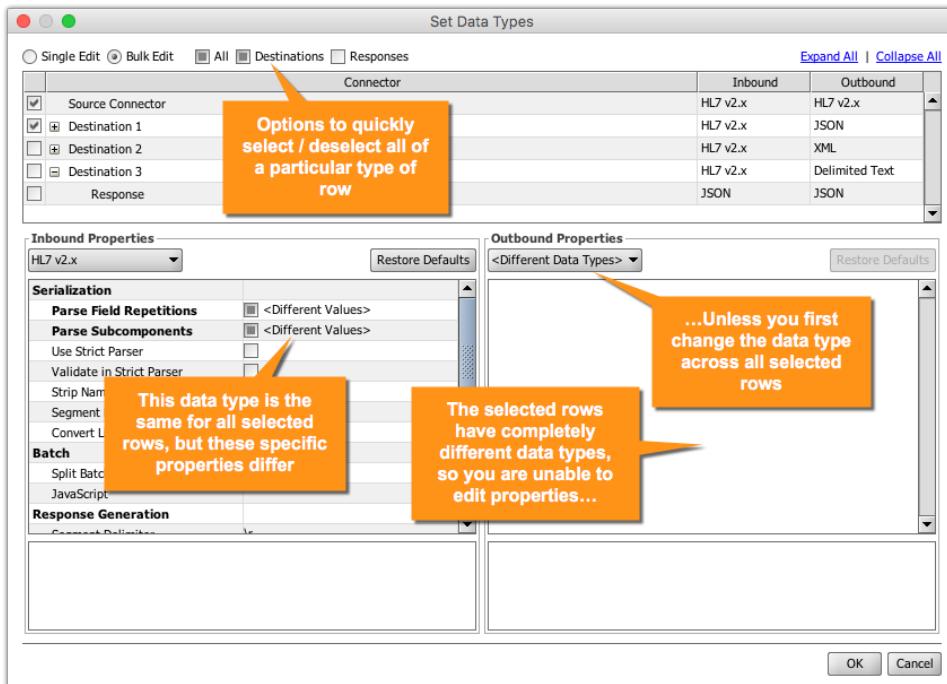
The above instructions so far have assumed that **Single Edit** is selected at the top-left of the dialog. This means that only a single connector / response row can be modified at a time. However you can also make common changes to multiple connectors all at once.

1. Click the **Bulk Edit** button at the top-left of the dialog:



2. After entering Bulk Edit mode, the drop-down menu arrows in the Inbound / Outbound columns disappear, and a new check box column appears at the left of the table.

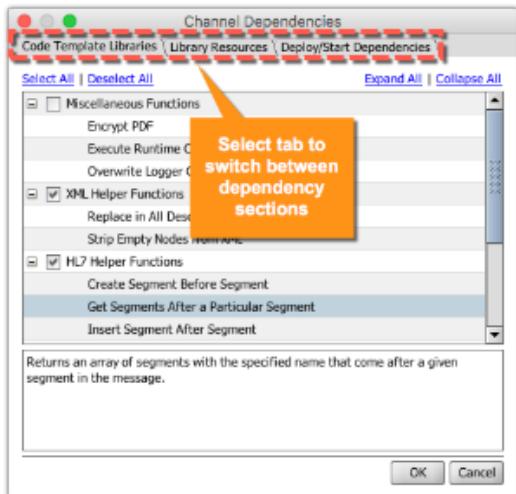
3. Select the connector / response rows you want to edit by checking the check box in the table next to the name. Once you've selected at least one, the Inbound / Outbound Properties sections below will become activated.



4. For either the Inbound or Outbound Properties section, if the data type is the same across all of the currently selected rows, then you have the option to edit properties. Any changes you make while in this mode will be applied across *all* of the currently selected connector / response rows.
5. If the data types are different, then you cannot bulk edit the properties. **However**, you can *change* the data type across all selected rows at once so that they *are* now the same, and then properties may be edited.
6. The **All**, **Destinations**, and **Responses** check boxes at the top are convenient ways to select multiple types of rows at once.

Set Dependencies Dialog

This dialog allows you to configure external dependencies for your channel / connectors. This may include [Code Template Libraries](#) or [Library Resources](#) you wish to use inside your channel, or even other channels that should be labeled as dependents / dependencies of this channel.

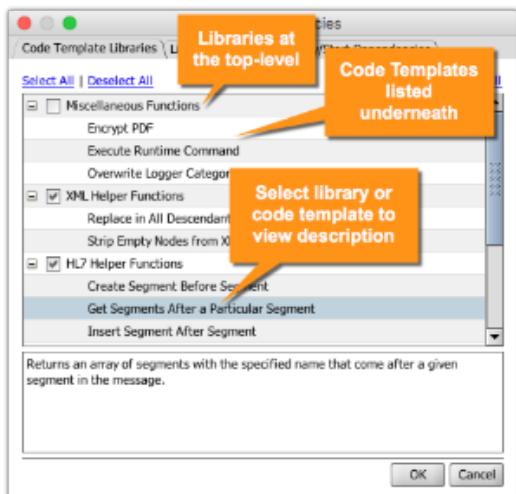


The Set Dependencies Dialog is separated into the following sections:

- [Code Template Libraries](#)
- [Library Resources](#)
- [Deploy / Start Dependencies](#)

Code Template Libraries

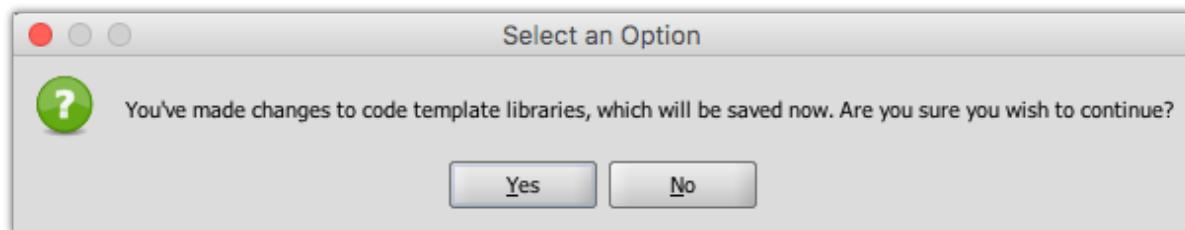
This tab allows you to link code template libraries to the current channel you're editing. Note that this accomplishes the same thing as including channels on the [Edit Library Panel](#) within the [Edit Code Templates View](#). It's included here as well within the [Edit Channel View](#) for convenience.



The tree within this tab is organized such that Libraries are the top-level nodes, and each Code Template underneath is a child of that library. Click on any library or code template row, and if a description is available, it will show in the bottom section of the dialog.

Linking Code Template Libraries

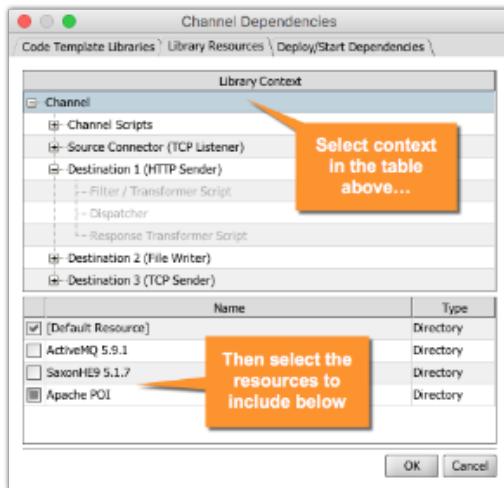
To include a library of code templates on the current channel, simply check the check box next to the library name. Then click the **OK** button. A confirmation dialog will be shown:



Because code template library / channel links are stored outside of the actual channel itself, when closing the Set Dependencies Dialog it prompts you to save those changes immediately. Click **Yes** to accept the changes and exit the dialog.

Library Resources

Resources are shared services that can be used in specific channels / connectors or in other places throughout the NextGen Connect Integration Engine server. They may include custom Java libraries to use within scripts, or services to handle outbound connections. This tab allows you to include resources throughout your entire channel, or in more specific contexts such as the channel scripts or certain connectors.

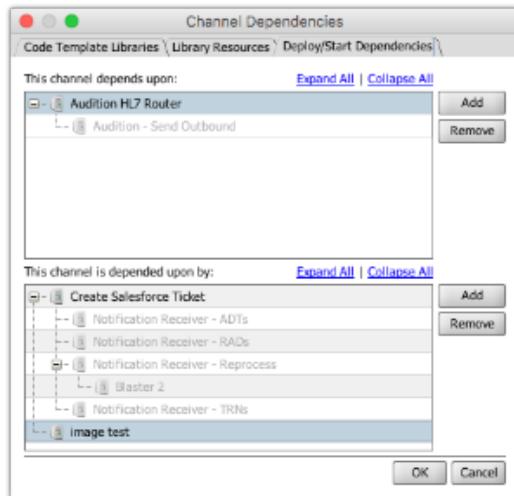


The **Library Context** tree in the top section of the tab allows you to select a specific context, or the **Channel** root-level node which represents *a//contexts* across your channel at once. The following library contexts are available:

Library Context	Description
Channel	This is the root-level node of the tree, and includes all contexts below it. If one of the resource check boxes in the table below is in the indeterminate state (grey box), it means that it is currently included on some but not all of the sub-contexts.
Channel Scripts	This includes the deploy, undeploy, preprocessor, postprocessor, attachment, and batch scripts.
Source Connector	This includes the source filter / transformer script, as well as the source connector itself. For example if you're using a Database Reader and you include a resource containing a custom JDBC Driver JAR, that will be available for use on the connector.
Destination Connectors	This includes the filter / transformer scripts, the response transformer scripts, and the destination connector itself. For example if you're using a JavaScript Writer and you include a resource containing custom Java classes, those will be available for use inside the destination script.

Deploy / Start Dependencies

Channel Dependencies allow you to mark one channel as a "dependency" of another channel. This means that the **dependency** channel *should* be started / deployed before the **dependent** channel. This could be for a variety of reasons, but one common use-case is when you have one channel sending to another channel. The upstream channel is the **dependent**, and the downstream would be the **dependency**.

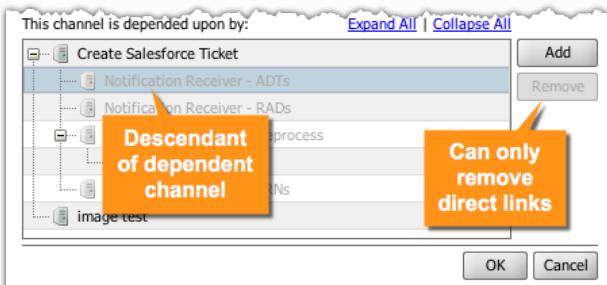


The tree in the top-half of the dialog shows **dependencies of** the current channel. In other words, channels that show up in the top tree are those that the current channel depends upon. By expanding nodes in this tree, you can see descendant dependencies, or "dependencies of the dependencies".

The tree in the bottom-half of the dialog shows channels that are **dependent on** the current channel. By expanding nodes in this tree, you can see descendant dependents, or "channels that are dependent on the channel that is dependent on the current channel".

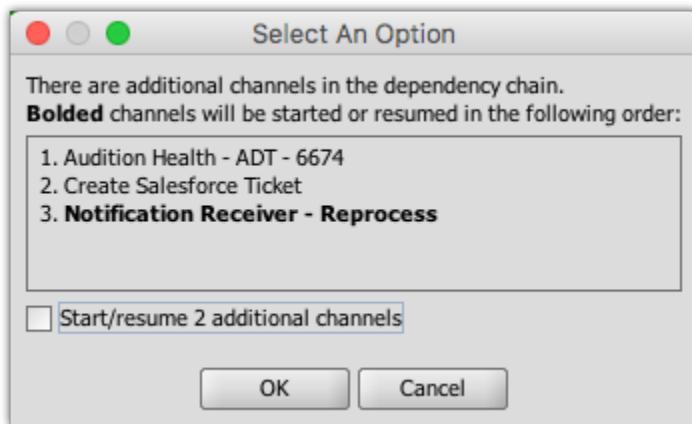
Click the **Add** / **Remove** buttons next to either tree to add or remove direct links.

- Only *direct* dependencies / dependents can be added or removed from this dialog. Descendants of the channels you add will be grey, and selecting them will disable the **Remove** button:



Deploying / Starting Channels

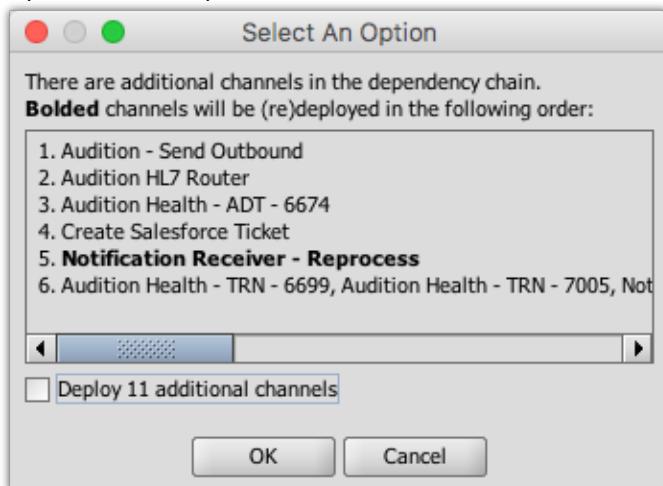
When you start a channel, the Mirth Connect Administrator will automatically check if there are any dependencies of the channel that should be started first. If there are, they will be shown in a special dialog:



By default only the channels you've selected will be **bolded** in the dialog, meaning that only they will be started. If you want to start the other channels in the dependency chain as well, check the "**Start/resume # additional channels**" check box before clicking the **OK** button. Upon pressing **OK** the channels will be deployed in the numeric order shown in the dialog.

The numbered list in this dialog indicates the different "tiers" in the dependency graph. Any downstream dependencies will be deployed / started *first*, and only then will their dependent channels be deployed / started. In the example picture above, if the checkbox is checked, the channel labeled **1.** will be started first, then the channel labeled **2.**, and then finally the channel labeled **3.**, which was the original channel selected from the Dashboard Table. If the checkbox remains unchecked, only the originally selected channel(s) will be started. You can always tell which channels will be affected, because they will appear as **bold** in the list.

Deploying channels is very similar in behavior, except that the chain of channels shown in the dialog will include both dependents and dependencies:

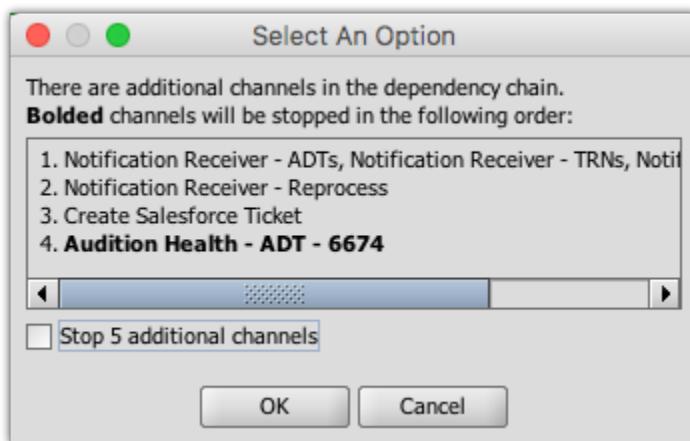


This is because deploying a channel may entail *undeploying* any selected channels first, and then *redeploying* them. So when (re)deploying a particular channel, this sequence is followed:

- All dependent channels are undeployed
- The selected channel is undeployed
- All dependency channels are undeployed
- All dependency channels are deployed
- The selected channel is deployed
- All dependent channels are deployed

Pausing / Stopping / Undeploying Channels

This follows similarly from deploying / starting channels, except that the logic is reversed. When pausing / stopping / undeploying a channel, the operation is performed on **dependent** channels (channels that are dependent on the selected channel) first.



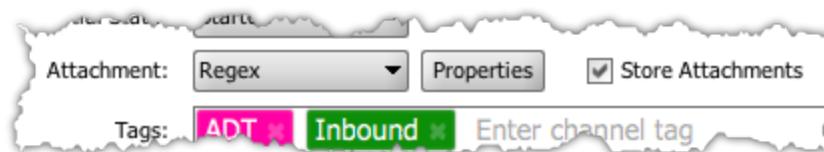
As in the deploy / start case, by default only the channels you've selected will be **bolded** in the dialog, meaning that only they will be paused / stopped / undeployed. If you want to perform this action on the other channels in the dependency chain as well, check the "**additional channels**" check box before clicking the **OK** button. Upon pressing **OK** the channels will be paused / stopped / undeployed in the numeric order shown in the dialog.

Attachment Handlers

An Attachment Handler allows you to extract pieces of any incoming message and store them separately. As a message processes through a channel, multiple copies of it will be held in memory at once (for the raw / transformed / encoded versions of a message, etc.). Attachments are stored only once, so by using them you can greatly reduce your channels' memory footprint. These are configured in the [Channel Properties](#) section of the [Summary Tab](#) within the [Edit Channel View](#).



See Also: [Attachment JavaScript Functions](#)



By default the attachment handler is set to **None**, meaning no attachments will be extracted. To extract attachments choose an attachment handler type from the drop-down menu, and click the **Properties** button to configure the handler.



If **Store Attachments** next to the Attachment menu is unchecked, then attachments will be extracted from the incoming message data, but not actually stored anywhere.

Extraction

When an attachment handler extracts data from a message, it leaves behind an **attachment replacement token** like this:



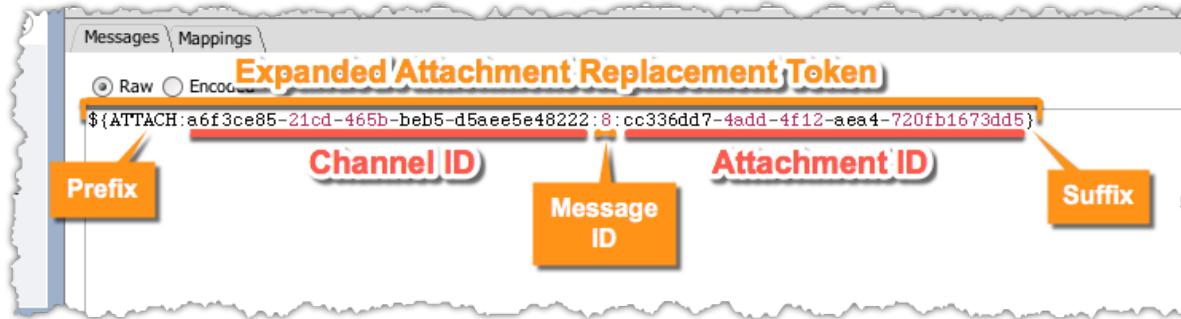
This value tells the destination connector where in your message the attachment should be re-inserted right before dispatching to an outbound system. If multiple attachments were extracted for a message, then there will be multiple replacement tokens in the raw data.

Reattachment

Right before a destination connector dispatches a message to the external system, it scans the outbound message for attachment replacement tokens and automatically re-inserts the actual attachment data. You can prevent a destination from reattaching data by disabling the **Reattach Attachments** option in the [Destination Settings](#).

Expanded Replacement Tokens

The standard attachment replacement token only includes the attachment ID, and is implicitly assumed to be tied to the current message / channel. However if you disable **Reattach Attachments** in the [Destination Settings](#), the destination will replace the token not with the actual attachment data, but instead with an **expanded token**:



The expanded token contains the channel ID, message ID, and attachment ID, so that you can uniquely identify an attachment even from a completely different channel or message. Because of this, you can use this replacement token in downstream channels and reattach attachments from earlier, upstream channels.

Attachment MIME Types

As explained in the [Attachments Tab](#) section, there are four types of attachment viewers in the [Message Browser](#): Text, Image, DICOM, and PDF. The **types** of attachment corresponding with these viewers are:

- **text/***: Plain textual data.
- **image/***: Image data (JPGs, PNGs, etc.).
- **DICOM**: A special (not strictly MIME) type reserved for DICOM attachment data.
- **application/pdf**: PDF data.

The * is a wildcard, signifying that anything can be present there. For example, if you're reading in RTF data, the appropriate MIME type would be **text/rtf**, which matches the **text/*** type when the message browser is searching for an attachment viewer.

Note that when extracting / creating attachments, you can use any type you want. It has no effect on how the data is stored or reattached (except for the DICOM special case), only how it's displayed in the message browser.

Attachment Handler Properties

The following attachment handlers are supported:

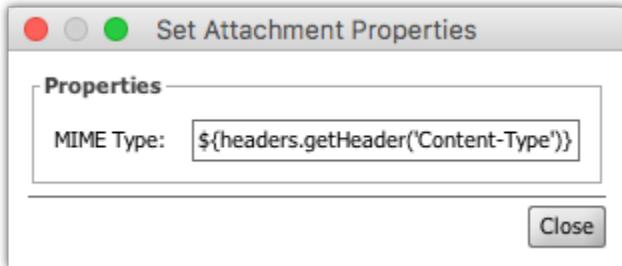
- [Entire Message Attachment Handler Properties](#)
- [Regex Attachment Handler Properties](#)
- [DICOM Attachment Handler Properties](#)
- [JavaScript Attachment Handler Properties](#)
- [Custom Attachment Handler Properties](#)

Entire Message Attachment Handler Properties

This attachment handler takes the entire incoming message data and stores it as a single attachment. The Raw message data afterwards will just be the attachment token.

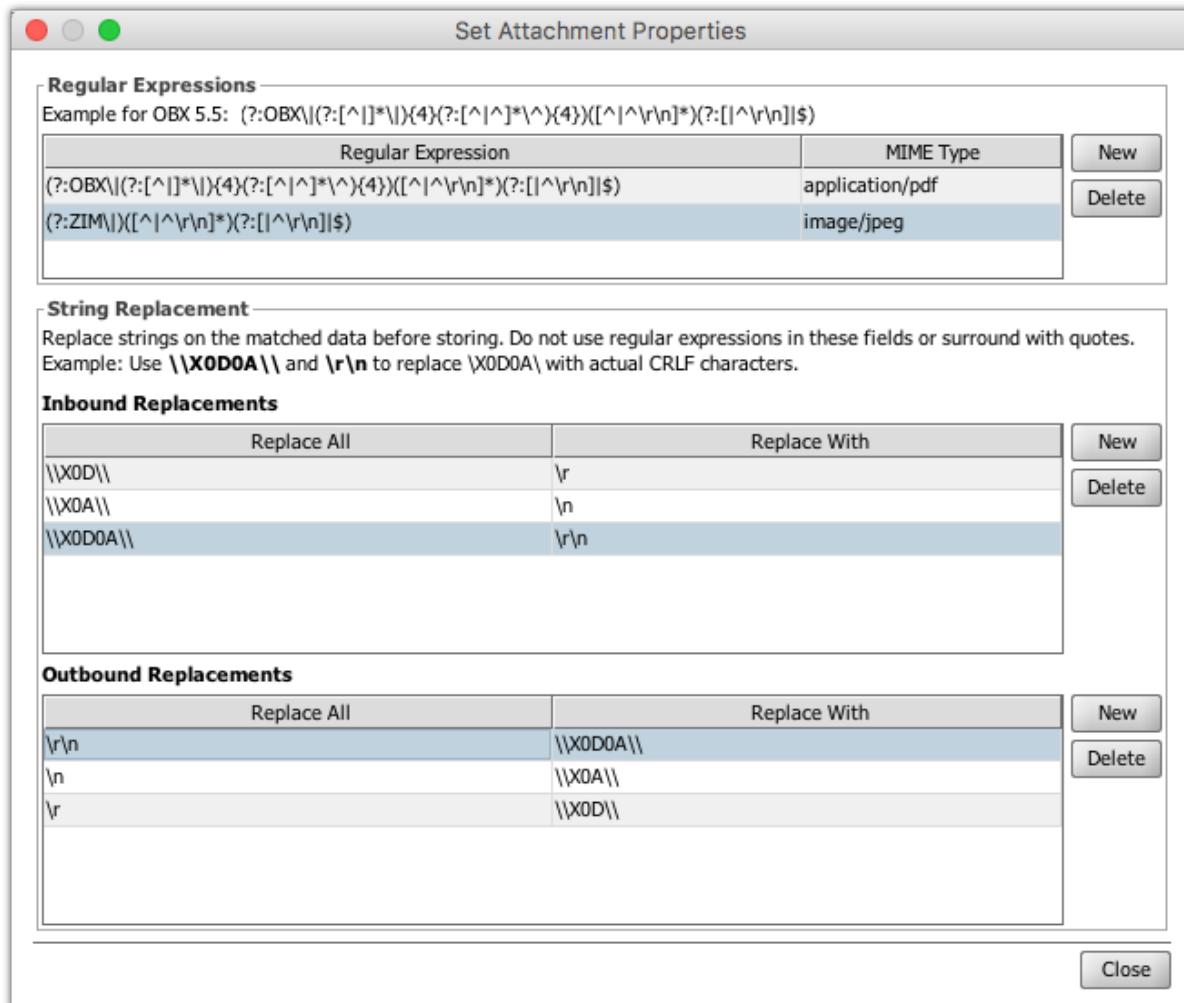


The handler has a single property, **MIME Type**, which specifies what type of attachment data you expect to receive. You can use a specific value like "text/plain", or you can use [Velocity replacement](#) to inject source map variables. For example if you're using an [HTTP Listener](#), you can use the MIME type coming in the Content-Type header:



Regex Attachment Handler Properties

This attachment handler extracts data from the incoming message using regular expressions. You can specify multiple expressions, each with their own MIME type. There are also options to replace certain values on the extracted attachment data before storing it in the database, and replace values in the attachment data right before reinserting it into the message for outbound dispatching.



Regular Expressions Table

The table at the top of the Regex Attachment Handler dialog shows the current regular expressions you have configured. The **first capture group** is used to determine what data to extract, so if you have other groups in the expression, make sure to include ":" to make them non-capturing. There is an example that shows how to extract data from the OBX.5.5 component in an HL7 v2.x message:

- (?:OBX\| (?:[^|]*\|){4} (?:[^|^]*^\|){4}) ([^|^r\n]*)(?:[|^r\n]|\$)

Click the **New / Delete** buttons to add or remove regular expressions from the table.

For each regular expression you can also specify a MIME type. This supports [Velocity Variable Replacement](#), so you can use source map variables here.

String Replacement Tables

The bottom section of the dialog has two tables, for **Inbound Replacements** and **Outbound Replacements**. The inbound table determines what replacements will be made on the attachment data after it's extracted from the message, but before it gets stored in the database. The outbound table determines what replacements will be made on the attachment data right before it gets reinserted into the message when a destination connector is about to dispatch data to an external system.



Standard Java String escape characters apply here. For example to replace a backslash, you'll actually want to use **two** backslashes ("\\") in the Replace All column.

DICOM Attachment Handler Properties

The DICOM attachment handler does not have properties to configure. When used, the handler will automatically take the incoming DICOM data and extract all pixel data into one or more attachments. The resulting raw data will **not** have an attachment replacement token, but instead will be the Base64 encoded representation of the DICOM message without the pixel data. If the [DICOM Data Type](#) is used in a filter / transformer, it will automatically serialize this Base64 data into an XML message containing all header / tag data.

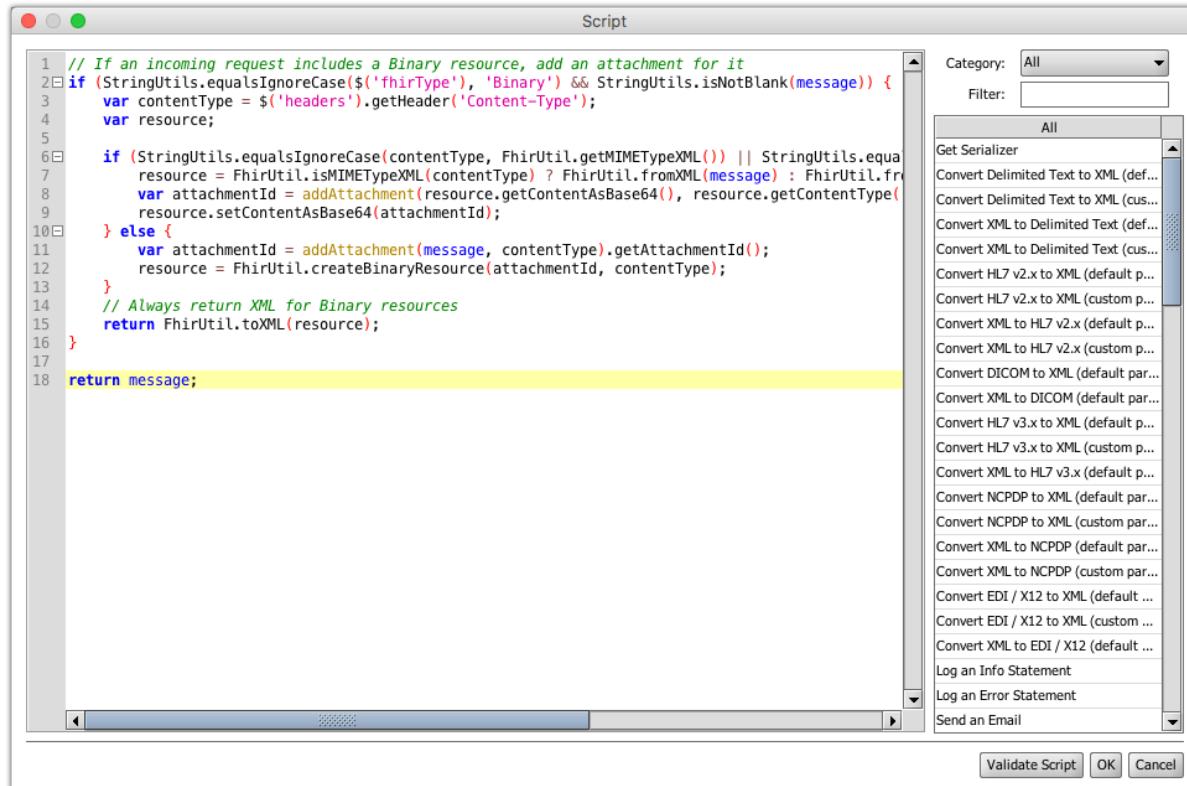
Because there is no attachment replacement token, to reattach DICOM messages on the destination connector side a special token is used:

- \${DICOMMESSAGE}

This indicates to the destination connector that the encoded data should be merged with any pixel data attachments into a final binary representation before being dispatched to the external system.

JavaScript Attachment Handler Properties

This attachment handler allows you to write a custom JavaScript script to handle extracting attachments. For additional information on how to work with JavaScript in NextGen Connect Integration Engine, see [NextGen Connect Integration Engine and JavaScript](#) and [Attachment JavaScript Functions](#).



Scope Variables

In addition to the standard global scope variables, the following are available from within the JavaScript attachment handler script:

- **message**: This is the raw inbound message string. If the data was passed in as a raw byte array, this variable will be the Base64 encoded string representation of the data.
- **binary**: This is a boolean that indicates whether the inbound data was passed in as a raw byte array.
- **sourceMap**: You have access to any variables in the source map. For additional information, see [Variable Maps](#).

Extracting Attachments

Use the following method to extract and store attachments from the attachment script:

addAttachment(data, type)

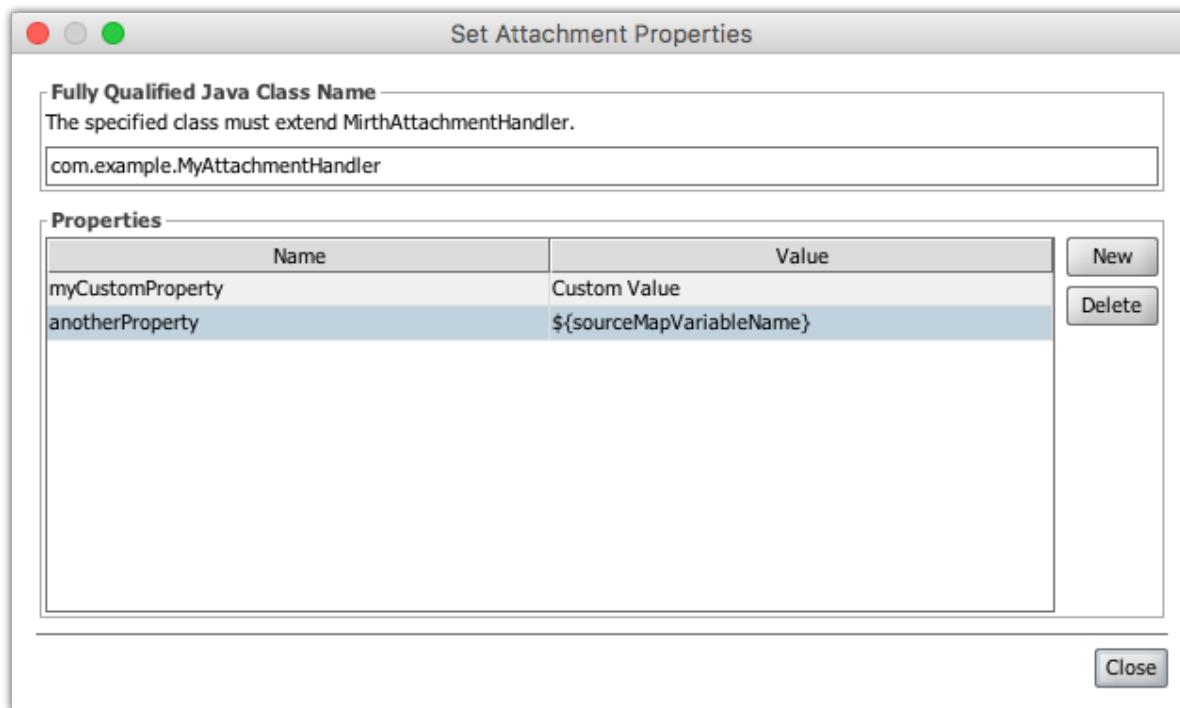
Creates and stores an attachment for the current message.

Parameters		
Name	Type	Description
data	String or Byte array	The actual attachment data to insert. May be either a String or a byte array. If a string is used, it's assumed to be a Base64 encoded representation of the actual attachment data.
type	String	The MIME type of the attachment. For additional information see Attachment Handlers .
<Return Value>	Attachment	The inserted Attachment object.

The resulting Attachment object contains the ID you need to inject back into the message. For more information, look at the [User API](#). The return value for the JavaScript attachment script should be the final message string, with any attachments extracted out and replaced with attachment replacement tokens.

Custom Attachment Handler Properties

This attachment handler gives you full control over the attachment extraction process by allowing you to provide a custom Java implementation of MirthAttachmentHandlerProvider. In the properties you specify the class name, and any properties you want to pass in.

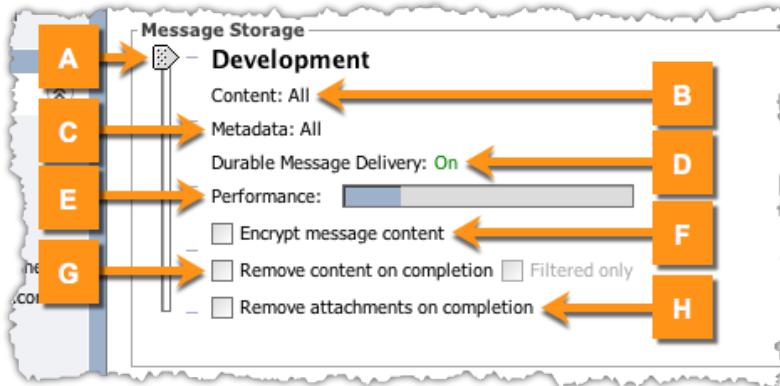


Properties

This is a map of string keys / string values that gets passed into your custom attachment handler implementation. When receiving a message, the entire RawMessage object including source map data will be available, but it will be up to your custom implementation to actually replace source map variables.

Message Storage Settings

This section of the [Summary Tab](#) allows you to determine how much message data your channel will store, whether to encrypt content, and whether to automatically delete content after messages are finished processing. Changing these settings may affect the availability of certain features, like queuing.

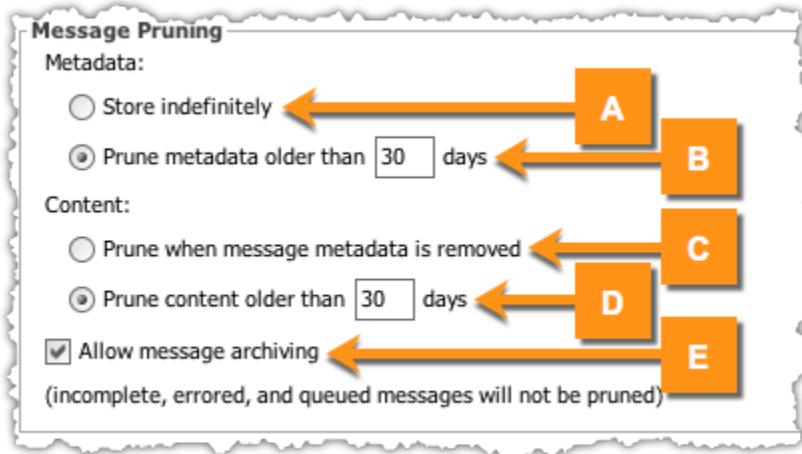


Item	Name	Description
A	Storage Slider	<p>Use this slider bar to change how much data to store as messages process through the channel. The options are:</p> <ul style="list-style-type: none"> • Development: All data will be stored. • Production: Everything except Processed Raw, Transformed, Response Transformed, and Processed Response content will be stored. • Raw: Only Raw / Source Map content and attachments will be stored. Destination queuing is not supported in this mode, but source queuing is still supported. • Metadata: No message content or attachments will be stored, only metadata (for additional information, see About Message Data). Source and destination queuing are not supported in this mode. • Disabled: No message metadata, content, or attachments will be stored. Source and destination queuing are not supported in this mode.
B	Content	Shows what message content will be stored for the currently selected storage settings.
C	Metadata	Shows what message metadata will be stored for the currently selected storage settings. Includes custom metadata columns.
D	Durable Message Delivery	Shows whether Durable Message Delivery is currently enabled based on the selected message storage settings. If enabled, unfinished messages will automatically be recovered and processed if the channel gets halted, or if the server suddenly goes down for any reason. If set to Reprocess only , unfinished messages will not be automatically reprocessed, but you still have the option of manually reprocessing them from the message browser.
E	Performance	Shows a relative estimation of performance for each storage option. When storage is Disabled , performance is highest, at the cost of not having durable message delivery, or the ability to view messages in the message browser.
F	Encrypt message content	If enabled, content stored in the database will be encrypted. Messages that are stored while this option is enabled will still be viewable in the message browser, but the content will not be searchable.

G	Remove content on completion	Remove message content once the message has completed processing. Not applicable for messages that are errored or queued. If Filtered only is also checked, only content for filtered connector messages will be removed.
H	Remove attachments on completion	Remove message attachments once the message has completed processing. Not applicable for messages that are errored or queued.

Message Pruning Settings

These settings allow you to decide when to prune and archive message data (if at all). Note that pruning does not happen automatically unless you enable the scheduler in the [Data Pruner Settings Tab](#). For more information about message data in general, see [About Message Data](#).



Item	Name	Description
A	Store indefinitely	If selected, message metadata will never be pruned.
B	Prune metadata	If selected, the field here determines how long to retain message metadata before it gets pruned.
C	Prune when metadata is removed	If selected, message content will follow the same pruning options set above for the metadata.
D	Prune content	If selected, the field here determines how long to retain message content before it gets pruned.
E	Allow message archiving	If checked and the data pruner and archiver are enabled, messages in this channel will be archived first before being pruned.

Custom Metadata Columns

Custom metadata columns are user-created columns that show up in the [Metadata Table](#) in the [Message Browser](#), and that are also searchable. As messages process through a channel, these columns are populated from the [Variable Maps](#) using a configurable map key.

Custom Metadata				
	Column Name	Type	Variable Mapping	
SOURCE		STRING	mirth_source	<input type="button" value="Add"/> <input type="button" value="Delete"/> <input type="button" value="Revert"/>
TYPE		STRING	mirth_type	
MRN		NUMBER	patient_mrn	
DOB		TIMESTAMP	patient_dob	
INPATIENT		BOOLEAN	inpatient	

Column	Description
Column Name	The name of the actual database column. This name will also be shown in the metadata table when viewing the message browser for the channel.
Type	<p>The type of column to create.</p> <ul style="list-style-type: none"> • STRING: A textual string value. • NUMBER: A numeric value. This may be an integer or floating point value. • BOOLEAN: A true/false value. • TIMESTAMP: A date/time value.
Variable Mapping	<p>The key to look up values in all available variable maps for each connector. As explained in the Variable Maps section, the following maps (if available) will be checked in this order:</p> <ul style="list-style-type: none"> • Response • Connector • Channel • Source • Global Channel • Global • Configuration

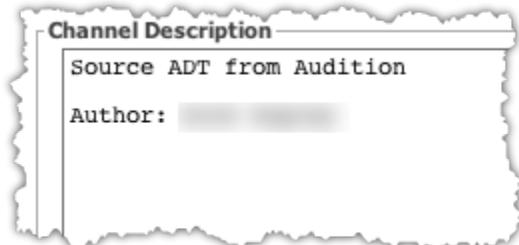
Modifying Custom Metadata Columns

Use the **Add** / **Delete** buttons to create and remove entries from the table. At any time you can press the **Revert** button which will restore the table back to the state it was when you first entered the Edit Channel View.

! Renaming, deleting or changing the type of existing custom metadata columns will delete all existing data for that column. This takes effect the next time the channel is deployed.

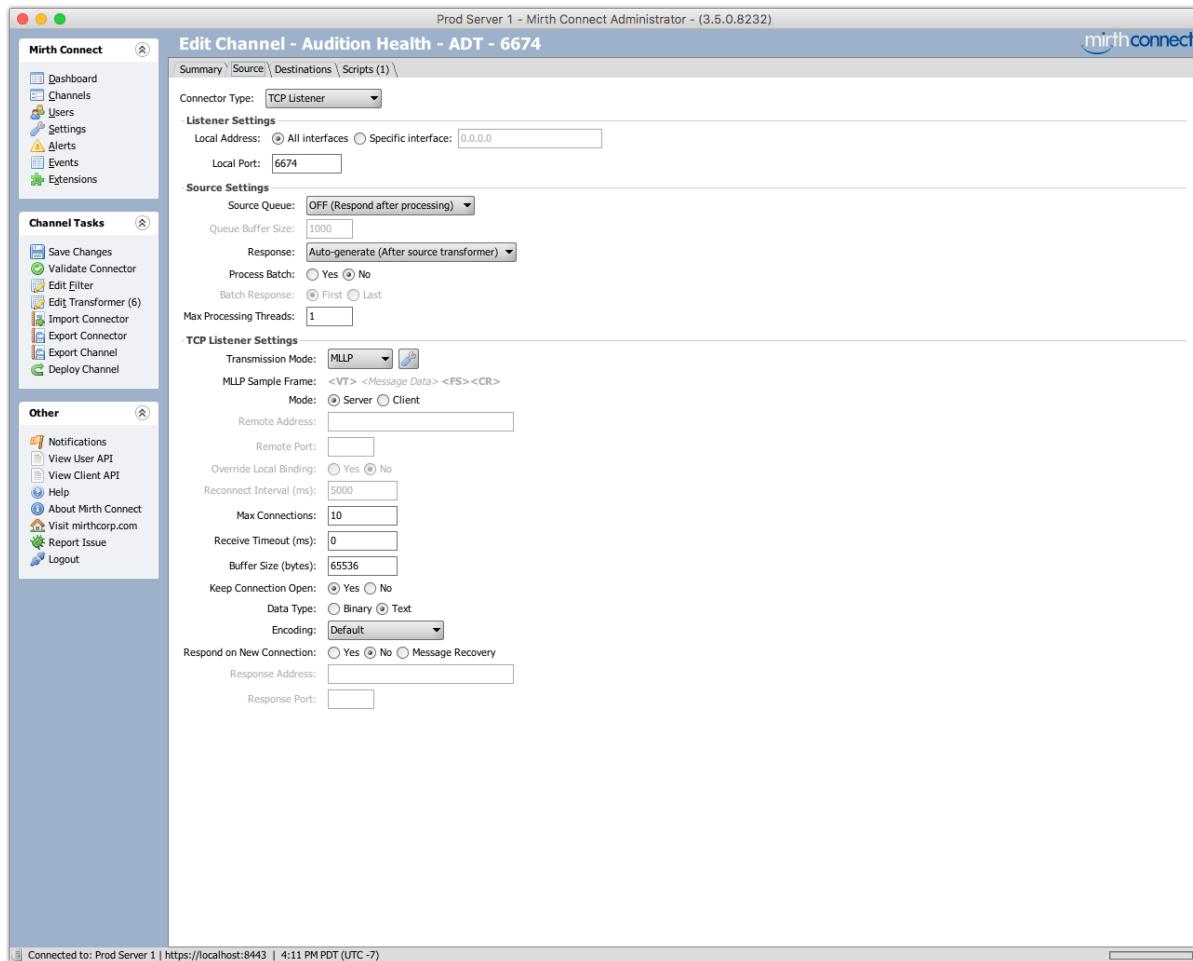
Channel Description

This is a general description text area for your channel, allowing you to describe the channel's purpose, who created it, who to contact in case of problems, etc.



Source Tab

This is where the source connector is configured. This includes the source connector properties, and the source filter / transformer scripts.



Choose a Source Connector

This is done simply by selecting a connector type from the drop-down menu at the very top of the tab:



After selecting a source connector type, the settings shown here are broken up into different categories depending on the type of connector chosen:

- Listener Settings
- Polling Settings
- Source Settings
- HTTP Authentication Settings
- Source Connector Properties

Listener Settings

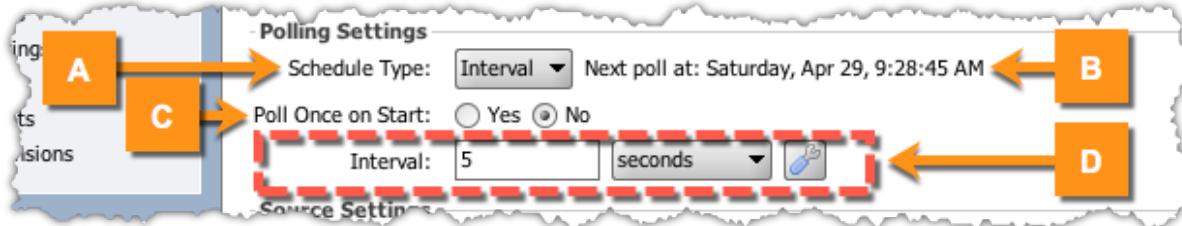
These properties are shown when the connector is a TCP socket-based listener (server), and allow you to specify the local network interface and port to listen on. Supported connectors include the [DICOM Listener](#), [HTTP Listener](#), [TCP Listener](#), and [Web Service Listener](#).



Item	Name	Description
A	Local Address	If All interfaces is selected, the connector will listen on all interfaces, using address 0.0.0.0. If Specific interface is selected, the connector will listen on the specific network interface address given in the accompanying field.
B	Local Port	The TCP port on which to listen for connections.

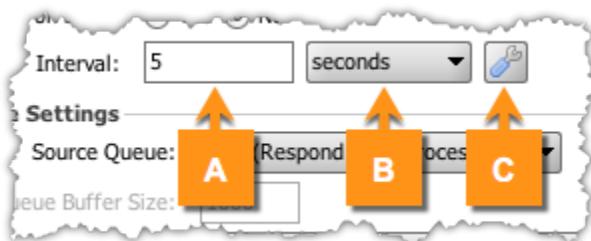
Polling Settings

These settings are for connectors that do not listen passively for messages, but instead actively poll according to a specific schedule. This may be once every few seconds, at a specific time of day, or something more complex. Supported connectors include the [Database Reader](#), [File Reader](#), and [JavaScript Reader](#).



Item	Name	Description
A	Schedule Type	Determines how the polling schedule is set. This could be on a specific interval, a time of day, or a custom expression.
B	Next poll at	Given the current polling settings, shows the next date and time that the source connector will poll for new messages (assuming it is deployed). This value is updated once when you navigate to the Source Tab , or when you update the schedule settings. It does not continue to update automatically.
C	Poll Once on Start	Select Yes to immediately poll once when the channel is started. All subsequent polling will follow the specified scheduling settings.
D	Schedule Settings	These are settings specific to the schedule type. This will be one of: <ul style="list-style-type: none"> • Interval Schedule Settings • Time Schedule Settings • Cron Schedule Settings

Interval Schedule Settings



Item	Name	Description
A	Value	The number of units to wait in between polling windows. The value, when resolved with the unit type, must be less than 24 hours of time.
B	Unit	The unit of time to use in conjunction with the value to determine how long to wait in between polling windows. Available units are:

		<ul style="list-style-type: none"> • Milliseconds • Seconds • Minutes • Hours
C	Advanced	Allows configuring day-of-week / day-of-month settings and time ranges. For additional information, see Advanced Settings .

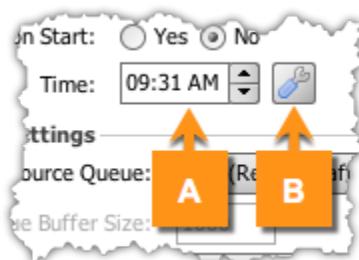
i Only one polling window will ever be active at any given time for a given source connector. For example, if you have a channel set to poll every 5 seconds, but a single message takes 6 seconds to process, a new polling job will **not** begin while the first one is still working. In this example the "effective" polling interval would be 10 seconds, because every other poll trigger gets skipped.

i The interval does not "start counting down" from the time when the channel is deployed/started. Instead, it starts on a consistent schedule based on the start of the time range set in the [Advanced Settings](#) (or 12:00 AM if the default value of All Day is selected).

Example: To poll every hour *on the hour*, you can just leave the advanced settings as their defaults (All Day range), and choose 1 hour for your interval.

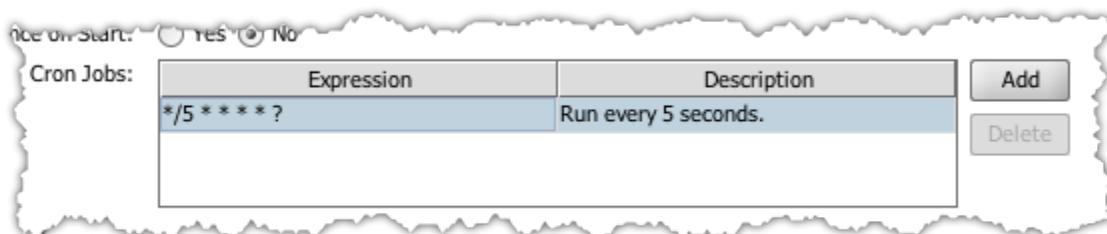
Example: To poll every hour *on the 30 minute mark* (12:30, 1:30, etc.), leave the interval as 1 hour, and set the Active Time Range to 12:30 AM - 11:30 PM.

Time Schedule Settings



Item	Name	Description
A	Time	The specific time of day to poll.
B	Advanced	Allows configuring day-of-week / day-of-month settings. For additional information, see Advanced Settings .

Cron Schedule Settings



This schedule type gives you full control to enable advanced, nuanced polling schedules using Cron-like expressions. Click the **Add / Delete** buttons to add or remove entries from the table. The expressions must abide by the following format:

Field	Required?	Allowed Values	Special Characters
Seconds	Yes	0-59	, - * /
Minutes	Yes	0-59	, - * /
Hours	Yes	0-23	, - * /
Day of Month	Yes	1-31	, - * ? / L W
Month	Yes	1-12 or JAN-DEC	, - * /
Day of Week	Yes	1-7 or SUN-SAT	, - * ? / L #
Year	No	1970-2099	, - * /

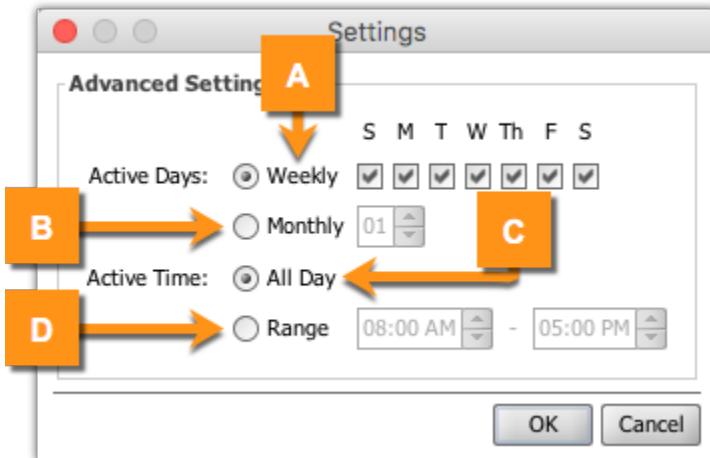
Special Characters	
*	All values
?	No specific range
-	Used to specify ranges
,	Used to specify list of values
/	Used to specify increments
L	Used to specify the "last of"
W	Used to specify the nearest weekday
#	Used to specify the Nth day of the month

Example: The expression "0 */5 8-17 * * ?" means to poll every 5 minutes starting at 8 AM and ending at 5 PM every day.



Specifying both a day-of-week and day-of-month is not supported. A "?" character must be used in one of these fields.

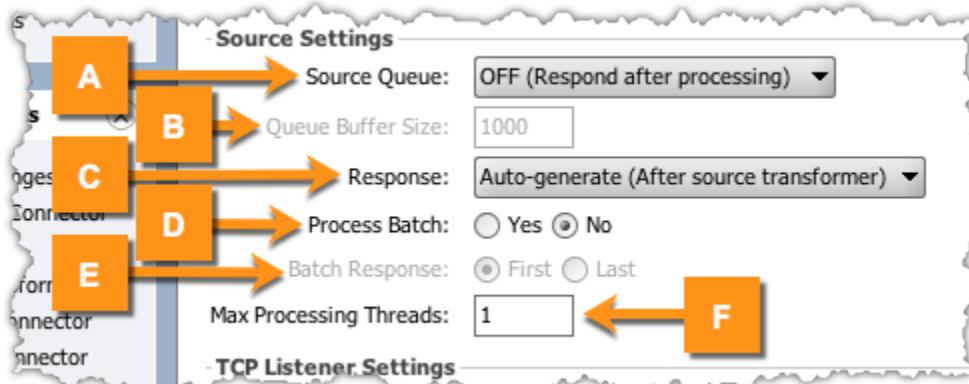
Advanced Settings



Item	Name	Description
A	Active Days: Weekly	Select this option to only poll on the days of the week that are checked to the right.
B	Active Days: Monthly	Select this option to only poll on the specified day of the month.
C	Active Time: All Day	If selected, polling may occur at any time during the day.
D	Active Time: Range	If selected, polling will only occur during the specified time range.

Source Settings

These are general settings that apply to *all* source connectors. It includes configuring the source queue, the response to send back to originating systems, batch processing, and maximum processing threads.

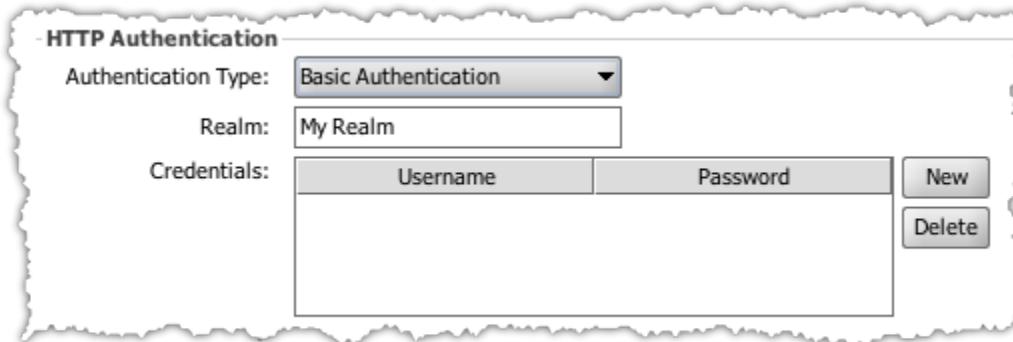


Item	Name	Description
A	Source Queue	<p>This determines whether the source queue is enabled. It also determines <i>when</i> the selected response will be sent back to the originating system.</p> <ul style="list-style-type: none"> OFF (Respond after processing): Selecting OFF will process the message before sending the response. In this scenario you may use the response from destinations, the response map, or the post-processor. This is the Default selection. ON (Respond before processing): Selecting ON will queue messages and immediately send a response. In this scenario you may only choose to not respond with anything, or to use an auto-generated response.
B	Queue Buffer Size	The buffer size for the source queue, only apply when Source Queue = ON . Up to this many connector messages may be held in memory at once when queuing. Default = 1000.
C	Response	<p>Determines what response to send back to the originating system. You may choose a destination's response, the post-processor return value, a response map variable, an auto-generation option, or None indicating that you don't want to send a response back at all.</p> <p>Select Auto-generate to send a response generated by the inbound data type using the raw message:</p> <ul style="list-style-type: none"> Before processing: Response generated before the channel processes the message (SENT status) After source transformer: Response generated after the channel processes the message (source connector status) Destinations completed: Response generated after the channel processes the message, with a status based on the destination statuses, using a precedence of <i>ERROR, QUEUED, SENT, FILTERED</i>.
D	Process Batch	Select Yes to enable batch processing. Batch messages are only supported if the source connector's inbound properties contains a Batch section. More information here: Batch Processing
E	Batch Response	Each message in the batch contains its own response that is generated via the method selected above. Select either the response from the first or last message in the batch to be

		sent back to the originating system.
F	Max Processing Threads	<p>The maximum number of messages that can process through the channel simultaneously. By default this is set to 1, meaning that only one message can process through a channel at any given time (does not include asynchronous processes like the destination queue). Increase this setting can greatly improve channel performance / throughput, at the cost of message order preservation.</p> <p> When this value is greater than 1, message order is NOT guaranteed.</p>

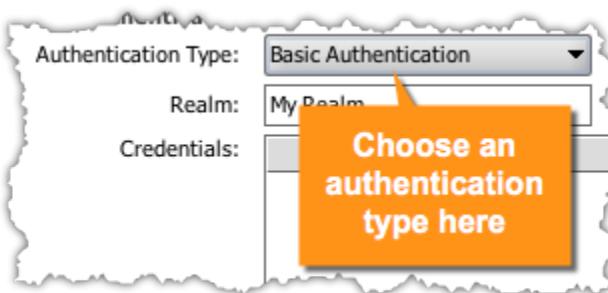
HTTP Authentication Settings

These are for HTTP-based source connectors, and provide automatic user authentication with a variety of supported mechanisms.



Choose an Authentication Type

This is done simply by selecting a type from the drop-down menu at the top of the settings:

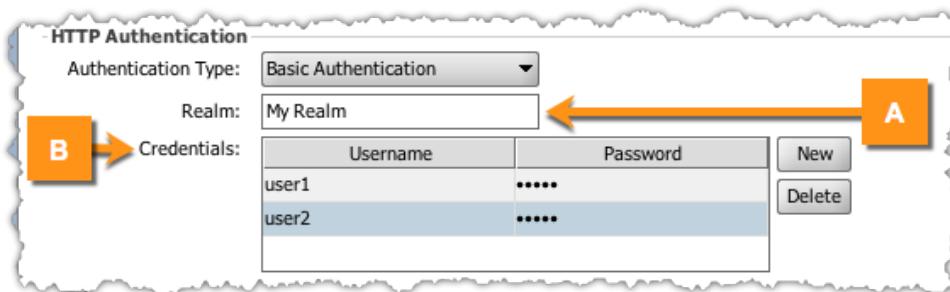


After selecting an authentication type, the settings shown below will change. The following authentication types are supported:

- Basic HTTP Authentication
- Digest HTTP Authentication
- JavaScript HTTP Authentication
- Custom Java Class HTTP Authentication
- OAuth 2.0 Token Verification

Basic HTTP Authentication

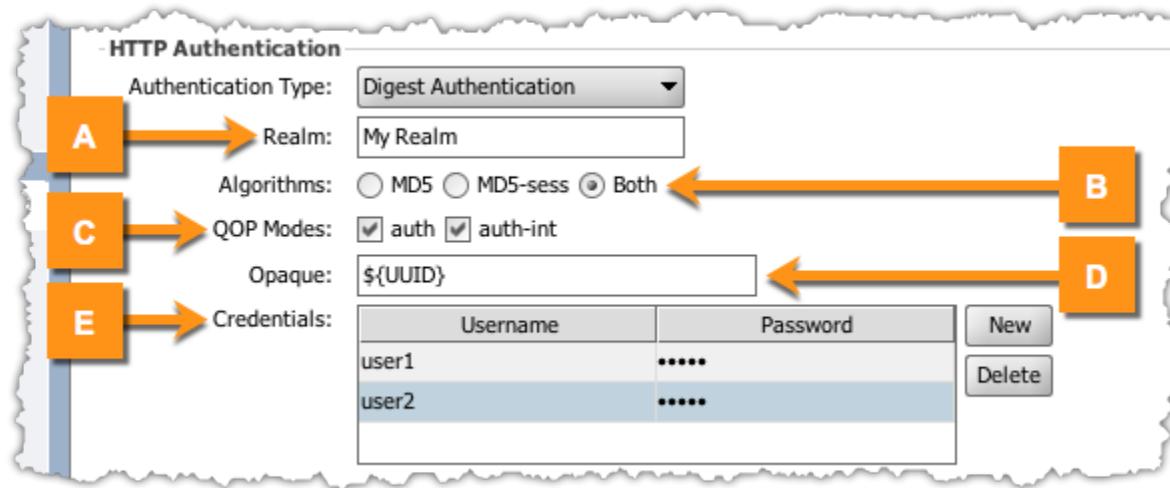
Provides [Basic Authentication](#) support for HTTP-based source connectors.



Item	Name	Description
A	Realm	The protection space for this server.
B	Credentials	Username and password pairs to authenticate users with. At least one pair is required.

Digest HTTP Authentication

Provides [Digest Authentication](#) support for HTTP-based source connectors.



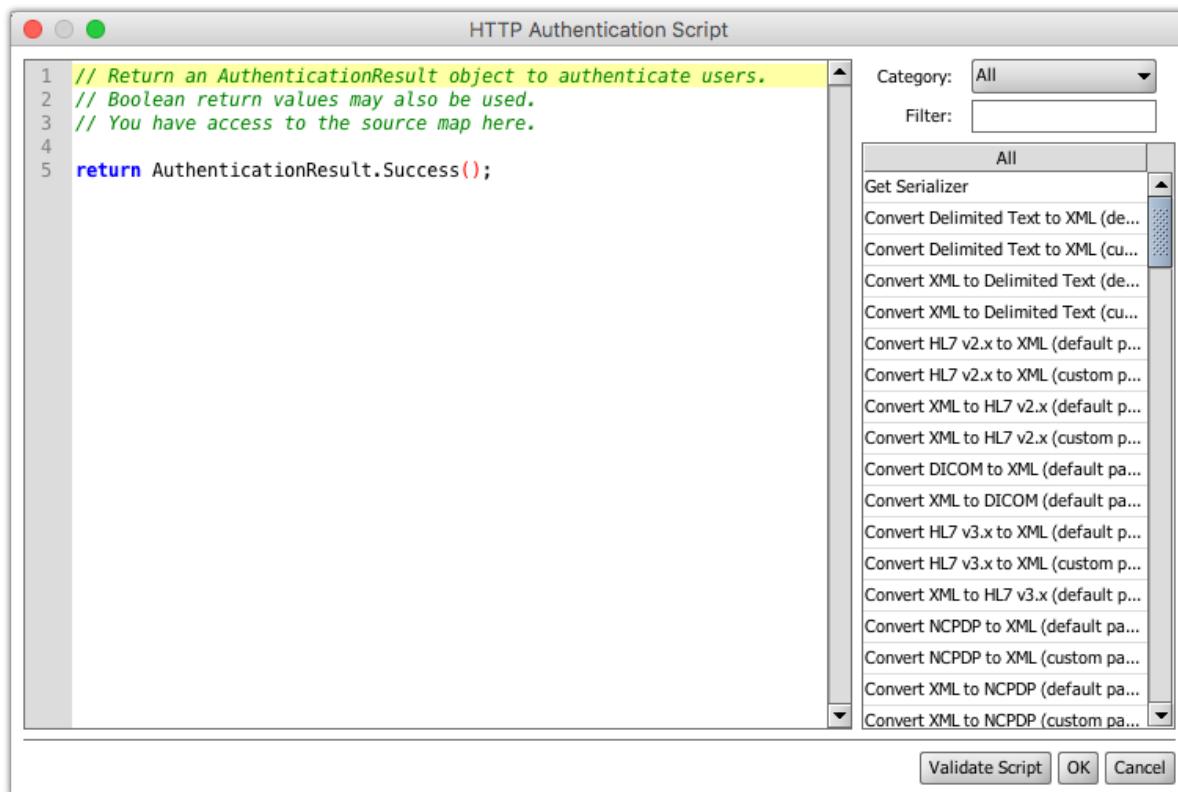
Item	Name	Description
A	Realm	The protection space for this server.
B	Algorithms	Specifies the digest algorithms supported by this server. <ul style="list-style-type: none"> MD5: The security data A1 will contain the username, realm, and password. MD5-sess: The security data A1 will also contain the server and client nonces.
C	QOP Modes	The quality of protection modes to support. <ul style="list-style-type: none"> auth: Regular auth with client none and count in the digest. auth-int: Same as auth, but also with message integrity protection enabled.
D	Opaque	A string of data that should be returned by the client unchanged. Velocity Variable Replacement is supported in this field. The default value, \${UUID}, means that a randomly generated universally unique identifier will be sent back on each digest challenge.
E	Credentials	Username and password pairs to authenticate users with. At least one pair is required.

JavaScript HTTP Authentication

Allows you to authenticate users with a custom JavaScript script. With this script you have access to source map variables, and can choose whether to send a challenged or failure response back to the client.



The default script simply allows all requests to pass. The **Script** field will show **<Default Script Set>** if the default script is currently being used. If you've made any modification to the script, the **Script** field will show **<Custom Script Set>**. Click on the **Script** field to edit the JavaScript:



This script expects either a boolean (**true** to accept the request, **false** to send back a failure response) or an **AuthenticationResult** object to be returned (for additional information, see [User API](#)). There are three types of results you can return:

- **AuthenticationResult.Success():** The request will be accepted and processed through the channel.
- **AuthenticationResult.Challenged(authenticateHeader):** The request will not be processed through the channel. A 401 response will be sent back to the client, with a given WWW-Authenticate header value.
- **AuthenticationResult.Failure():** The request will not be processed through the channel. A 401 response will be sent back to the client, without any WWW-Authenticate header or any additional information.

If a Challenged/Failure result is returned, the **AuthenticationResult** object also allows you to add custom headers to include on the HTTP response sent back to the client.

For more information on using JavaScript within NextGen Connect Integration Engine, see [NextGen Connect Integration Engine and JavaScript](#).

Custom Java Class HTTP Authentication

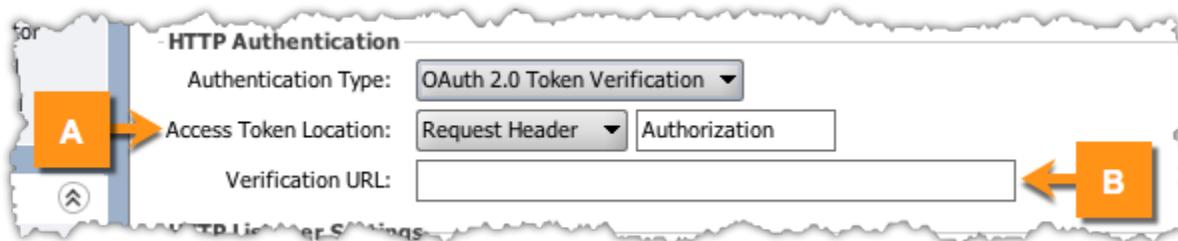
This authentication method gives you full control by allowing you to specify your own custom-developed Authenticator implementation.



Item	Name	Description
A	Class Name	The fully-qualified Java class name of the Authenticator class to use.
B	Properties	Custom string properties to pass into your class.

OAuth 2.0 Token Verification

This feature performs a GET request to an external HTTP endpoint, passing an OAuth access token as either a request header or query parameter. If the response code from this endpoint is ≥ 400 , the request will be rejected and not processed through the channel.



Item	Name	Description
A	Access Token Location	Determines where the access token is located in client requests. <ul style="list-style-type: none">Request Header: The field to the right specifies the HTTP header to pull the access token from. This same header will be sent in the request to the verification URL.Query Parameter: The field to the right specifies the query parameter to pull the access token from. This same parameter will be sent in the request to the verification URL.
B	Verification URL	The HTTP URL to perform a GET request to for access token verification. If the response code is ≥ 400 , the authentication attempt is rejected by the server, and the request will not process through the channel.



This token verification feature does **not** constitute a fully-functioning OAuth 2.0 server. It does not authenticate or authorize users directly, but simply delegates this to the *actual* OAuth server the Verification URL points to.

Source Connector Properties

This section refers to the actual connector-specific settings. Here is a list of source connectors supported by NextGen Connect Integration Engine

- [Source Connectors](#)
 - [Channel Reader](#)
 - [DICOM Listener](#)
 - [Database Reader](#)
 - [File Reader](#)
 - [HTTP Listener](#)
 - [JMS Listener](#)
 - [JavaScript Reader](#)
 - [TCP Listener](#)
 - [Web Service Listener](#)

Additional source connectors are made available as [commercial extensions](#):

- [Email Reader](#)
- [Serial Connector](#)

Destinations Tab

This is where destination connectors are configured. This includes the destination connector properties, the destination filter / transformer scripts, and the response transformer scripts. From this tab you can rename / reorder / enable / disable / clone destinations, and decide which ones belong in separate destination chains.

Status	Destination	Id	Connector Type	Chain
Enabled	Post to Internal API	1	HTTP Sender	1
Enabled	Append to log	2	File Writer	2
Enabled	Dispatch over MLP	3	TCP Sender	2

Configuration of destination connectors is separated into the following sections:

- [Destination Table](#)
- [Destination Tasks](#)
- [Destination Settings](#)
- [Destination Connector Properties](#)
- [Destination Mappings](#)

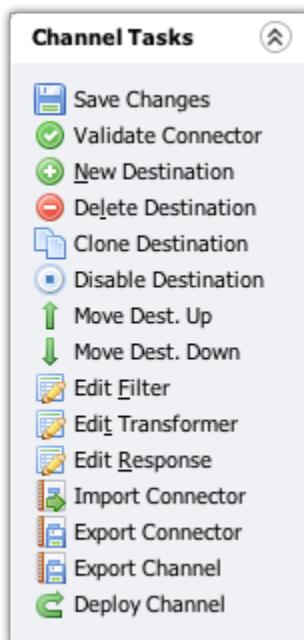
Destination Table

This table shows you all currently configured destinations for your channel. You can see at a glance the type of each destination, whether it's enabled, what chain it belongs to, and more.

Status	Destination	Id	Connector Type	Chain
<input checked="" type="radio"/> Enabled	Post to Internal API	1	HTTP Sender	1
<input checked="" type="radio"/> Enabled	Append to log	2	File Writer	2
<input checked="" type="radio"/> Enabled	Dispatch over MLLP	3	TCP Sender	2

Column	Description
Status	Indicates whether the destination is enabled or disabled. Only enabled destinations may process messages on a deployed channel. A channel must have at least one destination enabled.
Destination	Double-click this cell to edit the name of the destination. Note that while a destination may be renamed, its metadata ID will remain the same.
Id	The metadata ID that uniquely identifies this destination within the current channel.
Connector Type	The type of destination connector. To change this, select the destination from the table and choose a new type from the drop-down menu directly below the table.
Chain	The chain this destination connector belongs to. The first destination in the table is automatically placed into chain #1. To start a new chain, select a subsequent destination from the table, and uncheck the Wait for previous destination check box directly below the table. For additional information, see The Message Processing Lifecycle .

Destination Tasks

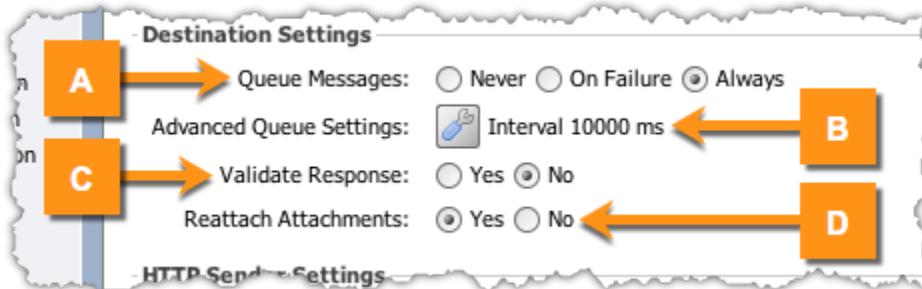


In addition to the general [Edit Channel Tasks](#), several context-specific tasks are unique to the [Destinations Tab](#):

Task Icon	Task Name	Description
	New Destination	Creates a new destination and adds it to the table above.
	Delete Destination	Deletes the currently selected destination and removes it from the table above. Note that a channel must have at least one enabled destination.
	Clone Destination	Copies the currently selected destination and adds it to the table above.
	Enable Destination	Marks this destination as ready to process messages at deploy time.
	Disable Destination	Marks this destination as not ready to process messages at deploy time. Note that a channel must have at least one enabled destination.
	Move Destination Up	Moves the currently selected destination one row higher in the table above.
	Move Destination Down	Moves the currently selected destination one row lower in the table above.
	Edit Response	Enters the Edit Transformer View for the currently selected destination's response transformer.

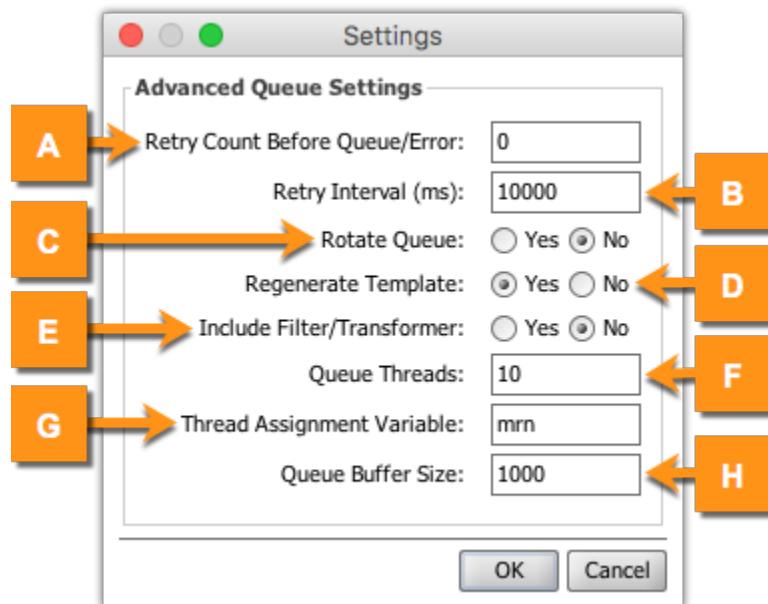
Destination Settings

These are general settings that apply to *all* destination connectors. They include configuring the destination queue, whether to validate responses, and whether to re-attach attachments on outbound messages.



Item	Name	Description
A	Queue Messages	<ul style="list-style-type: none"> Never: Disable the destination queue. On Failure: Attempt to send the message first before queuing it. This will allow subsequent destinations and the Postprocessor to use the response from this destination if it successfully sends before queuing. Always: Immediately queue the message. Subsequent destinations and the Postprocessor will always see this destination's response as QUEUED.
B	Advanced Queue Settings	Configure how often to re-attempt queued messages, increase queue threads, and more.
C	Validate Response	Select Yes to validate the response. Responses can only be validated if the response transformer's inbound properties contains a Response Validation section. If validation fails, the message will be marked as queued or errored. For additional information, see Data Types .
D	Reattach Attachments	If enabled, replacement tokens using the \${ATTACH:...} syntax will be automatically replaced with the associated attachment content before the message is sent. If disabled, the tokens will be expanded to the full \${ATTACH:channelId:messageId:attachmentId} syntax which can then be reattached in downstream channels. For additional information, see Attachment Handlers .

Advanced Queue Settings



Item	Name	Description
A	Retry Count Before Queue/Error	The maximum number of times the connector will attempt to send the message before queuing or erroring.
B	Retry Interval (ms)	The amount of time (in milliseconds) that should elapse between retry attempts to send messages. This interval applies to both the queue and initial retry attempts.
C	Rotate Queue	If enabled, when any message fails to be sent from the queue, the connector will place the message at the end of the queue and attempt to send the next message. This will prevent a single message from holding up the entire queue. If the order of messages processed is important, this should be disabled.
D	Regenerate Template	Regenerate the template and other connector properties by replacing variables each time the connector attempts to send the message from the queue. If this is disabled, the original variable replacement will be used for each attempt.
E	Include Filter /Transformer	If enabled, the filter and transformer will be re-executed before every queue send attempt. This is only available when the Regenerate Template setting is enabled.
F	Queue Threads	The number of threads that will read from the queue and dispatch messages simultaneously. Message order is <u>NOT</u> guaranteed if this value is greater than one, unless an assignment variable is used below.
G	Thread Assignment Variable	When using multiple queue threads, this map variable determines how to assign messages to specific threads. If rotation is disabled, messages with the same thread assignment value will always be processed in order.
H	Queue Buffer Size	The buffer size for the destination queue. Up to this many connector messages may be held in memory at once when queuing.

Destination Connector Properties

This section refers to the actual connector-specific settings. The section is labeled according to the connector type, e.g. "HTTP Sender Settings", "JavaScript Writer Settings". Here is a list of destination connectors supported by NextGen Connect Integration Engine:

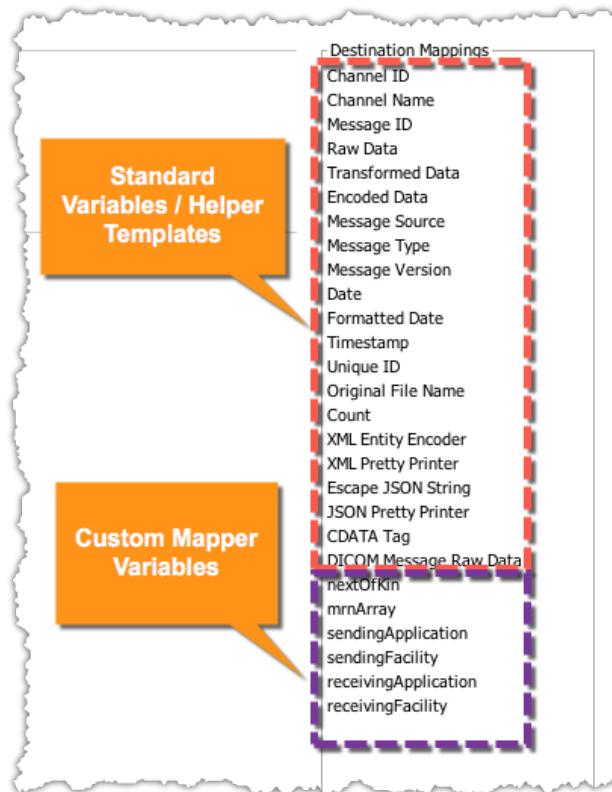
- Destination Connectors
 - Channel Writer
 - DICOM Sender
 - Database Writer
 - Document Writer
 - File Writer
 - HTTP Sender
 - JMS Sender
 - JavaScript Writer
 - SMTP Sender
 - TCP Sender
 - Web Service Sender

Additional destination connectors are made available as commercial extensions:

- Serial Connector
- NextGen Results CDR Connector

Destination Mappings

This section is to the right of the destination connector properties, and allows you to easily drag-and-drop common variables / templates into fields of the connector properties. **Standard** variables / templates are available across all destination connectors. **Custom** mapper variables come from the [Mapper Steps](#) you have added in the current destination or in any previous destinations. For additional information on Velocity replacement, see [Velocity Variable Replacement](#).



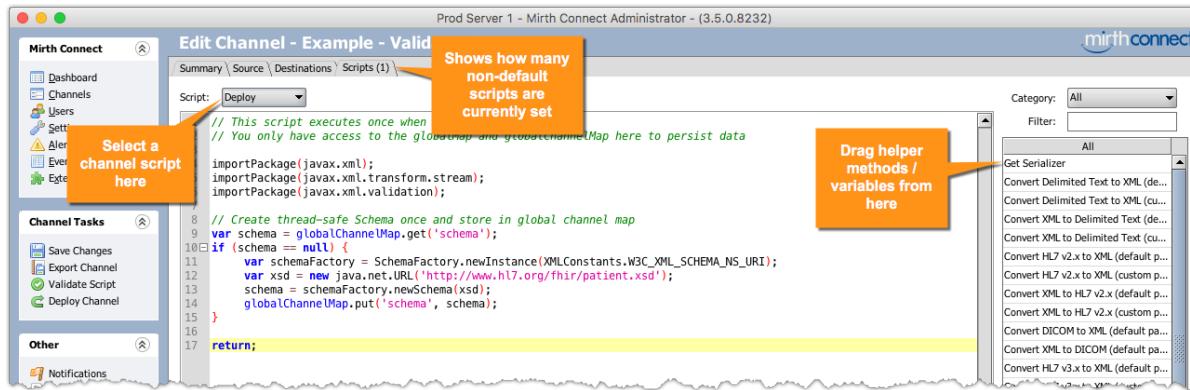
Standard Variables / Templates

Name	Description
Channel ID	The unique ID of the current channel.
Channel Name	The name of the current channel.
Message ID	The unique ID of the current message.
Raw Data	The raw content of the destination connector message (equal to the encoded content of the source connector message).
Transformed Data	The serialized internal representation of the post-transformer message data.
Encoded Data	The state of the message data as it exists in the transformer.

Message Source	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Sending Facility value in MSH.4.1.
Message Type	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Type and Trigger values in MSH.9.1 and MSH.9.2.
Message Version	Depends on the inbound data type for the connector. For HL7 v2.x messages this will usually be the Version value in MSH.12.1.
Date	The current date and time, printed using a standard format.
Formatted Date	The current date and time, printed using a custom user-specified format.
Timestamp	The current epoch time represented in milliseconds.
Unique ID	An auto-generated universally unique identifier string.
Original File Name	Only applicable when the source connector is a File Reader. The name of the file currently being processed.
Count	A number that automatically starts at 1 when the channel is deployed, and increments for each message, or for each time \${COUNT} is used.
XML Entity Encoder	Automatically encodes any special XML characters (like "&") into entities (like "&"). Useful when your message template is XML and you want to inject a custom variable into the inner text of a node.
XML Pretty Printer	Automatically indents and normalizes whitespace for the given XML string.
Escape JSON String	Automatically escapes any special JSON characters (like "{") with backslashes (like "\\{"). Useful when your message template is JSON and you want to inject a custom variable into a string property.
JSON Pretty Printer	Automatically indents and normalizes whitespace for the given JSON string.
CDATA Tag	Inserts a CDATA tag, inside which you can place custom data without having to encode entities.
DICOM Message Raw Data	A special replacement token telling the destination connector to merge the destination connector message with any DICOM pixel data attachments and dispatch the fully merged bytes to the outbound endpoint. Typically used by the DICOM Sender destination.

Scripts Tab

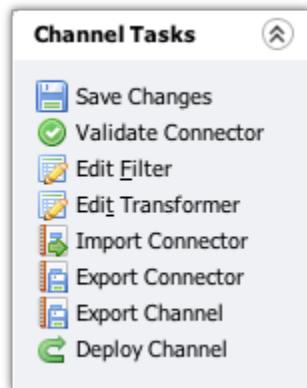
This is where channel-level scripts are configured. Select a script type from the drop-down and edit the script in the text area below. If a number appears next to "Scripts" in the tab, that number represents how many scripts have been edited from the default values. There is also a [Reference List](#) to the right for easy drag-and-drop of common helper methods / code templates.



The following channel-level script types can be edited:

- **Deploy Script:** This script executes once when the channel is deployed. You have access to the global / global channel / configuration maps here. Typically this script is used to perform a one-time operation for the given channel, like load a properties file from disk, or instantiate a database connection.
- **Undeploy Script:** This script executes once when the channel is undeployed. You have access to the global / global channel / configuration maps here. Typically this script is used to cleanup any data created from the deploy script, such as closing a database connection.
- **Preprocessor Script:** This script executes once for every message, after the [attachment handler](#) has run but before the message reaches the source filter/transformer. You have access to "message", a string variable containing the incoming data. Whatever you return from the preprocessor script will be stored as the Processed Raw content and used to feed into the source filter/transformer.
- **Postprocessor Script:** This script executes once for every message, after all destinations have completed processing (not including queued messages which are processed asynchronously). You have access to "message", which is an `ImmutableMessage` object containing information about the state of all connector messages. This script may be used as a general tool to perform a custom cleanup script. It can also be used to return a custom response that may be sent back to the originating system.

Edit Channel Tasks



The following context-specific tasks are available throughout the [Edit Channel View](#):

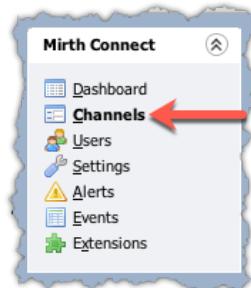
Task Icon	Task Name	Description
	Save Changes	Saves a new revision of the current channel, if anything was actually changed.
	Validate Connector	Validates the currently viewed connector, ensuring that all connector properties are valid and able to be saved.
	Edit Filter	Enters the Edit Filter View for the currently viewed connector.
	Edit Transformer	Enters the Edit Transformer View for the currently viewed connector.
	Import Connector	Imports a connector from an XML file into the current channel. For source connectors, this completely overrides all source connector properties and the source filter/transformer. For destination connectors a new destination will be added to the table.
	Export Connector	Exports the currently viewed connector to an XML file. This will include all connector properties and the connector's filter / transformer.
	Export Channel	Exports the current channel to an XML file. The channel must be saved first. For additional information, see Channel Tasks .
	Deploy Channel	Deploys the current channel.

Edit Filter / Transformer Views

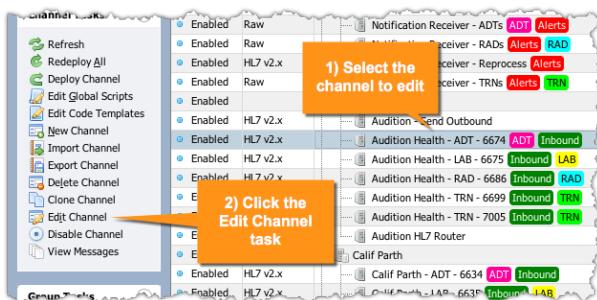
This is where filter rules and transformer steps are configured. Although Edit Filter and Edit Transformer are different views, they are very similar and so are combined into this section. Also, the filter and transformer are actually executed together as a single script for each connector.

Navigation

Click the Channels link in the Mirth Connect task pane at the top-left to enter the [Channels View](#):

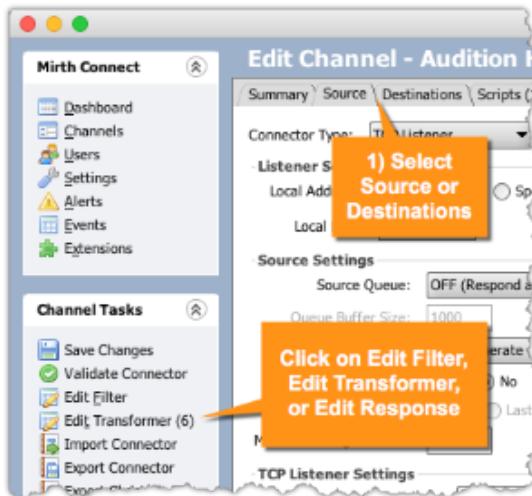


In the [Channel Table](#), select the channel you wish to edit, and click the **Edit Channel** task to the left:



OR, just double-click the channel row in the table.

Once in the [Edit Channel View](#), click on either the **Source Tab** or **Destinations Tab**. Select a destination if necessary. Then, click on the **Edit Filter**, **Edit Transformer**, or **Edit Response** tasks to the left. The **Edit Response** task (for editing the response transformer) will only be visible for destination connectors.

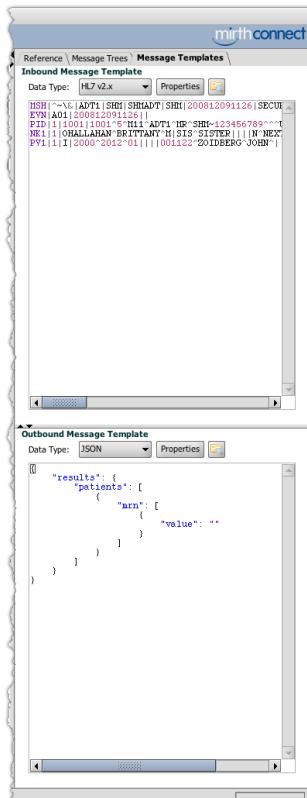


This section is separated into the following topics:

- [Message Templates Tab](#)
- [Message Trees Tab](#)
- [Reference Tab](#)
- [Creating New Rules / Steps](#)
- [Rule / Step Table](#)
- [Filter Rule Properties](#)
- [Transformer Step Properties](#)
- [Response Transformers](#)
- [Working With Iterators](#)
- [Viewing Generated Script](#)
- [Filter Tasks](#)
- [Transformer Tasks](#)

Message Templates Tab

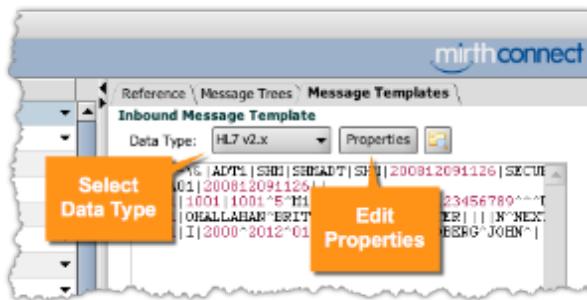
This is located on the right-hand side of the [Edit Filter / Transformer View](#) and allows you to edit the inbound/outbound message templates for your filter or transformer. Both templates can be used in the filter / transformer for easy reference and drag-and-drop capabilities, but only the **outbound** template has any effect on message processing. If an outbound template is specified, you'll see the Message Templates title appear **bold** in the tab list.



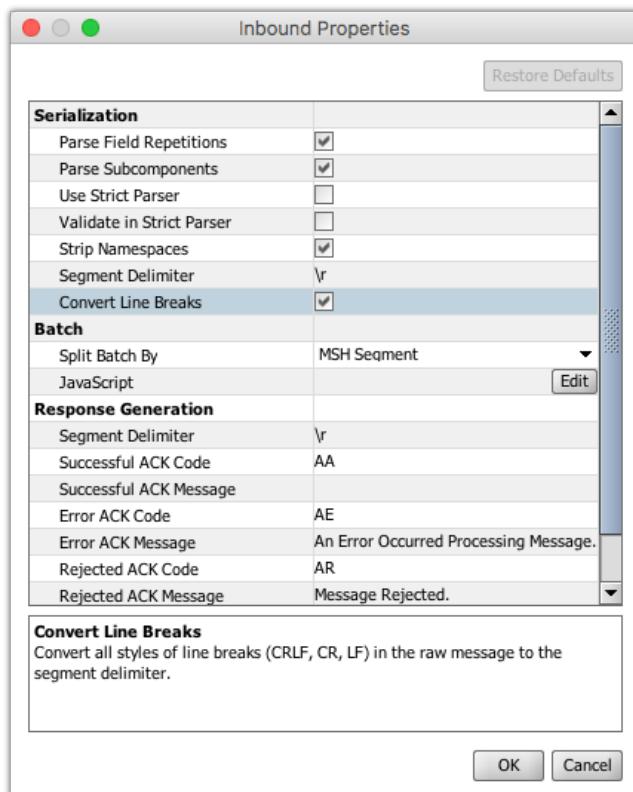
The **Outbound** template is only visible for transformers and response transformers. When you edit a connector's filter, you'll only see the **Inbound** template.

Editing Data Types

A filter only has access to the inbound data type, while a transformer has access to both inbound and outbound data types. Additionally, when editing the destination transformer the inbound data type may not be changed, since it must remain the same as the source connector's outbound data type. However, you can still edit the properties for the destination transformer's inbound data type.

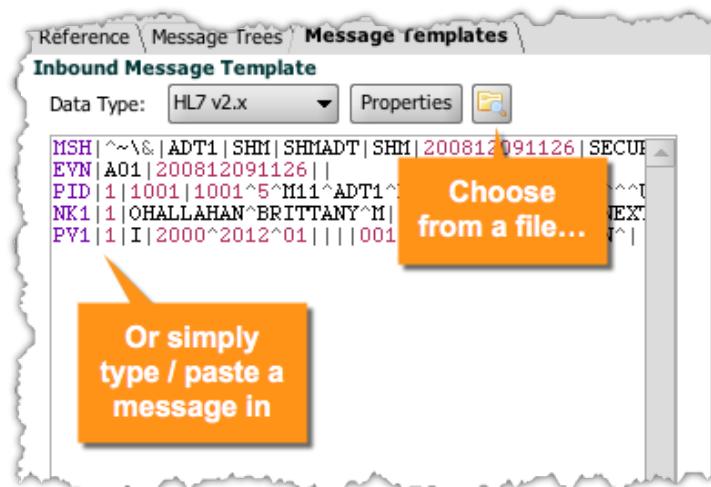


When clicking the **Properties** button, a dialog will open allowing you to edit the various inbound or outbound properties for the data type. This is a subset of the view shown in the [Set Data Types Dialog](#). For additional information, see [About Data Types](#) and [Data Types](#).



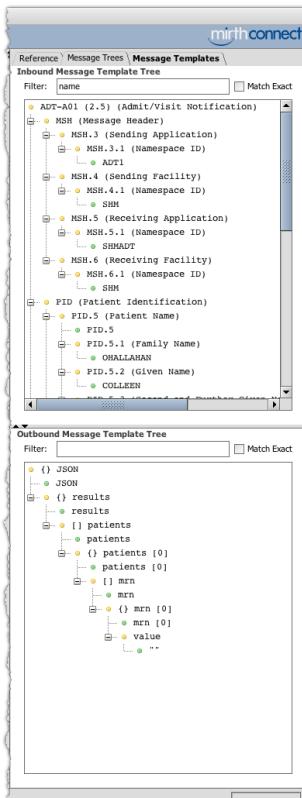
Specifying a Template

You can use the folder icon button next to the Properties button to select a message from your local filesystem. Or, simply type / paste a message into the text area.



Message Trees Tab

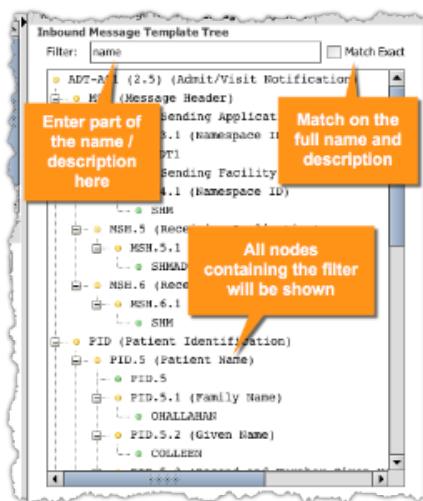
This is located on the right-hand side of the [Edit Filter / Transformer View](#) and allows you to view a tree representation of your inbound/outbound message templates set in the [Message Templates Tab](#).



i The **Outbound** template tree is only visible for transformers and response transformers. When you edit a connector's filter, you'll only see the **Inbound** template tree.

Filter By Node Name / Description

You can quickly find a particular part of your message by using the **Filter** field at the top:

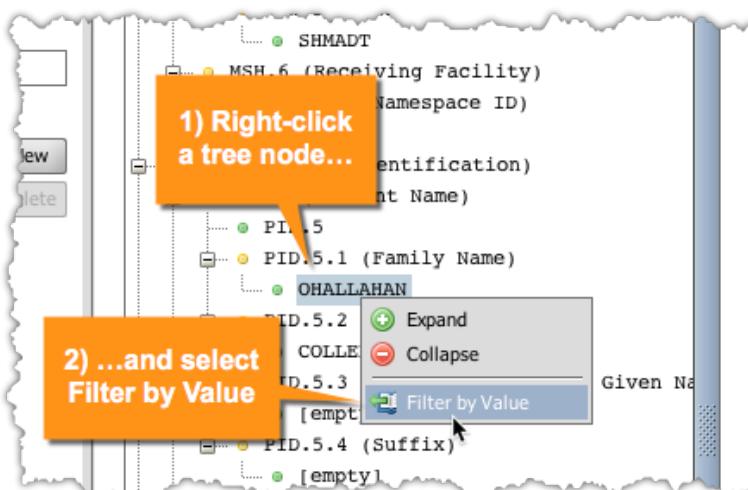


Create a Rule Builder Rule or Mapper Step

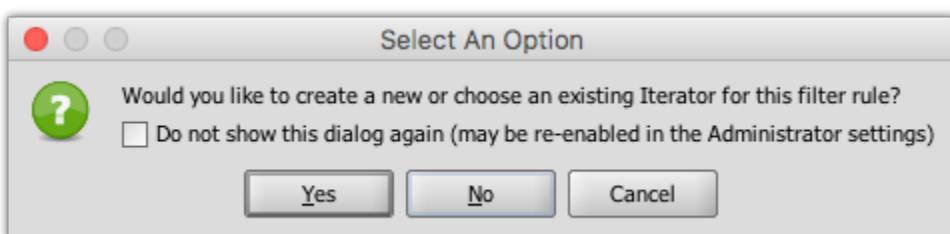
When editing a filter, you can create a [Rule Builder Filter Rule](#) directly from the message tree. When editing a transformer, you can create a [Mapper Transformer Step](#) in the same way.

Method 1:

Right-click the node you wish to filter on or map, and select either **Filter by Value** or **Map to Variable**:



A dialog may be shown asking whether you want to add the rule/step as part of an iterator. More information here: [Working With Iterators](#)

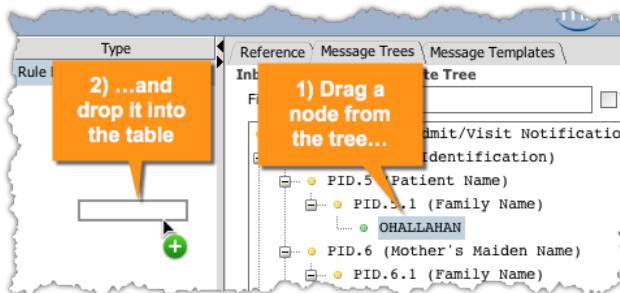


A Rule Builder or Mapper will automatically be added to the table with the selected node populated appropriately:

#	Name	Type
0	Accept message if "msg['PID']['PID.5']['PID.5.1'].toString()" exists	Rule Builder

Method 2:

Drag the node you wish to filter on or map from the tree into the table:



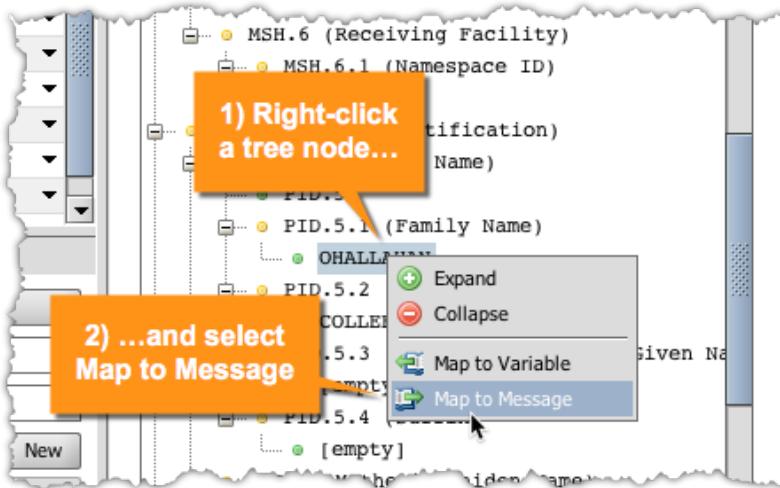
The same steps follow as in Method 1.

Create a Message Builder Step

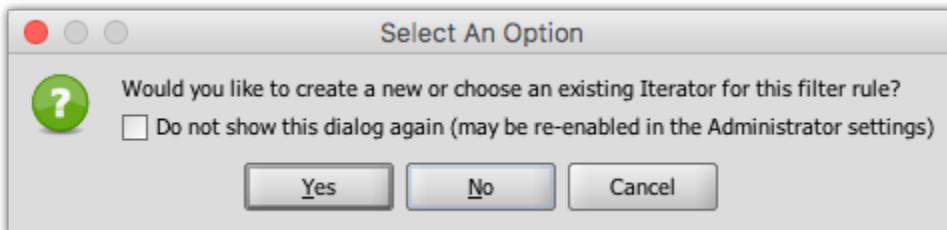
When editing a transformer, you can create a [Message Builder Transformer Step](#) directly from the message tree.

Method 1:

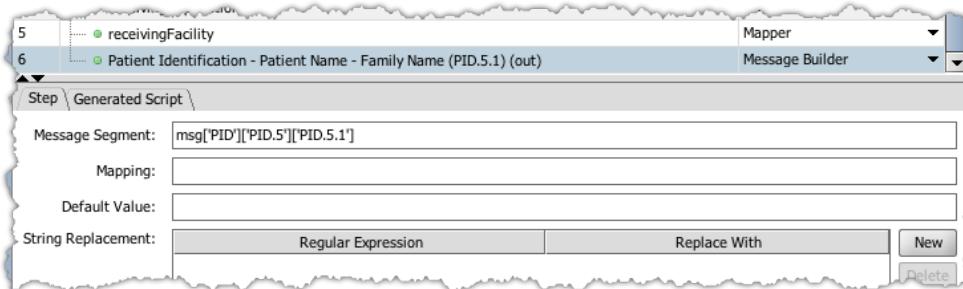
Right-click the node you want to modify, and select **Map to Message**:



A dialog may be shown asking whether you want to add the step as part of an Iterator. For additional information, see [Working With Iterators](#).

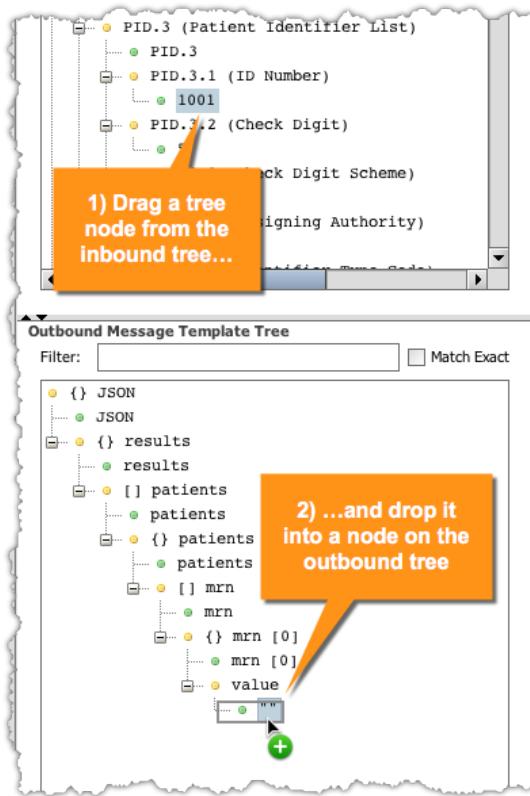


A Message Builder will automatically be added to the table with the selected node populated appropriately:



Method 2:

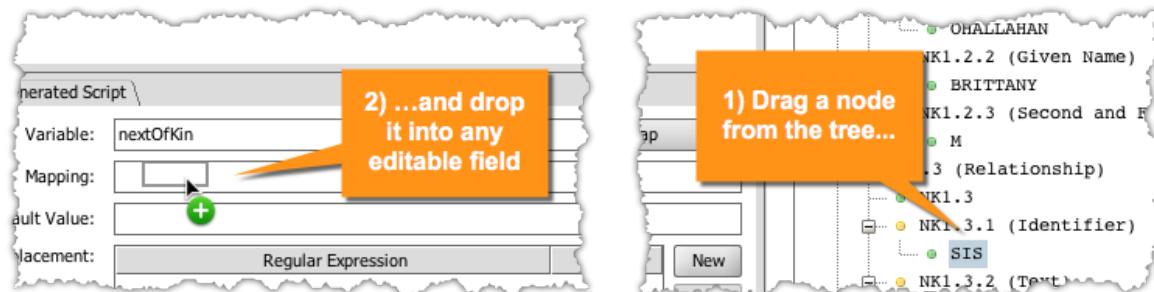
Drag the node you wish to map *from* the inbound message tree, and drop it into the node you wish to map *to* in the outbound message tree:



The same steps follow as in Method 1.

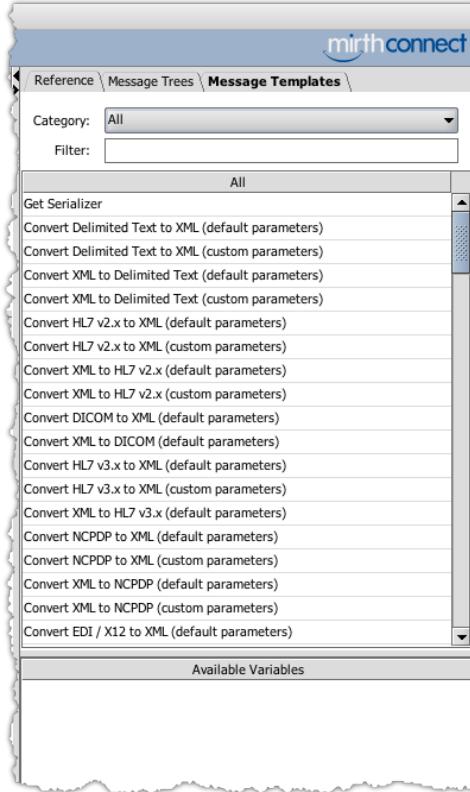
Drag-and-Drop Field Values

The message tree also allows you to easily populate values into your filter rules or transformer steps. Simply drag a node from the inbound or outbound tree, and drop it into any editable field:



Reference Tab

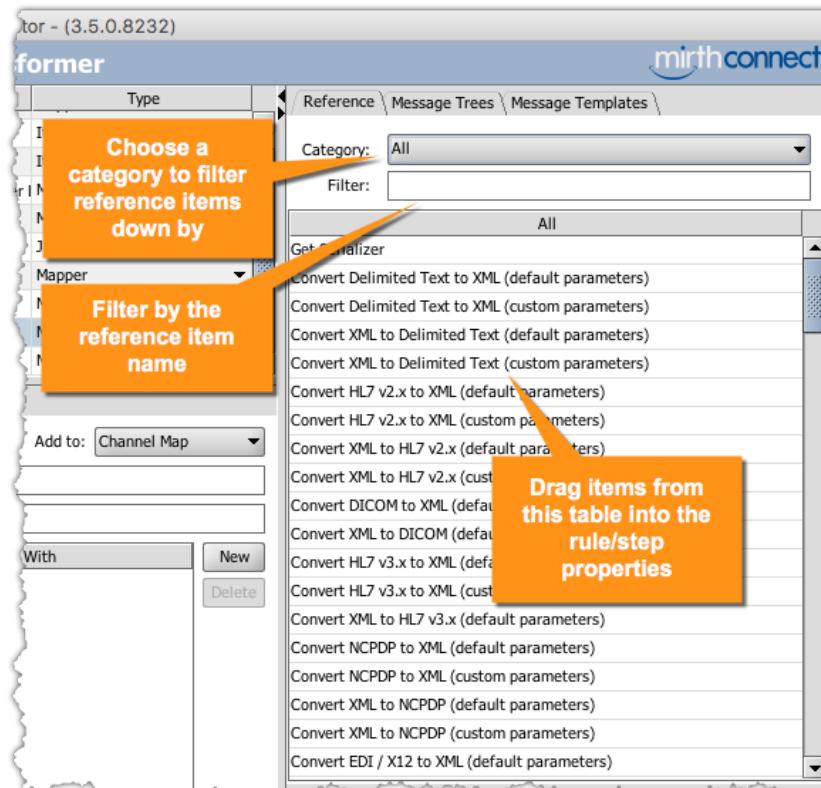
This is located on the right-hand side of the [Edit Filter / Transformer View](#) and provides helpful variables / templates you can use within your filter rules / transformer steps.



- [Reference List](#)
- [Available Variables](#)

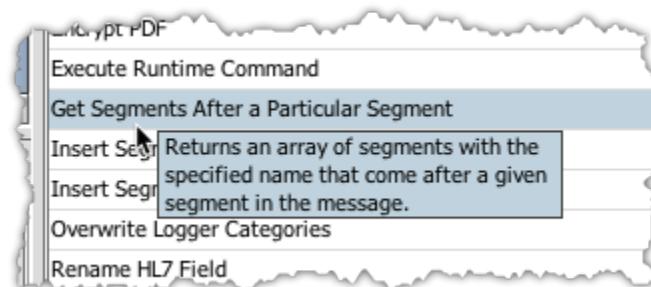
Reference List

The Reference List contains code templates (both built-in and custom) that you can drag into your filter rules / transformer steps. These include common operations like serializing to/from XML, performing a manual database query, or generating a unique ID. For additional information on code templates, see [Code Template Libraries](#).

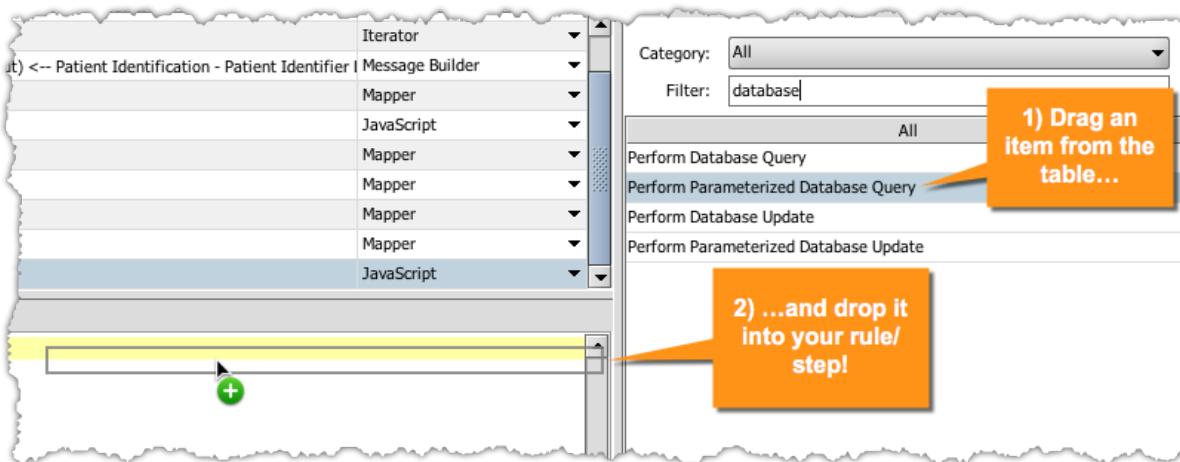


Use the **Category** and **Filter** components at the top to quickly find the template you're looking for. For custom code templates, choose the **User Defined Functions** or **User Defined Code** options.

Hover your mouse over an item in the list to view its description:



To use an item, simply drag it from the table into your rule/step properties window:



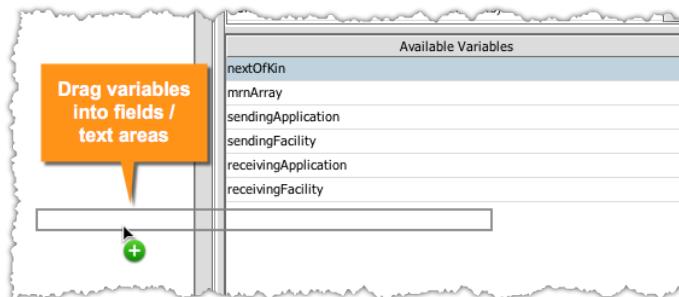
The corresponding code template / function call will automatically be pasted into the field / text area:

```
5 // ...
6 // ...
7 Step \ Generated Script
8
9
10
11 var dbConn;
12 var result;
13
14 try {
15     dbConn = DatabaseConnectionFactory.createDatabaseConnection('driver', 'address', 'username', 'password');
16     result = dbConn.executeCachedQuery('expression', paramList);
17 } finally {
18     if (dbConn) {
19         dbConn.close();
20     }
21 }
```

The screenshot shows the generated JavaScript code in the 'Step \ Generated Script' panel. The code is a try-finally block that creates a database connection, executes a query, and then closes the connection. The code is syntax-highlighted with colors for different language elements.

Available Variables

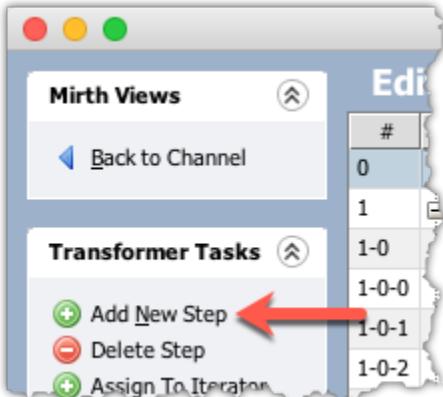
This is a convenience list shown in the [Edit Transformer View](#) when the [Reference Tab](#) is selected. It's automatically populated with the variables set in previous Mapper steps, allowing you to easily drag them into subsequent steps.



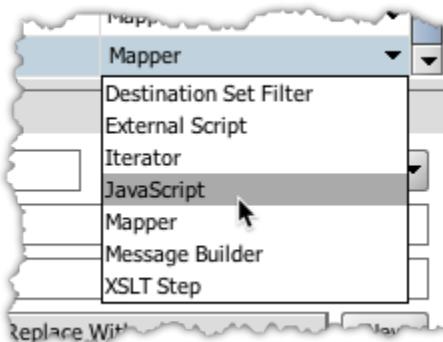
Creating New Rules / Steps

As outlined in the [Message Trees Tab](#) section, new rules / steps can be generated easily from the inbound/outbound message trees. They can also be created manually as well however.

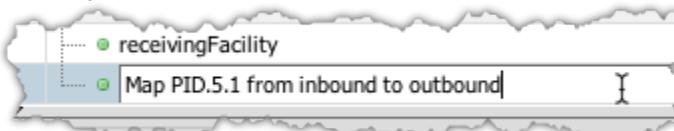
- Click the **Add New Rule/Step** task to the left:



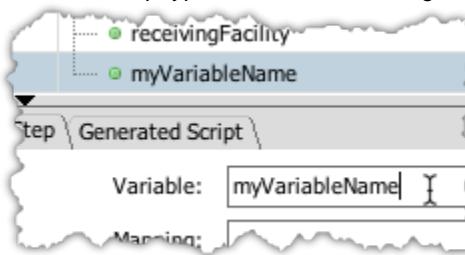
- A new rule/step will be created and added to the table. To change the type of rule/step, double-click the cell in the Type column:



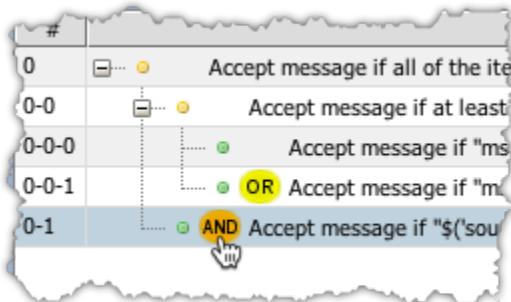
- To change the name, double-click the cell in the Name column:



- Some rule/step types have names auto-generated from the properties and so don't allow editing in the table:



- If you're creating a filter rule, you can change the **operator** of the rule by clicking the **AND/OR** button next to the name:



Rule / Step Table

The table in the top half of the [Edit Filter / Transformer Views](#) shows all the rules / steps you currently have configured. If you have [Iterators](#), the elements will be organized into a tree-table with the children underneath each Iterator.

Enabled	#	Name	Type
<input checked="" type="checkbox"/>	0	Accept message if at least one of the iterations returns true for each msg['PID']	Iterator
<input checked="" type="checkbox"/>	0-0	Accept message if at least one of the iterations returns true for each msg['PID'][i]['PID.3']	Iterator
<input checked="" type="checkbox"/>	0-0-0	Accept message if "msg['PID'][i]['PID.3'][j]['PID.3.1'].toString()" equals '1' or '4'	Rule Builder
<input checked="" type="checkbox"/>	0-0-1	OR Accept message if "msg['PID'][i]['PID.3'][j]['PID.3.2'].toString()" equals 'dummy'	Rule Builder
<input checked="" type="checkbox"/>	0-1	AND Accept message if "\${'sourceId'}" exists	Rule Builder
<input checked="" type="checkbox"/>	0-2	AND New Rule	JavaScript

Column	Description
Enabled	Determines whether the rule/step is currently enabled. When disabled, rules/steps are not executed at all, and act as if they were never there in the first place. For filters, this could change behavior as it follows the AND/OR order of operations. When all rules in a filter are disabled, the filter acts as if no rules exist, and is not executed at all.
#	This is the sequence number of the rule / step, starting at 0 for the first one. For child elements underneath an Iterator, this will be multiple numbers separated by a dash, to indicate the child index of each depth level.
Name	This is the name of the rule / step. If the type of rule / step allows user editing, you can double-click the cell to edit the name. For filter rules, you can also click on the AND / OR icon next to the name to change the operator.
Type	The type of rule / step. Double-click this cell to change to a different type.

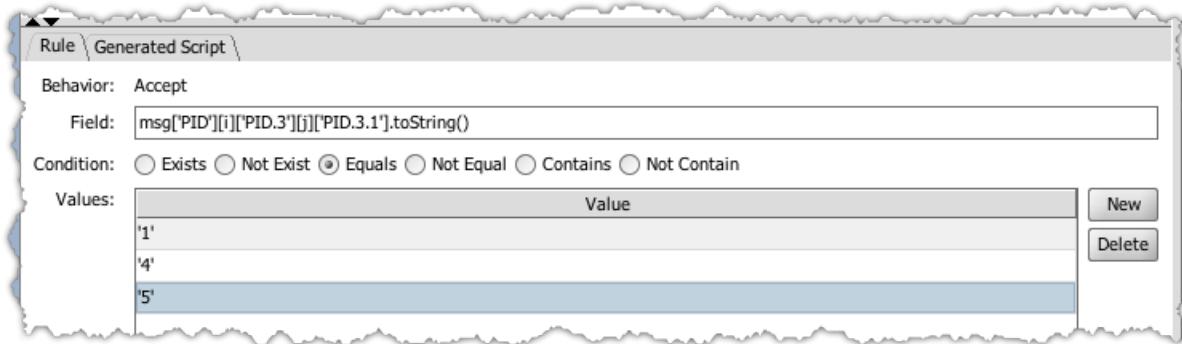
Filter Rule Properties

The following Filter Rules are supported by NextGen Connect Integration Engine:

- Rule Builder Filter Rule
- JavaScript Filter Rule
- External Script Filter Rule
- Iterator Filter Rule

Rule Builder Filter Rule

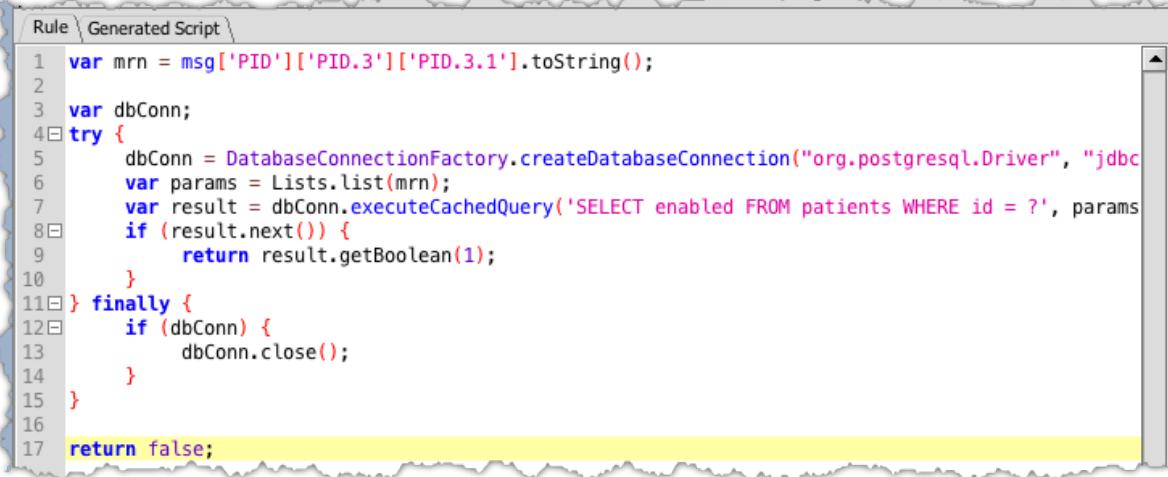
This rule allows you to build simple accept logic for a specific message field or expression.



Item Name	Description
Behavior	This is always set to "Accept", meaning that if the logical expression below evaluates to true , the message will be accepted.
Field	The message field or expression to test.
Condition	Determines how to test the Field set above. The following conditions are supported: <ul style="list-style-type: none"> Exists: Returns true if the length of the field is greater than 0. Not Exist: Returns true if the length of the field is 0. Equals: If the Values table is empty, returns true if the field is equal to an empty string. If the Values table is not empty, returns true if the field matches any of the values in the Values table below. Not Equal: If the Values table is empty, returns true if the field is not equal to an empty string. If the Values table is not empty, returns true if the field matches none of the values in the Values table below. Contains: Returns true if the field contains any of the values in the Values table below. Not Contain: Returns true if the field contains none of the values in the Values table below.
Values	A table of expressions that may be used in conjunction with the Condition to test the given field and decide whether or not to filter the message.

JavaScript Filter Rule

This rule allows you to write a completely custom script to decide whether to filter the message or not. For more information about using JavaScript, see [Using JavaScript in Mirth Connect](#).



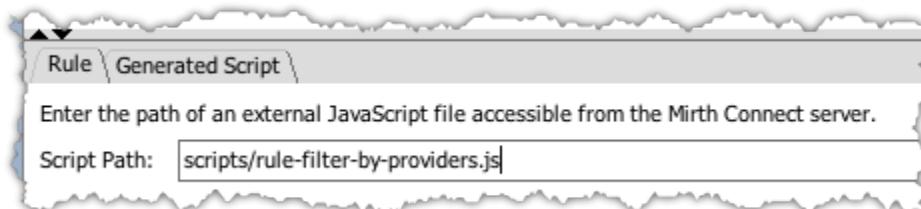
The screenshot shows a code editor window titled "Rule \ Generated Script \". The script content is as follows:

```
1 var mrn = msg['PID']['PID.3']['PID.3.1'].toString();
2
3 var dbConn;
4 try {
5     dbConn = DatabaseConnectionFactory.createDatabaseConnection("org.postgresql.Driver", "jdbc
6     var params = Lists.list(mrn);
7     var result = dbConn.executeCachedQuery('SELECT enabled FROM patients WHERE id = ?', params
8     if (result.next()) {
9         return result.getBoolean(1);
10    }
11} finally {
12    if (dbConn) {
13        dbConn.close();
14    }
15}
16
17 return false;
```

The code uses the Mirth Connect API to query a PostgreSQL database for a patient's enabled status based on their ID. If the patient is enabled, it returns true; otherwise, it returns false.

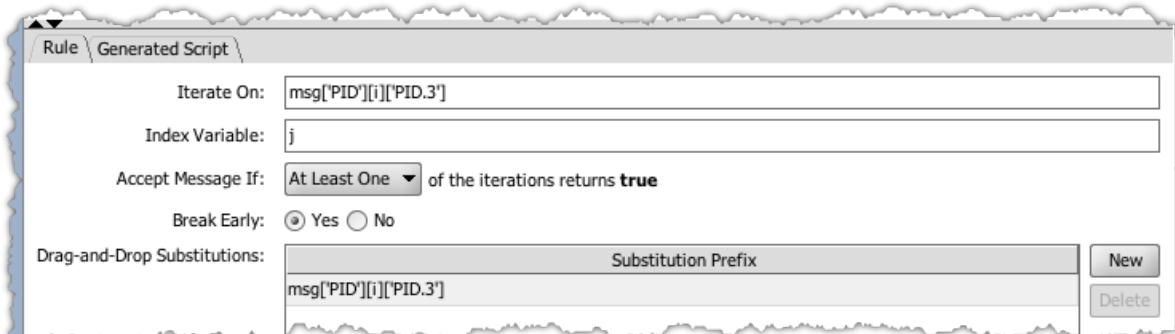
External Script Filter Rule

This rule functions the same way as the [JavaScript Filter Rule](#), except that the script is read from an external file when the channel is deployed. If the given path is not absolute, it will be relative to the NextGen Connect Integration Engine installation directory.



Iterator Filter Rule

This is a special type of rule that allows you to decide whether to filter a message or not by iterating through an array or list of XML nodes. The child rules underneath the Iterator determine the accept/filter behavior of the overall rule. For additional information, see [Working With Iterators](#).



Item Name	Description
Iterate On	The element to iterate on. This may be a list of E4X XML nodes, or a Java / JavaScript array.
Index Variable	The index variable to use for each iteration.
Accept Message If	Determines how to logically combine each iteration into the overall accept / filter behavior. <ul style="list-style-type: none"> At Least One: If the logical combination of the child rules returns true for <i>at least one</i> of the iterations, the overall Iterator behavior will be to accept the message. All: If the logical combination of the child rules returns true for <i>all</i> of the iterations, the overall Iterator behavior will be to accept the message.
Break Early	If this is enabled, the iterator loop will terminate as quickly as possible. For example if "At Least One" is chosen above, the loop will terminate as soon as the first iteration returns true .
Drag-and-Drop Substitutions	When drag-and-dropping values into the children underneath this Iterator, the index variable (e.g. "[i]") will be injected after any of these prefixes. For example if your index variable is i and you have msg['PID'] in the Drag-and-Drop Substitutions table, when you drag the value msg['PID']['PID.3']['PID.3.1'].toString() from the Message Trees Tab into a child rule, it will automatically be replaced with msg['PID'][i]['PID.3']['PID.3.1'].toString() .

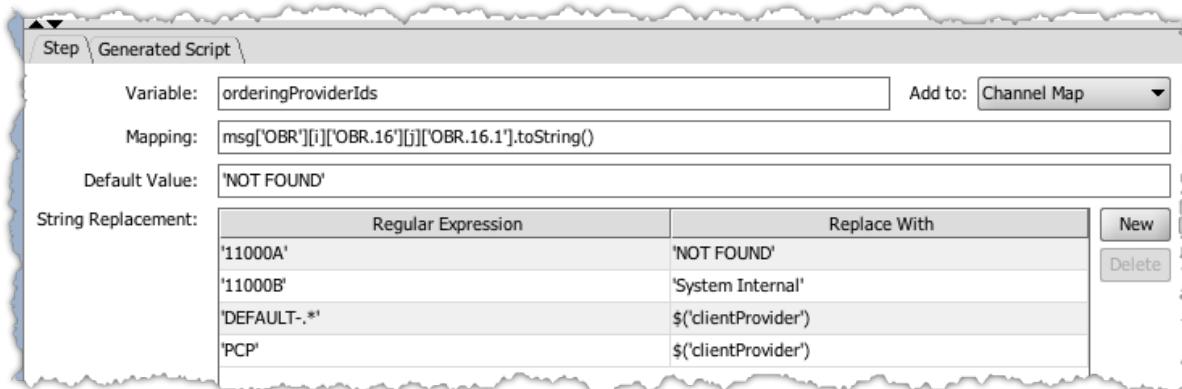
Transformer Step Properties

The following Transformer Steps are supported by NextGen Connect Integration Engine:

- [Mapper Transformer Step](#)
- [Message Builder Transformer Step](#)
- [JavaScript Transformer Step](#)
- [External Script Transformer Step](#)
- [XSLT Transformer Step](#)
- [Destination Set Filter Transformer Step](#)
- [Iterator Transformer Step](#)

Mapper Transformer Step

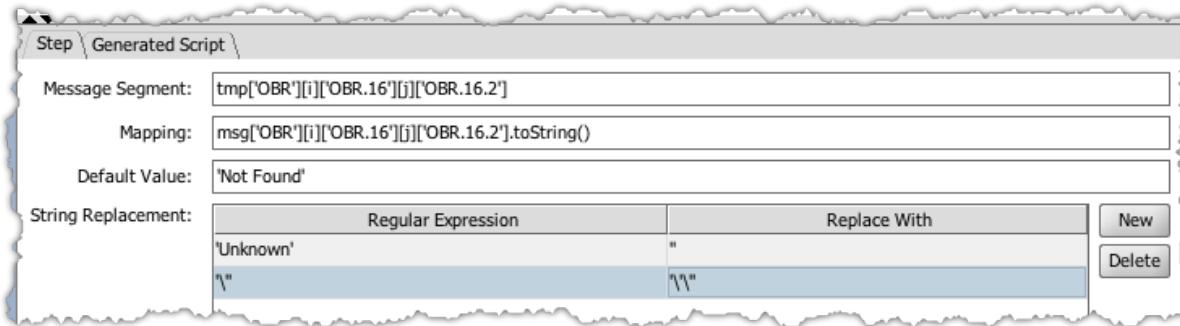
This step extracts data from a field in the message (or an expression) and places it into one of the available [Variable Maps](#). Depending on the scope of the map, this variable will be available in subsequent steps, in the destination connector properties, or even in subsequent connectors.



Item Name	Description
Variable	The variable name / key to use when inserting into the map. The Add to drop-down menu to the right determines which map to place the variable in. For additional information, see Variable Maps .
Mapping	The value to place into the map. This may be a field from the message, or any JavaScript expression.
Default Value	If the Mapping is not found or evaluates to an empty string, this value / expression will be used instead.
String Replacement	This table allows you to perform replacements on the value before it gets inserted into the map. <ul style="list-style-type: none"> Regular Expression: A Java-style regular expression to test against the value. This will implicitly set the global regex flag. Replace With: The value to replace any matched regions with.

Message Builder Transformer Step

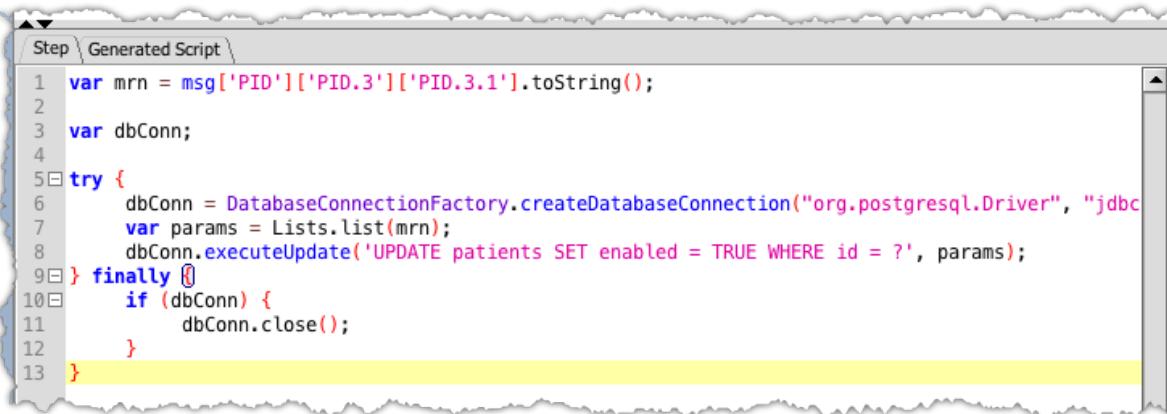
This step extracts data from a field in the message (or an expression) and maps it into a specific field in the inbound or outbound message. This can be used to simply modify a field in the inbound message, copy a field from one place to another, or map data from the inbound message to the outbound message.



Item Name	Description
Message Segment	The field/location in the inbound or outbound message to place the value into.
Mapping	The value to place into the given message segment. This may be a field from the message, or any JavaScript expression.
Default Value	If the Mapping is not found or evaluates to an empty string, this value / expression will be used instead.
String Replacement	<p>This table allows you to perform replacements on the value before it gets inserted.</p> <ul style="list-style-type: none"> Regular Expression: A Java-style regular expression to test against the value. This will implicitly set the global regex flag. Replace With: The value to replace any matched regions with.

JavaScript Transformer Step

This step allows you to write a completely custom script to extract / transform data, or to perform almost any intermediate action you need to. For additional information about using JavaScript, see [Using JavaScript in Mirth Connect](#).

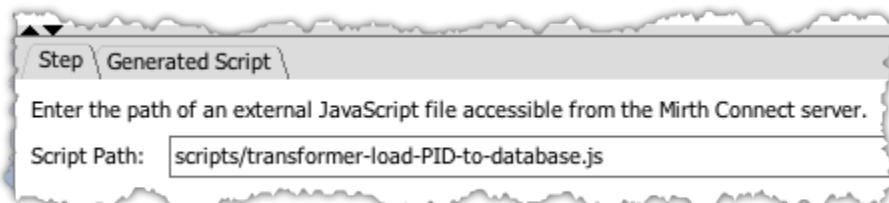


The screenshot shows a code editor window titled "Step \ Generated Script". The script content is as follows:

```
1 var mrn = msg['PID']['PID.3']['PID.3.1'].toString();
2
3 var dbConn;
4
5 try {
6     dbConn = DatabaseConnectionFactory.createDatabaseConnection("org.postgresql.Driver", "jdbc
7         var params = Lists.list(mrn);
8         dbConn.executeUpdate('UPDATE patients SET enabled = TRUE WHERE id = ?', params);
9     } finally {
10        if (dbConn) {
11            dbConn.close();
12        }
13    }
```

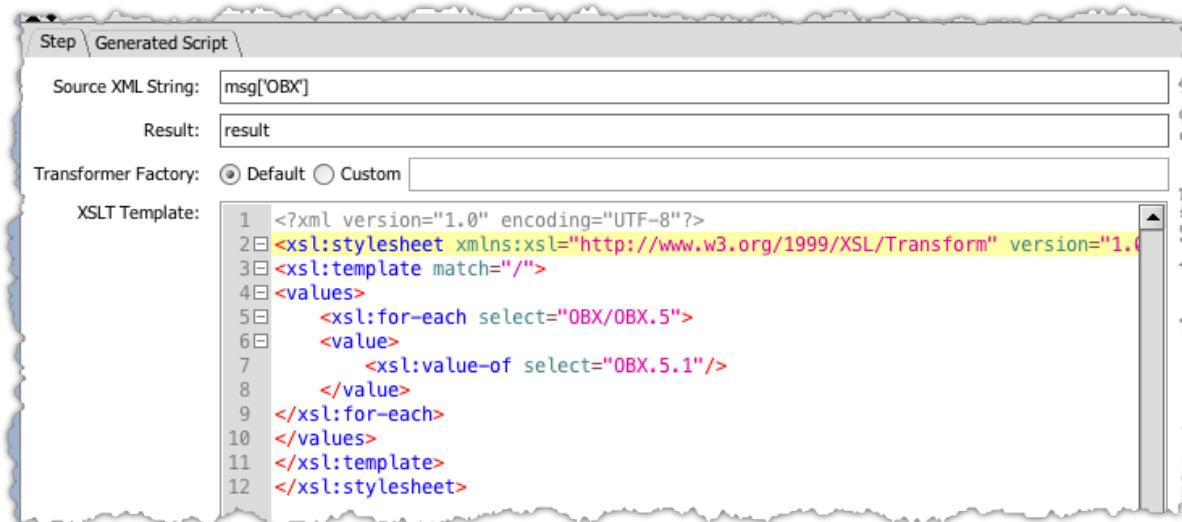
External Script Transformer Step

This step functions the same way as the [JavaScript Transformer Step](#), except that the script is read from an external file when the channel is deployed. If the given path is not absolute, it will be relative to the NextGen Connect Integration Engine installation directory.



XSLT Transformer Step

This step allows you to apply an XSLT (eXtensible Stylesheet Language Transformations) stylesheet to a given XML document. This may be msg/tmp (the internal XML representation of your message data), or some other variable containing an XML string. The result of the transformation will be stored in the channel map.

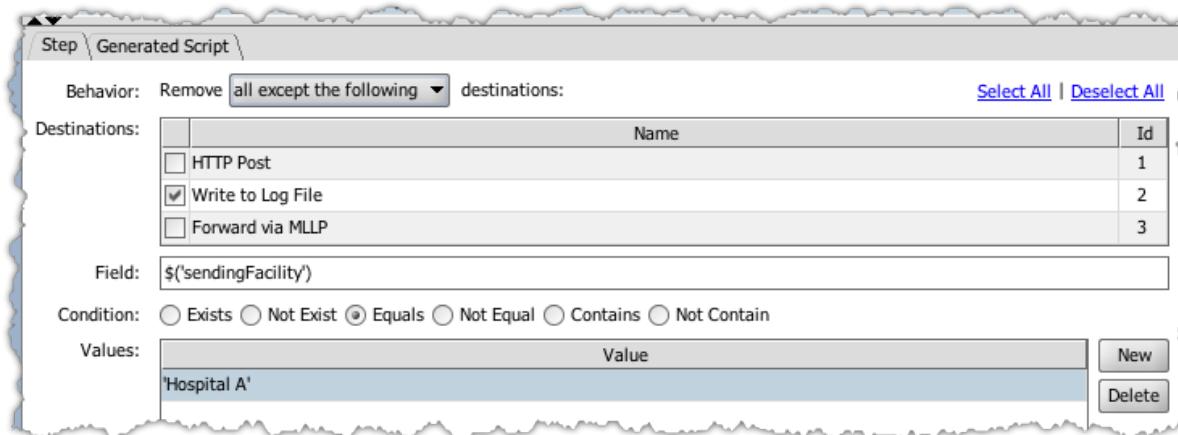


Item Name	Description
Source XML String	The XML string to transform.
Result	The key to use when storing the result into the channel map.
Transformer Factory	Select default to use the Java platform default TransformerFactory implementation class. Select custom to provide a custom TransformerFactory implementation class.
XSLT Template	The XSLT stylesheet to use to transform the source XML string.

Destination Set Filter Transformer Step

Destination Set Filtering is a powerful feature of the source transformer that allows you to decide in advance which destinations to exclude from message processing. Using each individual destination's filter to control where a message goes is still a valid workflow, but when you have many destinations all with mutually exclusive filters, the performance of the channel can be affected because message data will be stored to the database for each destination connector. Also filtered connector messages can clutter up the [message browser](#), making it harder to find what you're looking for. The advantage to using Destination Set Filtering in this case is that filtered destinations will not have any message data stored, and will not show up in the message browser. This can greatly increase message throughput.

Destination Set Filtering can be done manually with JavaScript (look at DestinationSet in the [User API](#)). However this step allows easier access to the feature without having to write any JavaScript.

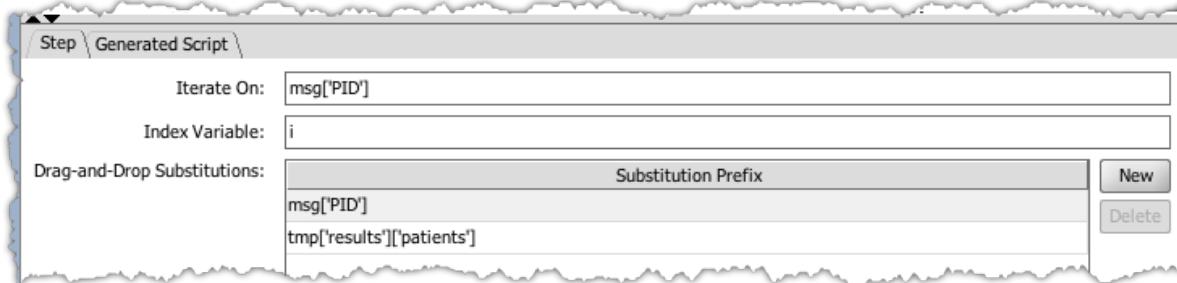


Item Name	Description
Behavior	Determines which destinations will be removed from the destination set (and so will not be processed). <ul style="list-style-type: none"> the following: The selected destinations will be filtered if the condition below evaluates to true. all except the following: All destinations (even new ones created after this step was created) except the selected ones will be filtered if the condition below evaluates to true. all: All destinations (even new ones created after this step was created) will be filtered if the condition below evaluates to true.
Destinations	Select the destinations to exclude or <i>not</i> /exclude, depending on the behavior above. Even if the destination is renamed later, these selections will still be correct since the metadata ID is used.
Field	The message field or expression to test.
Condition	Determines how to test the Field set above. The following conditions are supported: <ul style="list-style-type: none"> Exists: Returns true if the length of the field is greater than 0. Not Exist: Returns true if the length of the field is 0. Equals: If the Values table is empty, returns true if the field is equal to an empty string. If the Values table is not empty, returns true if the field matches any of the values in the Values table below.

	<ul style="list-style-type: none">• Not Equal: If the Values table is empty, returns true if the field is not equal to an empty string. If the Values table is not empty, returns true if the field matches none of the values in the Values table below.• Contains: Returns true if the field contains any of the values in the Values table below.• Not Contain: Returns true if the field contains none of the values in the Values table below.
Values	A table of expressions that may be used in conjunction with the Condition to test the given field and decide whether or not to filter the selected destinations.

Iterator Transformer Step

This is a special type of step that allows you to perform extract / transform operations while iterating through an array or list of XML nodes. For additional information, see [Working With Iterators](#).



Item Name	Description
Iterate On	The element to iterate on. This may be a list of E4X XML nodes, or a Java / JavaScript array.
Index Variable	The index variable to use for each iteration.
Drag-and-Drop Substitutions	When drag-and-dropping values into the children underneath this Iterator, the index variable (e.g. "[i]") will be injected after any of these prefixes. For example if your index variable is i and you have msg["PID"] in the Drag-and-Drop Substitutions table, when you drag the value msg['PID']['PID.3']['PID.3.1'].toString() from the Message Trees Tab into a child step, it will automatically be replaced with msg['PID'][i]['PID.3']['PID.3.1'].toString() .

Response Transformers

The response transformer is a special type of transformer only editable for destination connectors on the [Destinations Tab](#). It works the same as a regular transformer, except that the data being transformed is not the message flowing through the channel, but instead the response payload that the destination connector received from the external system (if applicable). For additional information about transformers, see [About Transformers](#).

A destination response is comprised not only of the response data, but also the **status** (e.g. SENT, ERROR), **status message**, and **error message**. Response transformers can be used to modify these latter pieces as well. For example if a message gets set to ERROR by the destination connector, in the response transformer you can choose to override that and set the status to SENT instead based on some custom logic.

- i** Response transformers will only execute if there's an actual response payload to transform. For example if you're using an [HTTP Sender](#) destination and it fails to connect to the remote server, then obviously there is no response payload. The one exception to this rule is if the response inbound data type is set to **Raw**. In that case, because the Raw data type doesn't need to perform any serialization, the response transformer will always execute even if there is no response payload.

Modifying the Response

Modifying the actual [response data](#) is done by using the normal features and steps available to a transformer. The internal representation of the response data is **msg**, while the internal representation of the outbound template (if set) is **tmp**. When the response transformer finishes processing, it will use the value of **tmp** (or **msg** if no outbound template is set) to create the Processed Response content.

There are three other pieces of the response that you can modify in the response transformer:

- **responseStatus**: This is the status that will be used to update the message after the response message finishes. You may set the status to SENT, QUEUED, or ERROR. If the status is set to QUEUED and queuing is not enabled for the destination connector, it will automatically be changed to ERROR.
- **responseStatusMessage**: This is a brief one-line message that displays alongside the status in the message browser. It's typically used to give a reason for the status.
- **responseErrorMessage**: This is the full error message associated with a response. Typically this is used to display large stacktrace messages.

In addition to the above variables, you have access to **response**, which is an `ImmutableResponse` object. For additional information, see the [User API](#).

Common Scenarios

Re-queue a message if the HL7 ACK has an AE code

```
if (msg['MSA']['MSA.1']['MSA.1.1'].toString() == 'AE') {  
    responseStatus = QUEUED;  
    responseStatusMessage = 'Application Error NACK received.';  
    responseErrorMessage = msg['MSA']['MSA.3']['MSA.3.1'].toString();  
}
```

Force a queuing message to error if the number of send attempts exceeds some threshold

```
if (responseStatus == QUEUED && connectorMessage.getSendAttempts() >= 5) {  
    responseStatus = ERROR;  
    responseStatusMessage = 'Maximum send attempts exceeded.';  
}
```

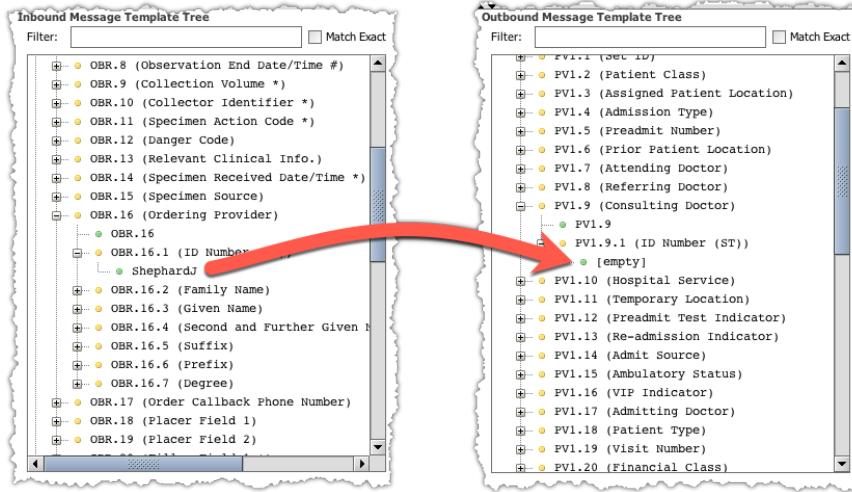
Route the response data to a downstream channel

```
if (responseStatus == SENT) {  
    router.routeMessageByChannelId('channel ID here', response.getMessage());  
}
```

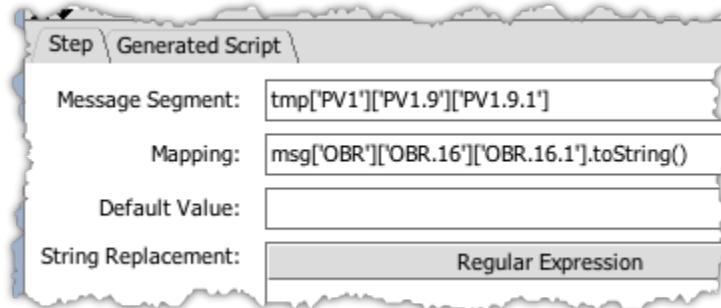
Working With Iterators

An Iterator is a special type of step that allows you to loop (iterate) through an array or list of XML nodes. For each array element or XML node (each "iteration"), you can execute multiple filter rules or transformer steps (the "children").

For example, let's say you are mapping inbound HL7 v2.x messages to an outbound HL7 v2.x template, and you want to copy **OBR.16.1** (ordering provider) component to a the **PV1.9.1** (consulting doctor) component in the outbound template.



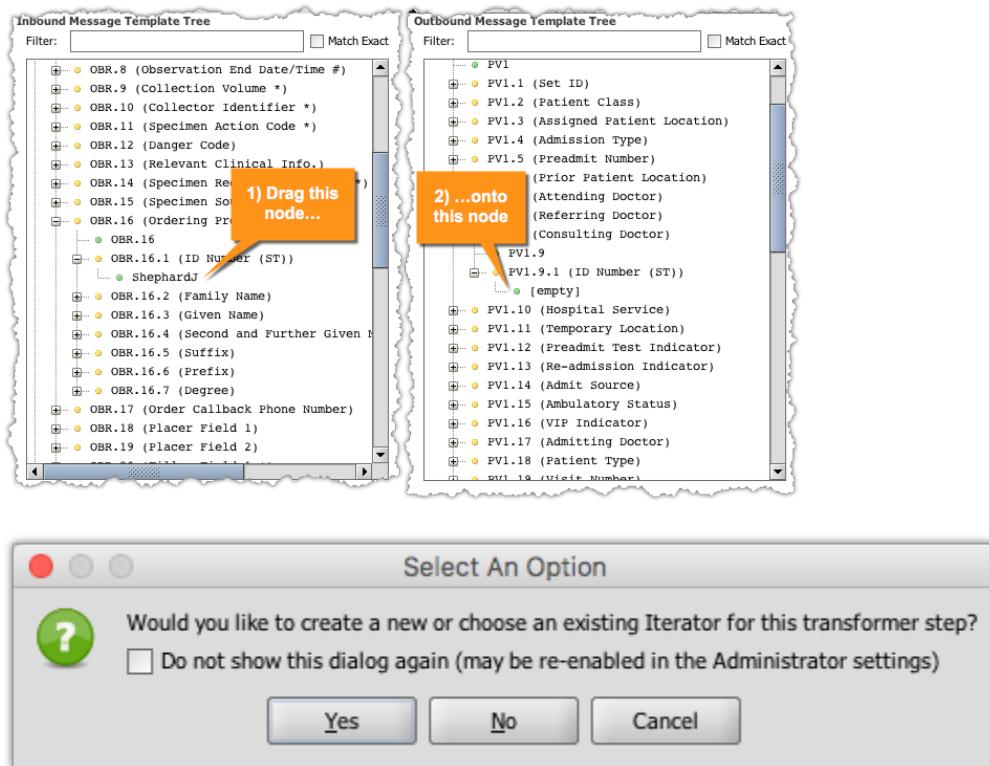
Typically you would do this with a [Message Builder Transformer Step](#):



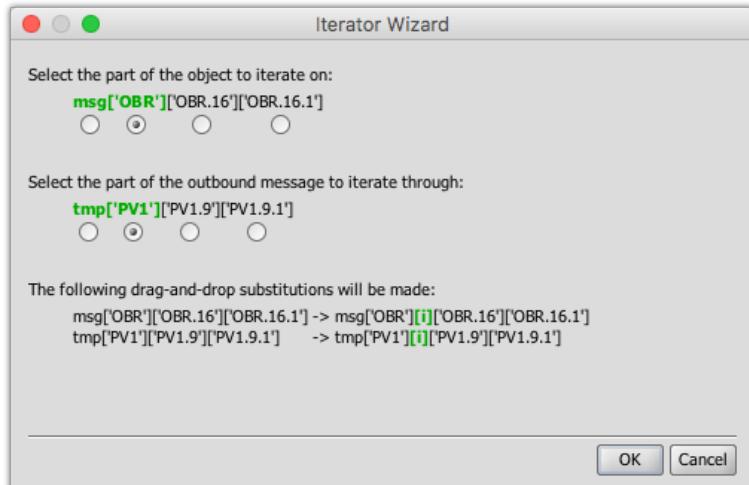
This works so far when there's only *one* OBR segment and *one* OBR.16 field. But what if you want to handle multiple segments or repeating fields? This is where Iterators come into play.

Creating Iterators From Drag-and-Drop

As explained in the [Message Trees Tab](#) section, new rules and steps can be created by right-clicking the node in the message tree, by dragging a node into the filter/transformer table, or by dragging a node from the inbound tree and dropping it onto a node in the outbound tree. In all of these cases, you will be presented with a prompt asking whether you want to create an Iterator automatically.

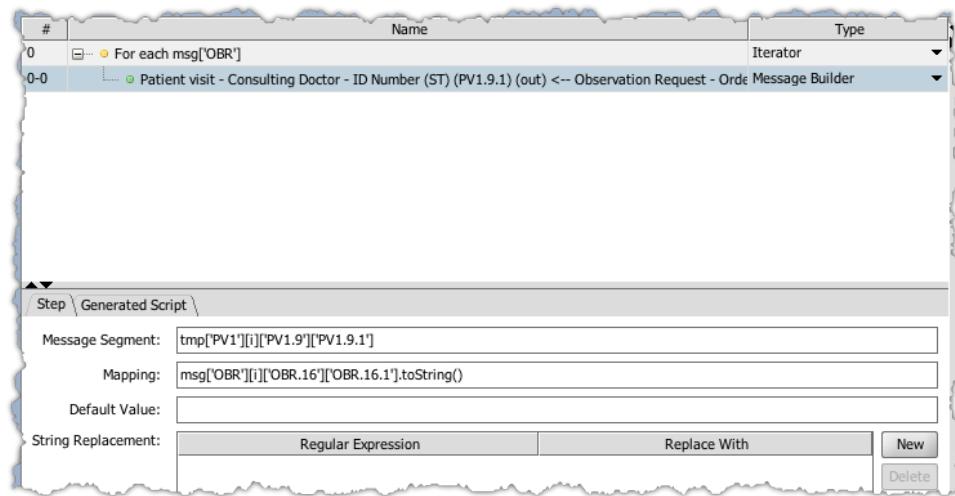


If **Yes** is chosen, the **Iterator Wizard** dialog will be shown:



In this wizard dialog you can select your iteration target (what to iterate on). If you've dragged from inbound to the outbound template, you will also have the option to select which part of the outbound (tmp) expression corresponds to the inbound (msg) target being iterated on. These options correspond directly to the drag-and-drop substitutions shown at the bottom of the dialog. You're essentially telling the wizard "where the /goes".

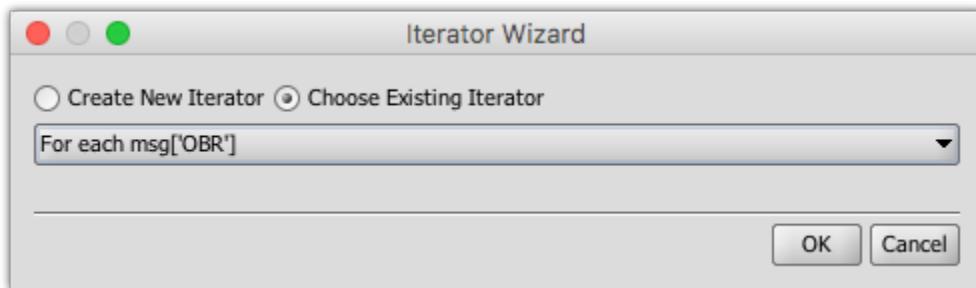
Once you click **OK**, the Iterator and subsequent rule/step will be created:



So far it's only iterating at one level though. In the example for this section, we wanted to iterate not only through each OBR segment, but also through each OBR.16 field repetition. So we want to take the current selected step and **assign** it to an additional, nested Iterator.

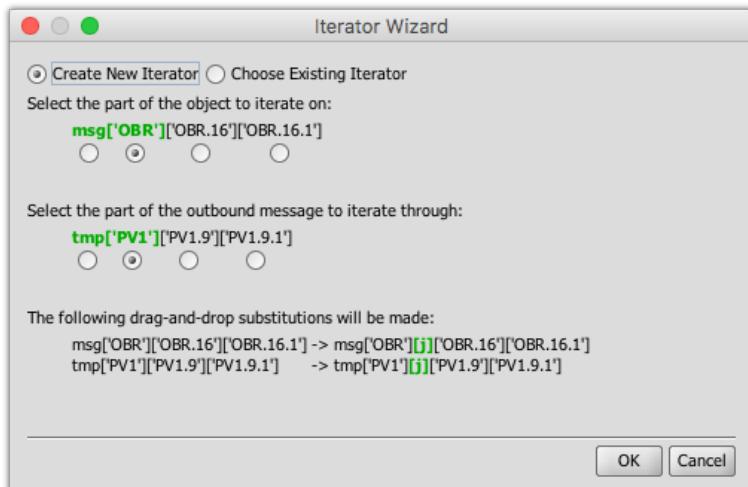
The Assign To Iterator Task

This task takes the currently selected rule/step, and either moves it to an existing Iterator, or creates an entirely new Iterator and puts the rule/step within it. When clicking on the task a dialog is shown:

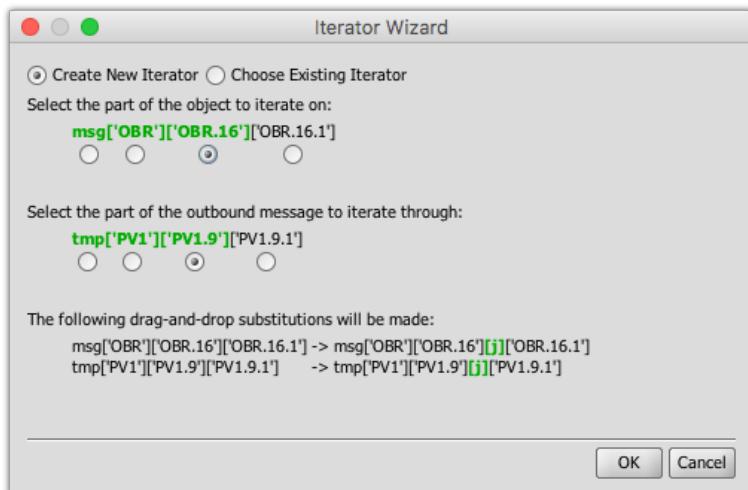


The **Choose Existing Iterator** option allows you to take the currently selected rule/step and move it underneath a specific pre-existing Iterator. If your rule/step already belongs to an Iterator, by default this option will be selected and the current parent will be selected in the drop-down menu.

The **Create New Iterator** option is the same dialog shown before, where you have the option of choosing what to iterate on:



Note that in this case, the variable *j* was automatically chosen, because the wizard detected that the currently selected step is part of a parent Iterator that is already using the *j*/index variable. We can then choose to iterate through each OBR.16 field instead of on each OBR segment:



After clicking **OK**, the step will now be placed inside a new nested Iterator. You can see from the below screenshot that OBR.16.1 is being mapped into PV1.9.1, but now it's being done for each OBR segment, and in turn **for each** OBR.16 field repetition.

The screenshot shows a hierarchical rule tree and a generated script panel. The tree has three levels:

- Level 0: For each msg['OBR'] (Iterator)
- Level 0-0: For each msg['OBR'][i]['OBR.16'] (Iterator)
- Level 0-0-0: Consulting Doctor - (PV1.9.1) (out) <-- Ordering Provider - (OBR.16.1) (in) (Message Builder)

The generated script panel at the bottom contains the following fields:

- Message Segment: tmp['PV1'][i]['PV1.9'][j]['PV1.9.1']
- Mapping: msg['OBR'][i]['OBR.16'][j]['OBR.16.1'].toString()
- Default Value: (empty)
- String Replacement: (New button)

Removing From Iterators

If you decide later that a rule or step shouldn't be part of an Iterator, it's easy to undo those changes. Select the rule or step, and click the **Remove From Iterator** task. The rule/step will be moved one level higher in the tree. So if the rule/step is currently nested under multiple Iterators, click the task multiple times until it's at the depth you want.

Viewing Generated Script

For all rule / step types, the properties panel shown in the bottom half of the screen is actually split up into two tabs: **Rule/Step**, and **Generated Script**. Clicking on the Generated Script tab shows the equivalent JavaScript that will execute when your channel is deployed and a message is sent through:

#	Name	Type
0	nextOfKin	Mapper
1	For each msg['PID']	Iterator
1-0	For each msg['PID'][i]['PID.3']	Iterator
1-0-0	_mrnArray	Mapper
2	sendingApplication	Mapper
3	sendingFacility	Mapper
4	receivingApplication	Mapper
5	receivingFacility	Mapper


```

Step } Generated Script }
1 var _mrnArray = Lists.list();
2
3 for (var i = 0; i < getArrayOrXmlLength(msg['PID']); i++) {
4
5   for (var j = 0; j < getArrayOrXmlLength(msg['PID'][i]['PID.3']); j++) {
6
7     var mapping;
8
9     try {
10       mapping = msg['PID'][i]['PID.3'][j]['PID.3.1'].toString();
11     } catch (e) {
12       logger.error(e);
13       mapping = '';
14     }
15
16     _mrnArray.add(validate(mapping, '', new Array()));
17   }
18 }
19
20
21 }
22 channelMap.put('mrnArray', _mrnArray.toArray());

```

The script pane is not editable, but you can still select code and expand/collapse code folds. As shown in the screenshot above, selecting an Iterator rule/step will show you the script for the Iterator and all of its children at once.



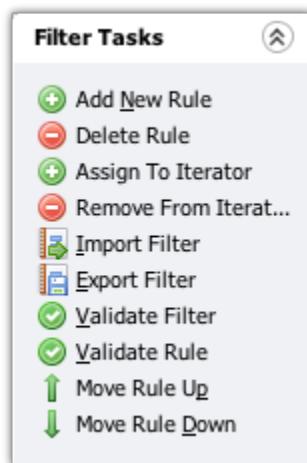
External Script rules/steps are an exception and will *not* show the actual script that resides on the server:

```

6 Load Providers External Script
Step } Generated Script
1 // External script will be loaded on deploy
2 // Path: /folders/scripts/load-providers.js

```

Filter Tasks



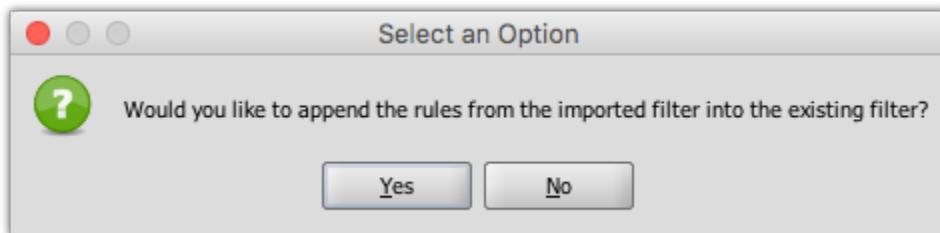
The following context-specific tasks are available throughout the [Edit Filter View](#):

Task Icon	Task Name	Description
	Add New Rule	Adds a new filter rule to the table. If an Iterator rule or any rule that is a child of an Iterator is currently selected, the new rule will be placed at the end of the children of the most immediate parent Iterator. Otherwise, the new rule will be placed at the very end of the list at the bottom of the table.
	Delete Rule	Removes the currently selected rule from the table. If an Iterator rule is deleted, all of its children will be deleted as well.
	Assign To Iterator	Adds the selected rule to a new or existing Iterator. For additional information, see Working With Iterators .
	Remove From Iterator	Removes the selected rule from its current Iterator. For additional information, see Working With Iterators .
	Import Filter	Imports a filter from an XML file. You can choose to completely replace the current filter, or simply append the rules to the current table.
	Export Filter	Exports the current filter (all rules) to an XML file.
	Validate Filter	Validates the entire filter and all rules. This includes property validation and script syntax validation.
	Validate Rule	Validates the currently selected rule. This includes property validation and script syntax validation.
	Move Rule Up	Moves the currently selected rule one slot higher in the table. If the rule is inside of an Iterator and is currently the first rule in the Iterator's children,

		this task will move the rule up and out of the Iterator, similar to the Remove From Iterator task. For additional information, see Working With Iterators .
 Move Rule Down	Move Rule Down	Moves the currently selected rule one slot lower in the table. If the rule is inside of an Iterator and is currently the last rule in the Iterator's children, this task will move the rule down and out of the Iterator, similar to the Remove From Iterator task. For additional information, see Working With Iterators .

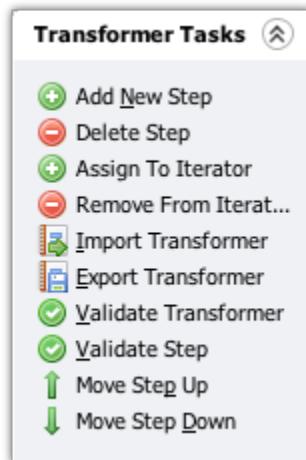
Import Filter

When this task is clicked, you'll be presented with a prompt:



If **Yes** is chosen, the rules from the filter will be added at the end of the current list at the bottom of the table. If **No** is chosen, all rules currently in the table will be deleted and replaced with the rules from the imported filter.

Transformer Tasks



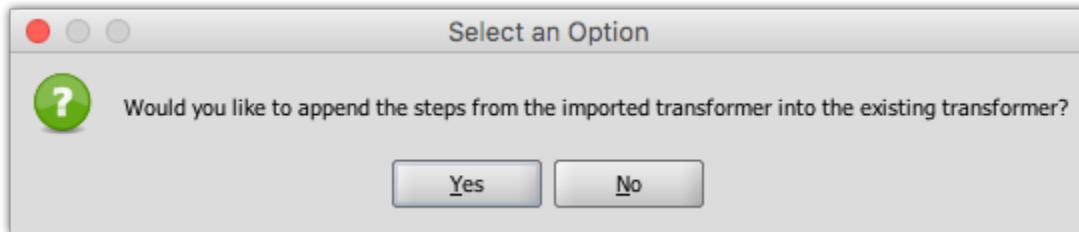
The following context-specific tasks are available throughout the [Edit Transformer View](#):

Task Icon	Task Name	Description
	Add New Step	Adds a new transformer step to the table. If an Iterator step or any step that is a child of an Iterator is currently selected, the new step will be placed at the end of the children of the most immediate parent Iterator. Otherwise, the new step will be placed at the very end of the list at the bottom of the table.
	Delete Step	Removes the currently selected step from the table. If an Iterator step is deleted, all of its children will be deleted as well.
	Assign To Iterator	Adds the selected step to a new or existing Iterator. For additional information, see Working With Iterators .
	Remove From Iterator	Removes the selected step from its current Iterator. For additional information: Working With Iterators
	Import Transformer	Imports a transformer from an XML file. You can choose to completely replace the current transformer, or simply append the steps to the current table.
	Export Transformer	Exports the current transformer (inbound/outbound data types and all steps) to an XML file.
	Validate Transformer	Validates the entire transformer and all steps. This includes property validation and script syntax validation.
	Validate Step	Validates the currently selected step. This includes property validation and script syntax validation.
	Move Step Up	Moves the currently selected step one slot higher in the table. If the step is inside of an Iterator and is currently the first step in the Iterator's children, this task will move the step up and out of the

		Iterator, similar to the Remove From Iterator task. For additional information, see Working With Iterators .
 Move Step Down	Move Step Down	Moves the currently selected step one slot lower in the table. If the step is inside of an Iterator and is currently the last step in the Iterator's children, this task will move the step down and out of the Iterator, similar to the Remove From Iterator task. For additional information, see Working With Iterators .

Import Transformer

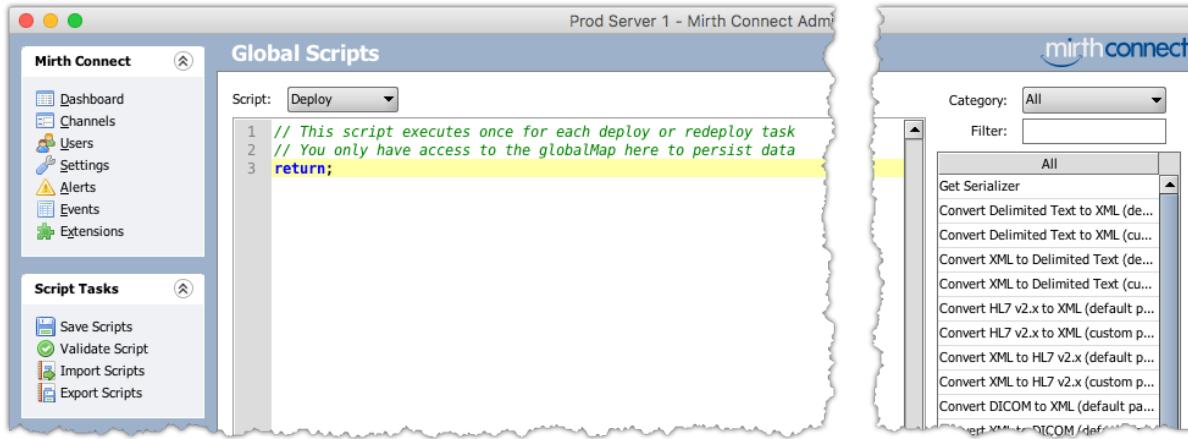
When this task is clicked, you'll be presented with a prompt:



If **Yes** is chosen, the steps from the filter will be added at the end of the current list at the bottom of the table. If **No** is chosen, all steps currently in the table will be replaced with the steps from the imported filter, and the inbound / outbound data type / template settings will be overwritten.

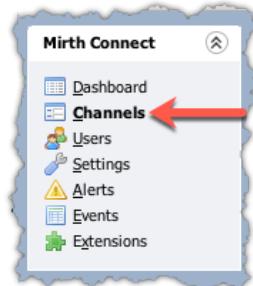
Edit Global Scripts View

This view is similar to the [Scripts Tab](#) in the [Edit Channel View](#), except that the scripts here are **global** across all channels.

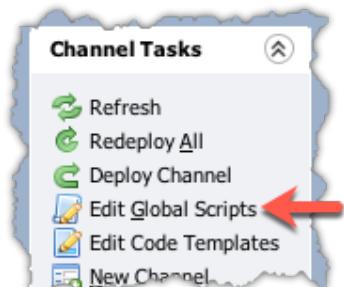


Navigation

Click the [Channels](#) link in the Mirth Connect task pane at the top-left to enter the [Channels View](#):



Click the **Edit Global Scripts** task to the left:



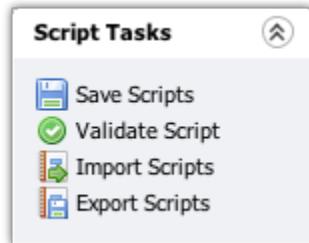
Editing Global Scripts

Select a script type from the drop down and edit the script in the text area below. There is also a [Reference List](#) to the right for easy drag-and-drop of common helper methods / code templates.

The following global script types can be edited:

- **Deploy Script:** This script executes once whenever any channel or set of channels is deployed. You have access to the global / configuration maps here. Typically this script is used to perform a one-time operation, like load a properties file from disk, or instantiate a database connection.
- **Undeploy Script:** This script executes once whenever any channel or set of channels is undeployed. You have access to the global / configuration maps here. Typically this script is used to cleanup any data created from the deploy script, such as closing a database connection.
- **Preprocessor Script:** This script applies across all channels, and executes once for every message, after the [attachment handler](#) has run but before the message reaches the source filter/transformer. You have access to "message", a string variable containing the incoming data. Whatever you return from the preprocessor script will be stored as the Processed Raw content and used to feed into the source filter /transformer. First the global preprocessor will be executed, and then the channel-level preprocessor.
- **Postprocessor Script:** This script applies across all channels, and executes once for every message, after all destinations have completed processed (not including queued messages which are processed asynchronously). You have access to "message", which is an ImmutableMessage object containing information about the state of all connector messages. The channel-level postprocessor script will execute first, and then the global postprocessor. If the channel-level postprocessor returned a response, that will be available as the variable "response" in the global postprocessor. This script may be used as a general tool to perform a custom cleanup script. It can also be used to return a custom response that may be sent back to the originating system.

Tasks



The following context-specific tasks are available:

Task Icon	Task Name	Description
	Save Scripts	Saves all global scripts.
	Validate Script	Validates the currently viewed script, ensuring that there are no syntax errors.
	Import Scripts	Imports all global scripts from an XML file.
	Export Scripts	Exports all global scripts to an XML file.

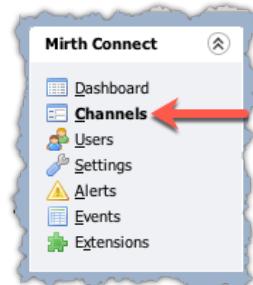
Edit Code Templates View

A **code template** is a function or snippet of code that can be used across multiple channels. A **code template library** is a group of code templates that is linked to one or more channels. When a library is linked to a channel, all code templates in the library will be available to the channel. This view is where code templates and libraries are configured for your NextGen Connect Integration Engine server.

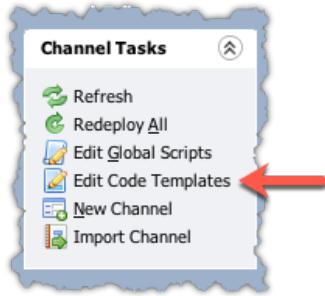
Name	Description	Revision	Last Modified
Create Segment Before Segment	Creates a new segment with the given name and inserts it before the given segment.	2	2017-04-17 08:01
Get Segments After a Particular Segment	Returns an array of segments with the specified name that come after a given segment in th...	1	2017-04-17 07:59
Insert Segment After Segment	Inserts a given segment after a particular segment.	1	2017-04-17 07:59
Insert Segment Before Segment	Inserts a given segment before a particular segment.	1	2017-04-17 07:59
Rename HL7 Field	This function returns a copy of the given HL7 field, changing all segment names to the given ...	1	2017-04-17 07:59
Unescape Xdd HL7 Sequences	Replaces any \xdd..\\HL7 escape sequences with the corresponding character(s).	1	2017-04-17 07:59
Template 1		1	2017-05-22 07:19
Encrypt PDF	Encrypts a PDF with a password. Either a Base64 string or by array can be passed in, and the ...	3	2017-05-22 07:19
Execute Runtime Command	Executes a command or array of command and arguments using a local OS shell. Returns an ...	1	2017-04-17 07:59
Overwrite Logger Categories	Overwrites the categories of Logger objects placed in JavaScript contexts.	1	2017-04-17 07:59
MomentJS	moment.js	2	2017-04-17 08:01

Navigation

Click the Channels link in the Mirth Connect task pane at the top-left to enter the [Channels View](#):



Click the Edit Code Templates link in the Channel Tasks task pane to the left:



This section is separated into the following topics:

- [Code Template Library Table](#)
- [Edit Library Panel](#)
- [Edit Code Template Panel](#)
- [Code Template Tasks](#)

Code Template Library Table

The table in the top section of the [Edit Code Templates View](#) shows all currently configured code templates and libraries. The name column is displayed as a tree, showing libraries at the hierarchical top level, and all child code templates at a lower level. The filter field at the bottom allows you to easily filter down to a particular code template or library by typing in all or part of the name. The status label to the left of the filter displays how many total code templates / libraries are configured, and how many are currently shown with the given filter. For additional information on tables in NextGen Connect Integration Engine, see [Mirth Connect Administrator](#).

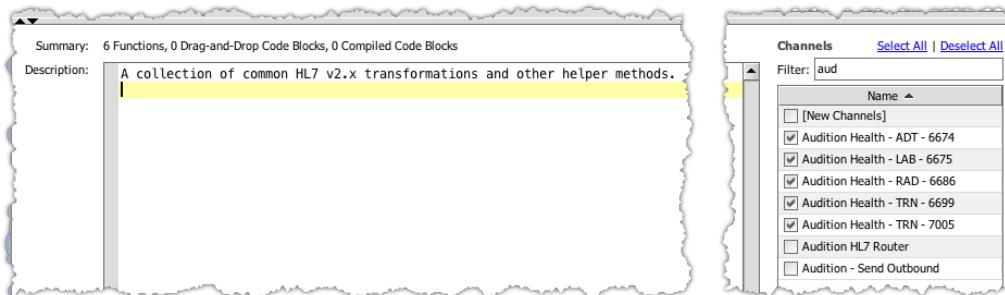
Code Templates					
	Name ▾	Description	Revision	Last Modified	
HL7 Helper Functions	Create Segment Before Segment	Creates a new segment with the given name and inserts it before the given segment.	2	2017-04-17 08:01	
	Get Segments After a Particular Segment	Returns an array of segments with the specified name that come after a given segment in th...	1	2017-04-17 07:59	
	Insert Segment After Segment	Inserts a given segment after a particular segment.	1	2017-04-17 07:59	
	Insert Segment Before Segment	Inserts a given segment before a particular segment.	1	2017-04-17 07:59	
	Rename HL7 Field	This function returns a copy of the given HL7 field, changing all segment names to the given ...	1	2017-04-17 07:59	
	Unescape Xdd HL7 Sequences	Replaces any Xdd.. HL7 escape sequences with the corresponding character(s).	1	2017-04-17 07:59	
Miscellaneous Functions	Encrypt PDF	Encrypts a PDF with a password. Either a Base64 string or by array can be passed in, and the ...	3	2017-05-22 07:19	
	moment.js	Encrypts a PDF with a password. Either a Base64 string or by array can be passed in, and the ...	2	2017-04-17 08:01	
	Recursive XML Node Iteration	Recursively iterates through all descendant nodes of an E4X XML object.	1	2017-04-17 07:59	
			3	2017-05-22 07:19	
			1	2017-04-17 07:59	

4 of 6 Libraries (2 filtered), 9 of 14 Code Templates (5 filtered) Filter: en

Column	Description
Name	The name of the code template or library. This column is displayed as a tree, showing libraries at the hierarchical top level, and all child code templates at a lower level. Double-click on this column to edit the name.
Id	The unique identifier for the code template or library.
Type	Only applicable for code templates, this is the type of template (Function, Drag-and-Drop Code Block, Compiled Code Block). For more information, see Edit Code Template Panel .
Description	A description for the code template or library.
Revision	The current revision for the code template or library. When code templates are modified in any way and saved, the revision is incremented. When libraries are modified in any way <i>or</i> when any code template is added or removed from the library and then saved, the library revision is incremented.
Last Modified	The timestamp at which the code template or library was last updated.

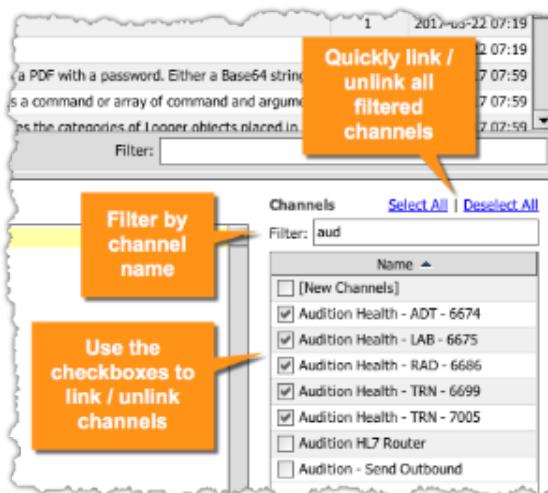
Edit Library Panel

When any library is selected in the [Code Template Library Table](#), the Edit Library Panel is shown below. This allows you to change the description of the library, and decide which channels to include the library in.



Linking Libraries to Channels

In order to use code templates within channels, the library must be linked to the channel. This is important because it allows you to "namespace" your code so that the same function name may appear in multiple libraries but serve different purposes. It also allows a degree of isolation, so that a particularly expensive code template need not be included in all channels, but instead *only* in the channels it needs to be. To edit the links between code template libraries and channels, use the Channels table on the right-hand side of this panel:

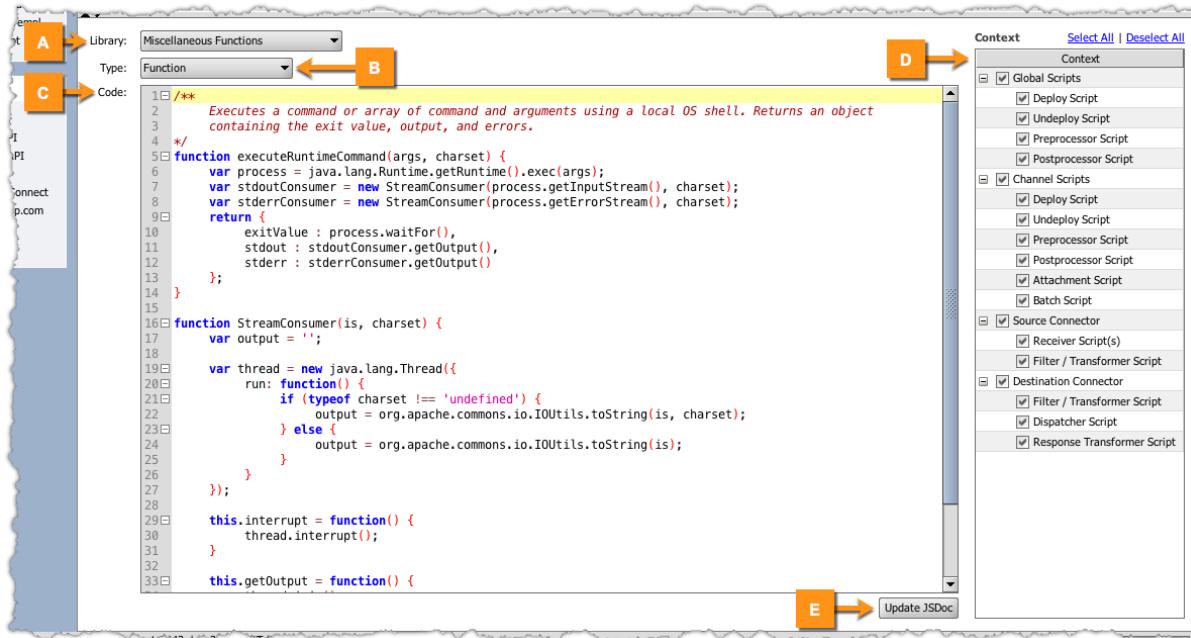


The **[New Channels]** check box indicates that the library should be included not only for the currently checked channels, but also on ***all*** new channels that get created later. For example if you have a library that you wish to always include on all channels no matter what, check this option.

Note that these links are the same as what is configured in the [Code Template Libraries dependency tab](#) in the [Summary Tab](#) of the [Edit Channel View](#).

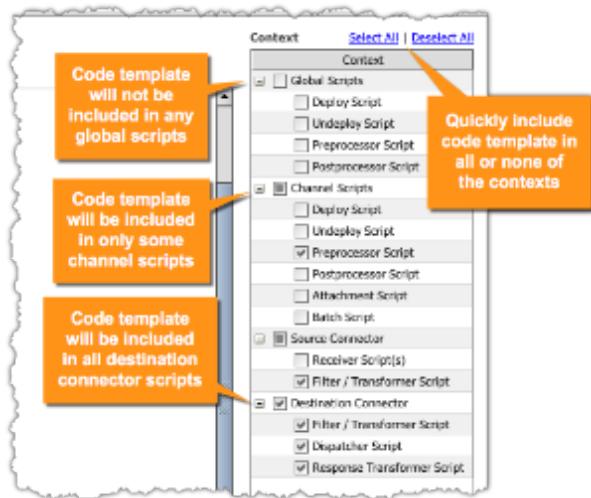
Edit Code Template Panel

When any code template is selected in the [table](#) above, the Edit Code Template Panel is shown below. This allows you to change which library it belongs to, the type of template, which contexts to include the code in, and the actual code.



Item	Name	Description
A	Library	The library that the current code template belongs to.
B	Type	<p>The type of code template to create.</p> <ul style="list-style-type: none"> Function: The template will be compiled in with scripts, and the drag-and-drop will include the function signature. Drag-and-Drop Code Block: The template will not be compiled in with scripts, and the drag-and-drop will include the entire code block verbatim (except for the initial documentation block). Use this option for snippets of boilerplate code you can drag into JavaScript scripts. Compiled Code Block: The template will be compiled in with scripts, but drag-and-drop will not be available at all. Use this to declare initial variables or anything else you want to have executed at the beginning of your scripts.
C	Code	The actual code to use. At the top of the code you may optionally provide a JSDoc comment block explaining the purpose and function of the template. This comment will be shown as the description for the template.
D	Context	Select which scripts should have access to this code template. Limiting contexts to only those scripts where a code template is actually <i>needed</i> can help optimize memory usage. For additional information, see Code Template Contexts .
E	Update JSDoc	Generates / updates a JSDoc at the beginning of your code, with parameter / return annotations as needed. For additional information, see Using JSDoc in Code Templates .

Code Template Contexts



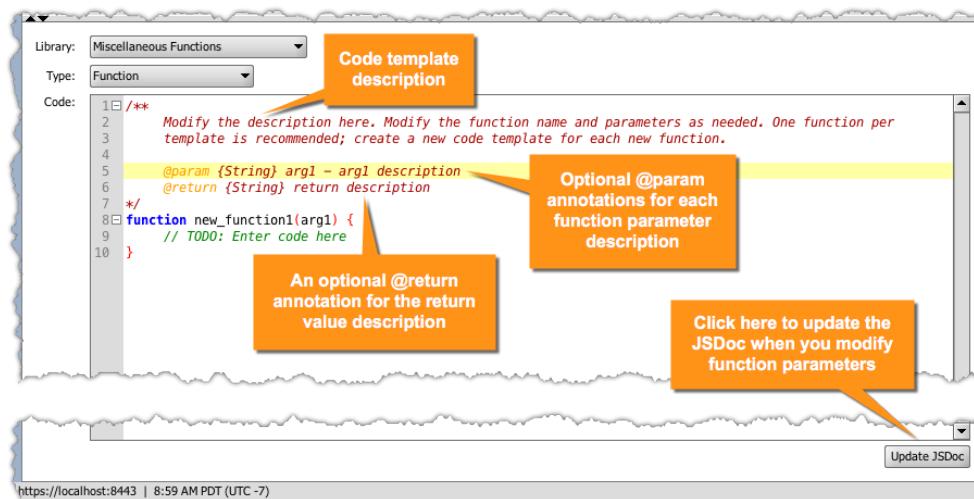
Similar to how a [code template library](#) can be included only on specific channels, code templates can further be isolated to specific scripts. This can be helpful for ensuring that there are no conflicts between function names, and also for better memory usage, since not all code templates need to be compiled in with all scripts across your entire server. The script contexts are organized into groups, allowing you to easily include a code template in, for example, `/global` scripts, in a single click. The following groups are displayed:

- **Global Scripts:** The global deploy/undeploy/preprocessor/postprocessor scripts, not specific to any particular channel. For additional information, see [Edit Global Scripts View](#).
- **Channel Scripts:** The channel-level deploy/undeploy/preprocessor/postprocessor scripts (more info [here](#)), as well as the [JavaScript Attachment Handler](#) and JavaScript Batch Adapter (for additional information, see [Data Types](#)).
- **Source Connector:** The source [filter/transformer script](#), and any script associated with the source connector. Examples of this include the [JavaScript Reader](#), [Database Reader](#) (in JavaScript mode), and the [JavaScript HTTP Authentication script](#).
- **Destination Connector:** The destination [filter/transformer script](#), the [response transformer](#), and any script associated with the destination connector. Examples of this include the [JavaScript Writer](#) and [Database Writer](#) (in JavaScript mode).

Using JSDoc in Code Templates

A [JSDoc](#) is a type of comment block used to annotate JavaScript scripts. For code templates in NextGen Connect Integration Engine, this is used not only for good documentation, but also for the code template description and the information that shows up in the auto-completion dialog in the [JavaScript Editor](#).

When you create a new code template, a sample JSDoc is created for you:

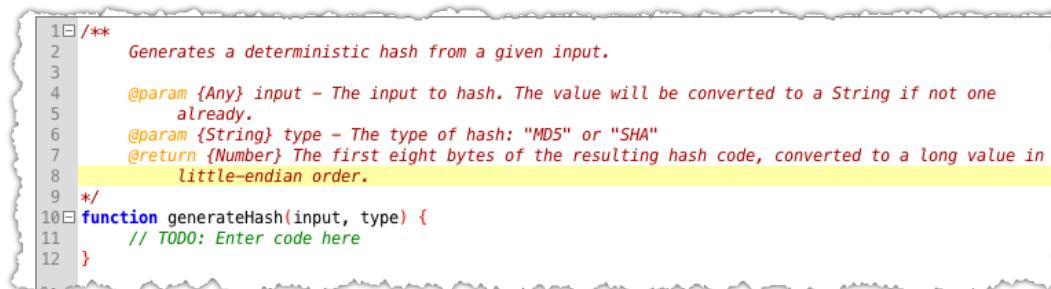


The first portion of the comment block is used for the description of the code template, and may contain multiple lines and blank lines. Following that, you can include any [JSDoc](#) annotations, your own custom annotations, or whatever you want, as long as it follows correct JavaScript syntax. Only the following annotations are recognized by NextGen Connect Integration Engine for the purpose of populating the auto-completion dialog in the [JavaScript Editor](#):

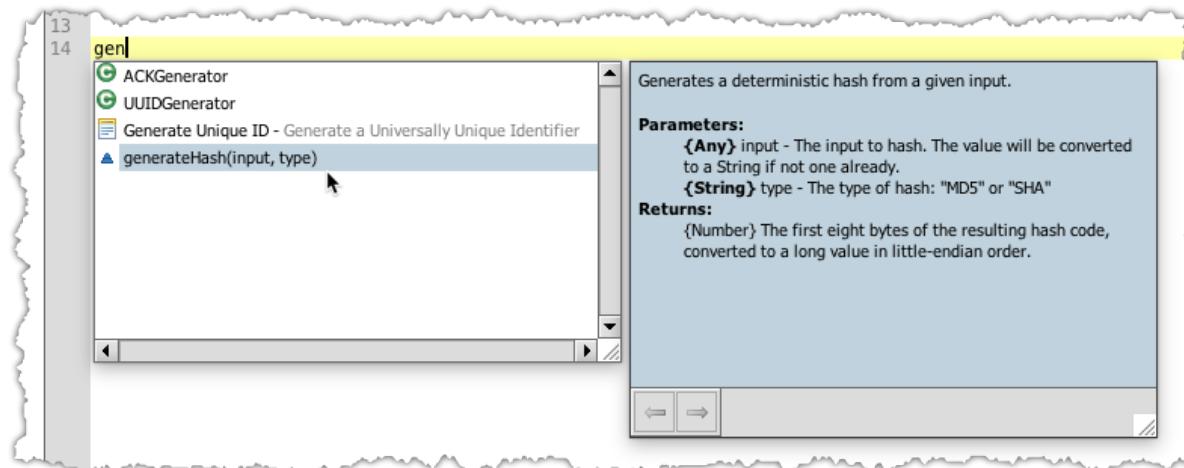
- **@param:** Documents the input arguments for your function, in the order that they appear. If you have multiple function arguments, you will want to add multiple @param annotations. The format is:
 - @param {Type} Name - Description

Note that the type doesn't have to be an actual JavaScript type. It can be anything you want as long as it doesn't contain the "{}" characters, like "MyObject", "String/Number", etc.
- **@return:** Documents the return value for your function, if applicable. The format is:
 - @return Description

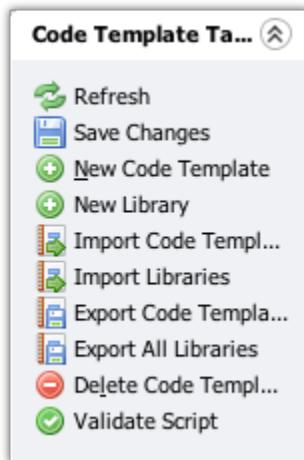
If you change the names or number of arguments in your function, you can use the **Update JSDoc** button at the bottom of the editor. This will automatically inject or update @param annotations as needed, after which you can change the Type and Description as needed.



Once you've filled out your JSDoc appropriately, the function will appear in the auto-completion dialog accessible from the [JavaScript Editor](#):



Code Template Tasks



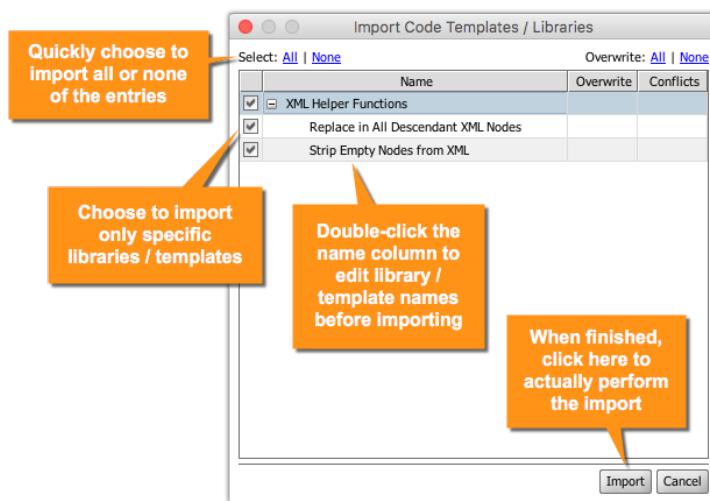
The following context-specific tasks are available throughout the [Edit Code Templates View](#):

Task Icon	Task Name	Description
	Refresh	Refreshes the list of code templates / libraries. If there are unsaved changes, you will be prompted to save first before refreshing.
	Save Changes	Saves all changes made to all code templates / libraries. Changes to a code template will cause its revision to increment. Changes to a library, or adding/removing code templates to/from the library, will cause its revision to increment.
	New Code Template	Creates a new code template in the currently selected library. If there are currently no libraries configured for your server, you must first create a new library before this task will become available.
	New Library	Creates a new code template library and adds it to the table. By default the library will not be included in any channels. Use the Channels table in the Edit Library Panel to link the library to specific channels.
	Import Code Templates	Import a single code template or a list of code templates from an XML file.
	Import Libraries	Import a single code template library or a list of libraries (including any code templates within) from an XML file.
	Export Code Template	Exports the currently selected code template to an XML file.
	Export Library	Exports the currently selected code template library to an XML file.
	Export All Libraries	Exports all libraries in the table to separate XML files.

 Delete Code Temp...	Delete Code Template	Deletes the currently selected code template, removing it from the table.
 Delete Library	Delete Library	Deletes the currently selected code template library, and all code templates belonging to the library.
 Validate Script	Validate Script	Validates the currently selected code template, ensuring that all properties are valid, and that the actual code has proper syntax.

Importing Code Templates / Libraries

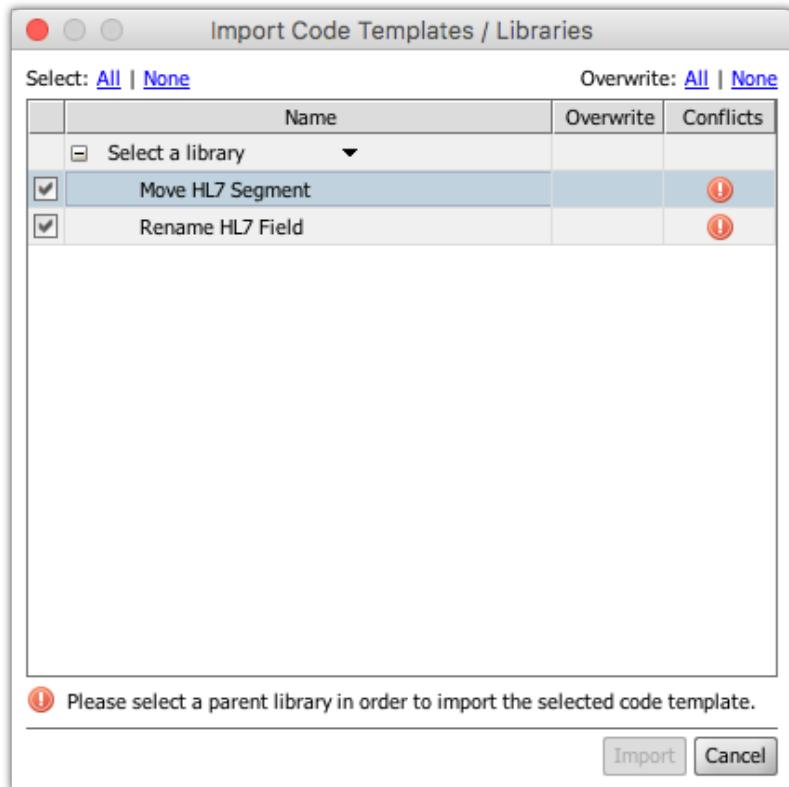
When importing code templates / libraries, either in the [Edit Code Templates View](#) or when [importing a channel](#) containing libraries, you'll be presented with a confirmation dialog.



All libraries and code templates contained in the import file/channel will be displayed in the dialog. From here you can choose to import everything, or individually select specific entries to import. Once you've made your selections, click the **Import** button to actually perform the import operation.

- If a code template or library already exists in your NextGen Connect Integration Engine server, you will see the Overwrite column display a checkbox. If checked, the entry will overwrite the current code template / library rather than creating a new one. If there are multiple entries with such conflicts, you can easily choose to overwrite all or none of them with the All / None links at the top-right of the dialog.
- If the library or code template has a name that conflicts with another entry in your current table, you will see the Conflicts column show a red error icon. In this case, you must choose to either overwrite the current entry, or update its name by double-clicking on the Name column.
- If a code template already exists in a different library than the one you're trying to import into, you'll see a yellow warning icon. In this case, you can choose to overwrite the current template, ignore the warning and continue, or simply cancel the operation.

When importing just code templates by themselves, you must first select a library to import them into:



Edit Alert View

An **alert** is a process that listens for certain types of events and triggers based on configurable settings. From these triggers you can take various actions, like dispatching an e-mail to a user or specific address, or sending a message to a channel. NextGen Connect Integration Engine comes with a built-in Error-based alerting system that listens for error events from selected channels. The [Advanced Alerting](#) extension adds on this by also including powerful metric-based alerts, escalation levels, scheduling, notification throttling, and other advanced features.

The Edit Alert view is where alerts are configured / modified. Once saved, an alert will be displayed on the [Alerts View](#), which is like a dashboard for your currently configured alerts.

The screenshot shows the 'Edit Alert - Connection Errors' configuration screen in the Mirth Connect Administrator. The left sidebar includes links for Dashboard, Channels, Users, Settings, Alerts, Events, and Extensions. Under 'Alert Edit Tasks', there are options for Save Alert and Export Alert. The 'Other' section contains links for Notifications, View User API, View Client API, Help, About Mirth Connect, Visit mirthcorp.com, Report Issue, and Logout.

The main configuration area has the following sections:

- Alert Name:** Connection Errors
- Enabled:**
- Errors (select all that apply):**
 - Error
 - Any
 - Source Connector
 - Destination Connector
 - Serializer
 - Filter
 - Transformer
 - User Defined Transformer
 - Response Validation
 - Response Transformer
 - Attachment Handler
 - Deploy Script
 - Preprocessor Script
 - Postprocessor Script
 - Undeploy Script
- Regex (optional):** Connection (Refused|Reset)|Read timed out
- Channels:**
 - Filter: lab
 - Enable Disable
 - [Expand All](#) [Collapse All](#)
 - Audition Health - LAB - 6675 (red dot)
 - Source (red dot)
 - Destination 1 (red dot)
 - [New Destinations] (red dot)
 - Calif Parth - LAB - 6635 (green dot)
 - Source (green dot)
 - Destination 1 (green dot)
 - [New Destinations] (green dot)
 - SamWorther Tech - LAB - 7713 (yellow dot)
 - Source (yellow dot)
 - Destination 1 (yellow dot)
 - [New Destinations] (yellow dot)
- Actions:**

Protocol	Recipient	Add
Channel	action Receiver - Reprocess	<input type="button"/> Remove
User	janeW	
User	susanc	
Email	helpdesk@hiemazing.com	
- Subject (only used for email messages):** Connection Error Detected in Channel: \${channelName}
- Template:**

```

${channelName} (${channelId})
Connector: ${connectorName}
Message ID: ${messageId}
Date: ${date}

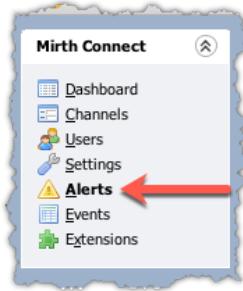
${error}

```
- Alert Variables:**
 - alertId
 - alertName
 - serverId
 - globalMapVariable
 - date
 - systemTime
 - error
 - errorMessage
 - errorType
 - channelId
 - channelName
 - connectorName
 - connectorType
 - messageId

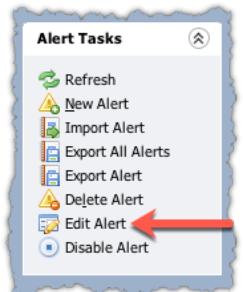
At the bottom left, it says "Connected to: Prod Server 1 | https://localhost:8443 | 10:34 AM PDT (UTC -7)".

Navigation

Click the Alerts link in the Mirth Connect task panel at the upper-left:



Select the alert you want to edit, and click the Edit Alert task in the panel on the left:



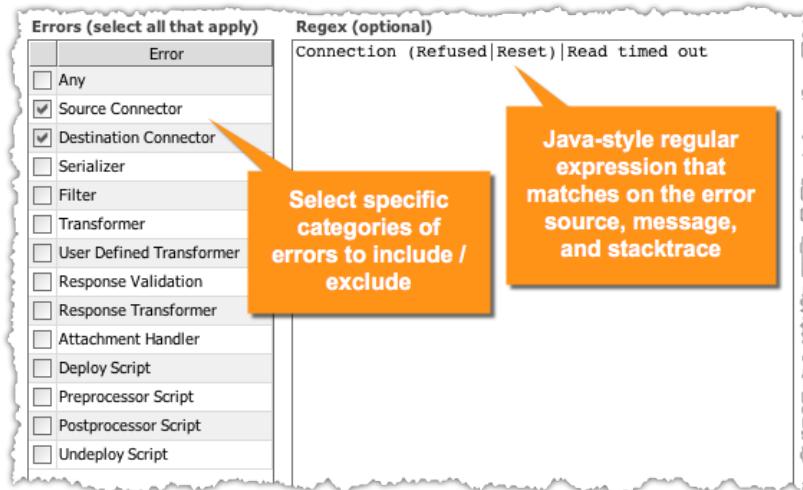
OR, simply double-click the alert from the [Alerts Table](#).

This section is separated into the following topics:

- Alert Error Types and Regex
- Alert Enabled Channels
- Alert Actions
- Edit Alert Tasks

Alert Error Types and Regex

This section of the [Edit Alert View](#) allows you to decide what type of errors you want your alert to trigger on. This includes choosing among categories of error types, and optionally using a regular expression that only allows events whose error source, error message, or exception stacktrace matches to cause an alert trigger to occur.



Alert Error Categories

- **Any:** Select this option to trigger on *all*/error categories.
- **Source Connector:** An error that originated from the source connector during deploy/start, receiving a message, or sending a response.
- **Destination Connector:** An error that originated from a destination connector during deploy/start, dispatching a message, or receiving a response.
- **Serializer:** An error that originated from the source or destination filter/transformer script, specifically when serializing the message to the internal representation (e.g. XML), or deserializing the message to the outbound data type format.
- **Filter:** An error that originated from a source or destination filter rule.
- **Transformer:** An error that originated from a source or destination transformer step.
- **User Defined Transformer:** An error that originated from a user-defined call to AlertSender (alerts.sendAlert) from any script. For additional information, see [The User API \(Javadoc\)](#).
- **Response Validation:** An error that originated from a destination connector after a response has been received, when the response fails validation. For example, an HL7 v2.x negative acknowledgement (NACK).
- **Response Transformer:** An error that originated from a destination connector response transformer step.
- **Attachment Handler:** An error that originated from an [Attachment Handler](#).
- **Deploy Script:** An error that originated from a channel deploy script.
- **Preprocessor Script:** An error that originated from a global or channel preprocessor script.
- **Postprocessor Script:** An error that originated from a global or channel postprocessor script.
- **Undeploy Script:** An error that originated from a channel undeploy script.

Alert Enabled Channels

This section of the [Edit Alert View](#) allows you to decide which channels and connectors may trigger an alert. You can choose to, for example, only listen for error events coming from source connectors, and ignore those coming from destination connectors.

Enabled

Filter by channel name

Nothing associated with this channel will trigger an alert

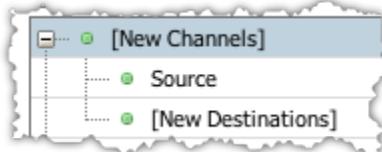
The entire channel and all connectors may trigger alerts

Only some connectors in the channel may trigger alerts

Expand All Collapse All

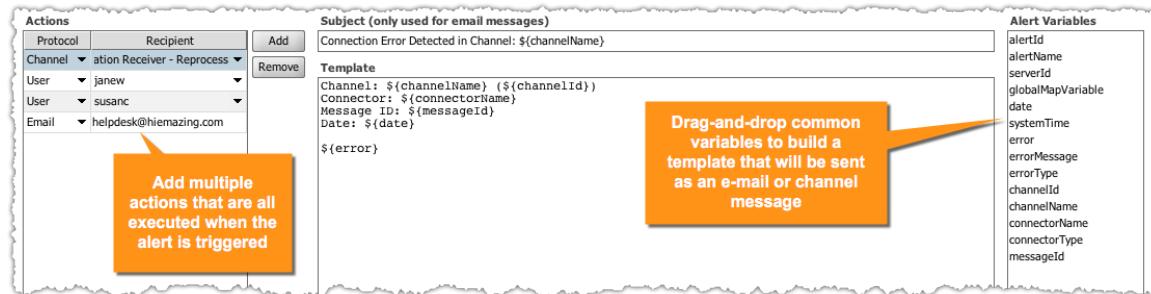
- Audition Health - LAB - 6675
 - Source
 - Destination 1
 - [New Destinations]
- Calif Parth - LAB - 6635
 - Source
 - Destination 1
 - [New Destinations]
- SamWorther Tech - LAB - 7713
 - Source
 - Destination 1
 - [New Destinations]

The **[New Channels]** option indicates that you want an alert to listen for error events on any *new* channels that are created for a channel after the alert was initially created. Similarly, the **[New Destinations]** option indicates that you want an alert to listen for error events on any *new* destinations that are created for a channel after the alert was initially created.



Alert Actions

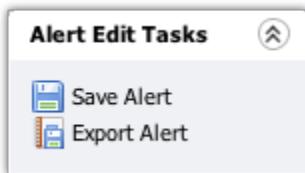
This section of the [Edit Alert View](#) allows you to decide what actions to take after your alert has been triggered. You can take multiple actions all at once for a single trigger event, which may include sending e-mails to multiple users and dispatching messages to multiple channels.



Alert Variables

- alertId:** The unique ID of the alert.
- alertName:** The name of the alert.
- serverId:** The unique ID of the server.
- globalMapVariable:** Indicates that you can use any variable in the global or configuration maps.
- date:** The current date, formatted as a human-readable string.
- systemTime:** The current epoch time in milliseconds.
- error:** The full error message, including (where applicable) the source code, line number, and error details.
- errorMessage:** If an exception is associated with the error event, this will be the message or detail of the exception.
- errorType:** The category of the error. For additional information, see [Alert Error Types and Regex](#).
- channelId:** The ID of the channel associated with the error, if applicable.
- channelName:** The name of the channel associated with the error, if applicable.
- connectorName:** The name of the connector associated with the error, if applicable.
- connectorType:** The type of the connector associated with the error, if applicable.
- messageId:** The ID of the message associated with the error, if applicable.

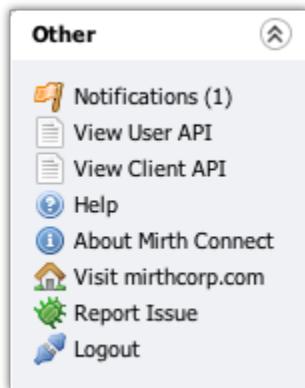
Edit Alert Tasks



The following context-specific tasks are available throughout the [Edit Alert View](#):

Task Icon	Task Name	Description
	Save Alert	Saves all changes made to the current alert. If the alert is enabled, it will automatically be listening for new events to trigger on. Note that since many alerts rely on dispatching e-mails, you will be warned if your global SMTP settings aren't configured on the Server Settings Tab .
	Export Alert	Exports the current alert to an XML file.

Other Tasks



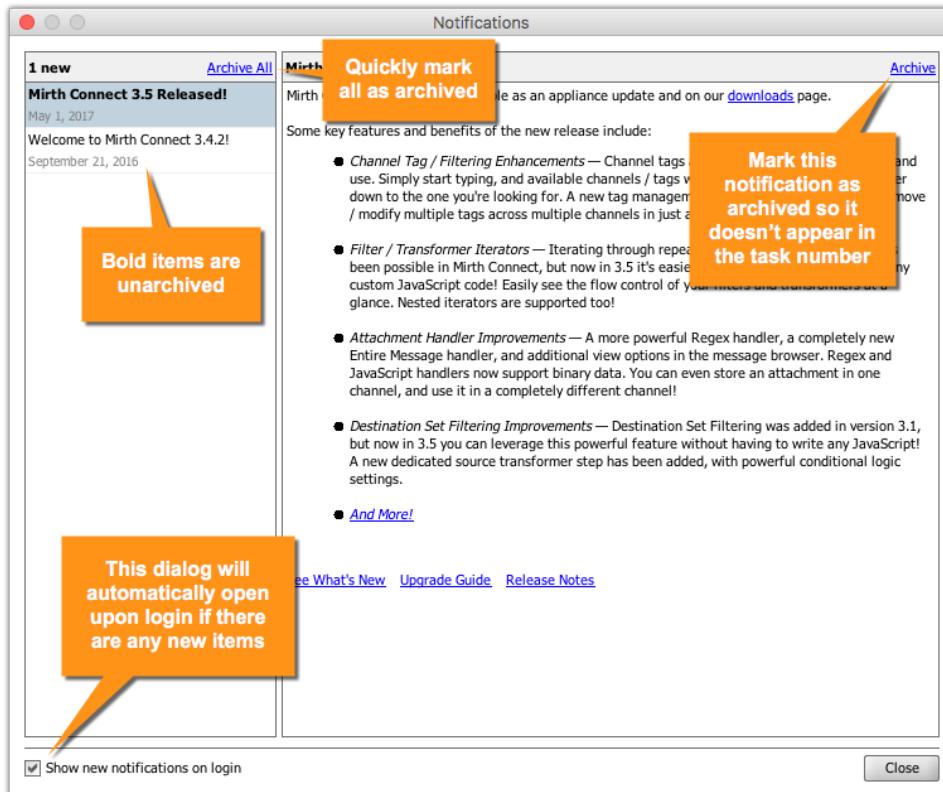
The following tasks are available at any time at the bottom-left of the task pane throughout the [Mirth Connect Administrator](#).

Task Icon	Task Name	Description
	Notifications (1)	View all notifications received from NextGen Connect Integration Engine headquarters. This allows you to be notified when there is a new version of NextGen Connect Integration Engine available, or for other news. The number shown in this task indicates how many unread or unarchived notifications you currently have. For additional information, see Notifications .
	View User API	Opens the User API (Javadoc) documentation in your default browser.
	View Client API	Open the NextGen Connect REST API documentation in your default browser.
	Help	Opens the Mirth Connect Wiki page in your default browser.
	About Mirth Connect	Opens the About dialog for NextGen Connect Integration Engine, which displays the current version, build date, server ID, version of Java being used, copyright information, and third-party acknowledgements. Note that the third-party libraries shown in this dialog are not an exhaustive list; look at the "docs/thirdparty" folder in the Installation Directory for a more complete list.
	Visit mirthcorp.com	Opens the Mirth website in your default browser.
	Report Issue	Opens the Mirth public issue tracker in your default browser. Please search the tracker for similar issues before creating a new issue. For defects / bugs, please try to include full reproduction steps, error stacktraces, the NextGen Connect Integration Engine / Java versions, and as much information as you can.
	Logout	Logs out of the Mirth Connect Administrator, and brings you back to the Login Dialog .



Notifications

This dialog shows all messages you've received from NextGen Connect Integration Engine headquarters. This allows you to be notified when a new version of NextGen Connect Integration Engine is available, or for other news.



As you read notifications, you can choose to archive them, which means they will not show up as **bold** anymore, and will not be counted in the number shown in the [Other Tasks](#) pane.

By default, this dialog will automatically pop-up when logging into the Administrator if there are new items. To turn this off, open the dialog and uncheck "**Show new notifications on login**".

Data Types

This section describes the various common properties configurable across data types, and specific properties for each data type. For an introduction to data types in general, see [About Data Types](#).

Whether a data type is used as **inbound** or **outbound**, and whether it is tied to a source connector, destination connector, or destination response, affects what properties it needs. The following groups of properties may be displayed in the [Set Data Types Dialog](#) depending on the data type and context:

Inbound Properties

- **Serialization Properties:** Determines how to convert data from the raw inbound format to the internal representation (e.g. XML). If a data type doesn't have serialization properties present, it either doesn't need any (DICOM, JSON), or it doesn't actually do any serialization (Raw).
- **Batch Properties:** Determines how to split an incoming message into multiple messages. Only supported when Process Batch is enabled in the [Source Settings](#). This will only be displayed for source connectors. Not all data types support batch processing (DICOM).
- **Response Generation Properties:** When an auto-generation option is chosen for the response on the [Source Settings](#), these properties determine how to generate an automatic response. This will only be displayed for source connectors. Not all data types support automatic response generation.
- **Response Validation Properties:** Determines how to validate responses received by a destination after dispatching a message. Only supported when Validate Response is enabled in the [Destination Settings](#). This will only be displayed for destination responses. Not all data types support automatic response generation.

Outbound Properties

- **Deserialization Properties:** After a transformer finishes, these properties determine how to convert data from the final internal representation (e.g. XML) into the outbound data format. If a data type doesn't have deserialization properties present, it either doesn't need any (DICOM, EDI/X12, XML, JSON) or it doesn't actually do any deserialization (Raw).
- **Template Serialization:** If an outbound template is specified for the transformer, these properties determine how to convert that template to its corresponding internal representation (e.g. XML).

NextGen Connect Integration Engine supports the following data types:

- [Delimited Text Data Type](#)
- [DICOM Data Type](#)
- [EDI / X12 Data Type](#)
- [HL7 v2.x Data Type](#)
- [HL7 v3.x Data Type](#)
- [JSON Data Type](#)
- [NCPDP Data Type](#)
- [Raw Data Type](#)
- [XML Data Type](#)
- [Batch Processing](#)
- [JavaScript Batch Script](#)

An additional data type is made available as a commercial extension: [ASTM E1394 Data Type](#)

Delimited Text Data Type

The Delimited Text data type is a very powerful and flexible data type that can satisfy many common formats like CSV, but also many proprietary formats that are dependent on custom delimiters, fixed-width columns, or other nuances. The following properties are configurable:

Serialization / Template Serialization / Deserialization Properties		
Name	Default Value	Description
Column Delimiter	,	If column values are delimited, enter the characters that separate columns. For example, this is a comma in a CSV file.
Record Delimiter	\n	Enter the characters that separate each record (a message may contain multiple records). For example, this is a newline (\n) in a CSV file.
Column Widths		If the column values are fixed width, enter a comma separated list of fixed column widths. By default, column values are assumed to be delimited.
Quote Token	"	Enter the quote characters that are used to bracket delimit column values containing embedded special characters like column delimiters, record delimiters, quote characters and/or message delimiters. For example, this is a double quote ("") in a CSV file.
Double Quote Escaping	Enabled	By default, two consecutive quote tokens within a quoted value are treated as an embedded quote token. Uncheck to enable escaped quote token processing (and specify the Escape Tokens).
Escape Token	\	Enter the characters used to escape embedded quote tokens. By default, this is a back slash. This option has no effect unless Double Quote Escaping is unchecked.
Column Names		To override the default column names (column1, ..., columnN), enter a comma separated list of column names.
Numbered Rows	Disabled	Check to number each row in the XML representation of the message.
Ignore Carriage Returns	Enabled	Ignores carriage return (\r) characters. These are read over and skipped without processing them.

Batch Properties		
Name	Default Value	Description
Split Batch By	Record	<p>Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings. The following options are available:</p> <ul style="list-style-type: none"> • Record: Treat each record as a message. Records are separated by the record delimiter from the serialization properties. • Delimiter: Use the Batch Delimiter to separate messages. • Grouping Column: Use a column to group multiple records into a single message. When the specified column value changes, this signifies the boundary between messages. • JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.

Number of Header Records	0	The number of header records to skip. By default, no header records are skipped.
Batch Delimiter		The delimiter that separates messages. The batch delimiter may be a sequence of characters.
Include Batch Delimiter	Disabled	Check to include the batch delimiter in the message returned by the batch processor. By default, batch delimiters are consumed.
Grouping Column		The name of the column used to group multiple records into a single message. When the specified column value changes, this signifies the boundary between messages.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

DICOM Data Type

This data type works in conjunction with the [DICOM Listener / DICOM Sender](#) and the [DICOM Attachment Handler](#) to consume and transformer DICOM messages. It has no configurable data type properties, but will automatically convert binary DICOM data to and from an XML format specified by the dcm4che parser library.

Example XML snippet:

```
<dicom>
    <tag00020000 len="4" tag="00020000" vr="UL">212</tag00020000>
    <tag00020001 len="2" tag="00020001" vr="OB">00\01</tag00020001>
    <tag00020002 len="26" tag="00020002" vr="UI">1.2.840.10008.5.1.4.1.1.4</tag00020002>
    <tag00020003 len="60" tag="00020003" vr="UI">1.
3.46.670589.11.30.9.1062531302827752870602.13.1.1.0.0.1</tag00020003>
    <tag00020010 len="18" tag="00020010" vr="UI">1.2.840.10008.1.2</tag00020010>
    <tag00020012 len="18" tag="00020012" vr="UI">1.3.46.670589.17.1</tag00020012>
    <tag00020013 len="14" tag="00020013" vr="SH">ARCVTS04NOV99</tag00020013>
    <tag00020016 len="14" tag="00020016" vr="AE">VTS_DCM_STORE</tag00020016>
    <tag00080005 len="10" tag="00080005" vr="CS">ISO_IR_100</tag00080005>
    <tag00080008 len="26" tag="00080008" vr="CS">ORIGINAL\PRIMARY\M_SE\M\SE</tag00080008>
```

Each node in the XML document contains the attribute length, tag code, value representation, and actual value. These can be used or modified within transformers, and the DICOM data type will automatically convert the finished XML to the native DICOM binary format.

EDI / X12 Data Type

This data type handles both [UN/EDIFACT](#) and [ASC X12](#) data formats, as well as custom formats similar to EDI / X12 that use a segment / element / subelement delimiter.

Serialization / Template Serialization / Deserialization Properties		
Name	Default Value	Description
Segment Delimiter	~	Characters that delimit the segments in the message.
Element Delimiter	*	Characters that delimit the elements in the message.
Subelement Delimiter	:	Characters that delimit the subelements in the message.
Infer X12 Delimiters	Enabled	This property only applies to X12 messages. If checked, the delimiters are inferred from the incoming message and the delimiter properties will not be used.

Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none"> • JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

HL7 v2.x Data Type

This data type allows powerful and flexible parsing and manipulation of HL7 v2.x messages. It features two modes, strict and non-strict. The strict mode parses messages into and from XML according to the official XSD, and allows automatic validation against the HL7 specifications. The non-strict mode parses messages into a simple, consistent XML structure consisting of the segment/field/component/subcomponent hierarchy, and allows quick and easy transformation in most use-cases.

The data type also features an automatic response (ACK) generator, and a response validator that can mark messages as failed when a negative acknowledgement is received or when the message control IDs don't match. It also has a batch adapter that can split messages based on the MSH segment, even while streaming over a TCP connection.

Serialization / Template Serialization / Deserialization Properties		
Name	Default Value	Description
Parse Field Repetitions	Enabled	Parse field repetitions (applies to Non-Strict Parser only).
Parse Subcomponents	Enabled	Parse subcomponents (applies to Non-Strict Parser only).
Use Strict Parser	Disabled	Parse messages based upon strict HL7 specifications.
Validate in Strict Parser	Disabled	Validate messages using HL7 specifications (applies to Strict Parser only).
Strip Namespaces	Enabled	Strips namespace definitions from the transformed XML message (applies to Strict Parser only).
Segment Delimiter	\r	This is the input delimiter character(s) expected to occur after each segment.
Convert Line Breaks	Enabled	Convert all styles of line breaks (CRLF, CR, LF) in the raw message to the segment delimiter.

Batch Properties		
Name	Default Value	Description
Split Batch By	MSH Segment	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none"> MSH Segment: Each MSH Segment indicates the start of a new message in the batch. JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

Response Generation Properties		
Name	Default Value	Description
Segment Delimiter	\r	These are the delimiter character(s) that will be used after each segment. This option has no effect unless an "Auto-generate" item has been selected in the response settings.
Successful ACK Code	AA	The ACK code to respond with when the message processes successfully. This value supports Velocity Variable Replacement with values from the current connector message.
Successful ACK Message		The ACK message to respond with when the message processes successfully. This value supports Velocity Variable Replacement with values from the current connector message.
Error ACK Code	AE	The ACK code to respond with when an error occurs during message processing. This value supports Velocity Variable Replacement with values from the current connector message.
Error ACK Message	An Error Occurred Processing Message.	The ACK message to respond with when an error occurs during message processing. This value supports Velocity Variable Replacement with values from the current connector message.
Rejected ACK Code	AR	The ACK code to respond with when the message is filtered. This value supports Velocity Variable Replacement with values from the current connector message.
Rejected ACK Message	Message Rejected.	The ACK message to respond with when the message is filtered. This value supports Velocity Variable Replacement with values from the current connector message.
MSH-15 ACK Accept	Disabled	This setting determines if Mirth should check the MSH-15 field of an incoming message to control the acknowledgment conditions. The MSH-15 field specifies if a message should be always acknowledged, never acknowledged, or only acknowledged on error.
Date Format	yyyyMMddHHmmss.SSS	This setting determines the date format used for the timestamp in the generated ACK. The default value is "yyyyMMddHHmmss.SSS".

Response Validation Properties		
Name	Default Value	Description
Successful ACK Codes	AA,CA	The ACK code(s) to expect when the message is accepted by the downstream system. By default, the message status will be set to SENT. Specify multiple codes with a list of comma separated values.
Error ACK Codes	AE,CE	The ACK code(s) to expect when an error occurs on the downstream system. By default, the message status will be set to ERROR. Specify multiple codes with a list of comma separated values.
Rejected ACK Codes	AR,CR	The ACK code(s) to expect when the message is rejected by the downstream system. By default, the message status will be set to ERROR. Specify multiple codes with a list of comma separated values.

Validate Message Control Id	Enabled	Select this option to validate the Message Control Id (MSA-2) returned from the response.
Original Message Control Id	Destination Encoded	Select the source of the original Message Control Id used to validate the response. If Destination Encoded is selected, the Id will be extracted from the MSH-10 field of the destination's encoded content. If Map Variable is selected, the Id will be retrieved from the destination's connector map or the channel map.
Original Id Map Variable		This field must be populated if the Original Message Control Id is set to Map Variable. The Id will be read from this variable in the destination's connector map or the channel map.

HL7 v3.x Data Type

This data type handles HL7 v3.x messages. No actual serialization or deserialization is needed because the data format is the same as the internal representation format (XML), but it still has options to strip namespaces if needed.

Serialization / Template Serialization Properties		
Name	Default Value	Description
Strip Namespaces	Enabled	Strips namespace definitions from the transformed XML message. Will not remove namespace prefixes. If you do not strip namespaces your default xml namespace will be set to the incoming data namespace. If your outbound template namespace is different, you will have to set "default xml namespace = 'namespace'" via JavaScript before template mappings.

Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none">• JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

JSON Data Type

This data type allows seamless integration between JSON messages and the filter/transformer scripts which are JavaScript-based. When using the JSON data type, the **msg** / **tmp** variables will be a standard JavaScript object, as opposed to an E4X XML object which some other data types use. Since JSON is a very lightweight data format, and the transition from String to JavaScript Object is all handled automatically, no serialization properties are needed.

Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none">• JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

NCPDP Data Type

This data type handles the flat file format for National Council for Prescription Drug Programs (NCPDP) pharmacy data.

Serialization / Template Serialization / Deserialization Properties		
Name	Default Value	Description
Field Delimiter	0x1C	Characters that delimit the fields in the message.
Group Delimiter	0x1D	Characters that delimit the groups in the message.
Segment Delimiter	0x1E	Characters that delimit the segments in the message.
Use Strict Validation	Disabled	Validates the NCPDP message against the appropriate schema. Only applicable for the Deserialization properties.

Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none">• JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

Raw Data Type

This data type allows a channel / connector to process any custom data not handled by any of the other data types. When using it, the **msg** / **tmp** variable accessible in the filter/transformer will be a String rather than an E4X XML object which some other data types use. This data type also has the special property that when used as the Response Inbound data type for a destination, the response transformer will *always* be executed, even if no actual response data was received. No serialization properties are needed since there is no conversion done.

Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none">• JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script

XML Data Type

This data type handles XML messages. No actual serialization or deserialization is needed because the data format is the same as the internal representation format (XML), but it still has options to strip namespaces if needed.

Serialization / Template Serialization Properties		
Name	Default Value	Description
Strip Namespaces	Enabled	Strips namespace definitions from the transformed XML message. Will not remove namespace prefixes. If you do not strip namespaces your default xml namespace will be set to the incoming data namespace. If your outbound template namespace is different, you will have to set "default xml namespace = 'namespace'" via JavaScript before template mappings.
Batch Properties		
Name	Default Value	Description
Split Batch By	JavaScript	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings . The following options are available: <ul style="list-style-type: none">• JavaScript: Use JavaScript to split messages. For additional information, see JavaScript Batch Script.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java BufferedReader, to read the incoming data stream. The script must return a string containing the next message, or a null/empty string to indicate end of input. For additional information, see JavaScript Batch Script .

Batch Processing

Batch Processing allows a channel to receive a single message, but split it into multiple messages that each get processed through the channel. When using this along with a source connector that supports streaming (File Reader, TCP Listener in MLLP mode), batch processing has the added benefit of not having to read the entire file into memory all at once, but instead only one message at a time. For example with the File Reader and batch processing you can read in files that are gigabytes in size, without causing any memory issues for your NextGen Connect Integration Engine server. Data types that support batch processing will have a **Batch Properties** section in the source inbound data type properties.

To enable batch processing, set **Process Batch** to **Yes** in the [Source Settings](#). To change how an incoming message is split into multiple messages, look at the **Batch Properties** section of the source inbound data type you're using. For example, the [Delimited Text Data Type](#) has options to split by record delimiter, a specific hard-coded delimiter, a grouping column, or a custom JavaScript script.

When batch messages are processing through a channel, some extra [Source Map variables](#) are available:

Key	Description
batchId	This is a unique ID that identifies the current overall batch file/message. It will be equal to the message ID of the first message that gets processed for the current batch. For example if you read in a single file and your batch processor splits it into 5 messages, all 5 of those messages will have the same batchId value in the source map.
batchSequenceId	This is an integer that starts at 1 and increments for each subsequent message in the batch.
batchComplete	This is a boolean value, equal to either true or false. It indicates whether the currently processing message is the last one in the current batch.

JavaScript Batch Script

The JavaScript batch adapter is an option common to almost all data types. When Process Batch is enabled on the [Source Settings](#), you can set this script from the [Set Data Types Dialog](#) to programmatically decide how to split the incoming data into multiple messages.

Within the batch script you have access to a variable called "reader", which is a Java [BufferedReader](#) object. Use this variable to consume from the underlying character stream and return a String for each message you want to process through the channel. When you decide that no more messages should be processed, or you reach the end of the stream, return null or an empty string.

Example 1

This script will simply send a message for every line in the input.

```
return reader.readLine();
```

Example 2

This script will split the input into multiple messages by assuming that each new message starts with a line break and the characters "MSH". Note that this is already a feature supported by the [HL7 v2.x Data Type](#), but is shown here to illustrate how the batch script can be used.

```
var message = new java.lang.StringBuilder();

var line;
while ((line = reader.readLine()) != null) {
    message.append(line).append('\r');

    // Mark the stream for 3 characters while we check for MSH
    reader.mark(3);
    // Check for the code points corresponding to MSH
    if (reader.read() == 77 && reader.read() == 83 && reader.read() == 72) {
        reader.reset();
        break;
    }
    reader.reset();
}

return message.toString();
```

Source Connectors

This section refers to the actual connector-specific settings for the source connector. The section is labeled according to the connector type, e.g. "HTTP Listener", "JavaScript Reader". For more information on connectors in general, go here: [About Channels and Connectors](#)

Here is a list of source connectors supported by NextGen Connect Integration Engine:

- Channel Reader
- DICOM Listener
- Database Reader
- File Reader
- HTTP Listener
- JMS Listener
- JavaScript Reader
- TCP Listener
- Web Service Listener

Additional source connectors are made available as [commercial extensions](#):

- Email Reader
- Serial Connector

Channel Reader

The **Channel Reader** is a connector that does nothing but wait for other channels / processes to send it messages. This can be useful if you split your message workflow into multiple channels, where one sends to another. Note that you **do not** need to use a Channel Reader source for the channel to be able to receive messages from other internal channels / processes. Channels using other source connector types can still receive messages from a [Channel Writer](#) or from a "router.routeMessage" call.

Supported property groups:

- [Source Settings](#)



Source Map Variables

If this connector receives a message from a [Channel Writer](#), the following [source map](#) variables will be available:

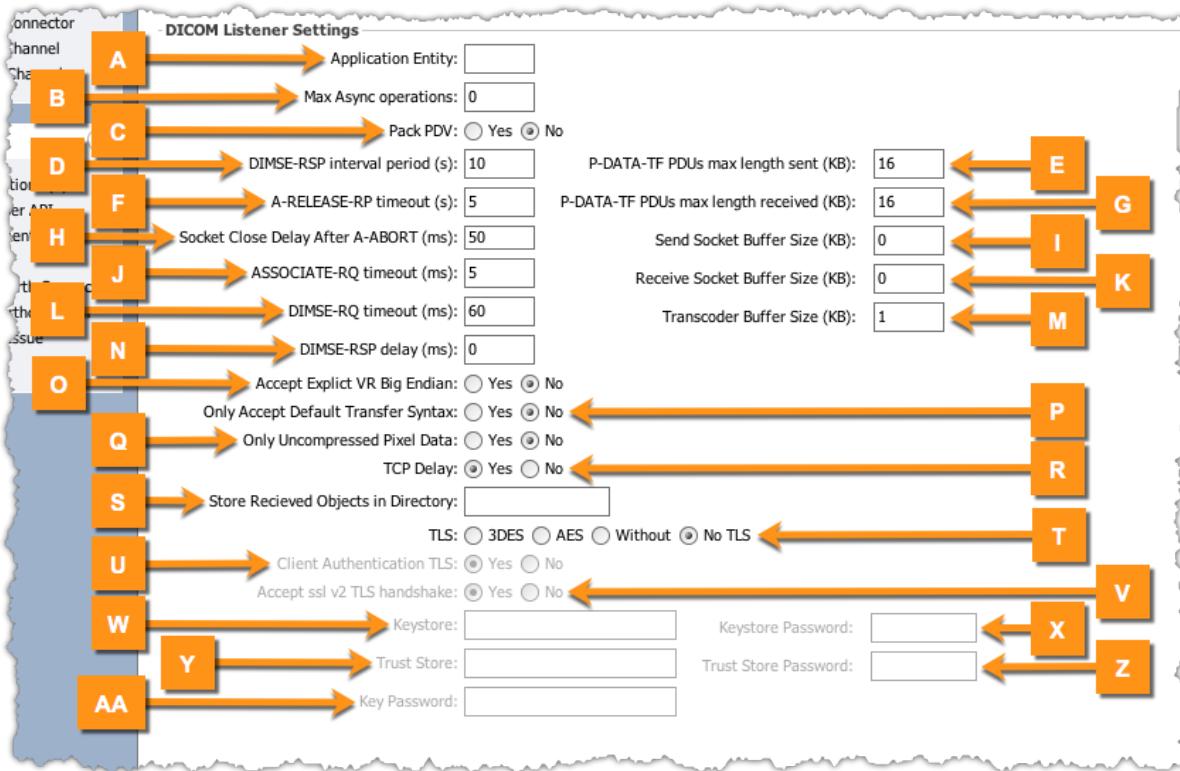
Key	Description
sourceChannelId	The unique ID of the channel that dispatched a message to the current channel.
sourceMessageId	The ID of the message from which the current message dispatch originated.
sourceChannelIds	If there are more than two channels in a Channel Writer -> Reader chain, this will be a List containing the IDs of all channels in the chain.
sourceMessageIds	If there are more than two channels in a Channel Writer -> Reader chain, this will be a List containing the IDs of all messages in the chain.

DICOM Listener

This source connector works in conjunction with the [DICOM Attachment Handler](#) and the [DICOM Data Type](#) to allow NextGen Connect Integration Engine to receive and consume DICOM data. This connector supports the C-STORE operation as a Service Class Provider (SCP). Additional options are available with the [SSL Manager](#) extension.

Supported property groups:

- Listener Settings
- Source Settings



Item	Name	Default Value	Description
A	Application Entity		If specified, only requests with a matching Application Entity title will be accepted.
B	Max Async operations	0	Maximum number of outstanding operations performed asynchronously, unlimited by default.
C	Pack PDV	No	Send only one PDV in one P-Data-TF PDU, pack command and data PDF in one P-DATA-TF PDU by default.
D	DIMSE-RSP interval period (s)	10	Period to check for outstanding DIMSE-RSP, 10 seconds by default.
E	P-DATA-TF PDUs max length sent (KB)	16	Maximal length in KB of sent P-DATA-TF PDUs, 16 KB by default.

F	A-RELEASE-RP timeout (s)	5	Timeout for receiving A-RELEASE-RP, 5 seconds by default.
G	P-DATA-TF PDUs max length received (KB)	16	Maximal length in KB of received P-DATA-TF PDUs, 16 KB by default.
H	Socket Close Delay After A-ABORT (ms)	50	Delay in ms for Socket close after sending A-ABORT, 50 ms by default.
I	Send Socket Buffer Size (KB)	0	Send socket buffer size in KB
J	ASSOCIATE-RQ timeout (ms)	5	Timeout in ms for receiving ASSOCIATE-RQ, 5 seconds by default.
K	Receive Socket Buffer Size (KB)	0	Receive socket buffer size in KB
L	DIMSE-RQ timeout (ms)	60	Timeout in ms for receiving DIMSE-RQ, 60 ms by default.
M	Transcoder Buffer Size (KB)	1	Minimal buffer size to write received object to file, 1 KB by default.
N	DIMSE-RSP delay (ms)	0	Delay in ms for DIMSE-RSP; useful for testing asynchronous mode.
O	Accept Explicit VR Big Endian	No	Accept explicit value representation Big Endian transfer syntax.
P	Only Accept Default Transfer Syntax	No	Accept only the default transfer syntax.
Q	Only Uncompressed Pixel Data	No	Accept only transfer syntax with uncompressed pixel data.
R	TCP Delay	Yes	Set TCP_NODELAY socket option to false, true by default.
S	Store Received Objects in Directory		Store received objects into files in specified directory.
T	TLS	No TLS	<p>Determines whether to receive data over an implicit SSL/TLS socket. The following options are available:</p> <ul style="list-style-type: none"> • 3DES: TLS will be used, with the <code>SSL_RSA_WITH_3DES_EDE_CBC_SHA</code> cipher suite. • AES: TLS will be used, with the following cipher suites: <ul style="list-style-type: none"> • <code>TLS_RSA_WITH_AES_128_CBC_SHA</code> • <code>SSL_RSA_WITH_3DES_EDE_CBC_SHA</code> • Without: TLS will be used without symmetric encryption, with the cipher suite <code>SSL_RSA_WITH_NULL_SHA</code>. DICOM messages will be received unencrypted. • No TLS: No TLS will be used. DICOM messages will be received over a regular unencrypted socket.
U	Client Authentication TLS	Yes	Enable client authentication for TLS. Only applicable if the TLS option is not set to No TLS.

V	Accept ssl v2 TLS handshake	Yes	Enable acceptance of the SSLv2Hello protocol in the TLS handshake.
W	Keystore		File path or URL of P12 or JKS keystore to use for the local server certificate keypair.
X	Keystore Password		Password for the configured Keystore.
Y	Trust Store		File path or URL of JKS truststore, used to trust remote client certificates.
Z	Trust Store Password		Password for the configured Truststore.
AA	Key Password		Password for accessing the key in the Keystore.

Source Map Variables

Key	Description
localApplicationEntityTitle	The Application Entity Title of the local Service Class Provider (SCP).
remoteApplicationEntityTitle	The Application Entity Title of the remote Service Class User (SCU).
localAddress	The IP address that the TCP socket is locally bound to.
localPort	The port that that TCP socket is locally bound to.
remoteAddress	The IP address of the remote connecting client.
remotePort	The TCP port of the remote connecting client.
associateACProtocolVersion	The associate protocol version of the local SCP.
associateACImplClassUID	The associate implementation class unique identifier of the local SCP.
associateACImplVersionName	The associate implementation version name of the local SCP.
associateACApplicationContext	The associate application context of the local SCP.
associateACPresentationContexts	A map containing all supported presentation contexts of the local SCP.
associateRQProtocolVersion	The associate protocol version of the remote SCU.
associateRQImplClassUID	The associate implementation class unique identifier of the remote SCU.
associateRQImplVersionName	The associate implementation version name of the remote SCU.
associateRQApplicationContext	The associate application context of the remote SCU.
associateRQPresentationContexts	A map containing all supported presentation contexts of the remote SCU.
username	The username presented by the remote SCU, if available.
passcode	The passcode presented by the remote SCU, if available.
userIdentityType	<p>The type of user identity presented by the remote SCU, if available. Will be one of the following values:</p> <ul style="list-style-type: none"> • USERNAME • USERNAME_PASSCODE

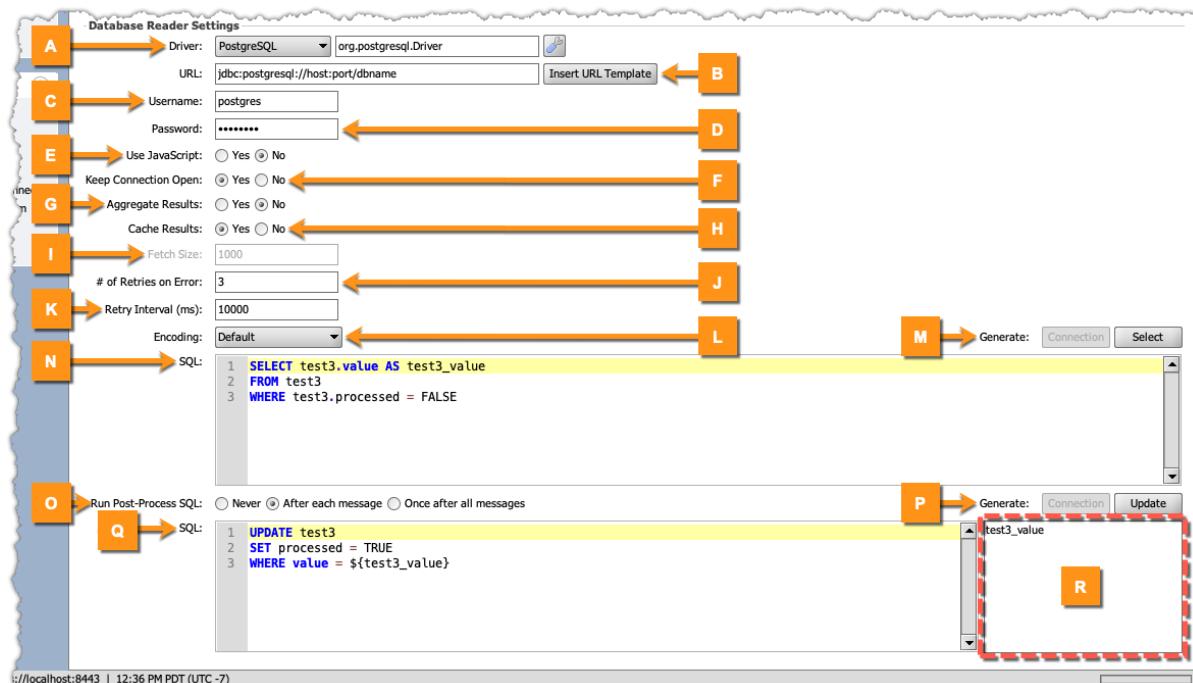
- KERBEROS
- SAML

Database Reader

This source connector connects to an external database, performs a query, and reads selected rows into messages that get dispatched to the channel. This can be done using a SQL statement, or by using JavaScript mode to perform the query manually. The database connection will automatically be kept open across multiple polling windows, unless otherwise specified. This connector also supports a Post-Process section where an update statement can be performed after each row is read in, for example to set a processed flag in the source table. The values selected from the query will be automatically converted into an XML document where each column will be a separate node. That XML document is what actually gets dispatched to the channel as a message.

Supported property groups:

- Polling Settings
- Source Settings



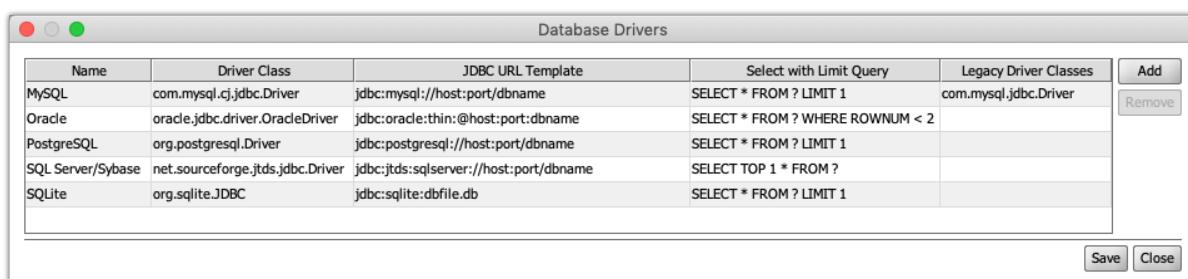
Item	Name	Default Value	Description
A	Driver		<p>Specifies the type of database driver to use to connect to the database. The following values are supported by default:</p> <ul style="list-style-type: none"> • MySQL • Oracle • PostgreSQL • SQL Server / Sybase • SQLite <p>A custom driver class name can be entered in the text field. Drivers can be added/modified by clicking the wrench icon, more information here: Editing Database Drivers</p>
B	URL		The JDBC URL to connect to the database with. This is not used when Yes for Use JavaScript is checked. However, it is used when the Generate

			Connection / Select feature is used to generate code. Use the Insert URL Template button above to populate the URL field with a starting template.
C	Username		The username to connect to the database with. This is not used when Yes for Use JavaScript is checked. However, it is used when the Generate Connection / Select feature is used to generate code.
D	Password		The password to connect to the database with. This is not used when Yes for Use JavaScript is checked. However, it is used when the Generate Connection / Select feature is used to generate code.
E	Use JavaScript	No	If enabled, the below JavaScript scripts will be used to select messages and run a post-process update. If disabled, SQL code (either standard or database-specific) may be used, and the connection will be handled automatically.
F	Keep Connection Open	Yes	Re-use the same database connection each time the select query is executed. If disabled, the connection will be closed after all selected messages have finished processing.
G	Aggregate Results	No	If enabled, all rows returned in the query will be aggregated into a single XML message. Note that all rows will be read into memory at once, so use this with caution.
H	Cache Results	Yes	Cache the entire result set in memory prior to processing messages.
I	Fetch Size	1000	The JDBC ResultSet fetch size to be used when fetching results from the current cursor position.
J	# of Retries on Error	3	The number of times to retry executing the statement or script if an error occurs.
K	Retry Interval	10000	The amount of time that should elapse between retry attempts.
L	Encoding	Default	Select the character set encoding used to convert binary data into message strings, or select Default to use the default character set encoding for the JVM NextGen Connect Integration Engine is running on.
M	Generate		<ul style="list-style-type: none"> • Connection: This button is enabled when Use JavaScript is enabled. When clicked, it inserts boilerplate Connection construction code into the JavaScript pane at the current caret position. • Select: Opens a window to assist in building a select query to select records from the database specified in the URL above.
N	SQL / JavaScript		The actual SQL or JavaScript code to execute for each polling window. When JavaScript mode is used, the return value of the script is expected to be a ResultSet or a List<Map<String, Object>> (a list of maps, where each entry in each map has a String key and any object value).
O	Run Post-Process SQL / JavaScript	Never	<p>Determines whether the post-process update script is active, and if so whether to execute it after each message or just once after all messages in the ResultSet have completed.</p> <p>If Aggregate Results is disabled:</p> <ul style="list-style-type: none"> • Never: Do not run the post-process statement/script. • After each message: Run the post-process statement/script after each message finishes processing.

		<ul style="list-style-type: none"> Once after all messages: Run the post-process statement/script only after all messages have finished processing. <p>If Aggregate Results is enabled:</p> <ul style="list-style-type: none"> Never: Do not run the post-process statement/script. For each row: Run the post-process statement/script for each row in the result set. Once for all rows: Run the post-process statement/script only once. If JavaScript mode is used, a List of Maps representing all rows in the result set will be available as the variable "results".
P	Generate (post-process)	<ul style="list-style-type: none"> Connection: This button is enabled when Use JavaScript is enabled and a post-process script is being used. When clicked, it inserts boilerplate Connection construction code into the JavaScript pane at the current caret position. Update: Opens a window to assist in building an update statement to update records in the database specified in the URL above. Only enabled if a post-process statement/script is being used.
Q	SQL / JavaScript (post-process)	The actual SQL or JavaScript code to execute after each row/message or after all rows/messages have completed.
R	Result Map	When using the After each message / For each row post-process option, values originally selected using the query above will be available in the SQL or JavaScript context. Drag the entries from this section into the post-process script to use them in your update statement. For example if you selected a unique ID column in your initial query, you may want to use that same value to update the table and set a processed flag.

Editing Database Drivers

Click on the wrench icon next to the **Driver** text field. A new dialog will appear:



From this dialog you can add/modify/remove the default drivers that will show up in the drop-down menu in Database Reader/Writer connectors. The following properties are used:

Column	Required	Description
Name	Yes	The name of the driver entry. This will appear in the drop-down menu for the Database Reader/Writer connectors.
Driver	Yes	The fully-qualified Java class name for the JDBC driver.

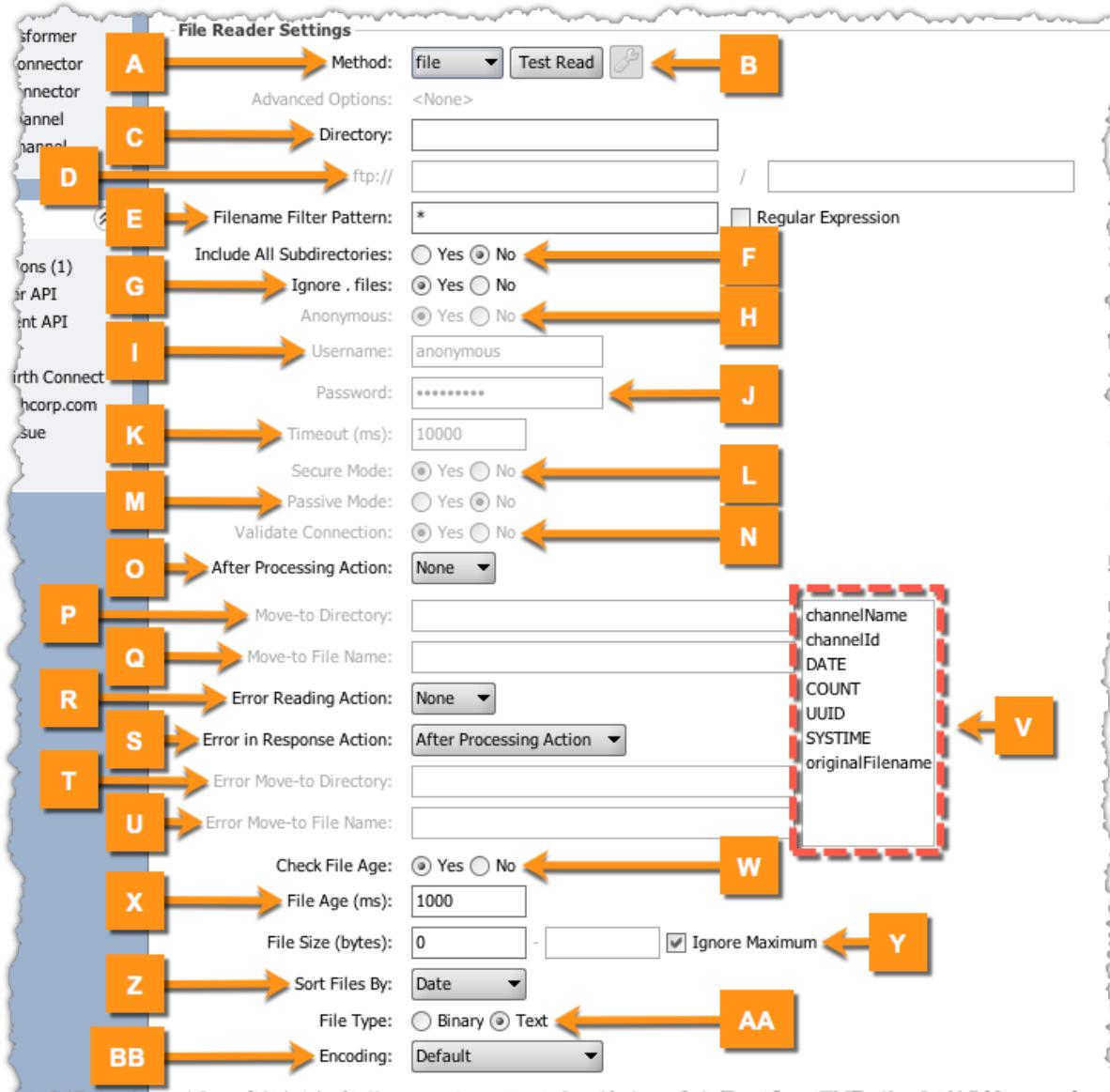
Class		
JDBC URL Template	Yes	The template for the JDBC connection URL that can be auto-populated from the Database Reader/Writer settings.
Select with Limit Query	No	A select query (with limit 1) that can be used to retrieve column metadata. If empty the driver-specific generic query will be used, which could be slow.
Legacy Driver Classes	No	A comma-separated list of alternate or legacy JDBC driver class names. Any Database Reader/Writer connector using one of these driver classes will have the corresponding entry selected in the Driver drop-down menu. The driver will be updated to the primary value if you select the entry from the drop-down menu again.

File Reader

This source connector reads files from a local or remote directory on a specified interval/time schedule. Several protocols are supported, including regular local file mode, FTP, SFTP, SMB, WebDAV, and Amazon S3. Files may be read in and converted to Base64, or converted to a message string using a specific character set encoding. After reading in files, the connector has options to either delete the original files, rename them, or move them to a separate directory. Additional options (like FTPS) are available with the [SSL Manager](#) extension.

Supported property groups:

- [Polling Settings](#)
- [Source Settings](#)



Item	Name	Default Value	Description

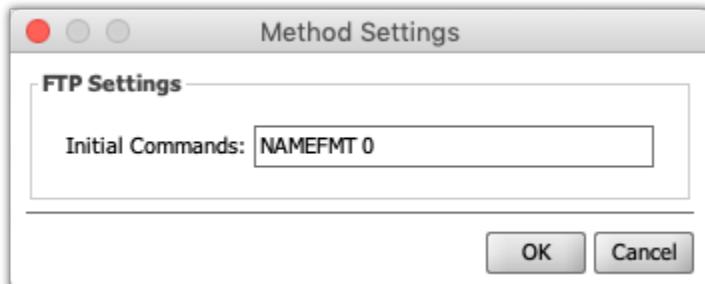
A	Method	file	The basic method used to access files to be read in. Options include File (local filesystem or NFS / mapped share), FTP , SFTP , SMB , WebDAV , or Amazon S3 . Once all necessary connection/directory information has been filled in before, use the Test Read button to attempt to actually connect and test the ability to read from the directory.
B	Advanced Options		If the file method supports advanced options, this button will be enabled. Any advanced options set will be summarized in the Advanced Options label below this.
C	Directory		Only applicable to the File method. The directory (folder) in which the files to be read can be found.
D	URL		Applicable to all methods except File . The domain name or IP address of the host (computer) on which the files to be read can be found. If this setting is enabled, the second text field specifies the directory (folder) to read from. When using the Amazon S3 method, the first text field will be the bucket name, and the second text field can be used for a directory prefix.
E	Filename Filter Pattern	*	Files with names that do not match this pattern will be ignored. If Regular Expression is disabled, regular wildcard (*) matching is supported.
F	Include All Subdirectories	No	Select Yes to traverse directories recursively and search for files in each one.
G	Ignore . files	Yes	Select Yes to ignore all files starting with a period.
H	Anonymous	Yes	Only applicable to the FTP / WebDAV / Amazon S3 methods. If enabled, connects to the remote server anonymously instead of using a username and password.
I	Username	anonymous	Applicable to all methods except File . The username used to connect to the remote server with. When using the Amazon S3 mode, this will be your AWS Access Key ID.
J	Password		Applicable to all methods except File . The password used to connect to the remote server with. When using the Amazon S3 mode, this will be your AWS Secret Access Key.
K	Timeout (ms)	10000	Applicable to the FTP / SFTP / SMB / Amazon S3 methods. The socket timeout (in ms) to use when connecting to the remote server.
L	Secure Mode	Yes	Only applicable to the WebDAV method. If enabled, HTTPS will be used instead of HTTP.
M	Passive Mode	Yes	Only applicable to the FTP method. If enabled, the server decides what port the client should connect to for the data channel. Passive mode sometimes allows a connection through a firewall that normal mode does not, because the client is initiating the data connection rather than the server.
N	Validate Connection	Yes	Only applicable to the FTP method. If enabled, the connection will be tested for validity before each operation.
O	After Processing Action	None	Select Move to move and/or rename the file after successful processing. Select Delete to delete the file after successful processing.

P	Move-to Directory		If successfully processed files should be moved to a different directory (folder), enter that directory here. The directory name specified may include template substitutions from the list to the right. If this field is left empty, successfully processed files will not be moved to a different directory.
Q	Move-to File Name		If successfully processed files should be renamed, enter the new name here. The filename specified may include template substitutions from the list to the right. If this field is left empty, successfully processed files will not be renamed.
R	Error Reading Action	None	Select Move to move and/or rename files that have failed to be read in (for example, if an out-of-memory error occurs, or the network connection drops). Select Delete to delete files that have failed to be read in.
S	Error in Response Action	After Processing Action	Select Move to move and/or rename the file if an <i>ERROR</i> response is returned. This action is triggered when the Response selected in the Source Settings has a status of <i>ERROR</i> . If After Processing Action is selected, the After Processing Action will apply. This action is only available if Process Batch is disabled in the Source Settings .
T	Error Move-to Directory		If files which cause processing errors should be moved to a different directory (folder), enter that directory here. This action is triggered when the Response selected in the Source Settings has a status of <i>ERROR</i> . The directory name specified may include template substitutions from the list to the right. If this field is left empty, files which cause processing errors will not be moved to a different directory.
U	Error Move-to File Name		If files which cause processing errors should be renamed, enter that directory here. This action is triggered when the Response selected in the Source Settings has a status of <i>ERROR</i> . The filename specified may include template substitutions from the list to the right. If this field is left empty, files which cause processing errors will not be renamed.
V	Move-to Variables		<p>The variables listed here can be dragged-and-dropped into the Move-to fields to the left.</p> <ul style="list-style-type: none"> • channelName: The name of the current channel. • channelId: The unique ID of the current channel. • DATE: The current date, formatted as a human-readable string. • COUNT: A numeric count that increases for each file read in, from the point when the channel was last deployed. • UUID: An auto-generated universally unique identifier. • SYSTIME: The current epoch time in milliseconds. • originalFilename: The name of the file that was read in. Use this to easily add on an extra file extension to the original name.
W	Check File Age	Yes	Select Yes to skip files that are created within the specified age below.
X	File Age (ms)	1000	If Check File Age is enabled, only the files with creation dates older than the specified value in milliseconds will be processed.
Y	File Size (bytes)	0, Ignore Maximum	The minimum and maximum size (in bytes) of files to be accepted. If Ignore Maximum is checked, the file size will only be bound by the minimum value.
Z	Sort Files By	Date	Selects the order in which files should be processed, if there are multiple files available. Files can be processed by Date (oldest last-modification date first), Size (smallest first), or Name (a before z, etc.).
AA	File Type	Text	Select Binary if files contain binary data; the contents will be Base64

			encoded before processing. Select Text if files contain textual data; the contents will be encoded using the specified character set encoding.
BB	Encoding	Default	If Text is chosen for the File Type , select the character set encoding (ASCII , UTF-8 , etc.) to be used in reading the contents of each file.

Advanced FTP Options

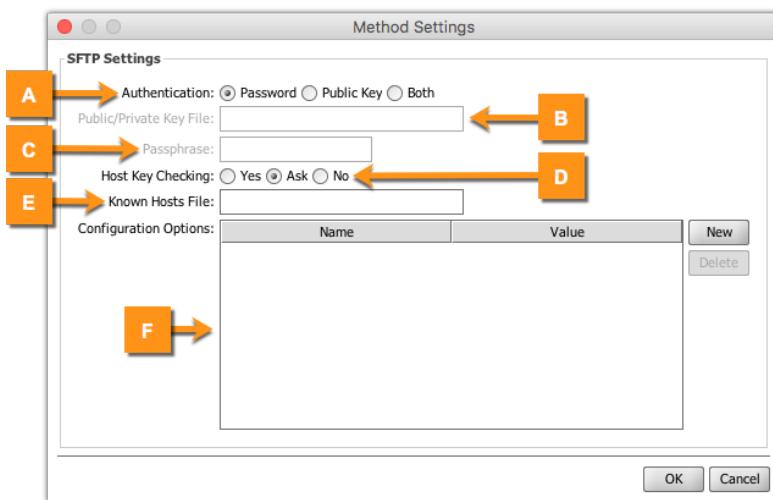
When the FTP file method is selected, these additional advanced options may be set:



Name	Default Value	Description
Initial Commands		A comma-separated list of custom commands to run upon initializing an FTP connection. For example when connecting to an AS/400 FTP server you may have to change the list format using the "NAMEFMT" command.

Advanced SFTP Options

When the SFTP file method is selected, these additional advanced options may be set:

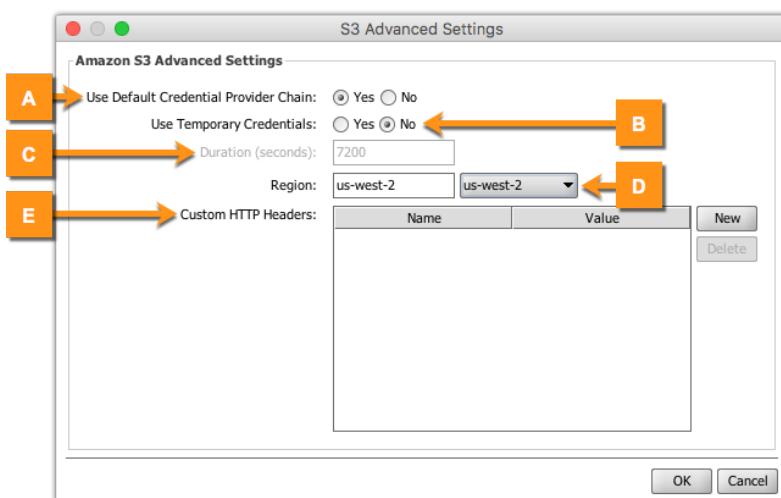


Item	Name	Default Value	Description
A	Authentication	Password	Determines how to authenticate to the SFTP server. Options include Password, Public Key, or Both.

B	Public /Private Key Files		The absolute file path of the public/private keypair used to gain access to the remote server.
C	Passphrase		The passphrase associated with the public/private keypair.
D	Host Key Checking	Ask	Select Yes to validate the server's host key within the provided Known Hosts file (or the system default). Otherwise the host key will always be automatically trusted.
E	Known Hosts File		The path to the local Known Hosts file used to trust remote host keys.
F	Configuration Options		Custom JSch configuration options used when connecting to the remote server. For example, these can be used to enable Kerberos authentication.

Advanced Amazon S3 Options

When the Amazon S3 file method is selected, these additional advanced options may be set:



Item	Name	Default Value	Description
A	Use Default Credential Provider Chain	Yes	<p>If enabled and no explicit credentials are provided, the default provider chain looks for credentials in this order:</p> <ul style="list-style-type: none"> • Environment variables: AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY • Java system properties: aws.accessKeyId and aws.secretKey • Default credentials profile file: Typically located at <code>~/.aws/credentials</code> (location can vary per platform) • ECS container credentials: Loaded from an Amazon ECS environment variable. • Instance profile credentials: Loaded from the EC2 metadata service. <p>Note that if your File Reader has Anonymous enabled, this option will not be enabled.</p>
B	Use Temporary Credentials	No	If enabled, the given credentials will be used to request a set of <i>temporary</i> credentials from the Amazon Security Token Service (STS). Those temporary credentials will then be used for all S3 operations.

C	Duration (seconds)	7200	The duration that the temporary credentials are valid. Must be between 900 seconds (15 minutes) and 129,600 seconds (36 hours).
D	Region	us-west-2	The AWS region that your S3 bucket is located in. Select a specific region from the drop-down menu, or enter one into the text field. You can also use Velocity Variable Replacement here.
E	Custom HTTP Headers		<p>These headers will be used on any S3 PUT operation. They are not used for GET operations.</p> <p>To add user-defined metadata tags to the S3 object, include a custom header that starts with "x-amz-meta-".</p> <p>For more information, check out the official Amazon S3 documentation.</p>

Source Map Variables

Key	Description
originalFilename	The name of the file that was read in.
fileDirectory	The absolute path of the directory in which the file resides.
fileSize	The size of the file in bytes.
fileLastModified	The last modified date of the file, as an epoch time in milliseconds.
pollId	A unique nanosecond timestamp that uniquely identifies the current polling window. If your File Reader polls 5 files, the messages for each file will have the same pollId.
pollSequenceId	An integer that starts at 1 and increments for every subsequent file in the current polling window. If your File Reader polls 5 files, the message(s) for the first file will have a pollSequenceId of 1, the message(s) for the second file will have a pollSequenceId of 2, and so on.
pollComplete	This is only present for the last file in the current polling window. The value of this entry is always equal to true. Use this to determine programmatically whether you are currently working with the <i>/last</i> file in a poll. Note that if you have Batch Processing enabled, you will want to look at both the pollComplete and batchComplete variables to determine whether the current message is truly the "last" one. If both are true, then you know that you're on the last file in the polling window, and also on the last message in that file.

Only Applicable to the Amazon S3 mode

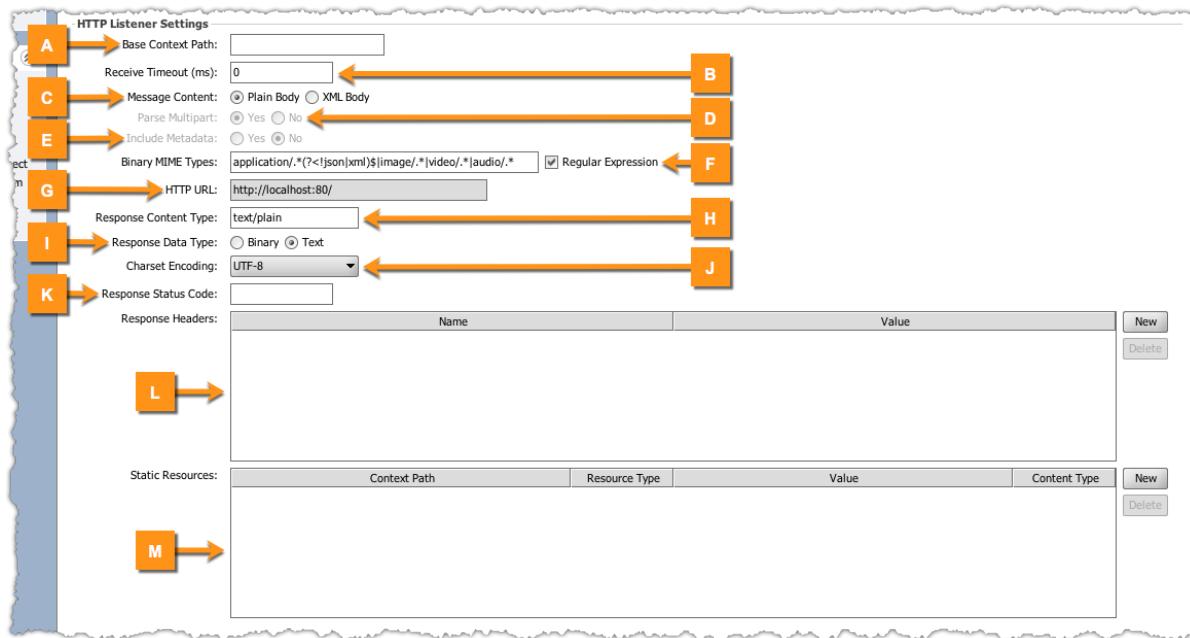
s3BucketName	The name of the S3 bucket that the file belongs to.
s3ETag	The hex encoded 128-bit MD5 hash of the file contents as computed by Amazon S3.
s3Key	The key under which the file is stored in S3.
s3Owner	The owner of the S3 object, if present.
s3StorageClass	The storage class used for this file in S3 (e.g. STANDARD, STANDARD_IA).
s3Metadata	A MessageHeaders object representing the map of metadata/headers for the S3 object.

HTTP Listener

This source connector acts as an HTTP server, listening for requests from one or more remote clients. The messages sent to the channel can be just the raw payload, or an XML document allowing multipart payloads to be parsed in a consistent and easy-to-use way. The HTTP payload can be either Base64 encoded or converted using a charset, depending on the Content-Type. Responses that go back to each client can be fully configured, including custom response headers. Finally, static resources or directories can be automatically hosted, to allow the connector to act as a simple web server that serves specific content. Additional options are available with the [SSL Manager](#) extension.

Supported property groups:

- Listener Settings
- Source Settings
- HTTP Authentication Settings



Item	Name	Default Value	Description
A	Base Context Path		The context path for the HTTP Listener URL. Note that if this is specified, any requests made at this base context path must have a trailing slash in the request URI.
B	Receive Timeout (ms)	30000	The maximum idle time in milliseconds for a connection.
C	Message Content	Plain Body	<ul style="list-style-type: none"> • Plain Body: The request body will be sent to the channel as a raw string. • XML Body: The request body will be sent to the channel as serialized XML.
D	Parse Multipart	Yes	Select Yes to automatically parse multipart requests into separate XML nodes. Select No to always keep the request body as a single XML node.

E	Include Metadata	No	Select Yes to include request metadata (method, context path, headers, query parameters) in the XML content. Note that regardless of this setting, the same metadata is always available in the source map.
F	Binary MIME Types	application/* (*<! json xml)\$ image/* video /* audio/*	When a response comes in with a Content-Type header that matches one of these entries, the content will be encoded into a Base64 string. If Regular Expression is unchecked, specify multiple entries with commas. Otherwise, enter a valid regular expression to match MIME types against.
G	HTTP URL	<auto-generated>	Displays the generated HTTP URL for the HTTP Listener. This is not an actual configurable setting, but is instead displayed for copy/paste convenience. Note that the host in the URL will be the same as the host you used to connect to the Administrator. The actual host that connecting clients use may be different due to differing networking environments.
H	Response Content Type	text/plain	The MIME type to be used for the response.
I	Response Data Type	Text	If Binary is selected, responses will be decoded from Base64 into raw byte streams. If Text is selected, responses will be encoded with the specified character set encoding.
J	Charset Encoding	UTF-8	Select the character set encoding to be used for the response to the sending system. Set to Default to assume the default character set encoding for the JVM NextGen Connect Integration Engine is running on.
K	Response Status Code		Enter the status code for the HTTP response. If this field is left blank, a default status code of 200 will be returned for a successful message, and 500 will be returned for an errored message. If a Response is chosen in the Source Settings , the status of that response will be used to determine a successful or errored response.
L	Response Headers		Enter custom headers to send back to the originating client.
M	Static Resources		Values in this table are automatically sent back to any request with the matching context path. There are three resource types: <ul style="list-style-type: none"> • File: The value field specifies the path of the file to return. • Directory: Any file within the directory given by the value field may be requested, but subdirectories are not included. • Custom: The value field itself is returned as the response.

Source Map Variables

Key	Description
remoteAddress	The IP address of the remote connecting client.
remotePort	The TCP port of the remote connecting client.
localAddress	The IP address that the TCP socket is locally bound to.

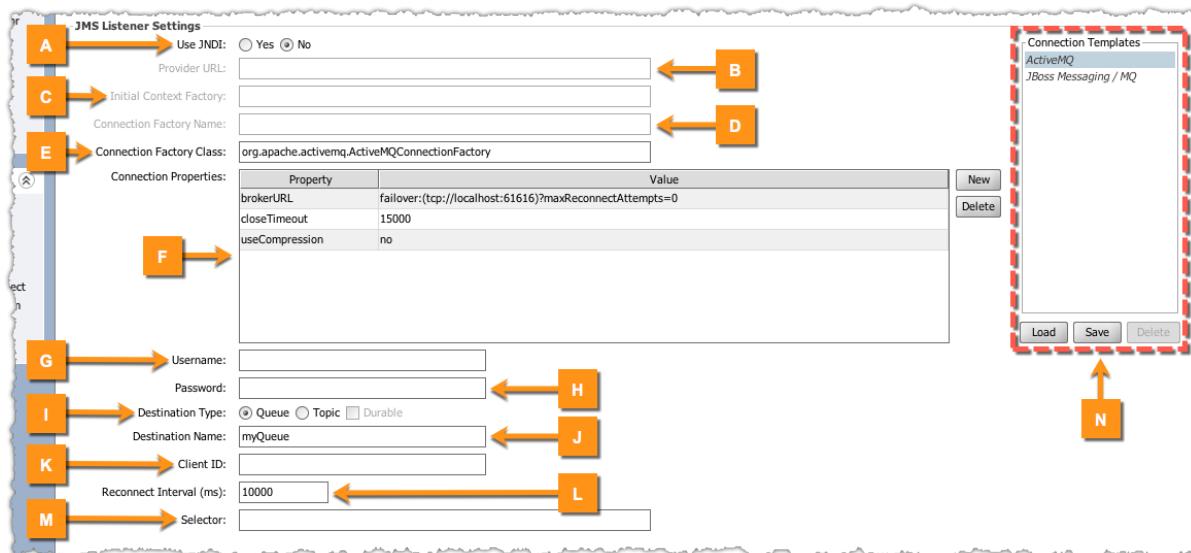
localPort	The port that that TCP socket is locally bound to.
method	The HTTP method used for the incoming request.
url	The URL of the client request, excluding any query parameters.
uri	The HTTP URI requested, including the context path and all query parameters.
protocol	The HTTP protocol version used in the request.
query	The query parameter portion of the request URI.
contextPath	The context path requested, without any query parameters.
headers	A MessageHeaders object containing all headers sent in the incoming request. Look in the User API for more information.
parameters	A MessageParameters object containing all headers sent in the incoming request. This will either be query parameters, or POST parameters if the application/x-www-form-urlencoded Content-Type is used.

JMS Listener

This source connector connects to an external JMS provider and reads messages from a queue or topic. It supports both JNDI and specifying a specific connection factory, as well as fine-tuned queries through a configurable selector. Once this connector is started, it will attempt to keep a persistent open connection to the JMS provider. If for any reason the connection is dropped, the connector will automatically reconnect without any intervention needed. The properties view also includes a mechanism to save configuration templates for common provider types, so that creating a new JMS Listener is as quick and easy as possible.

Supported property groups:

- Source Settings



Item	Name	Default Value	Description
A	Use JNDI	No	Select Yes to use JNDI to look up a connection factory to connect to the queue or topic. Select No to specify a connection factory class without using JNDI.
B	Provider URL		If using JNDI, enter the URL of the JNDI provider here.
C	Initial Context Factory		If using JNDI, enter the fully-qualified Java class name of the JNDI Initial Context Factory class here.
D	Connection Factory Name		If using JNDI, enter the JNDI name of the connection factory here.
E	Connection Factory Class		If using the generic JMS provider and not using JNDI, enter the fully-qualified Java class name of the JMS connection factory here.
F	Connection Properties		This table allows you to enter custom connection factory settings. The Property column is the key, while the Value column is the actual value for the setting. The specific properties used here will vary depending on what connection factory class / provider you're using.

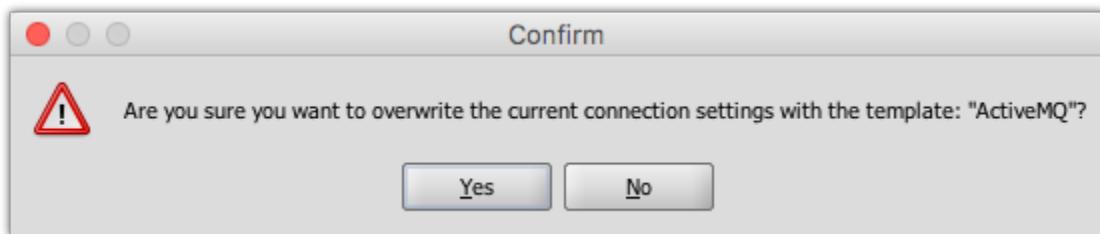
G	Username		The username for accessing the queue or topic.
H	Password		The password for accessing the queue or topic.
I	Destination Type	Queue	Specify whether the destination is a queue or topic. When connecting to a topic, you can check the Durable checkbox so that all messages published to the topic will be read, regardless of whether or not a connection to the broker is active. If unchecked, only messages published while a connection is active will be read.
J	Destination Name		The name of the queue or topic.
K	Client ID		The JMS client ID to use when connecting to the JMS broker.
L	Reconnect Interval (ms)	10000	The number of milliseconds between reconnect attempts in the case that a connection error occurs.
M	Selector		Enter a selector expression to select specific messages from the queue/topic. Leave blank to read all messages.
N	Connection Templates		This section allows you to save the current state of your JMS Listener properties into a template , which may then be restored later if you make changes, or may also be applied to other JMS Listener connectors. For additional information, see JMS Connection Templates .

JMS Connection Templates

This section allows you to save the current state of your JMS Listener properties into a **template**, which may then be restored later if you make changes, or may also be applied to other JMS Listener connectors. Custom templates can be updated and deleted.

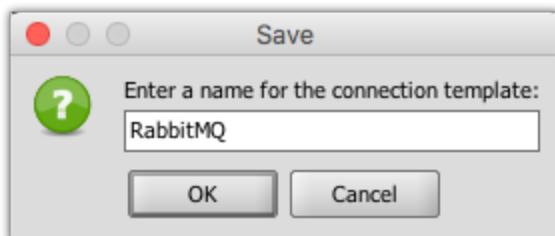
Loading Templates

Click a template in the Connection Templates list, then click the **Load** button. You will be prompted to overwrite your current JMS Listener settings:

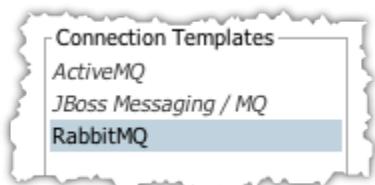


Creating New Templates

Configure your JMS Listener settings to the state you want to save, then click the **Save** button in the Connection Templates section. You will be prompted to give the template a name:

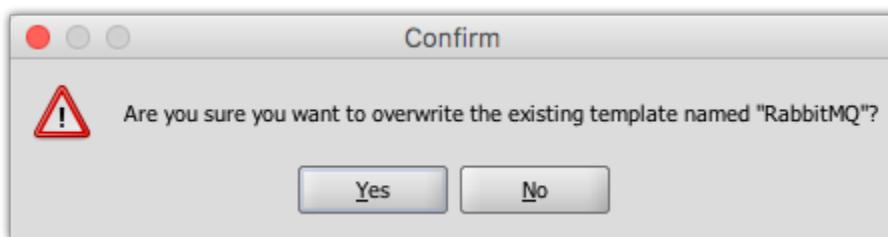


After clicking **OK**, the new template will appear in the Connection Templates list:



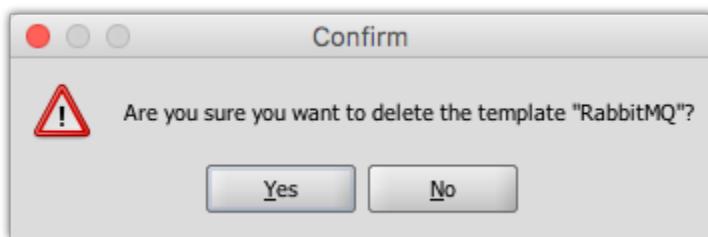
Updating Templates

To update a current template, follow the directions for creating a new template, then enter the same name as the template you wish to update. You will be prompted to overwrite:



Deleting Templates

Just click a template in the Connection Templates list, then click the **Delete** button to delete a template. You will be prompted to confirm the action:



i Updating / deleting an existing template does not affect any connectors currently using that template.

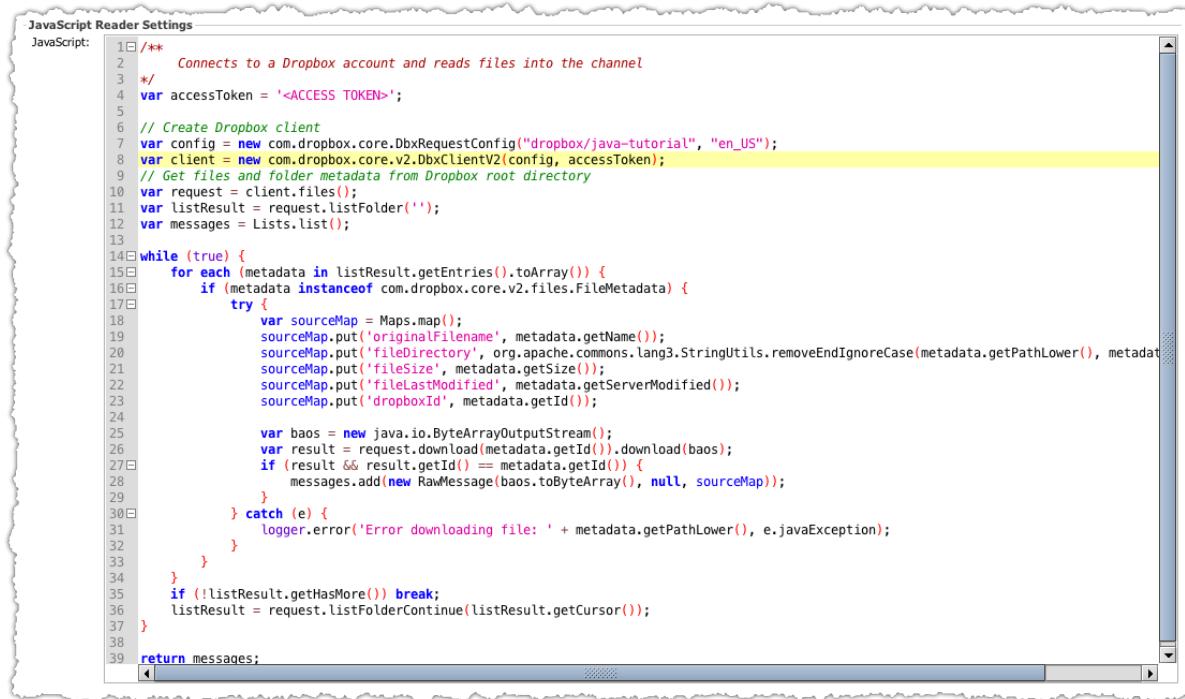
i For convenience, the JMS Listener comes with two reserved templates, "ActiveMQ" and "JBoss Messaging / MQ". These cannot be updated or deleted, however you can load the template, update the configuration as needed, and then save it as a new template.

JavaScript Reader

This source connector executes a custom user-defined JavaScript script on a specified schedule. This can be used in a wide variety of ways, such as calling out to external Java libraries or invoking a local OS shell script. You can return a message (or list of messages) to dispatch to the channel, or simply use the script as a scheduled job that doesn't necessarily produce messages. For example, you can use tools like [ChannelUtil](#) to programmatically start /stop/deploy channels from within the script.

Supported property groups:

- [Polling Settings](#)
- [Source Settings](#)



```

JavaScript Reader Settings
JavaScript:
1 /**
2  * Connects to a Dropbox account and reads files into the channel
3 */
4 var accessToken = '<ACCESS TOKEN>';
5
6 // Create Dropbox client
7 var config = new com.dropbox.core.DbxRequestConfig("dropbox/java-tutorial", "en_US");
8 var client = new com.dropbox.core.v2.DbxClientV2(config, accessToken);
9 // Get files and folder metadata from Dropbox root directory
10 var request = client.request();
11 var listResult = request.listFolder('');
12 var messages = Lists.list();
13
14 while (true) {
15     for each (metadata in listResult.getEntries().toArray()) {
16         if (metadata instanceof com.dropbox.core.v2.files.FileMetadata) {
17             try {
18                 var sourceMap = Maps.map();
19                 sourceMap.put('originalFilename', metadata.getName());
20                 sourceMap.put('fileDirectory', org.apache.commons.lang3.StringUtils.removeEndIgnoreCase(metadata.getPathLower(), metadata.getName()));
21                 sourceMap.put('fileSize', metadata.getSize());
22                 sourceMap.put('fileLastModified', metadata.getServerModified());
23                 sourceMap.put('dropboxId', metadata.getId());
24
25                 var baos = new java.io.ByteArrayOutputStream();
26                 var result = request.download(metadata.getId()).download(baos);
27                 if (result && result.getId() == metadata.getId()) {
28                     messages.add(new RawMessage(baos.toByteArray(), null, sourceMap));
29                 }
30             } catch (e) {
31                 logger.error('Error downloading file: ' + metadata.getPathLower(), e.javaException);
32             }
33         }
34     }
35     if (!listResult.getHasMore()) break;
36     listResult = request.listFolderContinue(listResult.getCursor());
37 }
38
39 return messages;

```

JavaScript Reader Return Values

If you're using the JavaScript Reader to produce messages for the channel, all you need to do is `return` those messages from the script. The following return values are accepted:

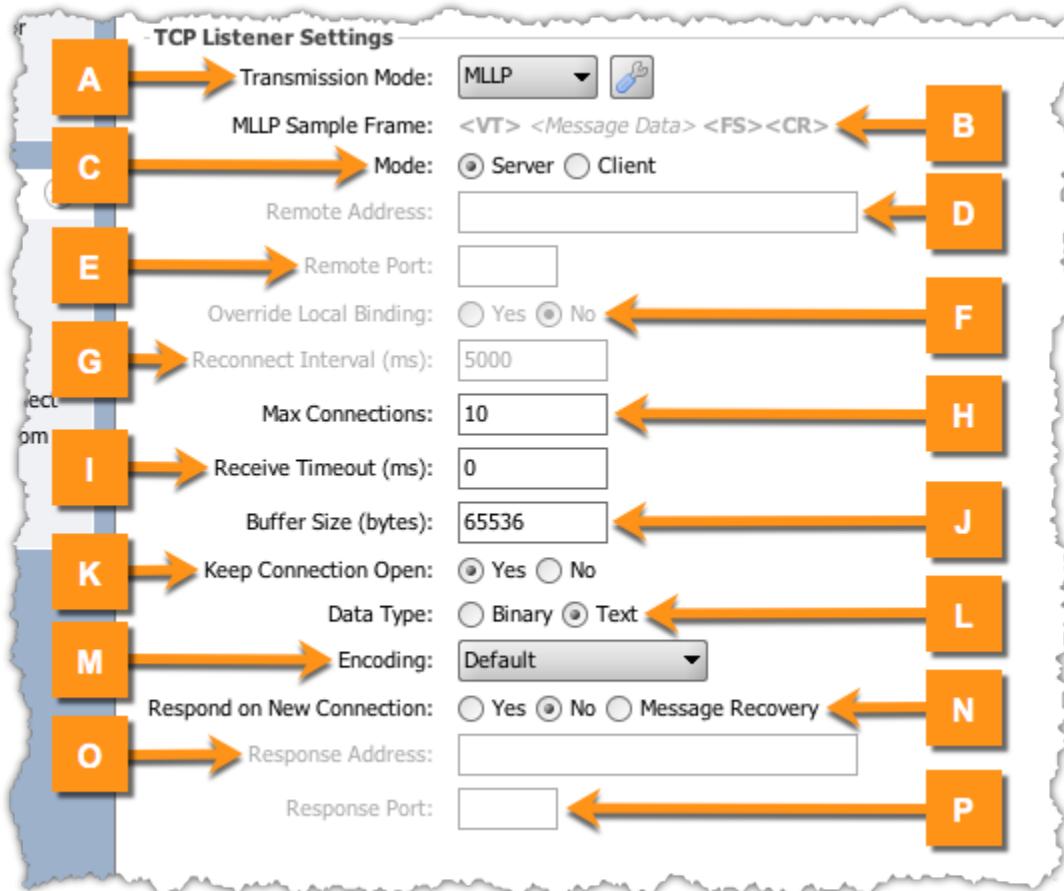
- **String:** Any non-empty string returned will be sent to the channel as a message.
- **RawMessage:** This is a special object that contains not only the string message data, but also information about which destinations to dispatch to, and any source map variables you wish to inject. For additional information, see [The User API \(Javadoc\)](#).
- **List:** If a Java List is returned, all values in the list will be sent to the channel as discrete messages. The list may contain a mix of Strings, RawMessage objects, or other objects.
- **Empty String / null / undefined:** Returning any of these (including just a "return;" statement or no return statement at all) will cause **no** messages to dispatch to the channel.
- **Any Object:** Any other object returned will be converted to a String via the `toString()` method, and that String representation will be sent to the channel as a message.

TCP Listener

This source connector listens for messages coming in over a TCP connection. It can either listen on a TCP interface /port and wait for clients to connect, or connect to an external server. There are options to decide when to keep connections open, and how many clients can connect at once. Configurable [transmission modes](#) allow you to decide how to receive inbound messages and send responses. When sending responses, you can choose to send the data back on the same connection, or on a new connection.

Supported property groups:

- [Listener Settings](#)
- [Source Settings](#)



Item	Name	Default Value	Description
A	Transmission Mode	MLLP	The transmission mode determines how to receive message data from the incoming byte stream, and how to send responses out. For additional information, see TCP Transmission Modes .
B	Sample Frame	<VT><Message Data><FS><CR>	An example of a valid incoming message. This is dependent on the Transmission Mode.

C	Mode	Server	Select Server to listen for connections from clients, or Client to connect to a TCP Server. In Client mode, the Listener Settings will only be used if Override Local Binding is enabled.
D	Remote Address		The domain name or IP address on which to connect. Only applicable for Client mode.
E	Remote Port		The port on which to connect. Only applicable for Client mode.
F	Override Local Binding	No	<p>Only applicable for Client mode. Select Yes to override the local address and port that the client socket will be bound to. Select No to use the default values of 0.0.0.0:0. A local port of zero (0) indicates that the OS should assign an ephemeral port automatically.</p> <p>Note that if a specific (non-zero) local port is chosen, after a socket is closed it is up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail.</p>
G	Reconnect Interval (ms)	5000	Enter the time (in milliseconds) to wait between disconnecting from the TCP server and connecting to it again. Only applicable for Client mode.
H	Max Connections	10	The maximum number of client connections to accept. After this number has been reached, subsequent socket requests will be rejected. Only applicable for Server mode.
I	Receive Timeout (ms)	0	The amount of time, in milliseconds, to wait without receiving a message before closing a connection.
J	Buffer Size (bytes)	65536	Useful when you expect to receive large messages. Generally, the default value is fine.
K	Keep Connection Open	Yes	Select No to close the socket after a received message has finished processing. Otherwise the socket will remain open until the sending system closes it. In that case, message will only be processed if data is received and either the receive timeout is reached, the remote system closes the socket, or an end-of-message byte sequence has been detected from the Transmission Mode.
L	Data Type	Text	Select Binary if the inbound messages are raw byte streams; the payload will be Base64 encoded. Select Text if the inbound messages are textual; the payload will be encoded with the specified character set encoding.
M	Encoding	Default	Select the character set encoding to use when decoding bytes from the TCP stream, or select Default to use the default character set encoding for the JVM NextGen Connect Integration Engine is running on.
N	Respond on New Connection	No	Select No to send responses only using the same connection the inbound message was received on. Select Yes to always send responses on a new connection (during normal processing as well as recovery). Select Message Recovery to only send responses on a new connection during message recovery. Connections will be bound locally on the same interface chosen in the Listener Settings with an ephemeral port.
O	Response Address		The domain name or IP address to send message responses to.
P	Response Port		The port to send message responses to.

Source Map Variables

Key	Description
localAddress	The IP address that the TCP socket is locally bound to.
localPort	The port that that TCP socket is locally bound to.
remoteAddress	The IP address of the remote system.
remotePort	The TCP port of the remote system.

TCP Transmission Modes

The transmission mode determines how to receive message data from the incoming byte stream, and how to send responses out. The following transmission modes are supported on the [TCP Listener / Sender](#) (and [Serial Listener / Sender](#) commercial extension):

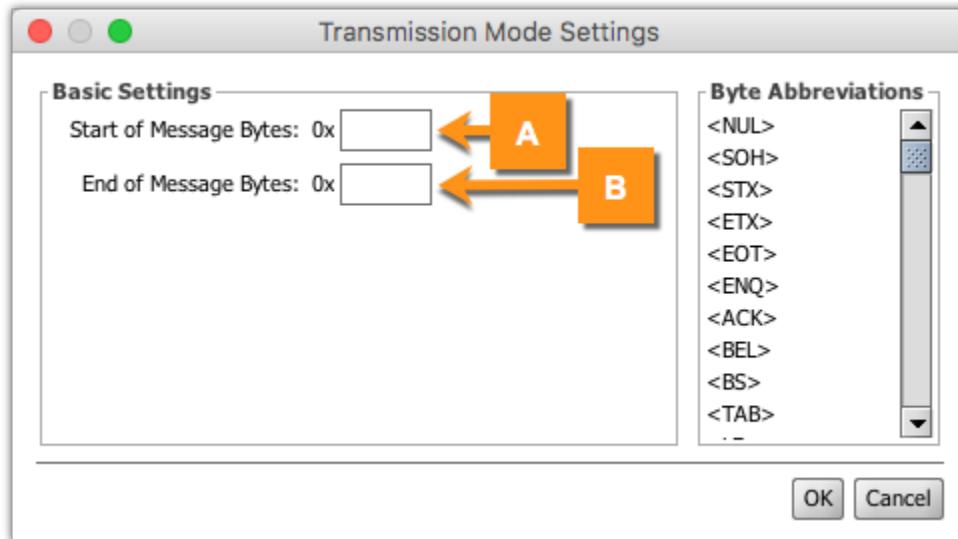
- [Basic TCP Transmission Mode](#)
- [MLLP Transmission Mode](#)

An additional transmission mode is made available via a commercial extension:

- [ASTM E1381 Transmission Mode](#)

Basic TCP Transmission Mode

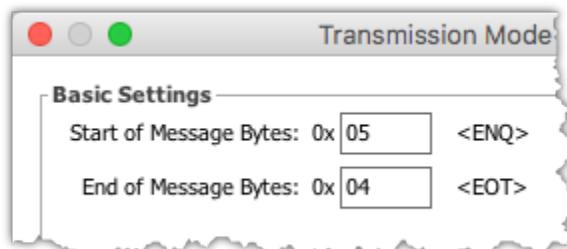
This transmission mode allows you to specify basic TCP frame data (beginning and ending byte sequences). This allows the source connector to know when a message has been fully received. Destination connectors also use these sequences when sending data outbound.



Item	Name	Default Value	Description
A	Start of Message Bytes		The bytes before the beginning of the actual message. Only valid hexadecimal characters (0-9, A-F) are allowed.
B	End of Message Bytes		The bytes after the end of the actual message. Only valid hexadecimal characters (0-9, A-F) are allowed. If this is not specified, the only way a connector knows whether a message has been received is if the socket timeout is reached or if the remote side closes the socket.

Byte Abbreviations

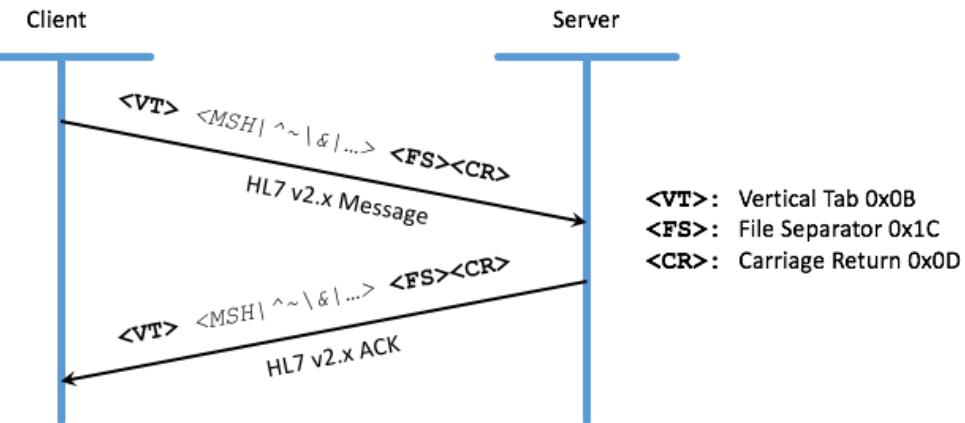
The Byte Abbreviations section to the right of the transmission mode dialog allows you to easily drag-and-drop bytes into the components to the left, without having to remember the actual hexadecimal values. These also show up as labels next to the byte fields:



MLLP Transmission Mode

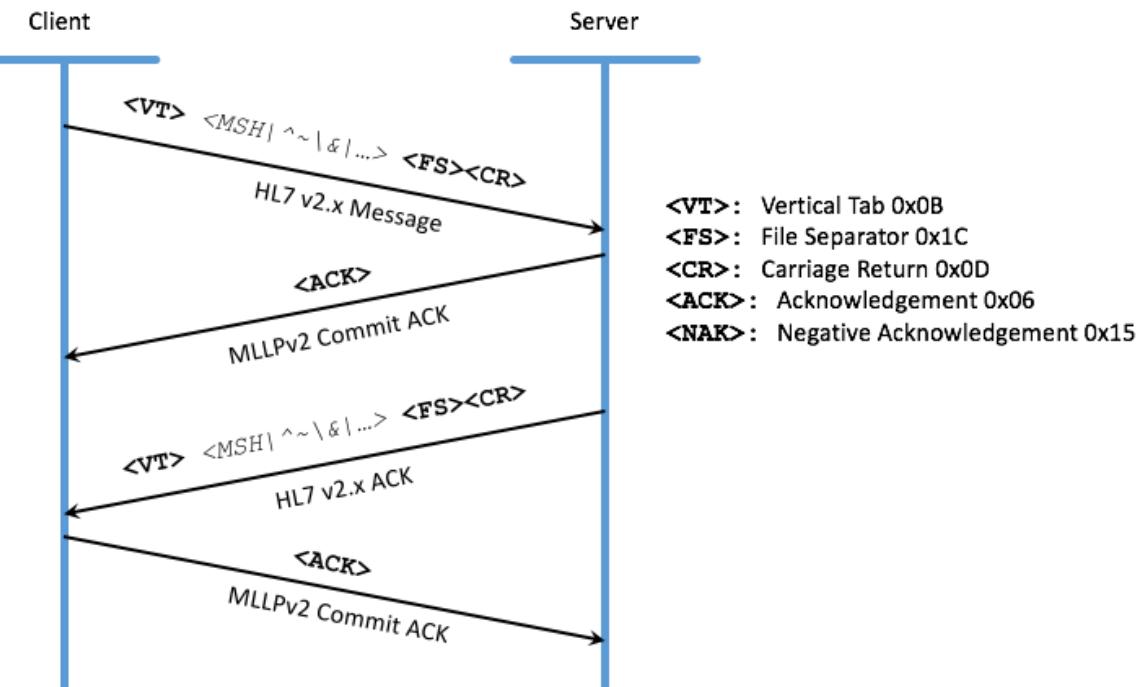
This transmission mode implements the Minimal Lower Layer Protocol (MLLP) specified by HL7, and is often used when transmitting [HL7 v2.x](#) messages. There are two versions of MLLP, v1 and v2. The first version is similar to the [Basic TCP Transmission Mode](#) in that it only specifies sequences for the start/end message bytes.

Minimal Lower Layer Protocol (MLLPv1)

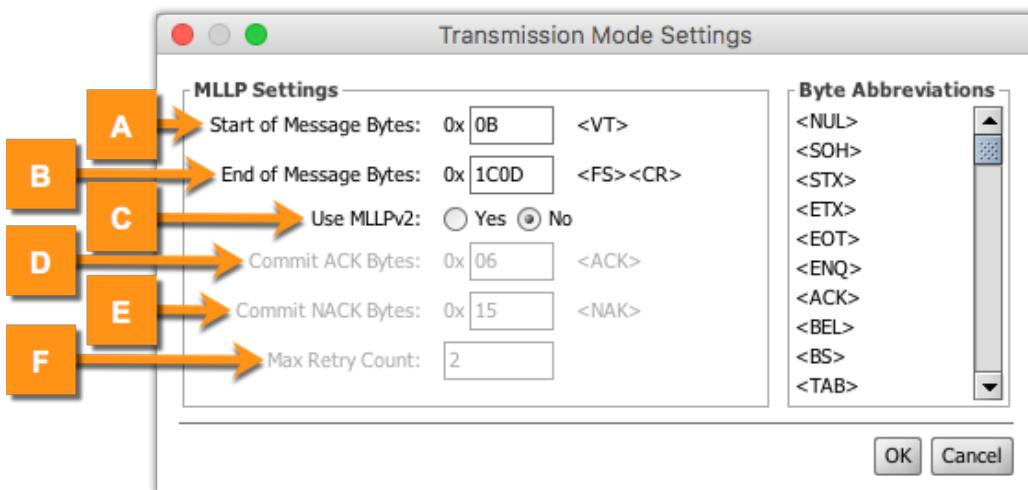


The second version builds on the first with "reliable delivery assurance", by having each system send a protocol-level ACK or NAK immediately after every received frame.

Minimal Lower Layer Protocol (MLLPv2)



By default only MLLPv1 is enabled, as it is the most common use-case.



Item	Name	Default Value	Description
A	Start of Message Bytes	0x0B (<VT>)	The MLLP Start Block bytes before the beginning of the actual message. Only valid hexadecimal characters (0-9, A-F) are allowed.
B	End of Message Bytes	0x1C0D (<FS><CR>)	The MLLP End Data/Block bytes after the end of the actual message. Only valid hexadecimal characters (0-9, A-F) are allowed.
C	Use MLLPv2	No	Select Yes to use the MLLPv2 bi-directional transport layer, which includes reliable delivery assurance as per the HL7 specifications.
D	Commit ACK Bytes	0x06 (<ACK>)	The MLLPv2 Affirmative Commit Acknowledgement bytes to expect after successfully sending a message, and to send after successfully receiving a message. Only valid hexadecimal characters (0-9, A-F) are allowed.
E	Commit NACK Bytes	0x15 (<NAK>)	The MLLPv2 Negative Commit Acknowledgement bytes to expect after unsuccessfully sending a message, and to send after unsuccessfully receiving a message. Only valid hexadecimal characters (0-9, A-F) are allowed.
F	Max Retry Count	2	The maximum number of time to retry unsuccessful dispatches before giving up and logging an error.

Byte Abbreviations

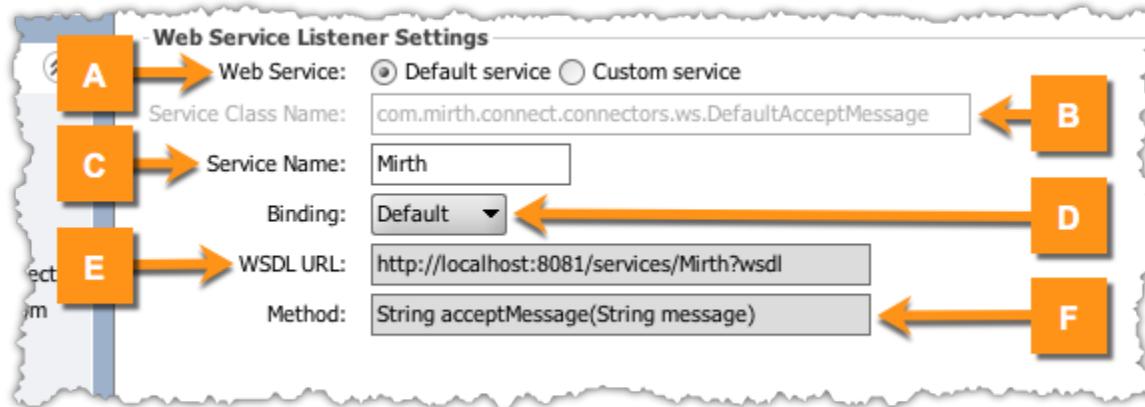
This section is the same as in the [Basic TCP Transmission Mode](#).

Web Service Listener

This source connector publishes a SOAP endpoint via [JAX-WS](#). By default it uses a simple service with one operation that takes in a message string and sends back a response string. The SOAP XML envelope is automatically handled, so that the actual data the channel receives is the content within the operation argument node. You also have the option to provide your own custom service, with custom operations.

Supported property groups:

- [Listener Settings](#)
- [Source Settings](#)
- [HTTP Authentication Settings](#)



Item	Name	Default Value	Description
A	Web Service	Default service	If Custom is selected, provide the fully-qualified class name of the Endpoint service you wish to publish.
B	Service Class Name	com.mirth.connect.connectors.ws.DefaultAcceptMessage	The fully-qualified class name of the Endpoint service to publish.
C	Service Name	Mirth	The name of the service, used to populate the URL context path.
D	Binding	Default	The selected binding version defines the structure of the generated envelope. Selecting Default will publish this endpoint with the value from the annotation in the Web Service class. If no annotation is found, the SOAP 1.1 binding will be used.
E	WSDL URL	<Auto-generated>	Displays the auto-generated WSDL URL for the web service. This is not an actual configurable setting, but is instead displayed for copy /paste convenience. Note that the host in the URL will be the same as the host you used to connect to the Administrator. The actual host that connecting clients use may be different due to differing networking environments.
F	Method	acceptMessage	If the default service is used, this will show the method "acceptMessage", which simply takes in a String and returns a String. For custom web services, this will display "<Custom Web Service Methods>".

Destination Connectors

This section refers to the actual connector-specific settings for destinations. The section is labeled according to the connector type, e.g. "HTTP Sender", "JavaScript Writer". For additional information on connectors in general, see [About Channels and Connectors](#).

Here is a list of destination connectors supported by NextGen Connect Integration Engine:

- Channel Writer
- DICOM Sender
- Database Writer
- Document Writer
- File Writer
- HTTP Sender
- JMS Sender
- JavaScript Writer
- SMTP Sender
- TCP Sender
- Web Service Sender

Additional destination connectors are made available as [commercial extensions](#):

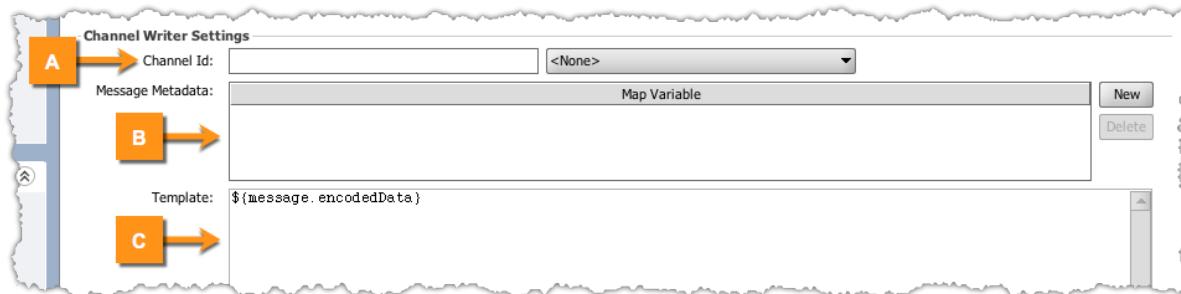
- Serial Connector
- NextGen Results CDR Connector

Channel Writer

The **Channel Writer** is a connector that simply dispatches messages to other internal channels. This can be useful if you split your message workflow into multiple channels, where one sends to another. If no target channel is specified, the connector acts as a "sink" where no message dispatching is done. Note that a channel does not need to use a [Channel Reader](#) source for a Channel Writer to be able to send messages to it. The connector also has options to inject source map variables into the downstream message of the target channel.

Supported property groups:

- [Destination Settings](#)



Item	Name	Default Value	Description
A	Channel Id	<None>	The unique ID of the target channel to send messages to. This may be a hard-coded ID, or may be a Velocity Variable Replacement . Use the drop-down menu to the right to quickly select a particular channel. If <None> is selected, the destination will act as a "sink" where messages are not dispatched anywhere.
B	Message Metadata		<p>The map variables entered here will be included in the source map of the destination channel's message.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> i This table expects only variable names, not Velocity replacement tokens. For example, do not use \${varName}, instead just use varName. The value will be extracted from all available Variable Maps. </div>
C	Template	\${message.encodedData}	The actual payload to send to the target channel. By default the encoded data of this destination will be used. Velocity Variable Replacement is supported here.

Source Map Variables

When this connector sends a message to another channel, the following [source map](#) variables will be available on the downstream message:

Key	Description
sourceChannelId	The unique ID of the channel that dispatched a message to the current channel.
sourceMessageId	The ID of the message from which the current message dispatch originated.

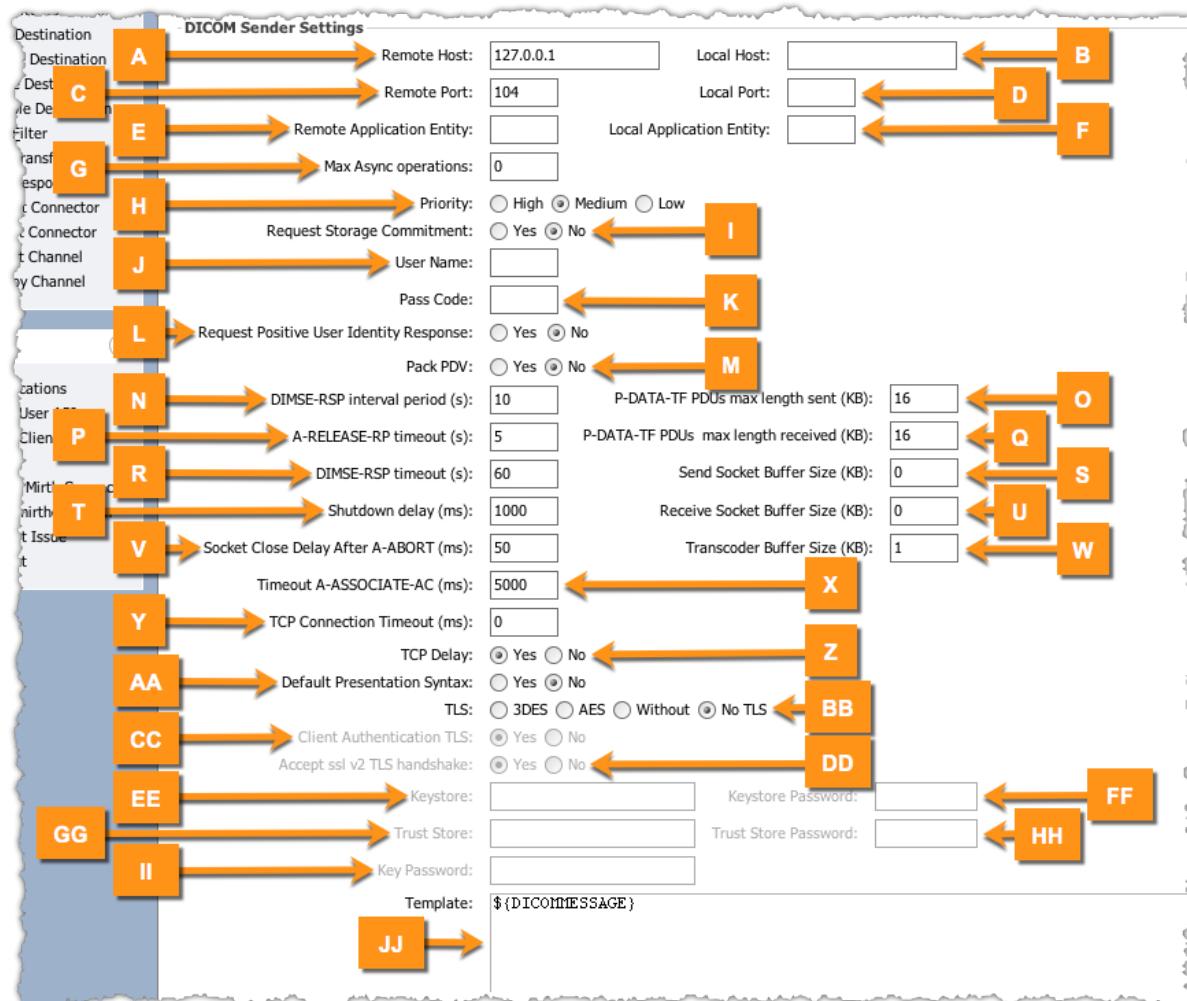
sourceChannelIds	If there are more than two channels in a Channel Writer -> Reader chain, this will be a List containing the IDs of all channels in the chain.
sourceMessageIds	If there are more than two channels in a Channel Writer -> Reader chain, this will be a List containing the IDs of all messages in the chain.

DICOM Sender

This destination connector works in conjunction with the [DICOM Attachment Handler](#) and the [DICOM Data Type](#) to allow NextGen Connect Integration Engine to send DICOM data. This connector supports the C-STORE operation as a Service Class User (SCU). Additional options are available with the [SSL Manager](#) extension.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Remote Host	127.0.0.1	The remote IP to send to.
B	Local Host	[]	The local address that the client socket will be bound to.
C	Remote Port	104	The remote port to send to.
D	Local Port	[]	The local port that the client socket will be bound to.
E	Remote Application	[]	The Application Entity title to sent to.

	Entity		
F	Local Application Entity		The Application Entity title to identify the local client with.
G	Max Async operations	0	Maximum number of outstanding operations performed asynchronously. Enter 0 for unlimited.
H	Priority	Medium	Priority of the C-STORE operation.
I	Request Storage Commitment	No	Request storage commitment of (successfully) sent objects afterwards.
J	User Name		Enable User Identity Negotiation with specified username and optional passcode.
K	Pass Code		Optional passcode for User Identity Negotiation, only effective when a username is set.
L	Request Positive User Identity Response	No	Request positive User Identity Negotiation response, only effective when a username is set.
M	Pack PDV	No	Send only one PDV in one P-Data-TF PDU, pack command and data PDF in one P-DATA-TF PDU by default.
N	DIMSE-RSP interval period (s)	10	Period to check for outstanding DIMSE-RSP, 10 seconds by default.
O	P-DATA-TF PDUs max length sent (KB)	16	Maximal length in KB of sent P-DATA-TF PDUs, 16 KB by default.
P	A-RELEASE-RP timeout (s)	5	Timeout for receiving A-RELEASE-RP.
Q	P-DATA-TF PDUs max length received (KB)	16	Maximal length in KB of received P-DATA-TF PDUs.
R	DIMSE-RSP timeout (s)	60	Timeout in milliseconds for receiving DIMSE-RSP.
S	Send Socket Buffer Size (KB)	0	Send socket buffer size in KB
T	Shutdown delay (ms)	1000	Delay in milliseconds for closing the listening socket.
U	Receive Socket Buffer Size (KB)	0	Receive socket buffer size in KB.
V	Socket Close Delay After A-ABORT (ms)	50	Delay in ms for Socket close after sending A-ABORT.

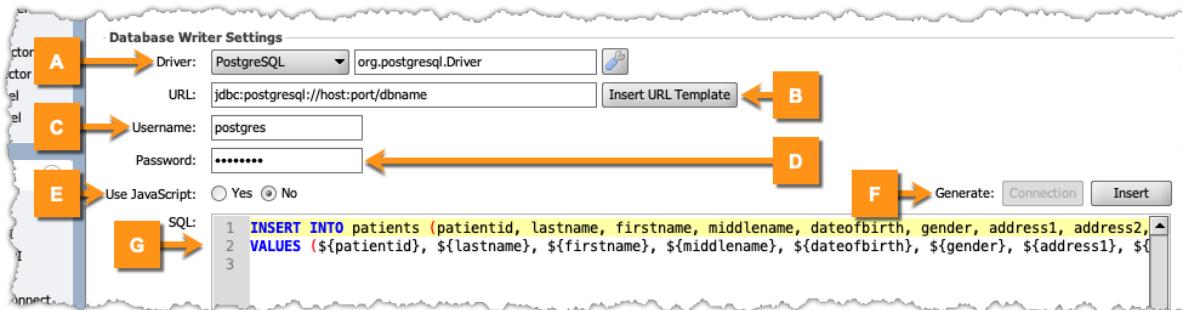
W	Transcoder Buffer Size (KB)	1	Transcoder buffer size in KB.
X	Timeout A-ASSOCIATE-AC (ms)	5000	Timeout in milliseconds for receiving A-ASSOCIATE-AC.
Y	TCP Connection Timeout (ms)	0	Timeout in milliseconds for TCP connection. Enter 0 for no timeout.
Z	TCP Delay	Yes	Set TCP_NODELAY socket option to false.
AA	Default Presentation Syntax	No	Offer Default Transfer Syntax in separate Presentation Context. By default offered with Explicit VR Little Endian TS in one PC.
BB	TLS	No TLS	Determines whether to receive data over an implicit SSL/TLS socket. The following options are available: <ul style="list-style-type: none"> • 3DES: TLS will be used, with the <i>SSL_RSA_WITH_3DES_EDE_CBC_SHA</i> cipher suite. • AES: TLS will be used, with the following cipher suites: <ul style="list-style-type: none"> • <i>TLS_RSA_WITH_AES_128_CBC_SHA</i> • <i>SSL_RSA_WITH_3DES_EDE_CBC_SHA</i> • Without: TLS will be used without symmetric encryption, with the cipher suite <i>SSL_RSA_WITH_NULL_SHA</i>. DICOM messages will be received unencrypted. • No TLS: No TLS will be used. DICOM messages will be received over a regular unencrypted socket.
CC	Client Authentication TLS	Yes	Enable client authentication for TLS. Only applicable if the TLS option is not set to No TLS .
DD	Accept ssl v2 TLS handshake	Yes	Enable acceptance of the SSLv2Hello protocol in the TLS handshake.
EE	Keystore		File path or URL of P12 or JKS keystore to use for the local server certificate keypair.
FF	Keystore Password		Password for the configured Keystore.
GG	Trust Store		File path or URL of JKS truststore, used to trust remote client certificates.
HH	Trust Store Password		Password for the configured Truststore.
II	Key Password		Password for accessing the key in the Keystore.
JJ	Template	\${DICOMMESSAGE}	The actual payload to send to the target channel. By default the encoded data of this destination (with all DICOM pixel data attachments reattached) will be used. Velocity Variable Replacement is supported here.

Database Writer

This destination connector connects to an external database and performs an INSERT/UPDATE statement (or any other statement, like calling a stored procedure). This can be done using a SQL statement, or by using JavaScript mode to execute the statement manually. The database connection will automatically be kept open across multiple dispatches.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Driver		<p>Specifies the type of database driver to use to connect to the database. The following values are supported by default:</p> <ul style="list-style-type: none"> • MySQL • Oracle • PostgreSQL • SQL Server / Sybase • SQLite <p>A custom driver class name can be entered in the text field. Drivers can be added /modified by clicking the wrench icon, more information in the Editing Database Drivers section of the Database Reader page.</p>
B	URL		The JDBC URL to connect to the database with. This is not used when "Use JavaScript" is checked. However, it is used when the Generate Connection / Insert feature is used to generate code. Use the Insert URL Template button to populate the URL field with a starting template.
C	Username		The username to connect to the database with. This is not used when "Use JavaScript" is checked. However, it is used when the Generate Connection / Insert feature is used to generate code.
D	Password		The password to connect to the database with. This is not used when "Use JavaScript" is checked. However, it is used when the Generate Connection / Insert feature is used to generate code.
E	Use JavaScript	No	If enabled, the JavaScript scripts will be used to run the insert/update statement. If disabled, SQL code (either standard or database-specific) may be used, and the connection will be handled automatically.
F	Generate		

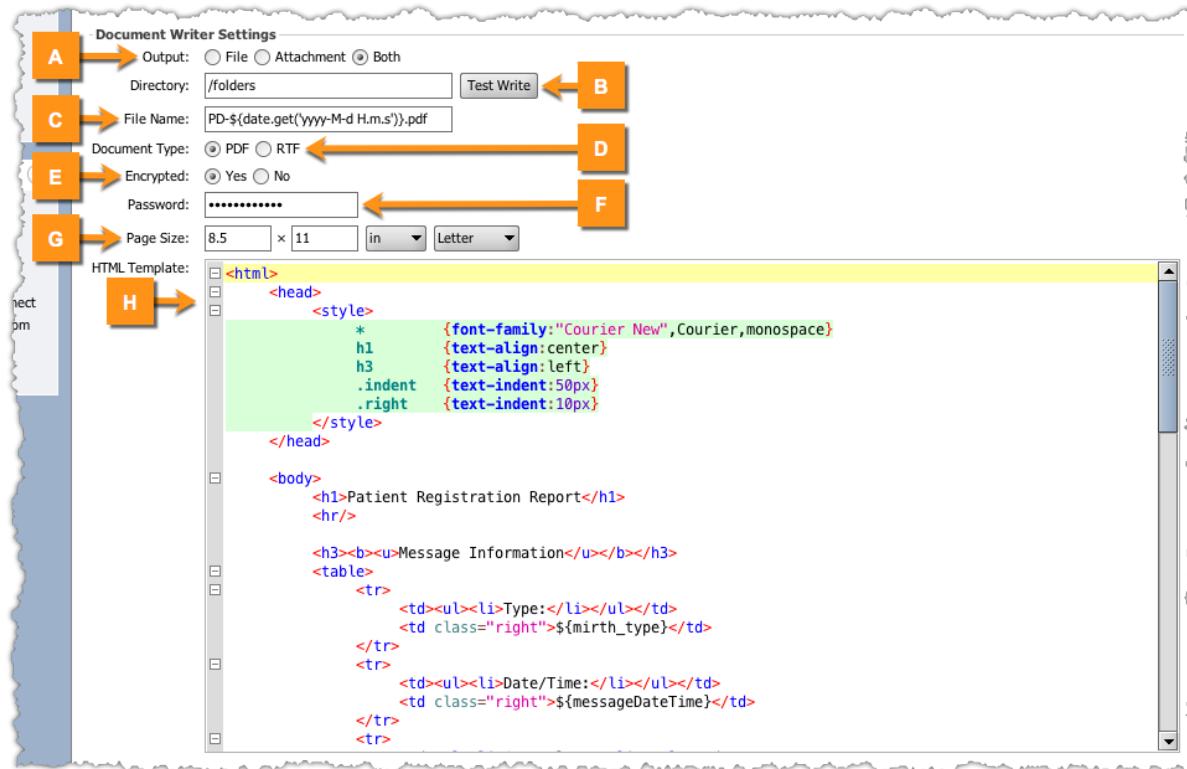
			<ul style="list-style-type: none">• Connection: This button is enabled when Use JavaScript is enabled. When clicked, it inserts boilerplate Connection construction code into the JavaScript pane at the current caret position.• Insert: Opens a window to assist in building an insert statement to insert records into the database specified in the URL above.
G	SQL / Template		The actual SQL or JavaScript code to execute.

Document Writer

This destination connector takes an HTML template and converts it into either a PDF or RTF document. Custom embedded stylesheets are supported. That document can then be written out to a file and/or stored as a message attachment. The page size can be specified, and for PDFs you can also encrypt the document with a password.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Output	File	<p>Choose how to output the document.</p> <ul style="list-style-type: none"> • File: Write the contents to a file. • Attachment: Write the contents to an attachment. The destination's response message will contain the attachment Id and can be used in subsequent connectors to include the attachment. • Both: Write the contents to both a file and an attachment.
B	Directory		The directory (folder) where the generated file should be written. Use the Test Write button to confirm that files can be written to the folder.
C	File Name		The file name to give to the generated file.
D	Document Type	PDF	The type of document to be created for each message.

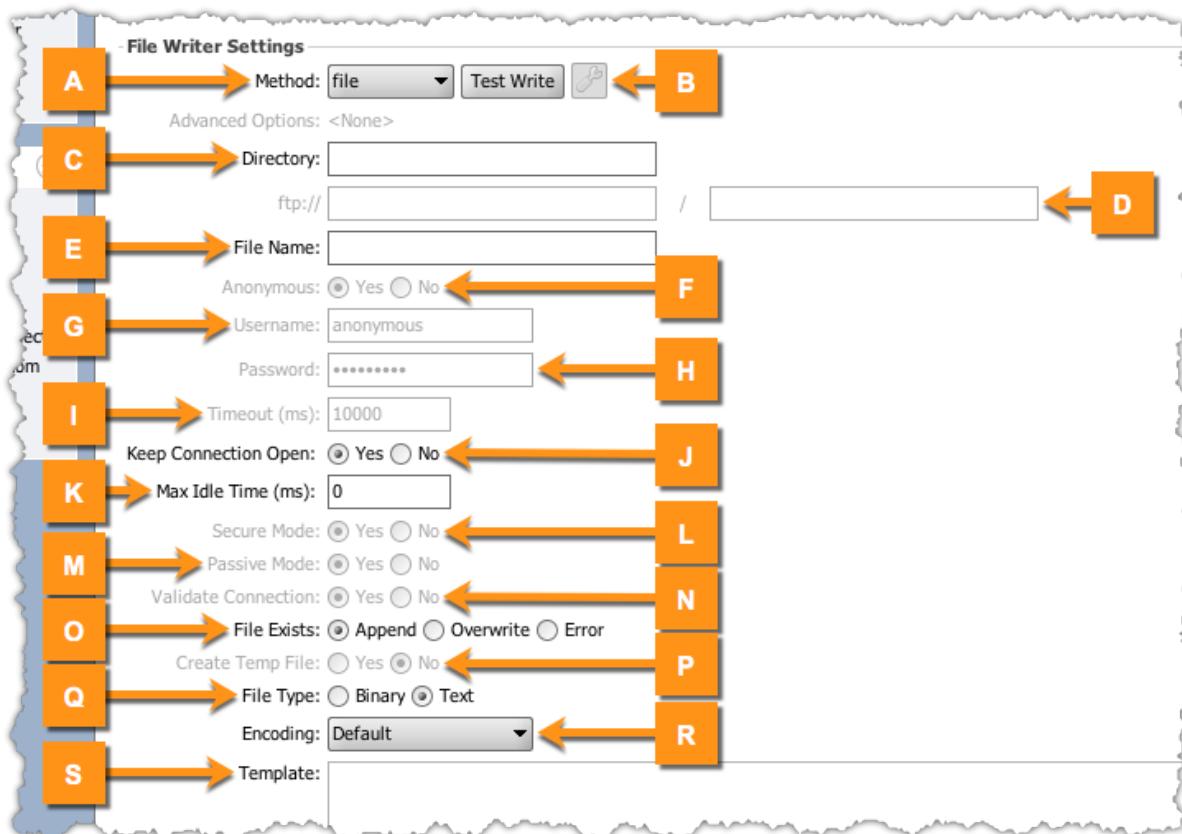
E	Encrypted	No	If the document type is PDF, generated documents can optionally be encrypted.
F	Password		If encryption is enabled, enter the password that must be used to view the document after encryption.
G	Page Size	8.5" x 11" (Letter)	The width and height of the document pages. The units for each are determined by the drop-down menu to the right. When rendering PDFs, a minimum of 26mm is enforced. Use the far-right drop-down menu to quickly select a page size among common US and UK formats.
H	HTML Template		This template is expected to be an HTML document, determining how to layout the PDF/RTF document. Custom embedded stylesheets are supported.

File Writer

This destination connector writes files out to the local filesystem, or to a remote directory. Several protocols are supported, including regular local file mode, FTP, SFTP, SMB, WebDAV, and Amazon S3. Files may be converted from Base64 and written out in raw binary format, or converted to bytes using a specific character set encoding. Additional options (like FTPS) are available with the [SSL Manager](#) extension.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Method	file	The basic method used to access the directory to write files to. Options include File (local filesystem or NFS / mapped share), FTP , SFTP , SMB , WebDAV , or Amazon S3 . Once all necessary connection/directory information has been filled in before, use the Test Write button to attempt to actually connect and test the ability to write to the directory.
B	Advanced Options		If the file method supports advanced options, this button will be enabled. Any advanced options set will be summarized in the Advanced Options label below this. For additional information, see File Reader .
C	Directory		Only applicable to the File method. The directory (folder) to write the files to.
D	URL		

			<p>Applicable to all methods except File. The domain name or IP address of the host (computer) to connect to. If this setting is enabled, the second text field specifies the directory (folder) to write to.</p> <p>When using the Amazon S3 method, the first text field will be the bucket name, and the second text field can be used for a directory prefix.</p>
E	File Name		The name to write the file out as.
F	Anonymous	Yes	Only applicable to the FTP / WebDAV / Amazon S3 methods. If enabled, connects to the remote server anonymously instead of using a username and password.
G	Username	anonymous	<p>Applicable to all methods except File. The username used to connect to the remote server with.</p> <p>When using the Amazon S3 mode, this will be your AWS Access Key ID.</p>
H	Password		<p>Applicable to all methods except File. The password used to connect to the remote server with.</p> <p>When using the Amazon S3 mode, this will be your AWS Secret Access Key.</p>
I	Timeout (ms)	10000	Applicable to the FTP / SFTP / SMB / Amazon S3 methods. The socket timeout (in ms) to use when connecting to the remote server.
J	Keep Connection Open	Yes	Select Yes to keep the connection to the file system open after writing to it.
K	Max Idle Time (ms)	0	The maximum amount of time that a connection can be idle/unused before it gets closed. A timeout value of zero is interpreted as an infinite timeout, meaning that connections will only be closed when the connector is stopped.
L	Secure Mode	Yes	Only applicable to the WebDAV method. If enabled, HTTPS will be used instead of HTTP.
M	Passive Mode	Yes	Only applicable to the FTP method. If enabled, the server decides what port the client should connect to for the data channel. Passive mode sometimes allows a connection through a firewall that normal mode does not, because the client is initiating the data connection rather than the server.
N	Validate Connection	Yes	Only applicable to the FTP method. If enabled, the connection will be tested for validity before each operation.
O	File Exists	Append	<p>Determines what to do when the file to be written already exists on the filesystem / remote server.</p> <ul style="list-style-type: none"> • Append: Messages will be appended to the end of the current file. • Overwrite: The current file will be completely overwritten with the current message content. • Error: The connector message will error out and the current file will remain unchanged.
P	Create Temp File	No	If enabled, the file contents will first be written to a temp file and then renamed to the specified file name. If disabled, the file contents will be written directly to the destination file. This option is not available if the file is being appended to (option M).

Q	File Type	Text	Select Binary if the Template contains Base64 data; the contents will be decoded into raw bytes. Select Text if the Template contains textual data; the contents will be decoded to bytes using the specified character set encoding.
R	Encoding	Default	If Text is chosen for the File Type, select the character set encoding (ASCII, UTF-8, etc.) to be used in writing out the contents of each file.
S	Template	\${message.encodedData}	The actual payload to send to the target channel. By default the encoded data of this destination will be used. Velocity Variable Replacement is supported here.

Connector Map Variables

When the Amazon S3 mode is used, these variables will be available in the connector map.

Key	Description
s3ETag	The hex encoded 128-bit MD5 hash of the file contents as computed by Amazon S3.
s3ExpirationTime	The expiration time for this object, only present if not null.
s3ExpirationTimeRuleId	The BucketLifecycleConfiguration rule ID for this object's expiration, only present if not null.
s3SSEAlgorithm	The server-side encryption algorithm, only present if the object is encrypted using AWS-managed keys.
s3SSECUSTOMERAlgorithm	The server-side encryption algorithm, only present if the object is encrypted using customer-provided keys.
s3SSECUSTOMERKEYMD5	The base64-encoded MD5 digest of the encryption key for server-side encryption, only present if the object is encrypted using customer-provided keys.
s3VersionId	The version ID of the newly uploaded object, only present if not null.
s3Metadata	A MessageHeaders object representing the map of metadata/headers for the S3 object.

HTTP Sender

This destination connector sends an HTTP request to an external web server. The method, parameters, and headers can all be fully customized. Both Basic and Digest authentication (preemptive or reactive) are supported. Sending requests through a proxy server is also supported, even when using HTTPS. The HTTP payload can be written out as raw bytes, or converted using a specified charset. Responses can be automatically converted to XML, allowing multipart payloads to be parsed in a consistent and easy-to-use way. Additional options are available with the [SSL Manager](#) extension.

Supported property groups:

- Destination Settings

The screenshot shows the 'HTTP Sender Settings' configuration screen. It includes the following fields and their corresponding labels:

- A:** URL (text input)
- B:** Use Proxy Server (radio buttons: Yes, No - Yes is selected)
- C:** Proxy Address (text input)
- D:** Proxy Port (text input)
- E:** Method (radio buttons: POST, GET, PUT, DELETE - POST is selected)
- F:** Multipart (radio buttons: Yes, No - Yes is selected)
- G:** Send Timeout (ms) (text input: 30000)
- H:** Response Content (radio buttons: Plain Body, XML Body - Plain Body is selected)
- I:** Parse Multipart (radio buttons: Yes, No - Yes is selected)
- J:** Include Metadata (radio buttons: Yes, No - Yes is selected)
- K:** Binary MIME Types (text input: application/*?(?<!json|xml)\$|image/*|video/*|audio/*) with a Regular Expression checkbox checked
- L:** Authentication (radio buttons: Yes, No - Yes is selected)
- M:** Authentication Type (radio buttons: Basic, Digest, Preemptive - Basic is selected)
- N:** Username (text input)
- O:** Password (text input)
- P:** Query Parameters (table with 'Name' and 'Value' columns, 'New' and 'Delete' buttons)
- Q:** Headers (table with 'Name' and 'Value' columns, 'New' and 'Delete' buttons)
- R:** Content Type (text input: text/plain)
- S:** Data Type (radio buttons: Binary, Text - Text is selected)
- T:** Charset Encoding (dropdown menu: UTF-8)
- U:** Content (text area)

Item	Name	Default Value	Description
A	URL		The URL of the HTTP server to send each message to.
B	Use Proxy Server	No	If enabled, requests will be forwarded to the proxy server specified in the address/port fields below.
C	Proxy Address		The domain name or IP address of the proxy server to connect to.
D	Proxy Port		The port on which to connect to the proxy server.

E	Method	POST	The HTTP operation (POST / GET / PUT / DELETE / PATCH) to send for each message.
F	Multipart	No	If enabled, the content will first be written to a local temp file. Then the contents will be wrapped in a single file part inside a multipart/form-data payload.
G	Send Timeout (ms)	30000	Sets the socket timeout (SO_TIMEOUT) in milliseconds to be used executing the method. A timeout value of zero is interpreted as an infinite timeout.
H	Response Content	Plain Body	<ul style="list-style-type: none"> Plain Body: The response body will be stored as a raw string. XML Body: The response body will be stored as serialized XML.
I	Parse Multipart	Yes	Select Yes to automatically parse multipart responses into separate XML nodes. Select No to always keep the response body as a single XML node.
J	Include Metadata	No	Select Yes to include response metadata (response code, headers) in the XML content. Note that regardless of this setting, the same metadata is always available in the connector map.
K	Binary MIME Types	application/* (?<! json xml)\$ image/* video/* audio/*	When a response comes in with a Content-Type header that matches one of these entries, the content will be encoded into a Base64 string. If Regular Expression is unchecked, specify multiple entries with commas. Otherwise, enter a valid regular expression to match MIME types against.
L	Authentication	No	If enabled, a Basic or Digest Authorization header will automatically be added to the request.
M	Authentication Type	Basic	Select between Basic or Digest auth. If the Preemptive option is checked, the Authorization header will be sent to the server with the initial request. Otherwise, the header will only be sent when the server requests it. When using Digest authentication, an Authorization header containing the realm/nonce/algorithm/qop values must be included in the Headers table.
N	Username		The username to use to authenticate to the HTTP server.
O	Password		The password to use to authenticate to the HTTP server.
P	Query Parameters		Entries in this table will automatically be added to the request URI as query parameters. Multiple parameters with the same name are supported. <i>(i)</i> If the "application/x-www-form-urlencoded" Content Type is used, this table will be used to populate the form data and will be sent in the entity payload rather than in the request URI.
Q	Headers		Entries in this table will be added to the request as HTTP headers. Multiple headers with the same name are supported.
R	Content Type	text/plain	The HTTP message body MIME type to use. If application/x-www-form-urlencoded is used, the query parameters specified above will be automatically encoded into the request body.

S	Data Type	Text	Select Binary if the outbound message is a Base64 string (will be decoded before it is sent out). Select Text if the outbound message is textual (will be encoded with the specified character set encoding).
T	Charset Encoding	UTF-8	Select the character set encoding to send with the Content-Type header, or Default to use the default character set encoding for the JVM NextGen Connect Integration Engine is running on. If None is selected, no charset will be included in the Content-Type header unless explicitly specified in the Content Type field.
U	Content		The actual payload to send. Only applicable for entity-enclosing requests (POST / PUT / PATCH). Velocity Variable Replacement is supported here.

Connector Map Variables

After a request finishes, the connector map will automatically have the following entries available. These can be used from within the [Response Transformer](#).

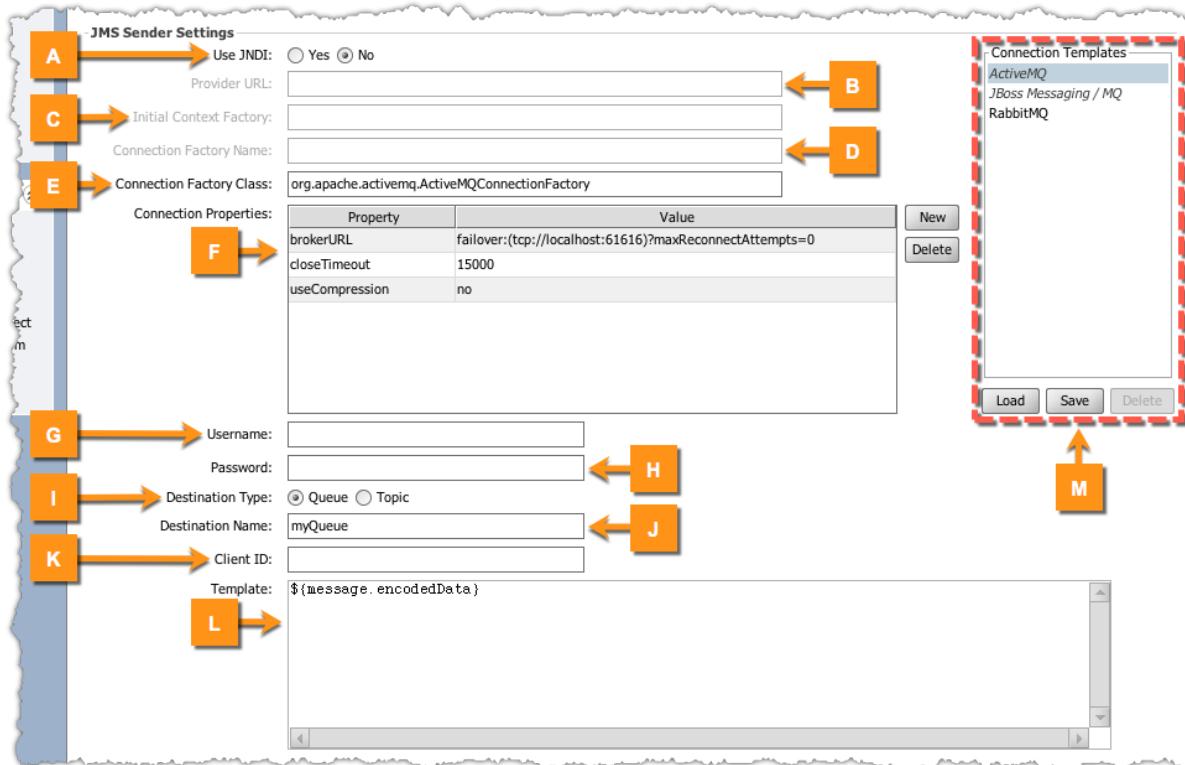
Key	Value
responseStatusLine	This is the full status line of the HTTP response, e.g. "HTTP/1.1 200 OK". It includes the HTTP version, the response code, and the response code reason.
responseHeaders	A MessageHeaders object containing all headers received in the response. Look in the User API for additional information.

JMS Sender

This destination connector connects to an external JMS provider and writes messages to a queue or topic. It supports both JNDI and specifying a specific connection factory. Once this connector dispatches a message, the connection to the JMS provider will be kept open and cached until the connector is stopped or an error occurs. The properties view also includes a mechanism to save configuration templates for common provider types, so that creating a new JMS Sender is as quick and easy as possible.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Use JNDI	No	Select Yes to use JNDI to look up a connection factory to connect to the queue or topic. Select No to specify a connection factory class without using JNDI.
B	Provider URL		If using JNDI, enter the URL of the JNDI provider here.
C	Initial Context Factory		If using JNDI, enter the fully-qualified Java class name of the JNDI Initial Context Factory class here.
D	Connection Factory Name		If using JNDI, enter the JNDI name of the connection factory here.

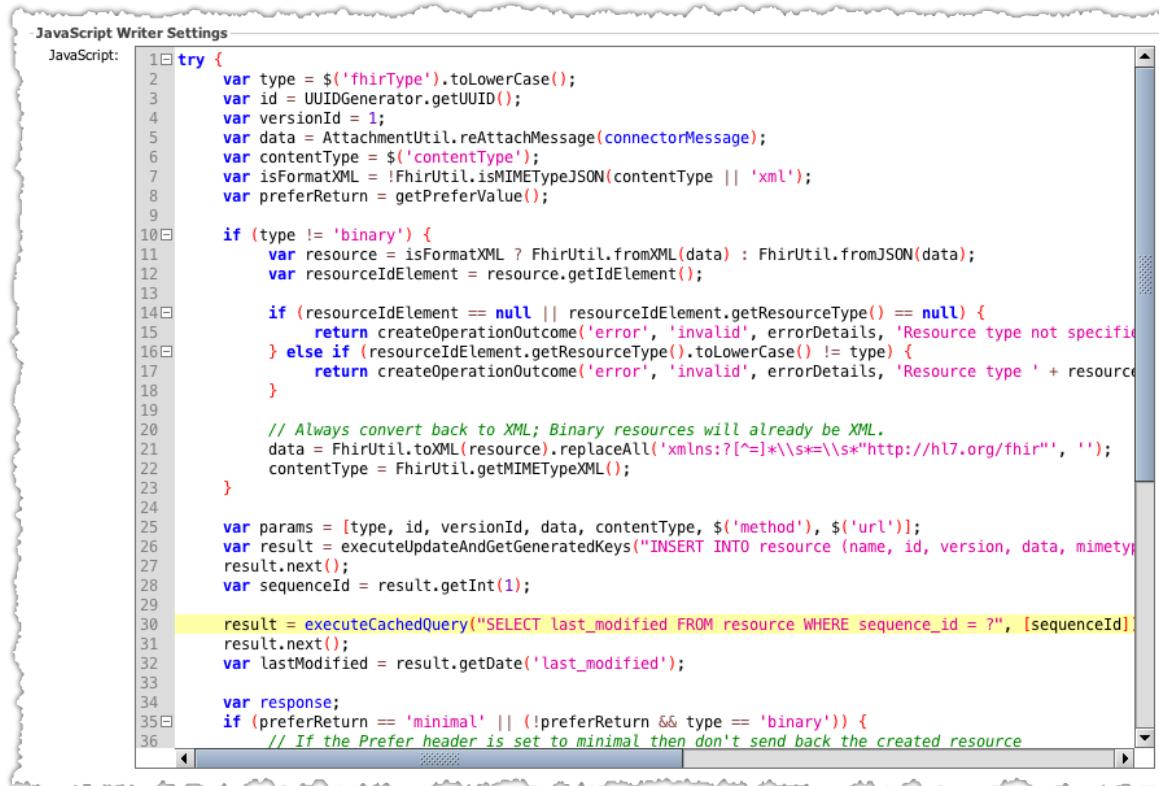
E	Connection Factory Class		If using the generic JMS provider and not using JNDI, enter the fully-qualified Java class name of the JMS connection factory here.
F	Connection Properties		This table allows you to enter custom connection factory settings. The Property column is the key, while the Value column is the actual value for the setting. The specific properties used here will vary depending on what connection factory class / provider you're using.
G	Username		The username for accessing the queue or topic.
H	Password		The password for accessing the queue or topic.
I	Destination Type	Queue	Specify whether the destination is a queue or topic.
J	Destination Name		The name of the queue or topic.
K	Client ID		The JMS client ID to use when connecting to the JMS broker.
L	Template	\${message.encodedData}	The actual payload to send to the JMS broker. By default the encoded data of this destination will be used. Velocity Variable Replacement is supported here.
M	Connection Templates		This section allows you to save the current state of your JMS Sender properties into a template , which may then be restored later if you make changes, or may also be applied to other JMS Sender connectors. More information here: JMS Listener

JavaScript Writer

This destination connector executes a custom user-defined JavaScript script. This can be used in a wide variety of ways, such as calling out to external Java libraries or invoking a local OS shell script. You can return custom values that determine what response data to store, and what status to put the destination connector message into. Or simply use the script as a generic job that doesn't necessarily produce responses. For example, you can use tools like [ChannelUtil](#) to programmatically start/stop/deploy channels from within the script.

Supported property groups:

- [Destination Settings](#)



```

JavaScript Writer Settings
JavaScript:
1  try {
2      var type = $('fhirType').toLowerCase();
3      var id = UUIDGenerator.getUUID();
4      var versionId = 1;
5      var data = AttachmentUtil.reAttachMessage(connectorMessage);
6      var contentType = $('contentType');
7      var isFormatXML = !FhirUtil.isIMETypeJSON(contentType || 'xml');
8      var preferReturn = getPreferValue();
9
10     if (type != 'binary') {
11         var resource = isFormatXML ? FhirUtil.fromXML(data) : FhirUtil.fromJSON(data);
12         var resourceIdElement = resource.getIdElement();
13
14         if (resourceIdElement == null || resourceIdElement.getResourceType() == null) {
15             return createOperationOutcome('error', 'invalid', errorDetails, 'Resource type not specified');
16         } else if (resourceIdElement.getResourceType().toLowerCase() != type) {
17             return createOperationOutcome('error', 'invalid', errorDetails, 'Resource type ' + resourceIdElement.getResourceType().toLowerCase() + ' does not match the specified type ' + type);
18         }
19
20         // Always convert back to XML; Binary resources will already be XML.
21         data = FhirUtil.toXML(resource).replaceAll('xmlns:?:[^=]*\\s*=\\s*\"http://hl7.org/fhir\"', '');
22         contentType = FhirUtil.getMIMETypeXML();
23     }
24
25     var params = [type, id, versionId, data, contentType, $('method'), $('url')];
26     var result = executeUpdateAndGetGeneratedKeys("INSERT INTO resource (name, id, version, data, mimeType, sequence_id) VALUES (?, ?, ?, ?, ?, ?)", params);
27     result.next();
28     var sequenceId = result.getInt(1);
29
30     result = executeCachedQuery("SELECT last_modified FROM resource WHERE sequence_id = ?", [sequenceId]);
31     result.next();
32     var lastModified = result.getDate('last_modified');
33
34     var response;
35     if (preferReturn == 'minimal' || (!preferReturn && type == 'binary')) {
36         // If the Prefer header is set to minimal then don't send back the created resource

```

JavaScript Writer Return Values

When you return from your script, you can choose to set a custom Response that will be stored for the destination. The following return values are accepted:

- **String:** Any string returned will be stored as the Response content for the destination, with a status of *SENT*.
- **Status:** An instance of the Status enum (for additional information, see [The User API \(Javadoc\)](#)) will cause no response content to be stored, but the connector message status will be updated to *SENT*, *QUEUED*, or *ERROR* depending on the status returned. Note that the status can only be set to *QUEUED* if queuing is enabled in the [Destination Settings](#).
- **Response:** A Response object contains a status, status message, error message, and the actual response content. If this object is returned, all of these things will be stored in the Response content, and the connector message status will be updated accordingly. For additional information, see [The User API \(Javadoc\)](#).
- **Empty String / null / undefined:** Returning any of these (including just a "return;" statement or no return statement at all) will cause **no** response data to be stored, and the message status will be updated to *SENT*.

- **Any Object:** Any other object returned will be converted to a String via the `toString()` method, and that String representation will be stored as the Response data. The message status will also be updated to `SENT`.

SMTP Sender

This destination connector sends an e-mail to a specified address (or list of addresses), through a given SMTP relay /host. Both implicit and explicit (STARTTLS) encryption modes are supported. The body can be either text or HTML. Custom headers and attachments can be added to the request as well.

Supported property groups:

- Destination Settings

The screenshot shows the 'SMTP Sender Settings' configuration screen. The interface includes the following fields and sections:

- SMTP Host:** smtp.gmail.com (labeled A)
- SMTP Port:** 587 (labeled B)
- Override Local Binding:** Yes No (labeled C)
- Local Address:** 0.0.0.0 (labeled D)
- Local Port:** 0 (labeled E)
- Send Timeout (ms):** 5000 (labeled F)
- Encryption:** None STARTTLS SSL (labeled G)
- Use Authentication:** Yes No (labeled H)
- Username:** test@hiemazing.com (labeled I)
- Password:** (redacted) (labeled J)
- To:** test@hiemazing.com (labeled K)
- From:** test@hiemazing.com (labeled L)
- Subject:** Test Subject (labeled M)
- Charset Encoding:** Default (labeled N)
- HTML Body:** Yes No (labeled O)
- Body:** Testing (labeled P)
- Headers:** (labeled Q)
- Attachments:** testattachment.png (labeled R)

Item	Name	Default Value	Description
A	SMTP Host		<p>The domain name or IP address of the SMTP server to use to send the e-mail messages. Note that sending e-mail to an SMTP server that is not expecting it may result in the IP of the machine running NextGen Connect Integration Engine being added to the server's "blacklist".</p> <p>After filling out the necessary information below, use the Send Test Email button to send a sample e-mail to the To address, to verify that everything is working as intended.</p>
B	SMTP Port	25	The port number of the SMTP server to send the e-mail messages to. Generally, the default port of 25 is used.
C	Override	No	

	Local Binding		Select Yes to override the local address and port that the client socket will be bound to. Select No to use the default values of 0.0.0.0:0. A local port of zero (0) indicates that the OS should assign an ephemeral port automatically. Note that if a specific (non-zero) local port is chosen, then after a socket is closed it's up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail.
D	Local Address	0.0.0.0	The local address that the client socket will be bound to, if Override Local Binding is enabled.
E	Local Port	0	The local port that the client socket will be bound to, if Override Local Binding is enabled. Note that if a specific (non-zero) local port is chosen, then after a socket is closed it's up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail.
F	Send Timeout (ms)	5000	The number of milliseconds for the SMTP socket connection timeout.
G	Encryption	None	Determines what type of encryption to use for the connection. <ul style="list-style-type: none"> • None: No encryption will be used. Messages will be sent over the connection in plain text. • STARTTLS: The connection will begin as unencrypted, and then the SMTP client will manually upgrade the connection through a STARTTLS command. • SSL: The connection will be encrypted from the very beginning. Use this when the server expects a TLS handshake after a client connects.
H	Use Authentication	No	Determines whether to use authentication when connecting to the SMTP server.
I	Username		The username to authenticate with.
J	Password		The password to authenticate with.
K	To		The e-mail address to send to. Multiple addresses can be specified with commas.
L	From		The e-mail address to send the message from.
M	Subject		The subject line of the e-mail.
N	Charset Encoding	Default	The character set encoding to use when converting the body, or Default to use the default character set encoding of the JVM NextGen Connect Integration Engine is running on.
O	HTML Body	No	Determines the MIME type of the message, either text/plain or text/html. If HTML is used, richer message formatting may be used.
P	Template		The actual body of the e-mail. Velocity Variable Replacement is supported here.
Q	Headers		Entries in this table will be included as SMTP headers in the e-mail dispatch.
R	Attachments		Entries in this table will be added as attachments with the e-mail. The following columns are configurable: <ul style="list-style-type: none"> • Name: The name of the attachment.

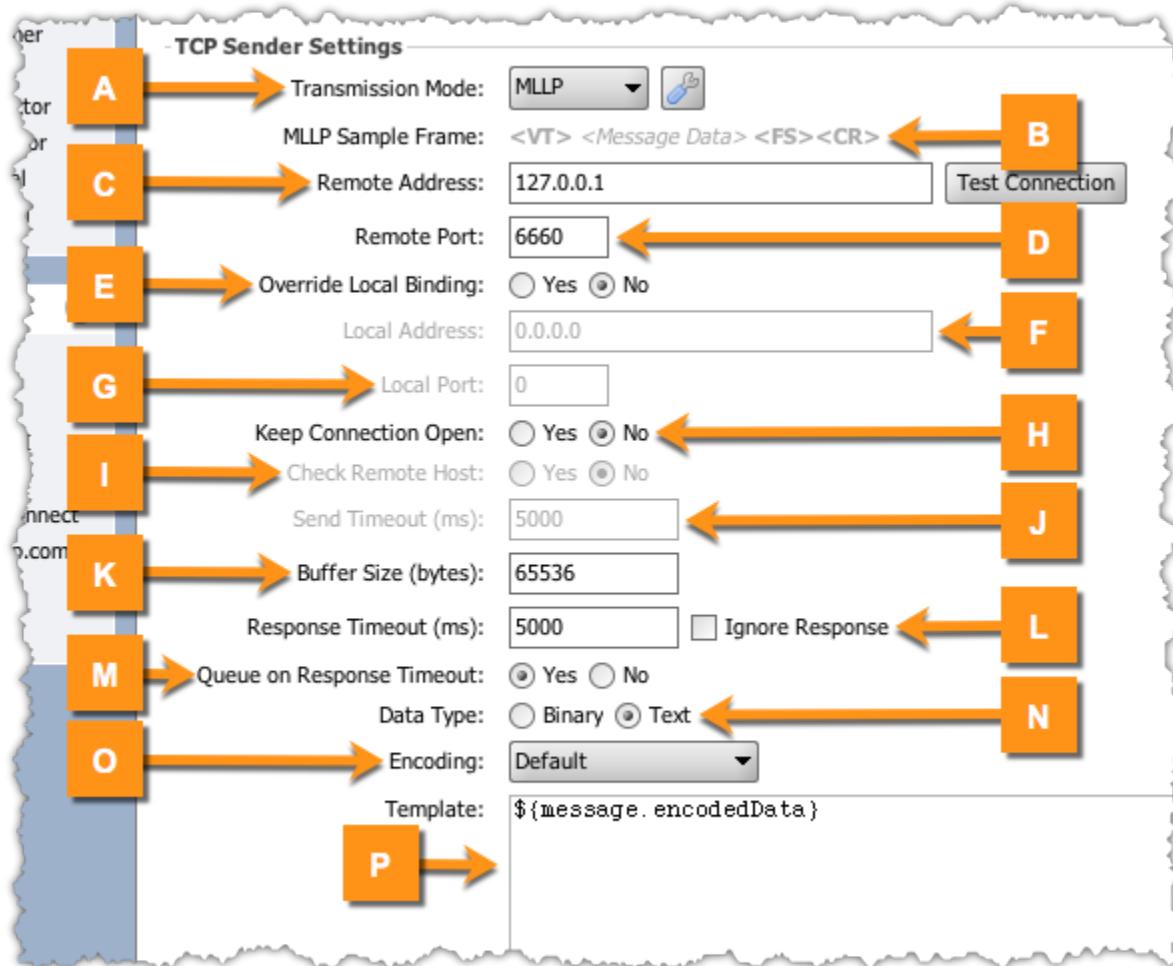
- **Content:** The Base64-encoded content of the attachment. You can also use a message attachment replacement token here.
- **MIME Type:** The MIME type of the attachment (e.g. "image/png").

TCP Sender

This destination connector opens a new TCP client connection and sends messages over it. You can decide whether to keep a connection open, and if so for how long. Configurable [transmission modes](#) allow you to decide how to send outbound messages and receive responses.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	Transmission Mode	MLLP	The transmission mode determines how to send message data out on the socket byte stream, and how to receive responses. For additional information, see TCP Listener
B	Sample Frame	<VT> <Message Data> <FS><CR>	This is dependent on the Transmission Mode and displays an example of how an outgoing message frame is expected to look.
C	Remote	127.0.0.1	The domain name or IP address on which to connect. Press the Test

	Address		Connection button to verify whether the server is able to open a TCP connection as the specified IP/port.
D	Remote Port	6660	The port on which to connect.
E	Override Local Binding	No	Select Yes to override the local address and port that the client socket will be bound to. Select No to use the default values of 0.0.0.0:0. A local port of zero (0) indicates that the OS should assign an ephemeral port automatically. Note that if a specific (non-zero) local port is chosen, then after a socket is closed it's up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail.
F	Local Address	0.0.0.0	The local address that the client socket will be bound to, if Override Local Binding is enabled.
G	Local Port	0	The local port that the client socket will be bound to, if Override Local Binding is enabled. Note that if a specific (non-zero) local port is chosen, then after a socket is closed it's up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail.
H	Keep Connection Open	No	Select Yes to keep the connection to the host open across multiple messages. Select No to immediately close the connection to the host after sending each message. <div style="border: 1px solid #f0e68c; padding: 10px; width: fit-content; margin-left: auto; margin-right: 0;">  When Keep Connection Open is enabled, the Send Timeout is used to determine how long to keep the connection open when there are no messages to send. By default the Send Timeout is set to 5 seconds, so even when Keep Connection Open is enabled, the connection may still be closed if there is a period of downtime where no messages are being dispatched. </div>
I	Check Remote Host	No	Select Yes to check if the remote host has closed the connection before each message. Select No to assume the remote host has not closed the connection. Checking the remote host will decrease throughput but will prevent the message from erroring if the remote side closed the connection and queuing is disabled.
J	Send Timeout (ms)	5000	The number of milliseconds to keep the connection to the host open, if Keep Connection Open is enabled. If zero, the connection will be kept open indefinitely.
K	Buffer Size (bytes)	65536	The size, in bytes, of the buffer to hold messages waiting to be sent. Generally, the default value is fine.
L	Response Timeout (ms)	5000	The number of milliseconds the connector should wait whenever attempting to create a new connection or attempting to read from the remote socket. If Ignore Response is checked, the connector will not wait for a response at all after sending a message.
M	Queue on Response Timeout	Yes	If enabled, the message is queued when a timeout occurs while waiting for a response. Otherwise, the message is set to <i>ERROR</i> when a timeout occurs. This setting has no effect unless queuing is enabled for the connector.
N	Data Type	Text	Select Binary if the outbound message is a Base64 string (will be decoded before it is sent out). Select Text if the outbound message is

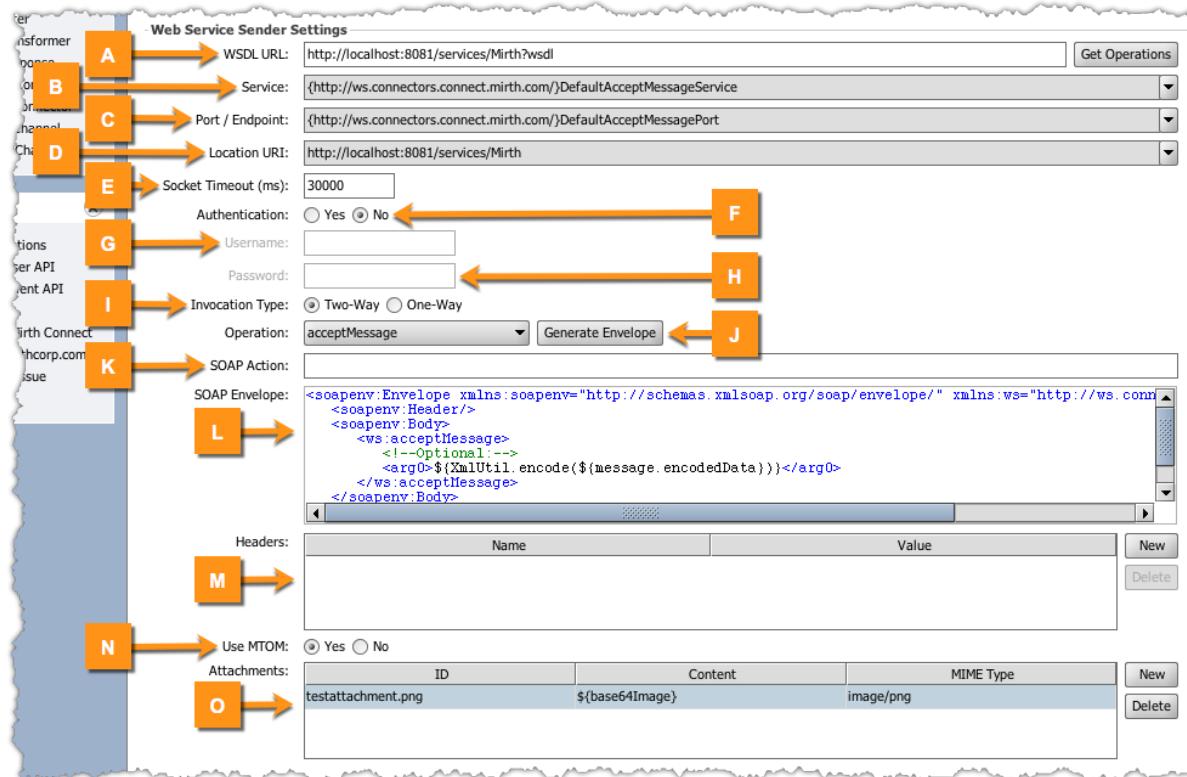
			textual (will be encoded with the specified character set encoding).
O	Encoding	Default	Select the character set encoding used by the message sender, or select Default to use the default character set encoding for the JVM NextGen Connect Integration Engine is running on.
P	Template	\${message.encodedData}	The actual payload to send to the remote server. By default the encoded data of this destination will be used. Velocity Variable Replacement is supported here.

Web Service Sender

This destination connector connects to a SOAP endpoint via [JAX-WS](#) and invokes a defined operation. When configuring this connector you can automatically fetch the WSDL from the remote server, and all the services / endpoints / operations will be filled out and modifiable from drop-down menus. Automatic generation of a sample SOAP envelope is supported too. Custom headers and MTOM attachments can be added to each request as well.

Supported property groups:

- Destination Settings



Item	Name	Default Value	Description
A	WSDL URL		The URL to the WSDL describing the web service and available operations. Click on Get Operations after entering the WSDL to automatically fill out the Service, Port, Location URI, and available Operations.
B	Service		The service name for the WSDL defined above. This field is filled in automatically when the Get Operations button is clicked and does not usually need to be changed, unless multiple services are defined in the WSDL.
C	Port / Endpoint		The port / endpoint name for the service defined above. This field is filled in automatically when the Get Operations button is clicked and does not usually need to be changed, unless multiple endpoints are defined for the currently selected service in the WSDL.
D	Location URI		The dispatch location for the port / endpoint defined above. This field is filled in automatically when the Get Operations button is clicked and does not

			usually need to be changed. If left blank, the default URI defined in the WSDL will be used.
E	Socket Timeout (ms)	30000	Sets the connection and socket timeout (SO_TIMEOUT) in milliseconds to be used when invoking the web service. A timeout value of zero is interpreted as an infinite timeout.
F	Authentication	No	Turning on authentication uses a username and password to get the WSDL, if necessary, and uses the username and password binding provider properties when calling the web service.
G	Username		The username used to get the WSDL and call the web service.
H	Password		The password used to get the WSDL and call the web service.
I	Invocation Type	Two-Way	<p>Determines how to invoke the operation selected below.</p> <ul style="list-style-type: none"> Two-Way: Invoke the operation using the standard two-way invocation function. This will wait for some response or acknowledgement to be returned. One-Way: Invoke the operation using the one-way invocation function. This will not wait for any response, and should only be used if the operation is defined as a one-way operation.
J	Operation		The web service operation to be called. This is used to generate the envelope along with the Generate Envelope button.
K	SOAP Action		The SOAPAction HTTP request header field can be used to indicate the intent of the SOAP HTTP request. This field is optional for most web services, and may be auto-populated when you select an operation.
L	SOAP Envelope		<p>The actual SOAP envelope to send to the remote web service. Use the Generate Envelope button above to generate a sample skeleton XML document that you can then fill out using Velocity Variable Replacement.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>i This is an XML SOAP document, so any variables used here may need to be properly entity-encoded. You can do this by dragging the XML Entity Encoder over from the Destination Mappings list first, and then dragging the map variable inside the "XmlUtil.encode()" function call.</p> </div>
M	Headers		Entries in this table will be added to the request as HTTP headers. Multiple headers with the same name are supported.
N	Use MTOM	No	Enables MTOM on the SOAP Binding. If enabled, attachments can be added to the table below and referenced from within the envelope.
O	Attachments		<p>Entries in this table will be added as MTOM attachments along with the request. The following columns are configurable:</p> <ul style="list-style-type: none"> ID: A unique ID for the attachment which can be referenced from within the SOAP envelope. Content: The Base64-encoded content of the attachment. You can also use a message attachment replacement token here. MIME Type: The MIME type of the attachment (e.g. "image/png").

NextGen Connect Integration Engine and JavaScript

JavaScript is a scripting language that can be used in a wide variety of places throughout NextGen Connect Integration Engine to perform advanced routing and transformation. This section is divided into the following topics:

- [About JavaScript](#)
- [Using JavaScript in Mirth Connect](#)
- [Using the JavaScript Editor](#)
- [Variable Maps](#)
- [Attachment JavaScript Functions](#)
- [The User API \(Javadoc\)](#)

About JavaScript

This section provides a basic explanation of how the language works:

- [Variables](#)
- [Comments](#)
- [Arrays](#)
- [Operators](#)
- [Conditional Statements](#)
- [Functions](#)
- [Loops and Iterations](#)
- [Exception Handling](#)

Variables

Unassigned variables have the value `undefined` by default; string literals can use single or double quotes; braces `{ }` create a block of statements that can be used for loops, conditionals, and statements. These are some examples of variable declarations:

```
var x;  
var y = r;  
z = "abc"
```

Comments

You can start a single-line comment with two forward slashes, or a multi-line comment with forward slashes and asterisks:

```
// This is a single-line comment  
  
/*  
   This is a  
   multi-line  
   comment  
*/
```

Arrays

Arrays are native objects indexed with bracket notation that can contain other objects, arrays, or primitive types. Arrays can be initialized using Java-like constructors or bracket notation and do not need to be sized upon construction:

```
var myArray1 = new Array(10);  
var myArray2 = new Array(5, "some string", new Array());  
var myArray3 = [ ];  
var myArray4 = [ "one", "two", "three" ];
```

Uninitialized elements in arrays are `undefined`, so use the `length` property (`var size = myArray.length;`) to get a size. Use the `delete` operator to remove the index value, which sets the element as `undefined`. There are several built-in methods for arrays: `concat()`, `reverse()`, `replace()`, `sort()`, `indexOf()`. Arrays in JavaScript are zero-based indexed, so use `myArray[0]` to access the first element. Other important native objects in JavaScript include:

- String
- Date
- Boolean
- RegExp (regular expressions)
- Math
- XML

Operators

The most common JavaScript operators can be put into the following categories:

- Arithmetic
- Assignment
- Comparison
- Logical

Arithmetic Operators

These operators are used to perform arithmetic between variables and/or values. In this table, the **y**-variable has a value of 5 to explain the JavaScript Arithmetic operators:

Operator	Description	Example	Result of x	Result of y
+	Addition	x=y+2 (5+2)	7	5
-	Subtraction	x=y-2 (5-2)	3	5
*	Multiplication	x=y*2 (5*2)	10	5
/	Division	x=y/2 (5/2)	2.5	5
%	Modulus (division remainder)	x=y%2 (remainder is 1 for the equation 5 /2)	1	5
++	Increment	x=++y (++6)	6 (a)	6 (b)
		x=y++ (6++)	5 (c)	6 (b)
--	Decrement	x=--y (-4)	4 (d)	4 (e)
		x=y-- (4--)	5 (f)	4 (e)

(a) The increment occurs *before* the variable, so **x** is the incremented value of **y**, which is 6; (b) The value is 6 because **y**, which is 5, is increased by 1 for this operator; (c) The increment occurs *after* the variable, so **x** is the original value of **y**, which is 5; (d) The decrement occurs *before* the variable, so **x** is the decremented value of **y**, which is 4; (e) The value is 4 because **y**, which is 5, is decreased by 1 for this operator; (f) The decrement occurs *after* the variable, so **x** is the original value of **y**, which is 5.

Assignment Operators

These operators are used to assign values to JavaScript variables. In this table, the **x**-variable has a value of 10, and the **y**-variable has a value of 5 to explain the JavaScript Assignment operators:

Operator	Description	Example	Same As	Result
=	assign	x=y	x=y (5)	x=5

<code>+=</code>	add and assign	<code>x+=y</code>	<code>x=x+y (10+5)</code>	<code>x=15</code>
<code>-=</code>	subtract and assign	<code>x-=y</code>	<code>x=x-y (10-5)</code>	<code>x=5</code>
<code>*=</code>	multiply and assign	<code>x*=y</code>	<code>x=x*y (10*5)</code>	<code>x=50</code>
<code>/=</code>	divide and assign	<code>x/=y</code>	<code>x=x/y (10/5)</code>	<code>x=2</code>
<code>%=</code>	modulus (division remainder) and assign	<code>x%==y</code>	<code>x=x%y (remainder is 0 for the equation 10/2)</code>	<code>x=0</code>

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values. In this table, the `x`-variable has a value of 5 to explain the JavaScript Comparison operators:

Operator	Description	Comparing	Returns
<code>==</code>	is equal to	<code>x==8</code>	false
		<code>x==5</code>	true
<code>!=</code>	is not equal to	<code>x!=8</code>	true
<code>></code>	is greater than	<code>x>8</code>	false
<code><</code>	is less than	<code>x<8</code>	true
<code>>=</code>	is greater than or equal to	<code>x>=8</code>	false
<code><=</code>	is less than or equal to	<code>x<=8</code>	true

Logical Operators

Logical operators are used to determine the logic between variables or values. In this table, the **x**-variable has a value of 6, and the **y**-variable has a value of 3 to explain the JavaScript Logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
	or	(x==5 y==5) is false
!	not	!(x==y) is true

Conditional Statements

This example shows the basic syntax structure for conditional statements in JavaScript:

```
if (condition1) {
    // Code to execute
} else if (condition2) {
    // Code to execute
} else {
    // Code to execute
}
```

Functions

This example shows the basic syntax structure for the creation of functions in JavaScript:

```
function functionName (p1, p2, ..., pN) {
    // Code to execute
    return someValue;
}
```

There are various ways to call a function [e.g., `var x = myFunction ("ABC", 100, myVar);`]. Parameters are optional and unlimited. The return statement is also optional.

Loops and Iterations

A *loop* is a type of programming-language statement that lets code be executed repeatedly; that is, a loop is a series of iterations. In programming language, there are four types of loops:

- `for`
- `for each...in`
- `while`
- `do...while`.

An *iteration* is a single execution of the inner loop process. If you loop from 1 to 10, the code inside the loop will be executed for 10 iterations.

Loops can be unconditionally exited with a *break* statement: **break**; The *continue* statement: **continue**; unconditionally skips to the next iteration of the loop.

for loops

These loops are often distinguished by an explicit loop *counter* (variable), which lets the body of the **for** loop (the code that is being repeatedly executed) know about the sequencing of each iteration. **for** loops are typically used when the number of iterations is known before the loop is entered.

Syntax (for loops)	Example
<pre>for (index=startValue; endCondition; inclIndex) { // Code to execute }</pre>	<pre>for (var i=0; i<10; i++) { logger.info("i = " + i); }</pre>

for each...in loops

These loops are part of the E4X standard. Unlike other **for** loop constructs, **for each...in** loops usually have no explicit counter; they essentially say "do this to everything in this set" rather than "do this X times." This avoids possible off-by-one errors and makes code easier to read. These loops are used to iterate through elements in an array (or collection) or in the property values of an object.

Syntax (for each...in loops)	Example
<pre>for each (var in object) { // Code to execute }</pre>	<pre>var sources = new Array (); sources [0] = "Customer 1"; sources [1] = "Customer 2"; sources [2] = "Customer 3"; for each (src in sources) { logger.info(src); }</pre>

while loops

These loops are **control-flow statements** that let code execute repeatedly based on a given Boolean (true/false) condition. A **while** loop can be thought of as a repeating "**if**" statement and consists of a block of code and a condition. Upon evaluation, if the condition is *true*, the code in the block is executed, repeating until the condition becomes *false*. A **while** loop checks the condition *before* the block is executed, in contrast to a **do...while** loop, which tests the condition *after* the block is executed.

Syntax (while loops)	Example
<pre>while (condition) { // Code to execute }</pre>	<pre>var index = 0; var found = false; while (!found) { if (myArray[index++] == "ABC") { found = true; } }</pre>

do...while loops

These loops let code execute *at least once* based on a Boolean (true/false) condition. A **do...while** loop consists of a process symbol and a condition. The code in the block executes, and the condition is evaluated. If the condition is *true*, the code in the block is executed again, repeating until the condition becomes *false*. A **do...while** loop checks the condition *after* the block is executed, in contrast to the **while** loop, which tests the condition *before* the block is executed. It is possible—and sometimes desirable—for the condition to always evaluate as *true*, which creates an infinite loop. When such a loop is created purposely, there is usually another control structure, such as a *break* statement, that terminates the loop.

Syntax (do...while loops)	Example
<pre>do { // Code to execute } while (conditional);</pre>	<pre>var index = -1; do { var val = getValue(++index); } while (val != "ABC");</pre>

Exception Handling

The variable in a *catch* statement is of the *Error* type or one of its subclasses. You can raise exceptions with a *throw* statement:

```
throw "This is my exception message!";
throw new RangeError("Var x is not between 1 and 100");
```

Uncaught exceptions inside a connector result in an error status for the processed message.

Syntax (Exception Handling)	Example
<pre>try { // Code to execute } catch (exception) { // Exception handling // code to execute } finally { // Code to always execute }</pre>	<pre>try { var x = undefinedVarName; } catch (e) { logger.error("Error: "+e); }</pre>

Using JavaScript in Mirth Connect

This section shows you how to use JavaScript to perform various messaging operations within NextGen Connect Integration Engine. Click a link to go to the operation:

- [About E4X](#)
- [Accessing Message Data with E4X](#)
- [Adding Segments to a Message](#)
- [Deleting a Segment](#)
- [Iterating Over Message Segments](#)
- [Iterating Over Repeating Fields](#)
- [Adding a New Repeating Field](#)
- [Message Variables](#)
- [Built-In Code Templates](#)
- [Using Java Classes](#)
- [Regular Expressions](#)
- [Logging with JavaScript](#)

About E4X

ECMAScript for XML (E4X), introduced in JavaScript 1.6, is a JavaScript extension that provides native XML support to *ECMAScript*.

```
var person1 = new XML("<person></person>");  
var person2 = <person></person>;
```

E4X supplies a simpler alternative to Document Object Model (DOM) interfaces for accessing XML documents. E4X also offers a new way to make XML visible. Prior to E4X, XML had to be accessed at an *object* level. E4X regards XML as *primitive* level, which suggests quicker access, improved support, and acknowledgment as a component (data structure) of a program. Provided below are several useful XML object methods:

- [appendChild\(\)](#)—appends a child element
- [name\(\)](#)—gets the name of an element
- [attribute\(\)](#)—gets an attribute of an element
- [children\(\)](#)—gets a list of all of an element's child elements
- [length\(\)](#)—gets the count of an element's child elements.

Use DOM-like syntax to access XML elements and use @ for element attributes:

```
<person>  
  <name>  
    <first/>  
    <last/>  
  </name>  
  <address type="home"/>  
</person>  
person.name.first = "Joe"  
person['name']['first'] = "Joe"  
...  
person.address.@type = "work";  
person['address'][@type] = "work";
```

Accessing Message Data with E4X

```
<?xml version="1.0" encoding="UTF-8"?>  
<HL7Message>
```

```

<MSH>
  <MSH.1>|</MSH.1>
  <MSH.2>|^~\&lt;/MSH.2>
  <MSH.3>
    <MSH.3.1>SENDAPP</MSH.3.1>
  </MSH.3>
  <MSH.4>
    <MSH.4.1>General Hospital</MSH.4.1>
  </MSH.4>
  <MSH.5>
    <MSH.5.1>RECAPP</MSH.5.1>
  </MSH.5>
  <MSH.6>
...
  </MSH>
...
</HL7Message>

```

The XML variables **msg** and **tmp** represent root-level elements. Use JavaScript bracket notation for each element level in the document below the root (this works with any message data, not just HL7):

```
var sendingFacility = msg['MSH']['MSH.4']['MSH.4.1'].toString();
```

Most values in an HL7 message go three levels deep. The first level is a *segment*; the second level is a *field* within the segment; the third level is a *component* within a segment field. If a field or component does not exist, it is created automatically. Examples for repeating segments and fields include using bracket notation to index:

```

var obx = msg['OBX'];
var obx5 = obx['OBX.5'];
var obx5_1 = obx5['OBX.5.1'].toString();

```

All E4X methods available on the **msg** and **tmp** variables can be accessed through the auto-completion dialog in the **JavaScript Editor**.

Adding Segments to a Message

To add a segment to a message, create a new XML object with a segment code, then add it after the segment it should follow:

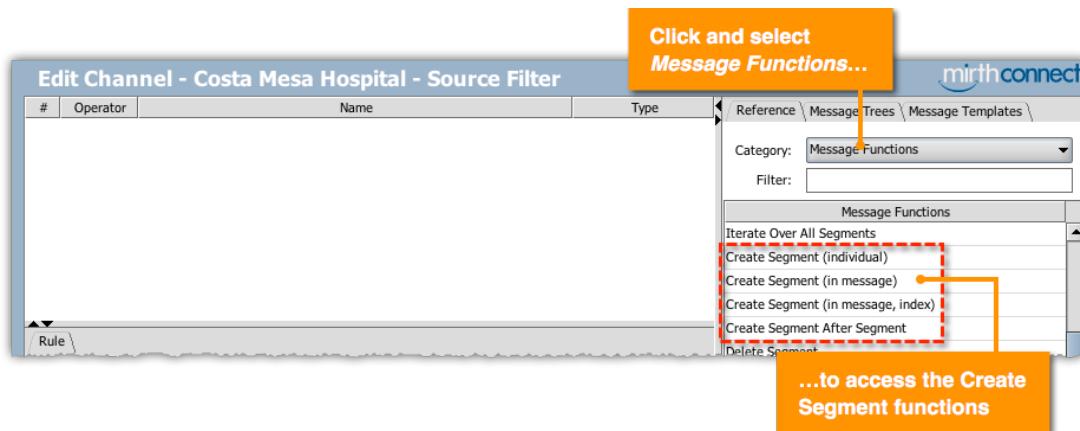
```
var seg = new XML("<ZZZ><ZZZ.1><ZZZ.1.1>My Value</ZZZ.1.1></ZZZ.1></ZZZ>");
msg['QRF'] += seg;
```

Insert the **`+=`** operator into a message after a particular segment or at the end of the message. NextGen Connect Integration Engine has four global functions by which you can create segments for messages:

Function	Function Description	Example
Create Segment (individual)	Creates an XML object for the segment that has not been inserted into a message.	<pre>var newSeg = createSegment('ZYX'); newSeg['ZYX']['ZYX.1']['ZYX.1.1'] = "My Value"; msg += newSeg;</pre>
Create Segment	Creates an XML object for the segment at the end of a specified message.	<pre>createSegment('ZYX', msg);</pre>

(in message)		msg ['ZYX']['ZYX.1']['ZYX.1.1'] = "My Value";
Create Segment (in message, index)	Creates an XML object for the segment in a specified message (msg or tmp) in a specified index and is issued for repeating segments; if a segment is already in the index, the new segment overwrites it.	createSegment ('OBX', msg, 4); msg ['OBX'][4]['OBX.3']['OBX.3.2'] = "Glucose";
Create Segment After Segment	Creates an XML object for the segment and adds it after the target segment.	createSegmentAfter ("ZZZ", msg ["QRF"]); msg ["ZZZ"]["ZZZ.1"]["ZZZ.1.1"] = "My Value"; which function call is equivalent to: msg ["QRF"] += createSegment ("ZZZ");

To access these functions, navigate to an **Edit Channel** page > **Channel Tasks** panel > **Edit Filter** or **Edit Transformer**. On the **Reference** tab, click the **Category** bar > *Message Functions*, among which you will find the *Create Segment* functions. For additional information, see [Reference List](#).



Deleting a Segment

To delete a segment, use the JavaScript delete keyword: `delete msg['ZZZ'];`

In repeating segments, the above code deletes **all** instances of that segment. When you delete a segment, all subsequent segments move up.

Iterating Over Message Segments

To iterate over all segments, follow this example:

```
for each (segment in msg.children()) {
    if (segment.name().toString() == "ORC") {
```

```
    } // Do something...
}
```

To iterate through specifically named segments, use this formula:

```
for each (segment in msg..OBX) {
    // Do something...
}
```

Note that although JavaScript can be used to iterate through message segments, you may find it easier to use the [Iterator Rule / Step](#) instead.

Iterating Over Repeating Fields

To iterate over repeating fields, check the **Handle Repetitions** property. First, use the XML object's **length()** method to get the number of repetitions to iterate over: `var reps = msg ['PV1']['PV1.7'].length();`

Then, use a **for each** loop to iterate over repetitions:

```
for each (attendingDr in msg['PV1']['PV1.7']) {
    lastName = attendingDr['PV1.7.2'].toString();
}
```

Note that although JavaScript can be used to iterate through repeating fields, you may find it easier to use the [Iterator Rule / Step](#) instead.

Adding a New Repeating Field

To add a new repeating field, you need to create an XML object at the segment's field level. Populate the fields, then link to the message at the repeating field level:

```
var newDr = new XML ("<PV1.7/>");
newDr ["PV1.7.1"] = "C3333";
newDr ["PV1.7.2"] = "Jones";
msg ['PV1']['PV1.7'] += newDr;
```

Message Variables

There are a wide variety of different JavaScript contexts throughout NextGen Connect Integration Engine, and each have various variables automatically available from the local scope. Here are some common examples:

- **message**: A string that represents a raw inbound message in native format.
- **msg**: An XML object that represents a transformed version of the inbound message.
- **tmp**: An XML object that represents an outbound message template; available only if an outbound template is defined.
- **connectorMessage**: An instance of the ImmutableConnectorMessage Java class; an internal representation of the message and its attributes.

Built-In Code Templates

These templates are used for *function calls* and *code snippets* for common JavaScript tasks. The templates are available in all JavaScript contexts on the **Reference** tab using the drag-and-drop function and can be extended with custom code templates. For additional information, see [Reference List](#).

Using Java Classes

You can access any Java class in any JavaScript context:

```
var map = new Packages.java.util.LinkedHashMap();
```

The "Packages." at the beginning may be omitted for common top-level package domains, like "com", "net", "org", and "java". To avoid having to type out the fully-qualified class name every time, you can import the package:

```
importPackage(org.apache.commons.io);
FileUtils.getUserDirectory();
```

For classes in custom or non-standard libraries, create a resource (see [Resources Settings Tab](#)) containing your .jar file. Then include it on the channel in the [Library Resources](#) dependencies tab.

Regular Expressions

These are used to search, match, or manipulate text strings based on patterns. In NextGen Connect Integration Engine, regular expressions may be used for:

- String replacement in the Mapper and Message Builder transformer steps
- File-reader filename filter patterns
- Error-condition matching in Alerts
- String methods in JavaScript contexts.

Regular expressions have their own set of rules.

Character	Character Name	Usage	Example
	Vertical bar /pipe	Separates alternatives	this that – matches “this” or “that”
()	Parentheses	Groups characters together	(A a)bc – matches “Abc” or “abc”
^	Caret	Matches characters only at the start position	^ab – matches “abc” but not “lab”
\$	Dollar sign	Matches characters only at the end position	Ab\$ - matches “lab” but not “abc”
?	Question mark	Indicates 0 instances or 1 instance of the previous character	Ab?cd – matches “acd” or “abcd”
*	Asterisk/star	Indicates 0 instances or multiple instances of the previous character	ab*c – matches “ac,” “abc,” “abbc,” “abbcc,” etc.
+	Plus	Indicates 1 instance or multiple instances of the previous character	ab+c – matches “abc,” “abbc,” “abbcc,” etc.
[]	Brackets	Denotes a set of characters that match	[abc] – matches “a,” “b,” or “c” [^abc] – matches any characters except “a,” “b,” or “c” [a-d] – matches “a,” “b,” “c,” or “d”

When you use regular expressions in JavaScript, define them with the regular expression object or with this syntax: [/pattern/attributes](#). Regular expression objects take two strings: *pattern* and *attributes*.

Attributes include “g” (global) and “i” (case insensitive): `var exp = new RegExp("abc", "gi");`

The [test\(\)](#) method matches an expression to a given string: `found = exp.test("I know my abc's");`

JavaScript string object methods that use regular expressions include: [match\(\)](#), [replace\(\)](#), [search\(\)](#), [split\(\)](#)

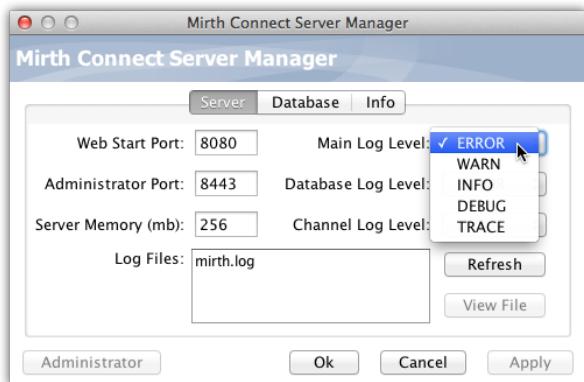
Example:

```
var myString = "Line1\nLine2\nLine3\n";
myString = myString.replace(/\n/g, "\r");
```

Logging with JavaScript

Use [logger.info\(\)](#) and [logger.error\(\)](#) to log information in JavaScript contexts: `logger.info("The value of x =" + x);`

You can view the output of this method in the Mirth Connect Dashboard's Server Log. The default logging level, **ERROR**, is the quickest, having the least amount of overhead. **INFO** and **DEBUG** levels give more details but have more overhead, so are somewhat slower. You can change the logging level via the Mirth Connect **Server Manager** for users who run the latest version of NextGen Connect Integration Engine on a Mac (**Applications > Mirth Connect > mcmanager**) or PC:



Mirth Appliance users can change the logging level from the Mirth Connect **Settings** page via **Applications > Mirth Connect > Manage** from the Appliance UI.

Using the JavaScript Editor

NextGen Connect Integration Engine has a rich JavaScript editor that includes automatic code completion, code folding, multi-line tabs/comment toggles, auto-indentation, bracket matching, macros, and much more.

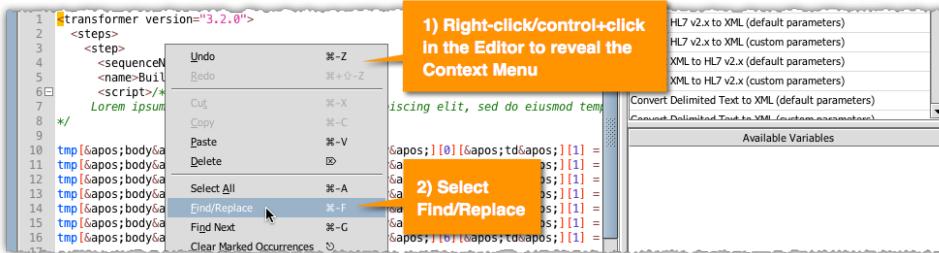
Using the Context Menu in the JavaScript Editor

The Context menu contains items that let you Undo/Redo actions; Select/Copy/Cut/Paste/Delete code; Find/Replace code; collapse/expand sections of code; and characterizes tabs/whitespace/line endings.

Finding/Replacing Code in the JavaScript Editor

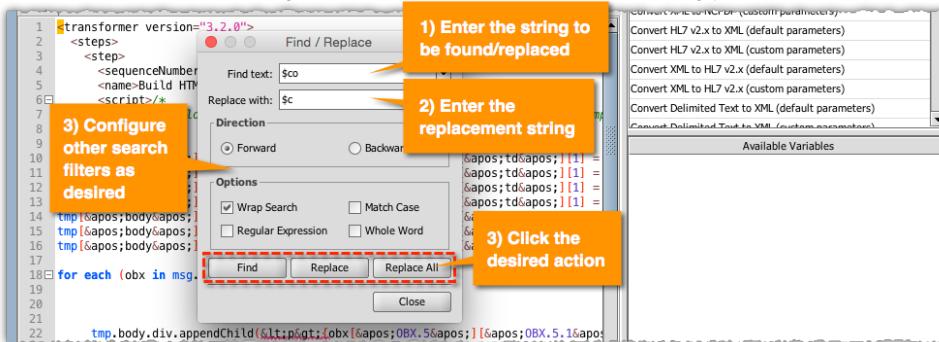
You can use the Context menu to find code, to find and replace certain found code, or to find and replace all found code.

1. Navigate to the page that has the desired JavaScript Editor, and **right-click/control+click** in the Editor.



2. On the Context menu, select **Find/Replace**.

3. On the **Find/Replace** dialog > **Find text** field, enter the code string you want to replace (in this case, **\$co**).



If you are merely searching for all incidents of "\$co" in the code, click the **Find/Replace** dialog's **Find** button > **Close** button. In the JS Editor, all incidents of the entered string are highlighted.

4. In the **Replace with** field, enter your replacement string (in this case, **\$c**).
5. Configure other search filters as desired, and, depending on the situation, click the **Replace** button (to replace only the first incident of the string) or the **Replace All** button (to replace all incidents of the string).



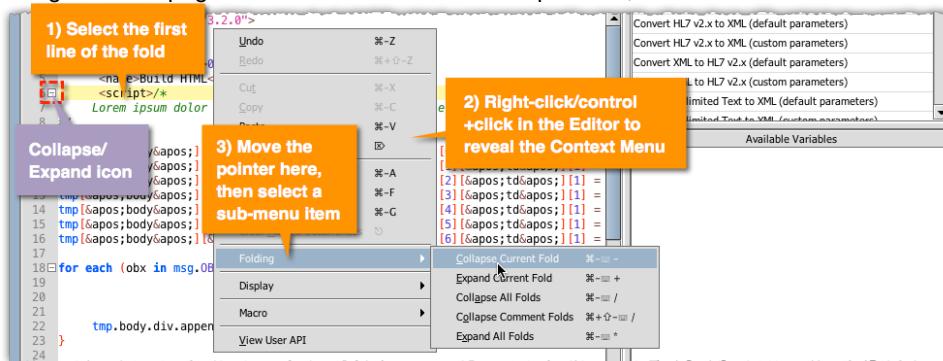
NextGen Connect Integration Engine performs the selected action and highlights the strings in the Editor.

6. Click the **Close** button.

Folding in the JavaScript Editor

Folding, which refers to collapsing/expanding portions of code in the JavaScript Editor, can be accomplished manually by clicking the +/- icon in the left margin of the JS Editor or from the Editor's Context menu (which offers more ways to collapse/expand folds faster than you can do manually). Use the following steps to collapse/expand a fold from the Context menu. (The procedures to perform the other folding actions are similar.)

1. Navigate to the page that has the desired JavaScript Editor, and click the first line of the desired fold.



⚠️ You need to select the first line in the desired fold, which lines are distinguished by a +/- icon in the grey margin of the JS Editor (previous graphic). If you do not select one of these lines, the Editor does not collapse/expand the fold.

You can see how many lines of code are in a fold by moving the pointer into the grey margin below a +/- icon, which action reveals a bracket that extends from the icon down to the last line of the fold.



2. Right-click/control+click in the Editor.

3. On the Context Menu, move the pointer over *Folding*, and select an option on the sub-menu (in this case, *Collapse Current Fold*).

⚠️ The *Collapse Current Fold* action is performed.

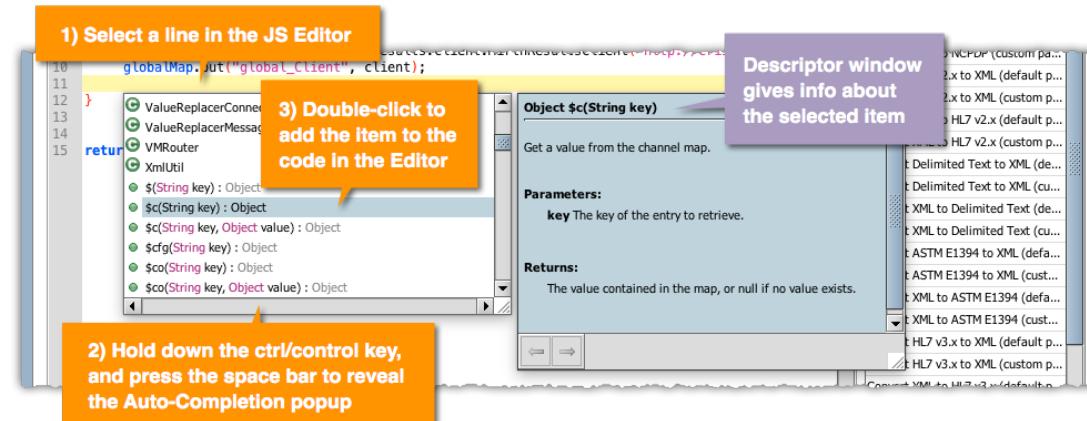


Using the Auto-Completion Popup in the JavaScript Editor

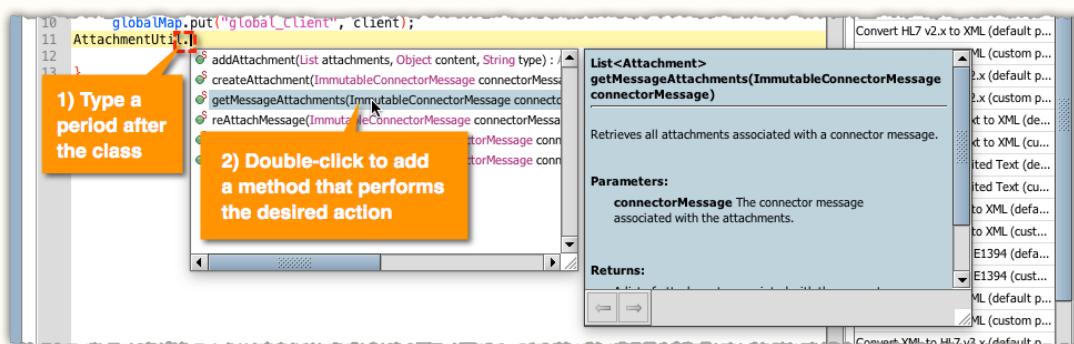
The Auto-Completion popup simplifies channel coding, eliminating the need to go back and forth between the JavaScript Editor and the [User API](#) or even the [Message Template](#) list on the same page as the Editor by putting all coding variables (e.g., code templates, classes, functions, variables) in one popup within the Editor itself.

Accompanying the Auto-Completion popup is a descriptor window that displays, depending on the selected item, its

name and various traits. To display the Auto-Completion popup, click on the beginning/end of a line in the JavaScript Editor, then hold down the **ctrl/control** button, and press the space bar. On the list, double-click the desired item to add it to the code in the Editor.



When adding classes in the JavaScript Editor, double-click the desired class on the Auto-Completion popup, and type a period after the class to reveal a list of its methods (that perform specific actions), then double-click the desired method to add it to the Editor.



Remapping Editor Shortcut Keys

All editor shortcut key mappings can be changed from the [Administrator Settings Tab](#) in the [Settings View](#). You only have to do this once, even if you log into multiple, separate NextGen Connect Integration Engine instances.

Variable Maps

Throughout the [Message Processing Lifecycle](#), your channels and messages have access to various maps. Depending on the scope, the map may only be available in the current channel/connector, or may be globally available across your entire system. These variable maps allow you to store a piece of information that can be used later (in a downstream channel, connector, or somewhere else). A common use for these variables is to provide easy drag-and-drop for connector properties. The [Destination Mappings](#) list will display all available connector/channel map variables for example. They are also used in other ways, such as populating [Custom Metadata Columns](#).

The following variable maps exist throughout NextGen Connect Integration Engine:

Name	JavaScript Variable	Get/Put Shortcut Variable
Response Map	responseMap	\$r
Connector Map	connectorMap	\$co
Channel Map	channelMap	\$c
Source Map	sourceMap	\$s
Global Channel Map	globalChannelMap	\$gc
Global Map	globalMap	\$g
Configuration Map	configurationMap	\$cfg

The table above also shows the precedence of these maps when referencing them in [Velocity](#) or when using the generic lookup function. For additional information, see [The Variable Map Lookup Sequence](#).

Connector Map

This map is isolated to the current message, and the current connector the message is processing through. For example, if you store a connector map variable in Destination 1, you will **not** be able to access that value in Destination 2. This is useful to avoid conflicts among common variable names, and to reduce message storage.

- Get connector map value:
 - `var value = connectorMap.get('key');`
 - `var value = $co('key');`
- Put connector map value:
 - `connectorMap.put('key', 'value');`
 - `$co('key', 'value');`

Channel Map

This map is isolated to the current message as it processes through a channel. If you store a connector map variable in the source connector, you will have access to that value in all subsequent destinations. However when the current message finishes and the next one begins, that next message will **not** have access to the value you stored for the previous message.

The channel map is useful for anything that needs to be shared among multiple destinations, or the source connector and all destinations. For example, you might have one [HTTP Sender](#) destination that makes a request to a remote service, and then in the [Response Transformer](#) you store a particular response HTTP header in the channel map. As long as the next destination connector is in the same [chain](#), it will have access to that channel map variable, and can do something else with it, like include it on a subsequent HTTP request.

- Get channel map value:

- `var value = channelMap.get('key');`
- `var value = $c('key');`
- Put channel map value:
 - `channelMap.put('key', 'value');`
 - `$c('key', 'value');`

Source Map

This map is isolated to the current message as it processes through a channel. Unlike the channel map however, this one is **read only**. The [Source Connector](#) or an upstream process can inject source map variables. For example, the [File Reader](#) will automatically inject the "originalFilename" variable.

- Get source map value:
 - `var value = sourceMap.get('key');`
 - `var value = $s('key');`

Response Map

This map is isolated to the current message as it processes through a channel. Unlike the channel map, this one is specifically used for storing Response objects. When a destination finishes processing, its Response will automatically be stored in the response map. Subsequent destinations and the [Postprocessor Script](#) will have access to these values. The source connector can also use values stored in the response map to send responses back to the originating system (set in the [Source Settings](#)).

- Get response map value:
 - `var value = responseMap.get('key');`
 - `var value = $r('key');`
- Put response map value:
 - `responseMap.put('key', 'value');`
 - `$r('key', 'value');`

Global Channel Map

This map is isolated to a specific channel, but *across* multiple messages. That means you can store a value during a message processing lifecycle, and it will be available during the lifecycle of the next message. You can also store global channel map values in the [channel scripts](#).

This map is useful for storing stateful, non-serializable objects like a database Connection. It is in-memory only, meaning that if NextGen Connect Integration Engine is restarted, the entries in this map are not preserved anywhere. It is also a concurrent map, which means that "null" values cannot be stored in it.

- Get global channel map value:
 - `var value = globalChannelMap.get('key');`
 - `var value = $gc('key');`
- Put global channel map value:
 - `globalChannelMap.put('key', 'value');`
 - `$gc('key', 'value');`



By default the "Clear global channel map on deploy" option is enabled on the [Summary Tab](#). You may want to uncheck this if you want the global channel map to remain unchanged when you redeploy the channel.

Global Map

This map is available across your *entire* server, across all channels and all messages. That means you can store a value during message processing in one channel, and use that value from a different channel, or somewhere else like an [Alert](#). You can also store global map values in the [global scripts](#).

Like the global channel map, this map is useful for storing stateful, non-serializable objects like a database Connection. It is in-memory only, meaning that if NextGen Connect Integration Engine is restarted, the entries in this map are not preserved anywhere. It is also a concurrent map, which means that "null" values cannot be stored in it.

- Get global map value:
 - `var value = globalMap.get('key');`
 - `var value = $g('key');`
- Put global map value:
 - `globalMap.put('key', 'value');`
 - `$g('key', 'value');`



By default the "Clear global map on redeploy" option is enabled on the [Server Settings Tab](#). You may want to disable this if you want the global map to remain unchanged when you redeploy all channels.

Configuration Map

This map is also available across your *entire* server, across all channels and all messages. Like the global map, that means you can use the values from the configuration map in any channel, or somewhere else like an [Alert](#). Unlike the global map however, this map is editable only from the [Configuration Map Settings Tab](#), and is read-only from the perspective of channels / messages. The values are also String key/values only.

This map is useful for global, static settings you want to persist across restarts of NextGen Connect Integration Engine. For example, you could store a variable like "clientAdtPort" and then use [Velocity](#) to reference that variable ("{\$clientAdtPort}") in a [TCP Listener](#). That way you can export the channel on one NextGen Connect Integration Engine installation, import it into a completely different installation, and then you would not have to edit anything in the channel settings as long as the configuration map is set on both instances.

- Get configuration map value:
 - `var value = configurationMap.get('key');`
 - `var value = $cfg('key');`
- Put configuration map value:
 - `configurationMap.put('key', 'value');`
 - `$cfg('key', 'value');`

The Variable Map Lookup Sequence

In many cases when referencing map variables in NextGen Connect Integration Engine , you don't call out to a *specific* map, but instead use the generic lookup function:

- `var value = ${'variableName'};`

Or, you might reference a variable using [Velocity Variable Replacement](#):

- `${variableName}`

When you do this, NextGen Connect Integration Engine will automatically look that key up in *all available* maps. That may only be the configuration/global map (in the case of the [global scripts](#)), or it may be *all/maps* (in the case of a [filter / transformer script](#)). This sequence is followed:

- Response Map
- Connector Map
- Channel Map
- Source Map
- Global Channel Map
- Global Map
- Configuration Map

For example, if you have stored a variable called "dataSource" in both the Connector Map and the Global Channel Map, the one from the Connector Map will be used. If you want the value specifically from the Global Channel Map instead, you can use the map-specific get function shown in the [Variable Maps](#) section:

- `var dataSource = globalChannelMap.get('dataSource');`

Attachment JavaScript Functions

[Attachment Handlers](#) are used to extract a portion of a message (or the entire message) and store it separately as an attachment. The portion of the message that was extracted is replaced with an attachment replacement token. When you use a destination connector to send a message downstream, NextGen Connect Integration Engine will automatically take the message and replace any attachment tokens with the actual attachment data (unless you have Reattach Attachments turned off in the [Destination Settings](#)).

Throughout the message lifecycle you can retrieve, modify, and add new attachments with built-in helper functions.

- [Built-In Attachment Functions](#)
- [The AttachmentUtil Class](#)
- [The Attachment Object](#)
- [Examples](#)

Built-In Attachment Functions

- **getAttachmentIds()**
 - Get a List containing the IDs of all Attachments associated with this message. Uses the current **connectorMessage** variable.
- **getAttachmentIds(channelId, messageId)**
 - Get a List containing the IDs of all Attachments associated with any channel / message.

• channelId	The ID of the channel associated with the attachments.
• messageId	The ID of the message associated with the attachments.
- **getAttachments(base64Decode)**
 - Get List of Attachments associated with this message. This will get all attachments that have been added in the source and destination(s).

• base64Decode	If true, the content of each attachment will first be Base64 decoded for convenience.
----------------	---
- **getAttachment(attachmentId, base64Decode)**
 - Get a specific Attachment associated with this message. Uses the current **connectorMessage** variable.

• attachmentId	The ID of the attachment to retrieve.
base64Decode	If true, the content of the attachment will first be Base64 decoded for convenience.
- **getAttachment(channelId, messageId, attachmentId, base64Decode)**
 - Get a specific Attachment associated with any channel / message. You can use this to retrieve an attachment from a completely different channel.

• channelId	The ID of the channel to retrieve the attachment from.
messageId	The ID of the message to retrieve the attachment from.
attachmentId	The ID of the attachment to retrieve.
base64Decode	If true, the content of the attachment will first be Base64 decoded for convenience.
- **addAttachment(data, type, base64Encode)**
 - Add attachment (String or byte[]) to the current message.

• data	The data to insert as an attachment. May be a string or byte array.
type	The MIME type of the attachment.
base64Encode	If true, the content will be Base64 encoded for convenience. If the content you're passing in is not already Base64 encoded, you should pass in true for this

argument.

- **updateAttachment(attachment, base64Encode)**
- **updateAttachment(attachmentId, data, type, base64Encode)**
- **updateAttachment(channelId, messageId, attachment, base64Encode)**
- **updateAttachment(channelId, messageId, attachmentId, data, type, base64Encode)**
 - Updates an attachment associated with the current connector message, or with any message from any channel.

• attachment	The Attachment object to update.
attachmentId	The unique ID of the attachment to update.
channelId	The ID of the channel the attachment is associated with.
messageId	The ID of the message the attachment is associated with.
data	The attachment content (must be a string or byte array).
type	The MIME type of the attachment.
base64Encode	If true, the content will be Base64 encoded for convenience. If the content you're passing in is not already Base64 encoded, you should pass in true for this argument.

The AttachmentUtil Class

All of the [built-in attachment functions](#) are also available from the AttachmentUtil utility class available from the [User API](#). In addition to this, AttachmentUtil has some extra methods available that you can use to re-attach attachment data into a message string.

For more information, check out the AttachmentUtil class in your locally hosted User API: [The User API \(Javadoc\)](#)

You can also view the documentation for this class here: <http://javadocs.mirthcorp.com/connect/3.6.0/user-api/com/mirth/connect/server/userutil/AttachmentUtil.html>

The Attachment Object

When using these methods, you will likely be working with a special class called Attachment. This class is documented in the [User API](#). A few of the available methods are documented here:

Create a new Attachment

- **new Attachment()**
- **new Attachment(id, content, type)**
- **new Attachment(id, content, charset, type)**

• id	The unique ID of the attachment.
content	The content (String or byte array) to store for the attachment.
charset	If the content passed in is a string, this is the charset encoding to convert the string to bytes with. If the content is a String and a charset is not used, UTF-8 will be used as the charset.
type	The MIME type of the attachment.

Retrieve Attachment Content



When you retrieve an attachment using the [built-in attachment functions](#) or [AttachmentUtil](#), you may want to pass in true for the **base64Decode** argument so that the attachment content is already decoded.

- **getContent()**
 - Retrieves the raw byte array content of the attachment. Note that this may be the actual raw bytes of the attachment, or it may be the Base64 byte array representation. See the note above.
- **getContentString()**
- **getContentString(charset)**
 - Returns the content of the attachment as a string, using the specified charset encoding. If not specified, UTF-8 will be used. Note that this may be the raw string value of the attachment, or it may be the Base64 string value. See the note above.

Examples

Modify a previously attached CCD attachment:

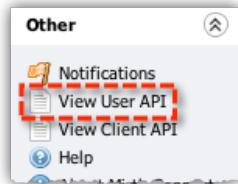
```
var ccd = new XML(getAttachment(attachmentId, true).getContentString());  
  
ccd.id.@root = 'testing';  
  
updateAttachment(attachmentId, ccd.toString(), 'text/xml', true);
```

Read in a PDF from the filesystem and add it as an attachment:

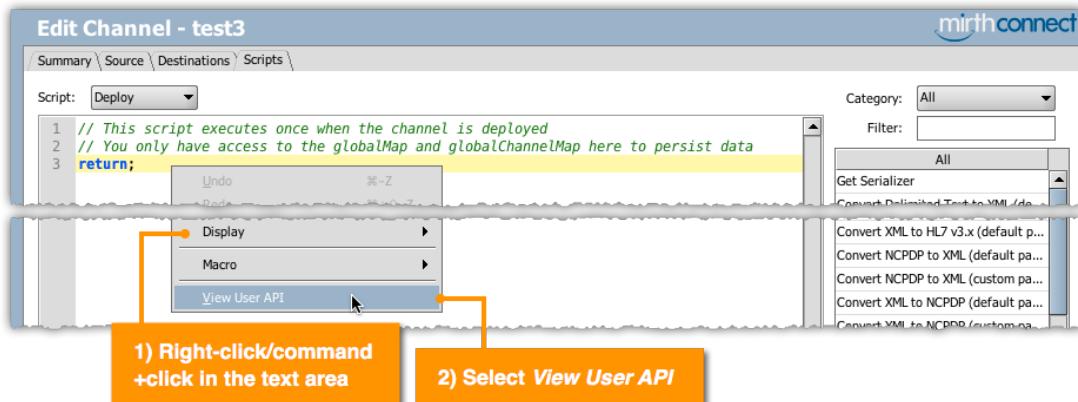
```
var fileBytes = FileUtil.readBytes('/path/to/file.pdf');  
  
// Pass in true for Base64Encode, since the content isn't already Base64 encoded  
addAttachment(fileBytes, 'application/pdf', true);
```

The User API (Javadoc)

The User API is a collection of classes and methods that help you interact with channels / messages, and also provides helper methods for common tasks such as date formatting. Wherever you are editing JavaScript code in NextGen Connect Integration Engine, you can view the API by clicking the **View User API** function in the **Other** panel, which is available on all Mirth Connect Administrator views.



You can also right-click/command+click in the text area of any [JavaScript Editor](#), and on the drop-down menu, select **View User API**.



The Javadoc appears in your default web browser, in which you can select classes to view their method signatures and descriptions.

All Classes
Packages

com.mirth.connect.plugins.httpauth.util
com.mirth.connect.server.util
com.mirth.connect.util

DateUtil

java.lang.Object
com.mirth.connect.server.util.DateUtil

public class DateUtil
extends Object

Provides date/time utility methods.

Method Summary

Methods	Static Methods	Concrete Methods
Method and Description		
static String convertDate(String inPattern, String outPattern, String date)		Parses a date string according to a specified input pattern, and formats the date back to a string according to a specified output pattern.
static String formatDate(String pattern, Date date)		Formats a java.util.Date object into a string according to a specified pattern.
static String getCurrentDate(String pattern)		Formats the current date into a string according to a specified pattern.
static Date getDate(String pattern, String date)		Parses a date string according to the specified pattern and returns a java.util.Date object.
Methods inherited from class java.lang.Object		
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait		

Method Detail

getDate

```
public static Date getDate(String pattern,
                           String date)
                           throws Exception
```

Parses a date string according to the specified pattern and returns a java.util.Date object.

Parameters:
pattern - The SimpleDateFormat pattern to use (e.g. "yyyyMMddHHmmss").
date - The date string to parse.

Returns:



You can also view the API via this link: <http://javadocs.mirthcorp.com/connect/3.6.0/user-api/>

Velocity Variable Replacement

The [Apache Velocity](#) template engine is used throughout NextGen Connect Integration Engine to allow dynamic variables to be injected into property fields. For example, you can use a transformer to programmatically select a TCP address/port to send to and store those values in the [channel map](#), and then use a Velocity template to inject those variables into the [TCP Sender](#) settings.

Basic Syntax

Basic syntax for a Velocity reference is as follows:

```
${variableName}
```

The brackets may be omitted if the identifier starts with a letter and contains only letters, numbers, hyphens, or underscores:

```
$variableName
```

The variable will be looked up in all available maps, according to [The Variable Map Lookup Sequence](#). The string representation of the value will then replace the Velocity reference. You can also access properties and methods from context variables:

```
${myArray.length}  
${myList.size()}  
${myObject.customMethod( 'param' )}
```

If a property has a corresponding *getter* method (like `getValue()`), the engine will automatically find that method when you attempt to access the property. Therefore these may be equivalent:

```
${myObject.value}  
${myObject.getValue()}
```

If the context variable doesn't exist, or if the value returned by the reference evaluation is *null*, no replacement will be done, so the final template will still have your "\${varName}" string within. In these cases you can put an exclamation mark after the dollar sign to tell the engine to replace null values with an empty string instead:

```
${!thisValueIsNull}
```

Conditional Statements

Velocity supports if..else statements, with the #if / #elseif / #else directives:

```
There #if($list.size()==1)is#${else}are#end ${list.size()} total value#if($list.size()!=1)s#end
```

The curly brackets ("{}") are only needed if the if/else/end might be confused with template data immediately before or after.

For Loops

Velocity supports iterating through Lists / Collections / Arrays with the #foreach directive:

```
<table>
#foreach ($item in $list)
    <tr>
        <td>${item.name}</td>
    <tr>
#end
</table>
```

NextGen Connect Command Line Interface

The Command Line Interface is an alternate, lightweight way to interact with your NextGen Connect Integration Engine server. Not everything you can do through the Administrator is available in the CLI (like editing channels), but common monitoring and management operations are supported:

- List deployed status / statistics of all channels
- Deploy/undeploy/start/stop/halt/pause/resume channels
- Enable/disable/remove channels
- Import/export channels, alerts, code templates, libraries, or the entire server configuration
- And more...

The CLI comes packaged with the standard NextGen Connect Integration Engine distribution. It is also available as a standalone client from the [Downloads page](#). If using the standalone client, simply extract the archive in a location of your choice.

Running the Command Line Interface

The CLI has both interactive and non-interactive modes. With the interactive mode, you can choose to store your username/password in a file, or enter your username/password every time you launch the CLI.

From a terminal / shell:

- In a terminal / shell, run the **mccommand** executable script, with the following options:
 - `mccommand -a https://localhost:8443 -u user -p pass`
- Change the address, username, and password as necessary

You can also set the address / username / password in the "conf/mirth-cli-config.properties" file, so that you don't need to enter it every time.

Once connected, you'll see a welcome prompt:

```
Connected to Mirth Connect server @ https://127.0.0.1:8443 (3.5.0.8232)
$
```

From here, you can type **help** to view all available commands.

Using Non-interactive Scripting

The CLI supports a **-s** option where you can pass in a script file containing multiple commands. This can be used to programmatically call the CLI from a scheduled job or anywhere else.

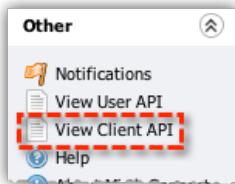
From a terminal / shell:

- In a terminal / shell, run the **mccommand** executable script, with the following options:
 - `mccommand -a https://localhost:8443 -u user -p pass -s script.txt`
- Change the address, username, and password as necessary

You can also add **script=script.txt** to the "conf/mirth-cli-config.properties" file, so that it will automatically be used any time the CLI is executed.

NextGen Connect REST API

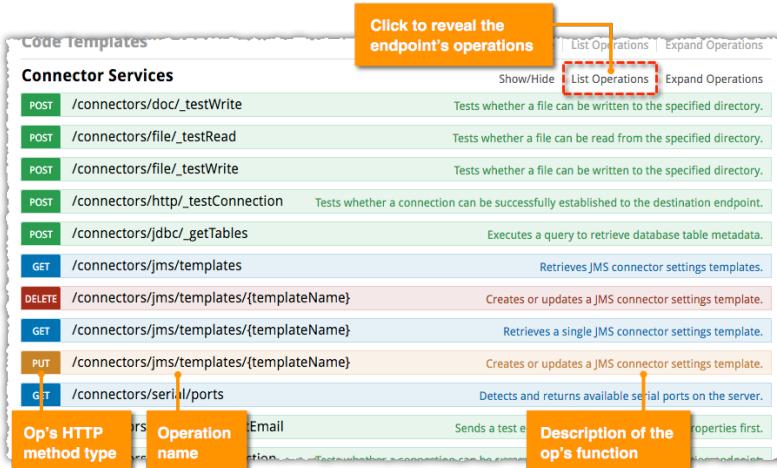
The [Administrator](#) and [Command Line Interface](#) both use a well-defined API to communicate with the NextGen Connect Integration Engine server. You can also use the same API to create custom integrations of your own. The API is documented at <https://localhost:8443/api> (change the IP / port as needed), or you can click on View Client API in Administrator via [Other Tasks](#):



- Click the **View Client API** action to open the API documentation in your default browser:

A screenshot of the MirthConnect API documentation. The left sidebar lists various API endpoints: Alerts, Channel Deployment Operations, Channel Groups, Channel Statistics, Channel Status Operations, Channels, Code Templates, Connector Services, Database Tasks, Events, Extension Services, Extensions, Messages, Server Configuration, System Information and Statistics, Usage Data, and Users. A red dashed box surrounds the first 12 items. An orange callout box points to the 'Channel Deployment Operations' item with the text 'The Client API's endpoints'. Another orange callout box points to the sidebar with the text 'Shows/Hides the previously viewed item type (whether List Ops or Expand Ops)'. Two more orange callout boxes point to the right side of the page with the text 'Displays the endpoint's ops' and 'Expands the endpoint's ops' respectively. The main content area shows a list of operations for each endpoint, with 'Show/Hide', 'List Operations', and 'Expand Operations' buttons for each row.

- Click an endpoint's **List Operations** action to view its operations. This list shows the operation's HTTP method type, name, and description of function.



- Click an endpoint's **Expand Operations** action to view the details of all its operations. If desired, you can click a specific operation to view only that op's details. Each operation's details include response class, model and model schema, response content type, parameters fields (channelId, channelName, body), and **Try it out!** button so you can try the op in your own instance.

Endpoint: Response Class (Status 200)

Operation: Operation's function

Response class: Toggle to view model/model schema

Model Schema: Select a response content type

Parameters: Input your parameters

Try it out!: Implement the op in your instance of Mirth Connect

Authentication

NextGen Connect Integration Engine supports both session-based cookie authentication and [Basic Authentication](#).

Basic Authentication:

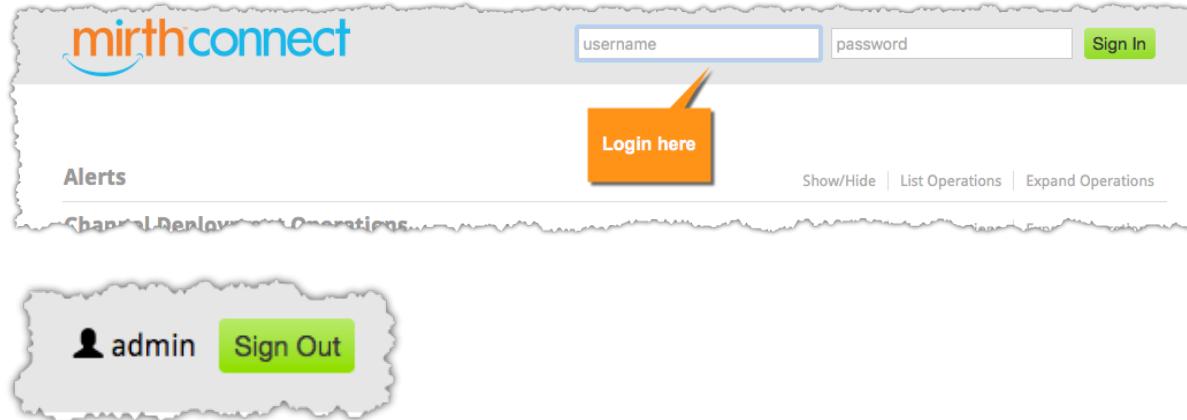
- Simply include an Authorization header on all API requests, with basic credentials. Example (for admin /admin):
 - Authorization: Basic YWRtaW46YWRtaW4=

Session-Based Authentication:

- First, invoke the [POST /users/_login](#) endpoint, passing in your login credentials. If successful, the server will respond with a cookie (Set-Cookie header) like the following:
 - Set-Cookie: JSESSIONID=uy5qrxtx91le36ernybizstps; Path=/api; Secure
- Invoke the actual API endpoints of your choice, passing in the same cookie as a header:
 - Cookie: JSESSIONID=uy5qrxtx91le36ernybizstps; Path=/api; Secure
- Once you're done, make sure to call the [POST /users/_logout](#) endpoint, making sure to pass in the same cookie.

Session-based Authentication is preferred since you only need to transmit your login credentials once.

Note that the API documentation page invokes the same endpoint automatically when you login at the top:



Installation Directory

This is the location you installed or extracted NextGen Connect Integration Engine into. Typically only administrators need to have access to this, and even then usually only once when first setting up. For example this is where you can edit initial configuration to [change the database](#) NextGen Connect Integration Engine points to. This section is separated into the following topics:

- Application Data Directory
- Configuration Directory
- Other Files and Folders

Application Data Directory

This directory stores configuration files and temporary data created by NextGen Connect Integration Engine after starting up. Usually the name of this directory is "appdata" and it resides directly inside your installation directory. However the name and location of this directory can be modified from [mirth.properties](#). The main files/folders of note here are as follows:

- [configuration.properties](#)
- [extension.properties](#)
- [keystore.jks](#)
- [server.id](#)
- [temp](#)

configuration.properties

This stores the current state of the [Configuration Map](#). The file is formatted as a standard properties file:

```
key = value  
  
# This is a comment for key "clientAddress"  
clientAddress = 10.0.1.123
```

The configuration map is edited from the [Configuration Map Settings Tab](#). If you instead edit the properties file manually, NextGen Connect Integration Engine will **not** pick up those changes until you restart the server.

extension.properties

This stores the current enabled/disabled state of all installed extensions. Typically you do not need to edit this file, as enabling / disabling can be done through the [Extensions View](#).

keystore.jks

This is a critical file that stores your server's local certificate keypair (for the web server and API), and also the secret key used for encrypting message data, exports, and anything else. Note that usually the name of this file is "keystore.jks" and it resides inside of appdata, but the keystore name and location can be modified from [mirth.properties](#).

! If you plan on making any changes to this file, **BACK IT UP** first! If you lose this file, any data (messages, exports, etc.) encrypted with it will be lost forever!

Changing The Server Certificate

When NextGen Connect Integration Engine starts up for the first time, it will automatically create a new self-signed certificate, which it will use for web server and secure API access. After installing NextGen Connect Integration Engine, you should replace this with an appropriate company certificate signed by a Certificate Authority (CA). Use the following steps to install a new certificate:

- **BACK UP** your current keystore.jks file just in case.
- Have your new keypair ready to import in a PKCS #12 format. Example: myservercert.p12
- In a terminal / shell, navigate to the location of your keystore.jks file.
- Use this command:
 - ```
keytool -importkeystore -srckeystore myservercert.p12 -srcstoretype PKCS12 -
srcstorepass mystorepass -srckeypass mykeypass -srcalias myalias -destkeystore keystore.
jks -deststoretype JCEKS -deststorepass 8luWxplDtB -destkeypass 8luWxplDtB -destalias
mirthconnect
```

Make sure to change the file names, passwords, and local alias as necessary. The **-destalias** option must be "mirthconnect" though in order to overwrite the current certificate.

- Restart the NextGen Connect Integration Engine server.

### **server.id**

This is the unique ID for your server instance. It is auto-generated when NextGen Connect Integration Engine starts up for the first time.

### **temp**

This stores all temporary files created by the NextGen Connect Integration Engine server. Note that usually the name of this file is "temp" and it resides inside of appdata, but the name and location can be modified from [mirth.properties](#).

## Configuration Directory

---

This directory contains the main configuration files NextGen Connect Integration Engine needs in order to start up correctly. Modifying the configuration requires a restart of NextGen Connect Integration Engine. The following files are found here:

- [The dbdrivers.xml File](#)
- [The log4j.properties File](#)
- [The log4j-cli.properties File](#)
- [The mirth.properties File](#)
- [The mirth-cli-config.properties File](#)

## The dbdrivers.xml File



This file has been deprecated as of version 3.8. Instead, these driver entries are stored in the database, and can be configured from either the Database Reader or Database Writer settings. More information in the [Editing Database Drivers](#) section of the [Database Reader](#) page.

This file is used by the [Database Reader](#) and [Database Writer](#) connectors to populate the Driver drop-down menu. The default values are as shown:

```
<!--
 Database driver information
 class = the driver class name, cannot be empty
 name = database driver name to be displayed as, cannot be empty
 template = the template for creating the database connection, cannot be empty
 selectLimit = defines the select statement used for retrieving column information, empty
means use the generic query (which could be slow)
 alternativeClasses = A comma-separated list of legacy driver classes (optional).
-->
<drivers>
 <driver class="com.mysql.cj.jdbc.Driver" name="MySQL" template="jdbc:mysql://host:port
/dbname" selectLimit="SELECT * FROM ? LIMIT 1" alternativeClasses="com.mysql.jdbc.Driver" />
 <driver class="oracle.jdbc.driver.OracleDriver" name="Oracle" template="jdbc:oracle:thin:
@host:port:dbname" selectLimit="SELECT * FROM ? WHERE ROWNUM < 2" />
 <driver class="org.postgresql.Driver" name="PostgreSQL" template="jdbc:postgresql://host:port
/dbname" selectLimit="SELECT * FROM ? LIMIT 1" />
 <driver class="net.sourceforge.jtds.jdbc.Driver" name="SQL Server/Sybase" template="jdbc:
jtds:sqlserver://host:port/dbname" selectLimit="SELECT TOP 1 * FROM ?" />
 <driver class="org.sqlite.JDBC" name="SQLite" template="jdbc:sqlite:dbfile.db" selectLimit="
SELECT * FROM ? LIMIT 1" />
</drivers>
```

### Adding a new entry to dbdrivers.xml

Simply add a new **<driver>** node to the list. The following attributes must be specified:

- **class:** The fully-qualified class name of the JDBC Driver implementation.
- **name:** The name you want to appear in the Driver drop-down menu.
- **template:** The JDBC URL template you want to have populated when you click on the **Insert URL Template** button in a Database connector.
- **selectLimit:** An optional query that selects one row from a given table. If specified, this will be used when attempting to retrieve database metadata.
- **alternativeClasses:** A comma-separated list of legacy driver classes (optional).

## The log4j.properties File

This file is used to tell NextGen Connect Integration Engine where to write log files out to, and how verbose the logs should be. It's split up into the following sections:

```
log4j.rootLogger = ERROR,stdout,fout
```

This is the "root" logger property, used to denote the appenders to include, and the root **level**. The following levels can be set, in order from least to most logging:

- OFF
- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- TRACE
- ALL

```
stdout appender
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern = %-5p %d [%t] %c: %m%n
```

This appender ensures that all logging can redirect to STDOUT, if you're running NextGen Connect Integration Engine from an attached shell.

```
file appender
dir.logs = logs
log4j.appender.fout = org.apache.log4j.RollingFileAppender
log4j.appender.fout.File = ${dir.logs}/mirth.log
log4j.appender.fout.MaxFileSize = 500KB
log4j.appender.fout.MaxBackupIndex = 20
log4j.appender.fout.layout = org.apache.log4j.PatternLayout
log4j.appender.fout.layout.ConversionPattern = %-5p %d [%t] %c: %m%n
```

This appender writes logs out to files in the "logs" directory. The directory and file names can be modified here. You can also modify the maximum file size and maximum file count to limit how much log data to retain. Finally, the pattern can be modified to change how the log entries will be formatted.

```
splash screen
log4j.logger.com.mirth.connect.server.Mirth = INFO

Mirth Connect server logging
log4j.logger.com.mirth.connect.donkey.server.channel.RecoveryTask = INFO
```

These entries help display certain helpful information when logging in, and when messages get recovered.

```
Mirth Connect channel logging
log4j.logger.transformer = DEBUG
log4j.logger.preprocessor = DEBUG
log4j.logger.postprocessor = DEBUG
log4j.logger.deploy = DEBUG
log4j.logger.undeploy = DEBUG
```

```
log4j.logger.filter = DEBUG
log4j.logger.db-connector = DEBUG
log4j.logger.js-connector = DEBUG
log4j.logger.attachment = DEBUG
log4j.logger.batch = DEBUG
log4j.logger.response = DEBUG
```

These determine what level verbosity to use when logging from within various JavaScript contexts. Note that these can also be set from the [Server Manager](#).

```
SQL Logging
log4j.logger.java.sql = ERROR
```

Turn the verbosity up here to see more information about which database statements are being executed across the server.

## The log4j-cli.properties File

This is the same as [the log4j.properties file](#), except that it's specific to the [Command Line Interface](#). You can edit the root logger level, and the STDOUT appender settings:

```
log4j.rootCategory=ERROR, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%-5p %d [%t] %c: %m%n
```

## The mirth.properties File

This is the main configuration file that tells NextGen Connect Integration Engine where to store application data, what web server ports to listen on, and which database to connect to. There are also other security and encryption options that may be set. The following properties are supported:

Property	Default Value	Description
<b>Directories</b>		
dir.appdata	appdata	The location of the <a href="#">Application Data Directory</a> .
dir.tempdata	\$(dir.appdata)/temp	The location of the temporary files directory, by default set inside of the <a href="#">Application Data Directory</a> .
<b>Ports</b>		
http.port	8080	<p>The HTTP port to make the web server available from. This is used to access the launch page and download signed client resources from.</p> <p>If this property is omitted or commented out, the web server will only start up on the HTTPS port.</p>
https.port	8443	The HTTPS port to make the web server available from. This is used to access the secure launch page, web dashboard, and all <a href="#">REST API</a> traffic (which includes the <a href="#">Administrator</a> and <a href="#">CLI</a> ).
<b>Password Requirements</b>		
password.minLength	0	Minimum password length, 0 for no minimum.
password.minUpper	0	Minimum uppercase characters, 0 for no minimum.
password.minLower	0	Minimum lowercase characters, 0 for no minimum.
password.minLength	0	Minimum numeric characters, 0 for no minimum.
password.minLength	0	Minimum special characters, 0 for no minimum.
password.retryLimit	0	Maximum number of times a user may retry a failed login, 0 for no maximum. If specified, the lockout period must be specified as well.
password.lockoutPeriod	0	Amount of time (in hours) to lockout user when the retry limit has been exceeded, 0 for no lockout.
password.expiration	0	After this amount of time (in days), passwords will expire.
password.gracePeriod	0	If user's password is expired, the amount of time (in days) to give the user to change password after the next login.
password.reusePeriod	0	The amount of time (in days) to wait before users can change passwords to one that was used in the past. Set to 0 to always allow reuse, and to -1 to never allow users to reuse the same password.
password.reuseLimit	0	The amount of times to allow users to reuse the same

		password. Set to 0 for no limit, and to -1 to never allow users to reuse the same password.
<b>Keystore</b>		
keystore.path	`\${dir.appdata}/keystore.jks	The location of the keystore file, which houses the server certificate and the secret encryption key. This is usually located in the <a href="#">Application Data Directory</a> .
keystore.storepass	81uWxplDtB	The password for the keystore file itself. <b>It's a good idea to change this from the default value.</b> On first startup when the keystore is created, if this value equals the default it will be replaced with a randomly generated password.
keystore.keypass	81uWxplDtB	The password for the keys within the keystore, including the server certificate and the secret encryption key. <b>It's a good idea to change this from the default value.</b> On first startup when the keystore is created, if this value equals the default it will be replaced with a randomly generated password.
keystore.type	JCEKS	The type of keystore. Usually this should not be changed.
<b>Server</b>		
http.contextpath	/	The base context path of the web server.
server.url		If set, this URL will be set in the webstart JNLP file, so that when users launch the Administrator it will be shown in the server URL field by default.
http.host	0.0.0.0	The network interfaces to listen on for the web server HTTP port. Use 0.0.0.0 for all interfaces.
https.host	0.0.0.0	The network interfaces to listen on for the web server HTTPS port. Use 0.0.0.0 for all interfaces.
server.id.ephemeral	false	If true, the server will auto-generate a server ID on startup. Otherwise, the appdata/server.id file will be used.
server.startupdeploy	true	Determines whether or not channels are deployed on server startup.
server.includecustomlib	false	Determines whether libraries in the custom-lib directory will be included on the server classpath. To reduce potential classpath conflicts you should create Resources and use them on specific channels/connectors instead, and then set this value to false.
administrator.maxheapsize	512m	The default maximum <b>client-side</b> heap size to set in the Java Web Start JNLP. Users may override this on the launch page. Note that this is <b>not</b> the same as the server-side max heap size.
administrator.maxheapsizeoptions	256m,512m,1g,2g	The client-side max heap size options to give the user from the launch page and from the <a href="#">Server Manager</a> .
configurationmap.location	file	Determines whether the configuration map will be stored as a file, or in the database. Valid values: file, database
configurationmap.path		The location of the <a href="#">configuration map</a> properties file. Usually

	<code>\$(dir.appdata)/configuration.properties</code>	this is in the <a href="#">Application Data Directory</a> .
<code>extension.properties.provider</code>		The fully-qualified class (extending <code>ExtensionStatusProvider</code> ) that controls extension enabled /disabled flags. If absent or set to "file", the default behavior is used, reading from <code>\$(dir.appdata)/extension.properties</code> .
<code>donkey.statsupdateinterval</code>	1000	The interval on which to update channel statistics across all channels.
<code>license.key</code>		If you have any <a href="#">commercial extensions</a> installed, enter your license key here. Contact the help desk through our <a href="#">Success Community</a> to get a license key.
<code>rhino.optimizationlevel</code>	-1	Sets the optimization level for Rhino (the JavaScript engine), 1 indicates that the engine should run in interpretive mode, which is less efficient but allows very large/complex scripts to compile. Set it to 0 or 1-9 to increase optimization, which may increase performance at the cost of limited script complexity.
<code>rhino.languageversion</code>	ES6	The ECMAScript/JavaScript version that the Rhino engine should use. Valid values: ES6, DEFAULT, 1.0-1.8.
<code>server.api.sessionstore</code>	false	If set to true, the web server sessions are stored in the database.
<code>server.api.sessionstoretable</code>	<code>sessiondata</code>	The table name to use for web server session data.
<code>server.api.sessioncache</code>	default	If absent or set to "default", an in-memory L1 cache is stored on each server for session data. If set to "none", then no cache is used, and every request for session data goes directly to the database.
<code>server.api.sessionmaxinactiveinterval</code>	259200 (72 hours)	The maximum amount of time (in seconds) that a session can be idle/inactive before it is invalidated and evicted from the cache. Minimum is 60 seconds.
<b>Security</b>		
<code>https.client.protocols</code>	TLSv1.3,TLSv1.2, TLSv1.1	The protocols to support by default for all TLS/SSL/HTTPS client traffic. <b>Changing this property could leave your server vulnerable to certain SSL-based attacks.</b> Note that the <a href="#">SSL Manager</a> allows you to tweak protocol settings on a per-connector basis, rather than having to change the value for the entire server.
<code>https.server.protocols</code>	TLSv1.3,TLSv1.2, TLSv1.1,SSLv2Hello	The protocols to support by default for all TLS/SSL/HTTPS server traffic. <b>Changing this property could leave your server vulnerable to certain SSL-based attacks.</b> Note that the <a href="#">SSL Manager</a> allows you to tweak protocol settings on a per-connector basis, rather than having to change the value for the entire server.
<code>https.ciphersuites</code>	<a href="#">See Below</a>	The cipher suites to support by default for all TLS/SSL /HTTPS server traffic. <b>Changing this property could leave your server vulnerable to certain SSL-based attacks.</b>

		Note that the <a href="#">SSL Manager</a> allows you to tweak cipher suite settings on a per-connector basis, rather than having to change the value for the entire server.
https.ephemeraldhkeysize	2048	The key size to use for all generated Diffie-Hellman parameters. <b>Changing this property could leave your server vulnerable to certain SSL-based attacks.</b>
server.api.allowhttp	false	If enabled, API access will be allowed through the regular HTTP port. Generally you should <b>not</b> enable this except for testing / development purposes.
server.api.accesscontrol.alloworigin	*	This value will be set on the Access-Control-Allow-Origin HTTP header on all API responses.
server.api.accesscontrol.allowcredentials	false	This value will be set on the Access-Control-Allow-Credentials HTTP header on all API responses.
server.api.accesscontrol.allowmethods	GET, POST, DELETE, PUT	This value will be set on the Access-Control-Allow-Methods HTTP header on all API responses.
server.api.accesscontrol.allowheaders	Content-Type	This value will be set on the Access-Control-Allow-Headers HTTP header on all API responses.
server.api.accesscontrol.exposeheaders		This value will be set on the Access-Control-Expose-Headers HTTP header on all API responses.
server.api.accesscontrol.maxage		This value will be set on the Access-Control-Max-Age HTTP header on all API responses.
server.api.contentsecuritypolicy	frame-ancestors 'none'	This value will be set on the Content-Security-Policy HTTP header on all API responses. Changing this property could leave your server vulnerable to clickjacking attacks if you're embedding API access in a webpage.
server.api.xframeoptions	DENY	This value will be set on the X-Frame-Options HTTP header on all API responses. Changing this property could leave your server vulnerable to clickjacking attacks if you're embedding API access in a webpage.
<b>Database</b>		
database	derby	<p>The database type to use for the NextGen Connect Integration Engine backend database. Options:</p> <ul style="list-style-type: none"> <li>• derby</li> <li>• mysql</li> <li>• postgres</li> <li>• oracle</li> <li>• sqlserver</li> </ul> <p>By default NextGen Connect Integration Engine ships with an embedded Apache Derby database for quick testing / development purposes. For production instances, you should <a href="#">change the database type</a> to one of the other supported options.</p>

database.url	jdbc:derby:\${dir.appdata}/mirthdb;create=true	The JDBC URL to use when connecting to the database.
database.driver		The fully-qualified JDBC Driver class to use when connecting to the database.
database.max-connections	20	The maximum number of connections to use for the internal messaging engine connection pool.
database.username		The username to use when connecting to the database.
database.password		The password to use when connecting to the database.
database.pool	HikariCP	The connection pool type to use for the internal messaging engine. By default HikariCP is used, but "DBCP" is supported as well.
database.jdbc4	true	Indicates whether the database driver supports JDBC 4 operations.
database.test-query	SELECT 1	A small test query (e.g. "SELECT 1") that the connection pool can use for validity checking.
database.enable-read-write-split	true	If enabled, the database connection pool will be split into read-only and read/write pools. More information <a href="#">here</a> .
database.write-pool-cache	false	If enabled, the channel / channel group / code template / library internal cache queries will use the read/write connection pool instead of the read-only pool. If your read-only pool is pointing to a read replica and there is significant replica lag, you may want to consider enabling this. Only applicable when "database.enable-read-write-split" is enabled.
database_READONLY		The database type to use for the read-only pool, if enabled. If not specified, defaults to the "database" setting.
database_READONLY.url		The JDBC URL to use when connecting to the database for the read-only pool, if enabled. If not specified, defaults to the "database.url" setting.
database_READONLY.driver		The fully-qualified JDBC Driver class to use when connecting to the database for the read-only pool, if enabled. If not specified, defaults to the "database.driver" setting.
database_READONLY.username		The username to use when connecting to the database for the read-only pool, if enabled. If not specified, defaults to the "database.username" setting.
database_READONLY.password		The password to use when connecting to the database for the read-only pool, if enabled. If not specified, defaults to the "database.password" setting.
database_READONLY.max-connections		The maximum number of connections to use for the read-only pool, if enabled. If not specified, defaults to the "database.max-connections" setting.
database_READONLY.pool		The connection pool type to use for the read-only pool, if

		enabled. If not specified, defaults to the "database.pool" setting.
database_READONLY.jdbc4		Indicates whether the database driver supports JDBC 4 operations for the read-only pool, if enabled. If not specified, defaults to the "database.jdbc4" setting.
database_READONLY.test-query		A small test query (e.g. "SELECT 1") used for validity checking for the read-only pool, if enabled. If not specified, defaults to the "database.test-query" setting.
<b>Encryption</b>		
encryption.export	0	If enabled, exported channels and other files from the Administrator will be encrypted.
encryption.properties	0	If enabled, the "database.password" property in this file will automatically encrypted and re-saved when the NextGen Connect Integration Engine server is next started. To update the password, simply overwrite database.password and on next server startup it will be automatically encrypted and updated again.
encryption.algorithm	AES	The algorithm to use for symmetric encryption. This applies to messages, exports, and anything that is used along with the keystore to encrypt / decrypt.
encryption.keylength	128	The key length to use for symmetric encryption.
digest.algorithm	SHA256	The algorithm to use for generating cryptographically secure hashes / digests. This is used for creating salted hash values for user passwords. If you change this, all current passwords will no longer be valid, and will have to be reset by an administrator.
security.provider	org.bouncycastle.jce.provider.BouncyCastleProvider	The fully-qualified JCE/JCA provider class name to use. This provider is used for both symmetric encryption and password hashing.

## Split Database Connection Pools

When the "database.enable-read-write-split" setting is enabled, the database connection pool is split into two: A **read-only** pool, and a **read/write** pool. The read-only pool is used for many of the Administrator API calls that only fetch data. The read/write pool is used for all backend message processing, as well as any operation that creates/modifies /deletes data.

When the connection pools are split, you can separately configure settings for the read-only pool, with the "database-readonly" options. By default none of the "database-readonly" options are set, meaning that the read-only pool will default to the same configuration as the main read/write pool.

For example, if "database-readonly.max-connections" is not set, it defaults to the "database.max-connections" setting. So if you have your max connections set to 20 for the read/write pool, the read-only pool will also have up to 20 connections, meaning the total number database connections will be at most 40.

These settings also allow you to point the read-only connection pool to a completely different database instance. You may want to do this if you have a master DB and a horizontally scaling cluster of read-replica DBs. You can point the main read/write pool to the master DB, and point the read-only pool to the read-replica cluster instead. By doing this you can potentially reduce the traffic and strain on your master database.

When using read replicas however, the concept of "replica lag" should be taken into account. That refers to the amount of time a read replica DB is behind the master DB. If your replica lag is sufficiently large, all the selects done by the read-only pool may return out-of-date information, which can lead to unintended results in the Administrator.

The server keeps internal caches for channels, channel groups, code templates, and code template libraries. By default these caches use the read-only pool. But if replica lag is a concern, a separate option, "database.write-pool-cache", allows you to switch the caches over to using the read/write pool instead.

## Default Supported Cipher Suites

The following cipher suites are supported by default for the overall server when using TLS / SSL / HTTPS:

- TLS\_CHACHA20\_POLY1305\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_CHACHA20\_POLY1305\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256
- TLS\_DHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256
- TLS\_AES\_256\_GCM\_SHA384
- TLS\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDH\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDH\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_DHE\_DSS\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDH\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDH\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_DHE\_DSS\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_ECDH\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_ECDH\_RSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256

- TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDH\_ECDSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDH\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDH\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDH\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_ECDH\_ECDSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_ECDH\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA
- TLS\_EMPTY\_RENEGOTIATION\_INFO\_SCCSV



Depending on your Java installation and other factors (like having the [JCE Unlimited Strength](#) policy files installed), not all the cipher suites listed above may be available.

## The mirth-cli-config.properties File

---

This properties file allows you to launch the [NextGen Connect Command Line Interface](#) without having to include any command line arguments like the username / password. The following properties are supported:

- **address:** The HTTPS address of the NextGen Connect Integration Engine server to connect to.
- **user:** The username to connect with.
- **password:** The password to connect with.
- **version:** The version to enforce, to ensure that you don't incorrectly connect to an unexpected server version. Use "0.0.0" to allow all.
- **script:** The file to use for non-interactive script mode.

## Other Files and Folders

---

Other files and folders present in the [Installation Directory](#) include the following:

- **cli-lib:** Libraries used by the [Command Line Interface](#). These should not be changed.
- **client-lib:** Libraries used by the [Mirth Connect Administrator](#). These should not be changed.
- **custom-lib:** Custom libraries can be dropped in this folder, to be used by the [default library resource](#).
- **docs:** Contains license information for NextGen Connect Integration Engine and all third-party libraries. Also contains the Javadoc documentation used by [User API](#).
- **extensions:** Extension libraries used by the Server, Administrator, and CLI. These should not be changed.
- **logs:** This is the default location that server logs will be written into. This can be changed from [the log4j properties file](#).
- **manager-lib:** Libraries used by the [Server Manager](#). These should not be changed.
- **mccommand(.exe):** The entrypoint script for launching the [Command Line Interface](#).
- **mcmanager(.exe):** The entrypoint script for launching the [Server Manager](#).
- **mcserver(.exe):** The entrypoint script for launching the NextGen Connect Integration Engine server in a shell or other custom job.
- **mcserver.vmoptions:** Entries set here will be added as command-line parameters to the Java process when running **mcserver**.
- **mcservice(.exe):** The entrypoint script for launching the NextGen Connect Integration Engine server as a service / daemon.
- **mcservice.vmoptions:** Entries set here will be added as command-line parameters to the Java process when running **mcservice**.
- **\*-launcher.jar:** These libraries are used by the entrypoint scripts to launch their respective applications.
- **public\_api\_html:** Contains the files used to serve up the [REST API documentation](#).
- **public\_html:** Contains the files used to serve up the launch page, if the Web Dashboard WAR fails to load.
- **server-lib:** Libraries used by the NextGen Connect Integration Engine server. These should not be changed.
- **server-launcher-lib:** JARs in this directory will be loaded into the main server thread context classloader upon startup. This is required if you are using any custom log4j appender libraries.
- **webapps:** All WAR files in this directory will automatically be published by the NextGen Connect Integration Engine web server upon startup.

## FAQ

- What is Mirth Connect?
- Is Mirth Connect the same as "Mirth?"
- Who develops Mirth Connect?
- What is the Mirth Connect license, and how much does it cost?
- How can Mirth Connect be free and open source?
- Is there a difference between the free, open-source Mirth Connect download and the supported version of Mirth Connect?
- How does Mirth Connect compare to commercial integration engines?
- How many production installations of Mirth Connect exist?
- What can I expect next from Mirth Connect?
- Is there data that shows how fast Mirth Connect operates?
- Is Mirth Connect hard to install and configure?
- Do I need Mirth Appliance to run Mirth Connect?
- As a member of the Mirth community, how can I get more help?
- How do I become a Mirth Connect expert?
- What are the system requirements for Mirth Connect?
- Which databases does Mirth Connect support for its data store?
- Does Mirth Connect use the Mule ESB?
- Do I need an application server to run Mirth Connect?
- Can Mirth Connect send data to \_\_\_\_\_ or transform data from \_\_\_\_\_ to \_\_\_\_\_?
- What message standards does Mirth Connect support?
- What transfer protocols does Mirth Connect support?
- How do I transform a data segment?

### What is Mirth Connect?

Mirth Connect is an open-source, standards-based healthcare integration engine that speeds message routing, filtering, and transformation between health-info systems over various messaging protocols (e.g., HL7, X12, EDI, DICOM, XML).

[Top](#)

### Is Mirth Connect the same as "Mirth?"

Yes. Mirth 1.0 was released in 2006 by WebReach, Inc. Based on the success of the Mirth application, WebReach, Inc. was renamed "Mirth Corporation" in 2009. To avoid confusion between the new company name and its products, the Mirth application was renamed *Mirth Connect*.

As of early 2016, Mirth Corporation, as a business entity, no longer exists, having been absorbed into the Quality Systems, Inc. family of products in Sept. 2013. QSI has since become NextGen Healthcare, and Mirth Connect lives on with other former Mirth Corporation products as part of the "Mirth Solutions" suite.

[Top](#)

### Who develops Mirth Connect?

NextGen Healthcare develops and fully sponsors Mirth Connect with the unofficial assistance of its users, who report bugs and contribute source-code patches, feature requests, and online support.

[Top](#)

### What is the Mirth Connect license, and how much does it cost?

Mirth Connect is released under the [Open Source Initiative \(OSI\)](#) approved MPL 1.1 (see [Mozilla Public License 1.1](#).), and it costs you absolutely nothing! You can download Mirth Connect free via the [Mirth download page](#).

[Top](#)

### How can Mirth Connect be free and open-source?

NextGen Healthcare, the primary developer of Mirth Connect, backs it with commercial [support, services, training, and appliances](#). Also, the contributions of thousands of users help keep Mirth Connect a leading HIT (health-information technology) integration engine.

[Top](#)

### Is there a difference between the free, open-source Mirth Connect download and the supported version of Mirth Connect?

No; however, [support subscriptions](#) make available to you various commercial plug-ins and connectors as well as day-to-day improvements that are unavailable to non-support users until the next software release.

[Top](#)

### How does Mirth Connect compare to commercial integration engines?

Based on user feedback, Mirth Connect's features compare favorably to commercial integration engines and are usually superior in head-to-head comparisons. Despite being free and open-source, Mirth Connect has NextGen Healthcare's full backing with support, training, and consulting services similar to those of commercially vended software. We are proud of our agile development process and frequent release cycle, which are possible because Mirth Connect is open-source and community-driven.

[Top](#)

### How many production installations of Mirth Connect are there?

NextGen Healthcare formally supports hundreds of specific production installations of Mirth Connect, but because it is free and open-source, the overall number of production installations is unknown. We do know, however, that Mirth Connect has been downloaded over 500,000 times, and the active online Mirth Connect community has over 40,000 members, so Mirth Connect installations in production worldwide are undoubtedly in the thousands.

[Top](#)

### What can I expect next from Mirth Connect?

Mirth Connect is continually evolving, and each new version includes various enhancements and new features. To see what's new in any given version, check out the release notes on our public wiki: [Mirth Connect 3.5.0 - What's New](#)

[Top](#)

### How fast does Mirth Connect operate?

Performance varies widely depending on your hardware setup and channel configuration. If there is a performance bottleneck, it is almost always not because of Mirth Connect, but rather because of the underlying database Mirth Connect is using, or the storage solution. For large-scale instances we recommend deploying Mirth Connect with a horizontally-scalable database solution, backed by SSDs. [Benchmarks](#) are available for the hardware appliances that Mirth Corporation offers.

[Top](#)

### Is Mirth Connect hard to install and configure?

No. In fact, we've heard from users that they installed Mirth Connect and minutes later were processing messages. There are several installation methods, but the easiest is the cross-platform GUI (graphical user interface) installer that guides you through the process. If you prefer a more hands-on method, however, we offer a **zip/tar.gz** distribution.

[Top](#)

### Do I need Mirth Appliance to run Mirth Connect?

No, but running Mirth Connect on the Appliance platform provides such advantages as easy updates, added security and reliability through clustering, and a platform that can host the entire Mirth Solutions suite. To view an extensive list of features, see our [data sheet](#) on Appliance solutions.

[Top](#)

### As a member of the Mirth Solutions community, how can I get more help?

NextGen Healthcare provides numerous [commercial support options](#) if you have a specific problem or would like support for running Mirth Connect for production. Sign up for support to access [online training videos](#) and [monthly Q&As](#) with Mirth Connect developers.

[Top](#)

### How do I become a Mirth Connect expert?

NextGen Healthcare provides extensive [Mirth Connect training](#) and a certification program. More information here: [Training](#)

[Top](#)

### What are the system requirements for Mirth Connect?

You can run Mirth Connect on any system that supports Java and requires the Sun/Oracle JRE (Java Runtime Environment) 8 or newer.

The Mirth Connect Server requires a database for its configuration and message store. For quick deployment, development, and testing, Mirth Connect already includes an embedded database (Apache Derby). For production use, the latest version of Mirth Connect supports the following databases:

- PostgreSQL 8.3+
- MySQL 5.6+
- Oracle 10gR2+
- SQL Server 2005+

Note that the above database requirements only apply to what is used for the configuration and message store of the Mirth Connect Server, and have no impact on which databases Mirth Connect can interface with.

[Top](#)

### Which databases does Mirth Connect support for its data store?

Apache Derby (default), PostgreSQL, MySQL, Oracle, and Microsoft SQL Server, but the Database Reader/Writer connectors can support any type of database if you add the right client libraries to the **custom** folder in the Mirth Connect installation directory.

[Top](#)

### Does Mirth Connect use the Mule ESB?

Not anymore. Prior to the release of Mirth Connect 3.0, Mule was eliminated from the application.

[Top](#)

### Do I need an application server to run Mirth Connect?

No. Mirth Connect runs as a stand-alone executable or service in its own JVM (Java virtual machine).

[Top](#)

Can Mirth Connect send data to \_\_\_\_\_ or transform data from \_\_\_\_\_ to \_\_\_\_\_ ?

Whichever variables fill these blanks, the answer is generally a resounding Yes! Even if a message standard (protocol) is foreign to Mirth Connect, advanced Java and JavaScript capabilities are so flexible that almost any data type can be transformed and transferred.

[Top](#)

What message standards does Mirth Connect support?

- [HL7 v2.x](#)
- [HL7 v3.x](#)
- [Delimited Text](#) (CSV, tab-delimited, fixed-width, etc.)
- [DICOM](#)
- [EDI / X12](#)
- [XML](#)
- [JSON](#)
- [NCPDP](#)
- [Raw](#) (supports *any* data format!)
- [ASTM E1394](#) (Commercial extension)

[Top](#)

What transfer protocols does Mirth Connect support?

- [MLLP](#) (v1 and v2)
- [TCP/IP](#)
- [HTTP](#)
- [Flat Files](#)
- [Databases](#)
- [SFTP](#)
- [FTP](#)
- [SMB](#)
- [WebDAV](#)
- [IMAP / POP3](#) (Email)
- [SMTP](#) (Email)
- [DICOM](#)
- [JMS](#)
- [SOAP Web Services](#)
- [PDF/RTF Documents](#)
- [Custom Java and JavaScript](#)
- [ASTM E1381](#)
- [Serial / RS-232](#)

[Top](#)

How do I transform a data segment?

Example: You have the data segment **20080624175854-0700**, and you want to eliminate the hyphen and the four digits after it. The easiest way to do this is to create a new **STEP** in the transformer with some JavaScript such as:

```
var originalValue =msg['PID']['PID.18']['PID.18.1'].toString(); //, which represents: 20080624175854-0700
var splittedValueArray = originalValue.split("-"); msg['PID']['PID.18']['PID.18.1'] = splittedValueArray[0]; //
splittedValueArray[0], which represents: 20080624175854
// splittedValueArray[1], which represents: 0700
```

Use of the split function returns an array of the values split by the delimiter.

[Top](#)

## Channel Development Best Practices and Tips

### Channel Performance

#### Adjust the Message Storage slider so that the channel only retains the message data that you will actually need.

One limitation on message throughput is the amount of disk writes that NextGen Connect Integration Engine's database must perform while processing each message. Reducing the amount of message data retained can increase throughput and decreases memory usage as well as disk usage. **Production** level guarantees that if an unexpected server failure occurs, any incomplete (received, but not yet fully processed) messages will be automatically recovered and processed on restart. If increased performance is needed, you can reduce this to the **Raw** level, which only retains the original raw message prior to being processed by any transformer steps. Incomplete messages can still be recovered on failure, but must be done so by manually reprocessing them. Keep in mind that using destination queues requires at least Production level.



#### If large messages are expected, use an Attachment Handler to improve throughput and reduce memory and disk usage.

When a channel processes a message, copies of the message content are made in memory at various steps in which the message is transformed. Depending on the message storage mode, multiple copies of the transformed content are also written to the disk/database. Due to this, large messages can quickly consume memory and can cause out-of-memory errors if one is not careful. An attachment handler can mitigate these problems by extracting the bulk of the message content that does not need to undergo transformation while passing through the channel.

For more information about attachment handlers, please see the official [User Guide](#).

#### Enable source queuing if an auto-generated acknowledgement is sufficient for the upstream system.

If you do not need to send back a response to the originating system, or the response does not need to come from one of your destinations, consider turning the "Source Queue" ON in your source connector. This lets the channel respond to the originating system immediately on message receipt, rather than waiting for the message to finish processing. This can increase your channel's overall throughput.

## Enable destination queuing if you don't need to respond to the originating system from your destination.

All destination connectors support queuing. Queuing is important for re-attempting to send messages if they do not succeed at first, but can also improve channel performance. Without queuing, the message send attempt will occur on the main message processing thread for the channel, which will block any new messages from being processed (unless the Max Processing Threads is increased, see below). This can significantly improve channel performance if the system you are sending to is slow to respond.

## Increase the Max Processing Threads value if message order is not important and you may receive a large number of messages in a short time.

This setting controls how many messages the channel can process simultaneously (on separate CPU threads). The default value is 1, which means that only one message can be processed by the channel at a time and also guarantees that messages are sent out from the channel in the order that they are received. Increasing this value eliminates the order guarantee, but can greatly increase throughput when a large number of messages is received in a short amount of time. Setting this value too high however, can harm throughput. Keep in mind that this setting controls how many CPU threads are used by the channel. So, it is important to consider how many threads the CPU can execute concurrently as well as how much of the CPU's resources are needed by other channels/processes.

- Source Settings

Source Queue:	OFF (Respond after processing) <input type="button" value="▼"/>
Queue Buffer Size:	1000
Response:	<input type="button" value="None"/> <input type="button" value="Success"/> <input type="button" value="Failure"/>
Process Batch:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Batch Response:	<input type="radio"/> First <input checked="" type="radio"/> Last
Max Processing Threads:	<input type="text" value="8"/>

## Increase the Queue Threads value if the downstream system can accept multiple concurrent connections.

This setting controls how many messages can be simultaneously retrieved from the outbound queue to attempt to send and is similar to the Max Processing Threads setting described above. As with the Max Processing Threads setting, the default value is 1, and increasing the value will no longer guarantee the order in which queued messages are processed. However, the "Thread Assignment Variable" setting can be used to assign messages to a particular queue thread, thus guaranteeing the send order of all messages assigned to a particular thread (unless "Rotate Queue" is set to "Yes"). Users wanting to increase outbound threads can start with 2-10 and increase or decrease from there until reaching maximum throughput.

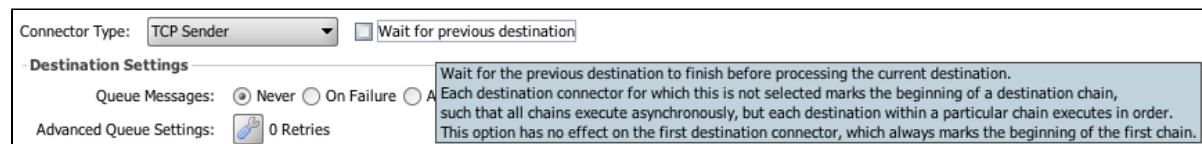
**Advanced Queue Settings**

Retry Count Before Queue/Error:	<input type="text" value="0"/>
Retry Interval (ms):	<input type="text" value="10000"/>
Rotate Queue:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Regenerate Template:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Include Filter/Transformer:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Queue Threads:	<input type="text" value="3"/>
Thread Assignment Variable:	<input type="text"/>
Queue Buffer Size:	<input type="text" value="1000"/>

**OK**    **Cancel**

**Uncheck the "Wait for previous destination" checkbox, unless you need destinations to process messages one after another.**

Unchecking this box will allow adjacent destinations to execute in parallel, potentially improving performance. When using several destinations, you can configure some to execute in order and others to execute in parallel.



**Use the "Destination Set Filter" feature in your source transformer when you need to route messages to just one (or a subset) of destinations in your channel.**

This feature has been available in the JavaScript user API since version 3.1.0:

```
if (sourceMap.get("contextPath") == "/path1") {
 destinationSet.removeAllExcept([1]);
}
```

A less efficient pattern used prior to version 3.1.0 was to filter the message on all destinations except the desired destination. This approach consumes more disk space as the message is copied to each destination before the filtering takes place. The Destination Set Filter will ensure that the message is only processed and stored by the desired destination.

Since version 3.5.0, it can also be leveraged without writing any code:

**Edit Channel - Test - Source Transformer**

#	Name	Type
0	Filter destination(s) if "sourceMap.get("contextPath")" equals "/path1"	Destination Set Filter
1	Filter destination(s) if "sourceMap.get("contextPath")" equals "/path2"	Destination Set Filter

Step \ Generated Script \

Behavior: Remove all except the following destinations: [Select All](#) | [Deselect All](#)

Destinations:

	Name	Id
<input checked="" type="checkbox"/>	Destination 1	1
<input type="checkbox"/>	Destination 2	2

Field: `sourceMap.get("contextPath")`

Condition:  Exists  Not Exist  Equals  Not Equal  Contains  Not Contain

Values:

Value	New
"/path1"	<a href="#">Delete</a>

## Channel Configuration

---

### Use tags to categorize your channels.

If you have a large number of channels to manage, tags can help you keep them organized. See the official User Guide at [mirth.com](#) or the [NextGen Connect Integration Engine 3.5.0 What's New](#) page for more information.

### Place any reusable JavaScript code into Code Template functions and organize them into Code Template Libraries.

This prevents code duplication and keeps your JavaScript code from being scattered across your channels. Put reusable code into Code Templates and keep only channel-specific code in your channels. Code template libraries can also be enabled for only specific channels, to prevent namespace clashes and other confusion.

Read more about Code Templates in the official [User Guide](#).

### Use Custom Meta Data Columns to increase the performance of searching the message history on a particular message field.

If there is a particular element of data from your messages that you may need to search by later, configure the channel to map the data into a custom meta data field. NextGen Connect Integration Engine will then index this data to improve search performance when searching for particular values. To do this, first create a mapper transformer

step that maps the value into a Channel Map variable ("myField" for example). Then in the "Custom Metadata" section of the channel summary screen, add a new column and assign it to the variable mapping that you defined ("myField").

To search for messages based on values in your custom meta data column, click "Advanced..." to open the Advanced Search Filter. Then add search criteria for the custom field in the table

Advanced Search Filter		
<a href="#">Select All</a>   <a href="#">Deselect All</a>		
Id	Current Connector Name	Included
0 Source		<input checked="" type="checkbox"/>
1 Destination 1		<input checked="" type="checkbox"/>
2 Destination 2		<input checked="" type="checkbox"/>
-- Deleted Connectors		<input checked="" type="checkbox"/>

Message Id:	<input type="text"/>	-	<input type="text"/>
Original Id:	<input type="text"/>	-	<input type="text"/>
Import Id:	<input type="text"/>	-	<input type="text"/>
Server Id:	<input type="text"/>		
Send Attempts:	<input type="text"/> 0	-	<input type="text"/>
Has Attachment:	<input type="checkbox"/>		
Has Error:	<input type="checkbox"/>		
Content Type	Contains		
<input type="button" value="New"/> <input type="button" value="Delete"/>			

Metadata	Operator	Value	Ignore Case
MYFIELD	=	<input type="text"/> example value	<input type="checkbox"/>

<input type="button" value="New"/>	<input type="button" value="Delete"/>
------------------------------------	---------------------------------------

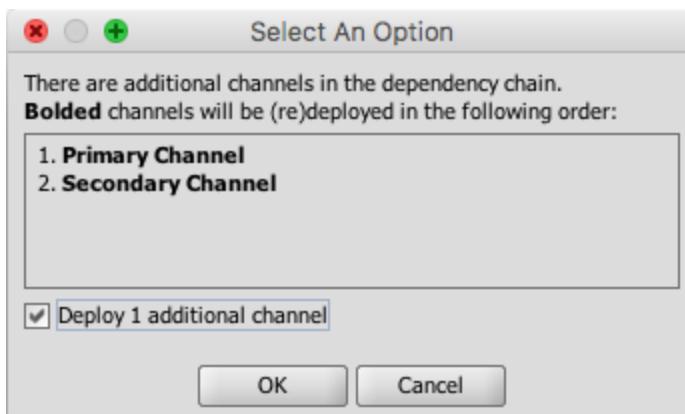
<input type="button" value="OK"/>	<input type="button" value="Cancel"/>
-----------------------------------	---------------------------------------

## Use Deploy/Start Dependencies if one channel depends on another to operate correctly.

This can be configured in the Channel Summary view > Set Dependencies > Deploy/Start Dependencies.

Channel Dependencies	
<a href="#">Code Template Libraries</a> \ <a href="#">Library Resources</a> \ <a href="#">Deploy/Start Dependencies</a> \	
This channel depends upon:	
<a href="#">Expand All</a>   <a href="#">Collapse All</a>	
<ul style="list-style-type: none"> <li>..... Primary Channel</li> </ul>	
<input type="button" value="Add"/> <input type="button" value="Remove"/>	

By defining channel dependencies, you can protect yourself against deploying a channel without deploying another channel that it depends on to operate correctly. NextGen Connect Integration Engine will alert you when you attempt to deploy a channel that depends on another:



## Use multiple resource folders for your Java libraries if you need to limit library usage to specific channels or if you have many Java libraries.

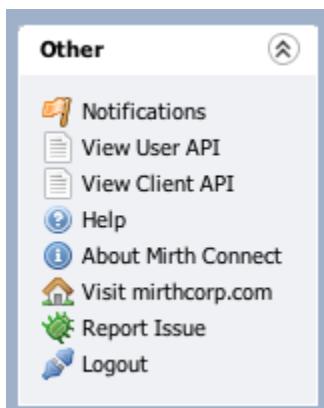
(Settings > Resources) Avoid putting all of your custom Java libraries in the Default Resource folder ("custom-lib"). You may have added a particular Java library to be used by a single channel. If so, place that library into a separate resource and enable that resource for that one channel under the "Dependencies" section of the Channel Properties. A danger of having too many libraries in the default resource is that unexpected runtime conflicts could occur between different libraries.

## Other Tips

### Don't reference NextGen Connect Integration Engine's internal Java classes in your JavaScript code.

These classes are not intended to be used directly by your channels and could result in unexpected behavior. They are also subject to change, so upgrading to a new version of NextGen Connect Integration Engine could immediately break your channels.

Instead, make use of the user API. There is a link to browse the user API documentation in the left-hand navigation menu:



Any methods/functions in the user API are intended to be used by channels and are guaranteed to be supported and available when you upgrade NextGen Connect Integration Engine. Future versions of NextGen Connect Integration Engine may deprecate certain functions, in which case a warning message will appear in your server log.

## Troubleshooting

This page is meant to capture some of the most common errors that may occur when using NextGen Connect Integration Engine. It is by no means an exhaustive list. For additional help, you may be interested in our [Commercial Support / Extensions](#) or [Training](#) offerings.

- [Logs](#)
- [Configuration](#)
- [NextGen Connect Integration Engine fails to start up](#)
- [Unable to launch Mirth Connect Administrator](#)
- [Clearing your Java Cache](#)
- [Opening the Java Client Console](#)
- [Out of Memory Errors](#)

### Logs

Whenever an issue occurs, it may be helpful to look at the server logs. These are stored in the "logs" folder inside your [Installation Directory](#). The "mirth.log" file will be the most recent log.

### Configuration

Often times issues occur because of mistakes in initial configuration. The main configuration for NextGen Connect Integration Engine is done in [the mirth.properties file](#) in the "conf" folder inside your [Installation Directory](#).

### NextGen Connect Integration Engine fails to start up

- A common cause of this is port conflicts. Check the [logs](#), you may see something like this:

 ERROR 2017-06-23 09:17:22,757 [Main Server Thread] com.mirth.connect.server.Mirth: http.port port is already in use: 8080  
ERROR 2017-06-23 09:17:22,758 [Main Server Thread] com.mirth.connect.server.Mirth: https.port port is already in use: 8443

- Change the "http.port" and "https.port" in your [configuration](#) to ones that aren't already used.
- Another common cause is database connectivity / problems. If you see this:

 ERROR 2017-06-23 09:26:07,888 [Main Server Thread] com.mirth.connect.server.Mirth: Error establishing connection to database, aborting startup. The connection attempt failed.

It probably means you have an error in your database [configuration](#) settings, or the machine running NextGen Connect Integration Engine does not have connectivity to the correct IP.

- If you see an error like this:

 ERROR 2017-06-23 09:32:35,405 [Main Server Thread] com.mirth.connect.server.Mirth: Failed to migrate database schema  
com.mirth.connect.model.util\MigrationException: Failed to execute script:

If may mean that you've changed the "database.url" setting correctly, but forgot to also change the "database" setting. For example if you're switching from Derby to PostgreSQL, make sure to change the "database" setting to "postgres".

- Another possibility is that you have all the database settings correct, but the schema you're trying to connect to doesn't yet exist:

 ERROR 2017-06-23 09:35:10,668 [Main Server Thread] com.mirth.connect.server.Mirth: Error establishing connection to database, aborting startup. FATAL: database "mirthdb" does not exist

- In your database management tool / command line, make sure to create the schema ("mirthdb" by default). Then NextGen Connect Integration Engine will automatically create all the necessary tables once it starts up.

## Unable to launch Mirth Connect Administrator

The login dialog will generally tell you if there is any issue:



- If you don't have connectivity from your local machine to the NextGen Connect Integration Engine server over the HTTPS port (default 8443), you'll see this:

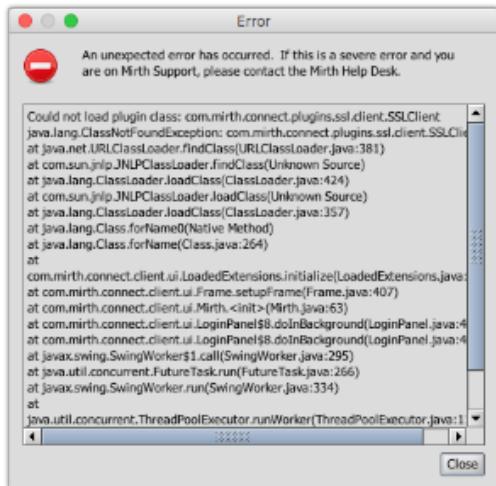
 There was an error connecting to the server at specified address. Please verify that the server is up and running.

Make sure the URL you're using is correct, and verify whether you have network connectivity to the server.

- Also verify that the NextGen Connect Integration Engine server is actually running. If you installed as a service, you can view the current status from the [Server Manager](#).

## Clearing your Java Cache

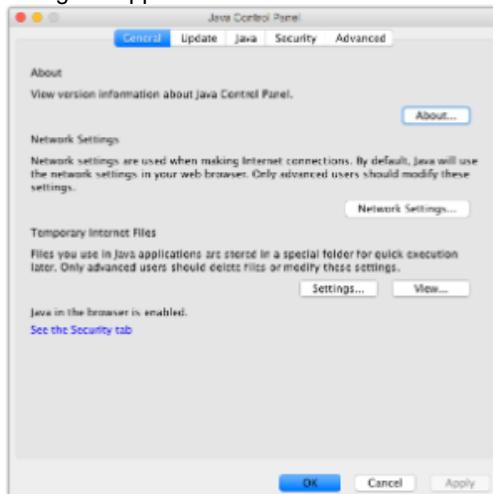
If you've recently upgraded NextGen Connect Integration Engine or [installed a new extension](#), you may find that your current Administrator shortcut no longer works:



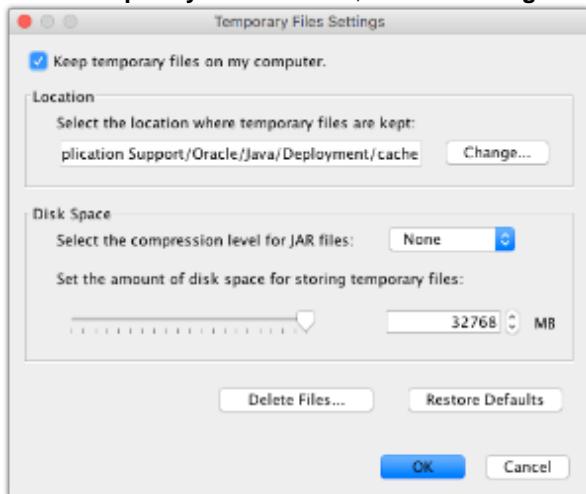
This can be resolved by clearing your Java cache. You can do that directly from the command line, or by using the Java Control Panel.

### Using the Java Control Panel

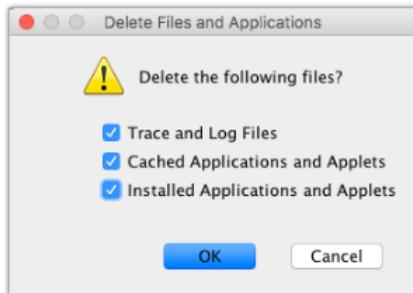
- In your System Preferences (on OSX) or the Control Panel (on Windows), open the Java section. A new dialog will appear:



- Under **Temporary Internet Files**, click on **Settings...**



- Click on **Delete Files...**



- Check all three boxes and then click OK.

### Using the Command Line

- Open up a terminal / command prompt, and type this:

- javaws -uninstall

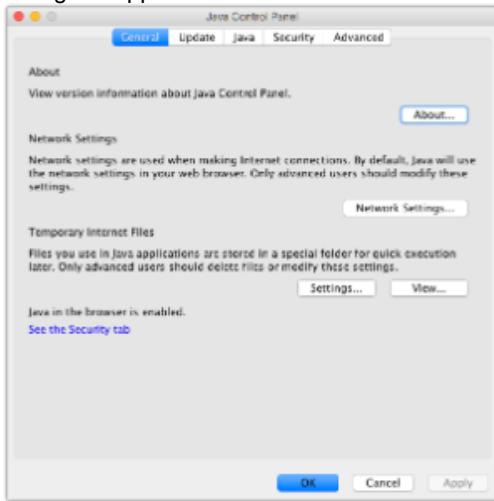
- Wait for the operation to finish. Note that if you have multiple versions of Java installed on your machine, you may have to use the **javaws** executable specific to the version you launch the Mirth Connect Administrator with.

## Opening the Java Client Console

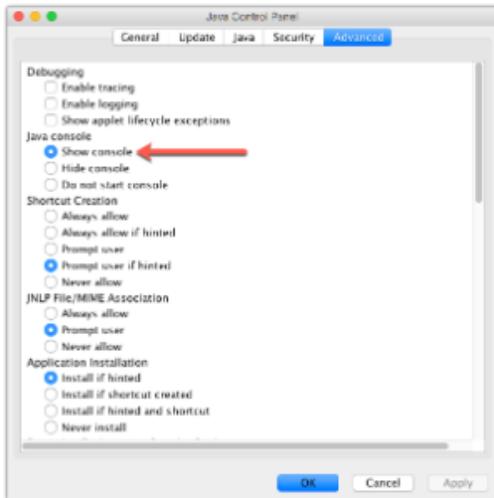
If an error occurs on the NextGen Connect Integration Engine server, you will typically see one or more entries appear in the [logs](#). If an error occurs in the [Mirth Connect Administrator](#), it's possible that it's isolated to the client-side only, and so nothing will appear in the server-side logs. In cases like this, it's useful to enable the **Java Client Console** to see any exceptions that occur on your client-side Java virtual machine (JVM).

To enable the console, first open the Java Control Panel:

- In your System Preferences (on OSX) or the Control Panel (on Windows), open the Java section. A new dialog will appear:

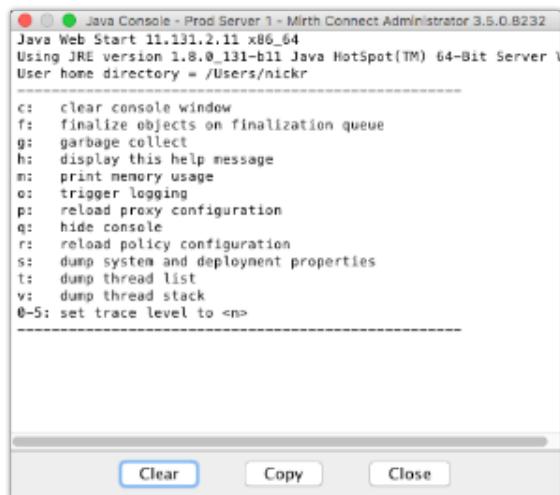


- Click on the **Advanced** tab.
- Select **Show console**.



- Click **Apply**, then click **OK**.

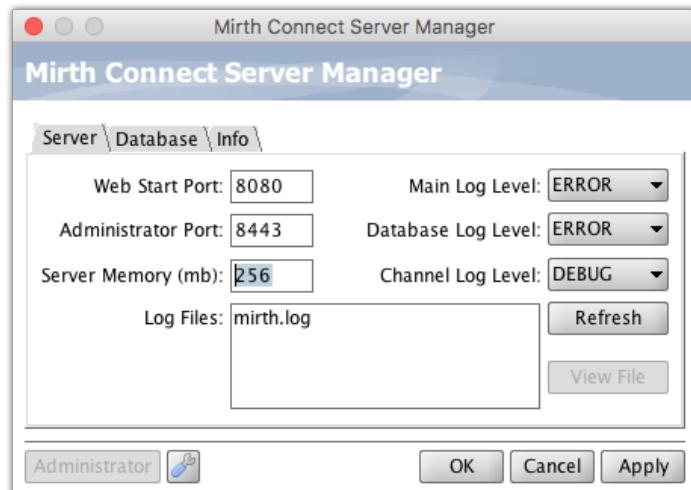
Now when you launch the Administrator, you'll see a second window also pop-up:



Any exceptions that are logged to stdout / stderr on the client side will appear in that console.

## Out of Memory Errors

By default, the max Java heap size for the NextGen Connect Integration Engine server is set to 256 MB. For large production instances you will typically want to increase this value. You can do this from the [Server Manager](#):



Or, you can edit the appropriate **\*.vmoptions** file in your [Installation Directory](#).



When changing the max Java heap size, you must restart your NextGen Connect Integration Engine server / service for it to take effect.

Out of memory errors can also happen on the client-side, usually when attempting to view very large messages through the message browser:



If this is the case, first you may want to consider using [Attachment Handlers](#) on your channel. To change the max heap size for the client-side Administrator, open the 8080 launch page in a browser:

Click on the cog icon next to the Launch button, and change the max heap size. Then click the launch button. The JNLP file you download will have the "max-heap-size" attribute set to the value you chose.

The default value for the client-side max heap size is 512 MB. To change this default, edit the **administrator.maxheapsize** setting in [the mirth.properties file](#). To change the options that show up in the drop-down on the 8080 launch page, change the **administrator.maxheapsizeoptions** setting.

## Commercial Support / Extensions

A NextGen Connect Integration Engine commercial license includes:

- **Enterprise-class Support:** Most importantly, you'll be safely under our support umbrella. You can contact us directly if for any reason you run into a production incident.
- **Advanced Commercial Extensions:** Our commercial extensions are built specifically for large healthcare organizations that require secure and scalable solutions. Streamline SSL certificate management, add role-based user restrictions, define metric-based advanced alerts, and much more! Look [below](#) for a full list.
- **Option For Professional Services:** We have a professional services team that can build interfaces from scratch to your (or your vendor's) specifications! We offer this option (as a separate engagement) to commercial support customers in case your organization doesn't have the necessary developer resources.



Don't hesitate to [Contact Us](#) if you have any questions about commercial support, training, or any of the extensions!

The following extensions are available as part of commercial licenses:

- [Advanced Alerting](#)
- [Advanced Clustering](#)
- [ASTM E1381 Transmission Mode](#)
- [ASTM E1394 Data Type](#)
- [Channel History](#)
- [Email Reader](#)
- [FHIR Connector](#)
- [Interoperability Connector Suite](#)
- [LDAP Authorization](#)
- [Message Generator](#)
- [Multi-Factor Authentication](#)
- [Serial Connector](#)
- [SSL Manager](#)
- [User Authorization](#)
- [NextGen Results CDR Connector](#)

## Advanced Alerting

Advanced Alerting provides metric, exception, and state-based monitoring of channels and connectors. Additional features include automatic escalation and de-escalation, scheduling, and notification throttling. Using advanced alerts, dynamically send different alert messages to different user groups based on the current escalation level, time, and day. The new alert dashboard provides a view of all alert statistics and logs. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the 'Edit Alert - No messages sent for 10 minutes' configuration page in the Mirth Connect Administrator. The alert is triggered by 'No. of messages (sliding window)' with a limit of 0 over a 10m window length. It is set to 'Sent' status and aggregate by 'Connector'. The 'Enabled' checkbox is checked. The 'Channels' section lists various destinations, including [New Channels], ADT Receiver 6661, ADT Receiver 7661, ADT Receiver 8661, Alert Receiver, LLP Sender 1 (with Source and Destination 1), LLP Sender 2 (with Source and Destination 1), and several MDM and ORU Receivers. The 'Escalation Levels' table shows three levels: Normal (triggered after 1h, returns to normal after 30m), Warning, and Error. The 'Action Groups' section contains four groups: Warning Group (Tier 1), Error Group (Tier 2), After Work Hours, and Level Change. The 'Actions' section shows an alert receiver as the recipient for email messages. The 'Subject' field is 'Subject (only used for email messages)'. The 'Template' field contains the message: 'Escalation level for alert \${alertName} has changed from \${oldLevel} to \${newLevel}.'. The 'Alert Variables' section lists alertId, alertName, groupName, serverId, globalMapVariable, date, oldLevel, and newLevel. The bottom status bar indicates 'Connected to: Production Server 1 | https://localhost:8443 | 8:54 AM PST (UTC -8)'.

## Advanced Clustering

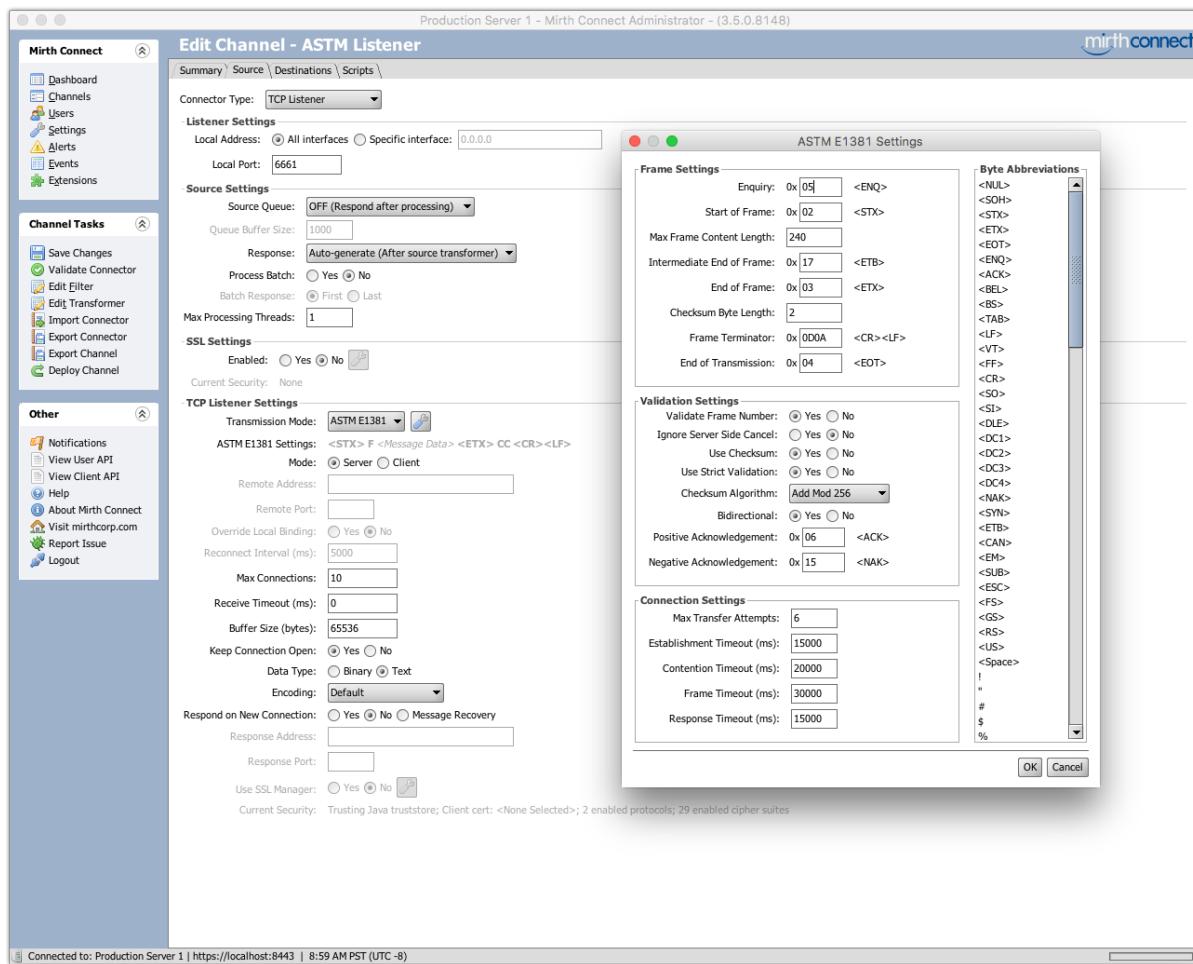
The Advanced Clustering plug-in improves the availability of your NextGen Connect Integration Engine cluster with automatic message takeover. If one server in the cluster fails, another server will automatically resume processing of any queued or unfinished messages from the failed server. Managing your cluster becomes easier as you can monitor the status of each server, deploy/start/stop/pause channels across all servers, and view message statistics for the whole cluster or a particular server from the Mirth Connect dashboard. This plug-in can be used with standalone NextGen Connect Integration Engine instances using your own load-balancing solution, but also seamlessly integrates with the load-balancing and failover services provided on Mirth Appliances. Feel free to [Contact Us](#) if you have any questions.

The screenshot displays the Mirth Connect Administrator interface with the following sections:

- Dashboard:** Shows a list of servers and their channel status. For example, Hospital A has an ADT Receiver (HL7) and MDM Receiver (HL7) both marked as "Inpatient". Hospital B has an ADT Receiver (HL7), MDM Receiver (HL7), and ORU Receiver (HL7) all marked as "Inpatient". LLP Sender 1 and LLP Sender 2 are listed under "Mixed".
- Task Errors:** A modal window for "Production Server 2" shows a stack trace for a "Test Exception" at line 48. The exception details mention a deployment script and a sleep operation.
- Cluster Task History:** A table showing tasks submitted to the cluster. The last entry is "Undeploy channel 'RS-232 Sender'" at 2016-12-15 09:51:47.

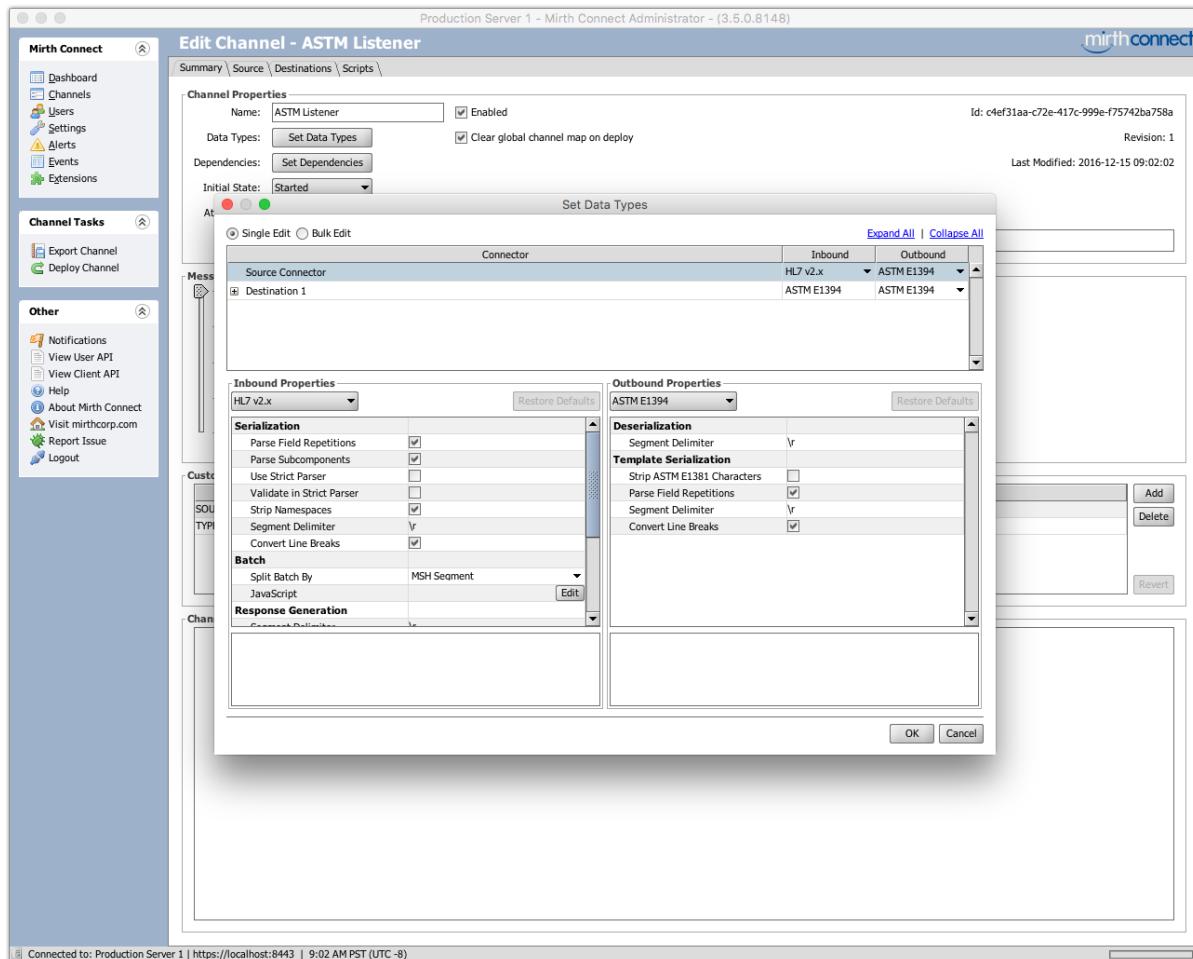
## ASTM E1381 Transmission Mode

The ASTM E1381 [transmission mode](#) for NextGen Connect Integration Engine allows you to send and receive data using the ASTM E1381 lower-layer protocol standard. It can be used in conjunction with the [TCP Listener / TCP Sender](#) or the [Serial Connector](#). By default the official ASTM E1381 standard is strictly followed, but there are several settings available to support variations from the standard. The extension supports specifying the bytes used for frame delimiters, the checksum algorithm used, various timeouts, and much more. Feel free to [Contact Us](#) if you have any questions.



## ASTM E1394 Data Type

The ASTM E1394 [data type](#) for NextGen Connect Integration Engine allows you to easily accept, parse, and transform messages following the ASTM E1394 data standard. As with other data types, an incoming message will be serialized into a simple XML format, and all the usual transformer steps may then be used to transform the message, or to convert it to or from a different data type. Several data type properties are also provided for specifying how messages should be converted to and from XML. A standard vocabulary for the ASTM E1394 data type is also included to help identify fields in the message template trees of a transformer. Feel free to [Contact Us](#) if you have any questions.



## Channel History

Get configuration management for all of your critical channels using the Channel History plug-in. You can view and compare past revisions of channel configurations—and identify the user making changes. Plus, revert to a past revision from the embedded viewer. Feel free to [Contact Us](#) if you have any questions.

Status	Data Type	Name	Id	Description	Rev Δ	Last Deployed	Last Modified
Enabled		[Default Group]	d32ac48c-b5e5-41bc-9316-2392a7074202	Channels not part of a group will appear...	--	--	--
Enabled	HL7 v2.x	Alert Receiver Channel	c1ef31aa-c72e-417c-999e-f75742ba758a	--	3	2016-12-15 09:04	2016-12-15 09:04
Enabled	HL7 v2.x	ASTM Listener	067f170b-9d73-47bd-b345-5741bcd773e	--	--	2016-12-15 09:02	2016-12-15 09:02
Enabled	HL7 v2.x	Hospital A	b582863e-5667-4aa3-92e2-426077930940	--	--	2016-12-15 08:40	2016-12-15 08:40
Enabled	HL7 v2.x	ADT Receiver 6661	ADT [Inpatient]	--	--	2016-12-15 08:39	2016-12-15 08:39
Enabled	HL7 v2.x	MDM Receiver 6662	HL7 [Inpatient] MDM	e77d50e2-85ba-4134-a6a4-beb303531f7	--	--	2016-12-15 08:41
Enabled	HL7 v2.x	ORU Receiver 6663	HL7 [Inpatient] ORU	9c5382b2-2a72-4bb9-9d71-356eb79ffce0	--	--	2016-12-15 08:41
Frothland		MHealthNet R	71ad758f-read-4f9a-48a0-4c3a9d19206	--	--	2016-12-15 08:42	2016-12-15 08:42

**Channel changes from revision 11 to 12**

```
<channel version="3.5.0">
 <id>d32ac48c-b5e5-41bc-9316-2392a7074202</id>
 <nextMetaId>2</nextMetaId>
 <name>Alert Receiver</name>
 <description></description>
 <revision>12</revision>
 <sourceConnectorVersion>3.5.0</sourceConnectorVersion>
 <metaId>0</metaId>
 <name>sourceConnector</name>
 <properties><property class="com.mirth.connect.connectors.vm.VmReceiverProperties" version="3.5.0">
 <pluginProperties/>
 <sourceConnectorProperties version="3.5.0">
 <responseVariable>None</responseVariable>
 <respondAfterProcessing>true</respondAfterProcessing>
 <processBatch>false</processBatch>
 <firstResponse>false</firstResponse>
 <processingThreads>1</processingThreads>
 <resourceIds class="linked-hash-map">
 <entry>
 <string>Default Resource</string>
 <string>Default Resource</string>
 </entry>
 </resourceIds>
 <queueBufferSize>1000</queueBufferSize>
 </sourceConnectorProperties>
 </properties>
</transformer version="3.5.0">
 <steps/>
 <inboundDataType>HL7V2</inboundDataType>
 <outboundDataType>HL7V2</outboundDataType>
 <inboundProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2DataProperties">
 <serializationProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2SerializationProperties">
 <handleRepetitions>true</handleRepetitions>
 <handleSubcomponents>true</handleSubcomponents>
 <useStrictValidation>false</useStrictValidation>
 <stripNamespaces>true</stripNamespaces>
 <segmentDelimiter>\r\n</segmentDelimiter>
 <convertLineBreaks>true</convertLineBreaks>
 </serializationProperties>
 <deserializationProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2DeserializationProperties">
 <useStrictValidation>false</useStrictValidation>
 <stripNamespaces>true</stripNamespaces>
 <segmentDelimiter>\r\n</segmentDelimiter>
 <convertLineBreaks>true</convertLineBreaks>
 </deserializationProperties>
 </inboundProperties>
 <outboundProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2DataProperties">
 <serializationProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2SerializationProperties">
 <handleRepetitions>true</handleRepetitions>
 <handleSubcomponents>true</handleSubcomponents>
 <useStrictValidation>false</useStrictValidation>
 <stripNamespaces>true</stripNamespaces>
 <segmentDelimiter>\r\n</segmentDelimiter>
 <convertLineBreaks>true</convertLineBreaks>
 </serializationProperties>
 <deserializationProperties class="com.mirth.connect.plugins.datatypes.hl7v2.HL7v2DeserializationProperties">
 <useStrictValidation>false</useStrictValidation>
 <stripNamespaces>true</stripNamespaces>
 <segmentDelimiter>\r\n</segmentDelimiter>
 <convertLineBreaks>true</convertLineBreaks>
 </deserializationProperties>
 </outboundProperties>
</transformer>
<channel>
```

12 revisions      Prune      Revert      Compare      Close

Filter: Enter channel tag or name      5 Groups, 13 Channels, 13 Enabled

Connected to: Production Server 1 | https://localhost:8443 | 9:05 AM PST (UTC -8)

## Email Reader

Securely connect to a POP3 or IMAP email server and download email messages for processing in a channel using the Email Reader. With the Message Content parameter, you can specify if an individual email message should be read as XML, including the metadata and the body, as just the body, or as a set of attachments. Access numerous options for specifying behavior once a message is read. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the 'Edit Channel - Email Reader' configuration page within the Mirth Connect Administrator. The left sidebar contains navigation links for Dashboard, Channels, Users, Settings, Alerts, Events, and Extensions. The main panel is titled 'Edit Channel - Email Reader' and shows the following settings:

- Connector Type:** Email Reader
- Polling Settings:**
  - Schedule Type: Interval (selected), Next poll at: Thursday, Dec 15, 9:06:30 AM
  - Poll Once on Start: No
  - Interval: 5 seconds
- Source Settings:**
  - Source Queue: OFF (Respond after processing)
  - Queue Buffer Size: 1000
  - Response: None
  - Process Batch: Yes (selected)
  - Batch Response: First (selected)
  - Max Processing Threads: 1
- SSL Settings:**
  - Use SSL Manager: Yes (selected)
  - Current Security: Trusting 3 certs and Java truststore; Client cert: <None Selected>; 2 enabled protocols; 29 enabled cipher suites
  - Use STARTTLS: Yes (selected)
- Email Reader Settings:**
  - Protocol: IMAP (selected)
  - Host: imap.gmail.com
  - Port: 993
  - Username: myemail@gmail.com
  - Password: [REDACTED]
  - Message Content: All (selected)
  - Filter Expression: keyword
  - Read Folder: INBOX
  - Process: Unread (selected)
  - After Processing: Leave (selected)
  - Mark as Read: Yes (selected)
  - Error Move-to Folder: Errors

At the bottom of the page, a status bar indicates: Connected to: Production Server 1 | https://localhost:8443 | 9:07 AM PST (UTC -8).

## FHIR Connector

FHIR (Fast Healthcare Interoperability Resources) is a new set of HL7 healthcare standards. Its main focus is on the ease of implementation, based on RESTful HTTP using XML or JSON. Components called "Resources" are used to store and exchange data between systems. The resources are just XML or JSON documents following a standard definition, and include the actual resource data (like patient demographics), metadata, tags, and a human-readable description. However they are also fully extensible, meaning that additional implementation-specific data elements can be added if needed.

This extension provides FHIR Listener and FHIR Sender connectors, a new FHIR data type, a FHIR Resource Builder available as both a transformer step and code template type, and some helper/utility classes for working with responses. For more information, check out the user guide on the public wiki: [https://www.mirthcorp.com/community/wiki/display/mirth/FHIR+Connector+Extension+\(3.6\)](https://www.mirthcorp.com/community/wiki/display/mirth/FHIR+Connector+Extension+(3.6))

The screenshot shows the 'Edit Channel - FHIR Listener' configuration page in the Mirth Connect Administrator. The page includes sections for Listener Settings, Source Settings, SSL Settings, HTTP Authentication, and FHIR Listener Settings. In the FHIR Listener Settings section, the 'Base Context Path' is set to '/r3', 'Receive Timeout (ms)' is 0, and the 'HTTP URL' is 'http://localhost:9001/r3'. The 'Resource Interactions' table lists various FHIR resources and their supported interactions (read, read, update, patch, delete, history-instance, history-type, create, search-type). A sidebar on the left provides navigation links for Dashboard, Channels, Users, Settings, Alerts, Events, Extensions, and other tasks like Save Changes, Validate Connector, and Import Connector.

Resource	Interactions
Account	read,read,update,patch,delete,history-instance,history-type,create,search-type
ActivityDefinition	read,read,update,patch,delete,history-instance,history-type,create,search-type
AllergyIntolerance	read,read,update,patch,delete,history-instance,history-type,create,search-type
AdverseEvent	read,read,update,patch,delete,history-instance,history-type,create,search-type
Appointment	read,read,update,patch,delete,history-instance,history-type,create,search-type
AppointmentResponse	read,read,update,patch,delete,history-instance,history-type,create,search-type
AuditEvent	read,read,update,patch,delete,history-instance,history-type,create,search-type
Basic	read,read,update,patch,delete,history-instance,history-type,create,search-type
Binary	read,read,update,patch,delete,history-instance,history-type,create,search-type
BodySite	read,read,update,patch,delete,history-instance,history-type,create,search-type
Bundle	read,read,update,patch,delete,history-instance,history-type,create,search-type
CapabilityStatement	read,read,update,patch,delete,history-instance,history-type,create,search-type
CarePlan	read,read,update,patch,delete,history-instance,history-type,create,search-type
CareTeam	read,read,update,patch,delete,history-instance,history-type,create,search-type

https://localhost:8443 - Mirth Connect Administrator - (3.6.0)

### Edit Channel - Convert HL7 to FHIR Patient - Example 1 - Source Transformer

Mirth Views    Back to Channel

Transformer Tasks

- Add New Step
- Delete Step
- Assign To Iterator
- Import Transformer
- Export Transformer
- Validate Transformer
- Validate Step
- Move Step Up
- Move Step Down

Other

- Notifications
- View User API
- View Client API
- Help
- About Mirth Connect
- Visit mirthcorp.com
- Report Issue
- Logout

Step \ Generated Script \

Store As: Local Variable Map Variable Channel Map

Variable: patient

Resource: Patient

Object Builder Variable: Patient

resourceType: Enum Patient Variable: String

identifier: Array Builder Variable: Identifier[] identifiers

active: Boolean

name: Array Builder Variable: HumanName[] combineArrays(names, aliases)

telecom: Array Builder Variable: ContactPoint[] combineArrays(homePhoneNumb)

gender: Enum male Variable: String convertToFhirGender(msg["PID"]["PID.8"]?)

birthDate: String convertToFhirDate(msg["PID"]["PID.7"]|PID|)

deceasedBoolean: Boolean convertYesNoIndicator(msg["PID"]["PID.2"]?)

deceasedDateTime: String convertToFhirTimestamp(msg["PID"]["PID.4"]|PID|)

address: Array Builder Variable: Address[] addresses

maritalStatus: Object Builder Variable: CodeableConcept

coding: Array Builder Variable: Coding[]

text: String msg["PID"]["PID.16"]|PID.16.1|toString()

id: String

extension: Array Builder Variable: Extension[]

multipleBirthBoolean: Boolean convertYesNoIndicator(msg["PID"]["PID.2"]?)

multipleBirthInteger: Number

photo: Array Builder Variable: Attachment[]

contact: Array Builder Variable: Patient\_Contact[] contacts

animal: Object Builder Variable: Patient\_Animal

communication: Array Builder Variable: Patient\_Communication[]

Reference \ Message Trees \ Message Templates \

Category: All

Filter:

All

- Get Serializer
- Convert XML to JSON
- Convert JSON to XML
- Convert Delimited Text to XML (default parameters)
- Convert Delimited Text to XML (custom parameters)
- Convert XML to Delimited Text (default parameters)
- Convert XML to Delimited Text (custom parameters)
- Convert HL7 v2.x to XML (default parameters)
- Convert HL7 v2.x to XML (custom parameters)
- Convert XML to HL7 v2.x (default parameters)
- Convert XML to HL7 v2.x (custom parameters)
- Convert DICOM to XML (default parameters)
- Convert XML to DICOM (default parameters)
- Convert HL7 v3.x to XML (default parameters)
- Convert HL7 v3.x to XML (custom parameters)
- Convert XML to HL7 v3.x (default parameters)
- Convert NCPDP to XML (default parameters)
- Convert NCPDP to XML (custom parameters)
- Convert XML to NCPDP (default parameters)

Available Variables

## Interoperability Connector Suite

This extension provides Listener and Sender connectors that can help kickstart your [eHealth Exchange](#) onboarding and integration. The following protocols and operations are supported:

- **PIX** (v2 and v3)
  - PRPA\_IN201301UV02
  - PRPA\_IN201302UV02
  - PRPA\_IN201304UV02
  - PRPA\_IN201309UV02
- **PDQ** (v2 and v3)
  - PRPA\_IN201305UV02
  - QUQI\_IN000003UV01\_Cancel
  - QUQI\_IN000003UV01\_Continue
- **XDS.b**
  - RegistryStoredQuery
  - RegisterDocumentSet-b
  - RetrieveDocumentSet
  - ProvideAndRegisterDocumentSet-b
- **XCA**
  - CrossGatewayQuery
  - CrossGatewayRetrieve
- **XCPD**
  - PRPA\_IN201305UV02

The connectors accept SOAP XML from external systems, and convert it to JSON for transforming within a channel. They also support automatic generation and validation of NHIN (eHealth Exchange) compliant WS-Security/SAML. In addition, this extension includes a new UDDI Provider [resource](#) which automatically downloads and syncs business endpoint information from a remote UDDI provider.

[https://localhost:8443 - Mirth Connect Administrator - \(3.6.0\)](https://localhost:8443)

**Edit Channel - XCPD and XCA Sender**

Summary \ Source \ Destinations \ Scripts \

Status	Destination	Id	Connector Type	Chain
Enabled	Patient Discovery	4	XCPD Sender (SSL Configured)	1
Enabled	Cross Gateway Query	5	XCA Sender (SSL Configured)	1
Enabled	Cross Gateway Retrieve	6	XCA Sender (SSL Configured)	1

**Connector Type:** XCPD Sender  Wait for previous destination

**Destination Settings**

- Queue Messages:  Never  On Failure  Always
- Advanced Queue Settings:  0 Retries
- Validate Response:  Yes  No
- Reattach Attachments:  Yes  No

**SSL Settings**

- Use SSL Manager:  Yes  No
- Current Security: Trusting Java truststore; Client cert: mysequoiacert; 2 enabled protocols; 22 enabled cipher suites

**XCPD Sender Settings**

- Use Default WSDL:  Yes  No
- WSDL URL: [Using Button Default WSDLs...](#)
- Use UDDI:  Yes  No [AEGIS DIL, apple pie test system, PatientDiscovery 2.0](#)
- Location URI: <https://www.apple.pie.com:443/patientdiscovery>
- Socket Timeout (ms): 30000
- Validate WS-Security:  Yes  No
- Generate WS-Security:  Yes  No
- Operation: RespondingGateway\_PRPA\_IN201305UV02
- Use SOAP Template:  Yes  No [Generate Envelope](#)
- SOAP Envelope:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:h17-org:v3">
<soap:Header>
<PRPA_IN201305UV02 ITSPVersion="XHL_1.0" xmlns="urn:h17-org:v3" xmlns:ns2="urn:h17-org:sdtc" xmlns:ns3="urn:h17-org:ccat" xmlns:ns4="urn:h17-org:evn">
<acceptAckCode code="NE"/>
<controlActProcess statusCode="CACT" moodCode="EVN">
<authorOrPerformer typeCode="AUT">
<assignedDevice statusCode="ASSIGNED">
<id value="1.2.3.4.5"/>
</assignedDevice>
</authorOrPerformer>
```

- Use MTOM:  Yes  No

**Attachments:**

ID	Content	MIME Type

**Destination Mappings**

- Channel ID
- Channel Name
- Message ID
- Raw Data
- Transformed Data
- Encoded Data
- Message Source
- Message Type
- Message Version
- Date
- Formatted Date
- Timestamp
- Unique ID
- Original File Name
- Count
- XML Entity Encoder
- XML Pretty Printer
- Escape JSON String
- JSON Pretty Printer
- CDATA Tag
- DICOM Message Raw Data
- lastName
- firstName
- dob

## LDAP Authorization

The LDAP Authorization replaces the existing authentication mechanism and instead authenticates against an LDAP server so you can manage user accounts on a centralized LDAP server. Any user contained within the specified User Base DN on the LDAP server can log in to NextGen Connect Integration Engine. When a user logs in, NextGen Connect Integration Engine copies the user's attributes from the LDAP server. The connection to the LDAP server can optionally use SSL or STARTTLS encryption. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the Mirth Connect Administrator interface with the following details:

- LDAP Configuration:**
  - LDAP Enabled: Yes (radio button selected)
  - Host: localhost
  - Port: 10389
  - Base DN: ou=system
  - Encryption: None (radio button selected)
  - Certificate Validation: Enabled (radio button selected)
  - Hostname Verification: Enabled (radio button selected)
  - Admin User DN: uid=admin,ou=system
  - Change Admin Password: [empty input field]
  - Manage Roles: Yes (radio button selected)
- Search Contexts:** A table listing search contexts with their DN, Recursive setting, and assigned Role:
 

DN	Recursive	Role
ou=users	Yes	Help Desk
uid=admin	No	Administrator
ou=groups	Yes	Power Users
ou=partitions,ou=configuration	No	Read Only
- Local Recovery Administrator Account:**
  - Username: recoveryuser
  - Change Password: [redacted]
  - Confirm Password: [redacted]
- LDAP Attribute Mapping:**

Username (uid)	First Name (cn)	Last Name (sn)	Organization (ou)	Email (mail)	Phone # (telephoneNumber)	Description (displayName)	Roles
test2	john	doe			123-456-7890		Help Desk
test3	test	testusername					Help Desk
test	test	testusername					Help Desk
admin	system adm...	administrator				Directory Superuser	Administrator

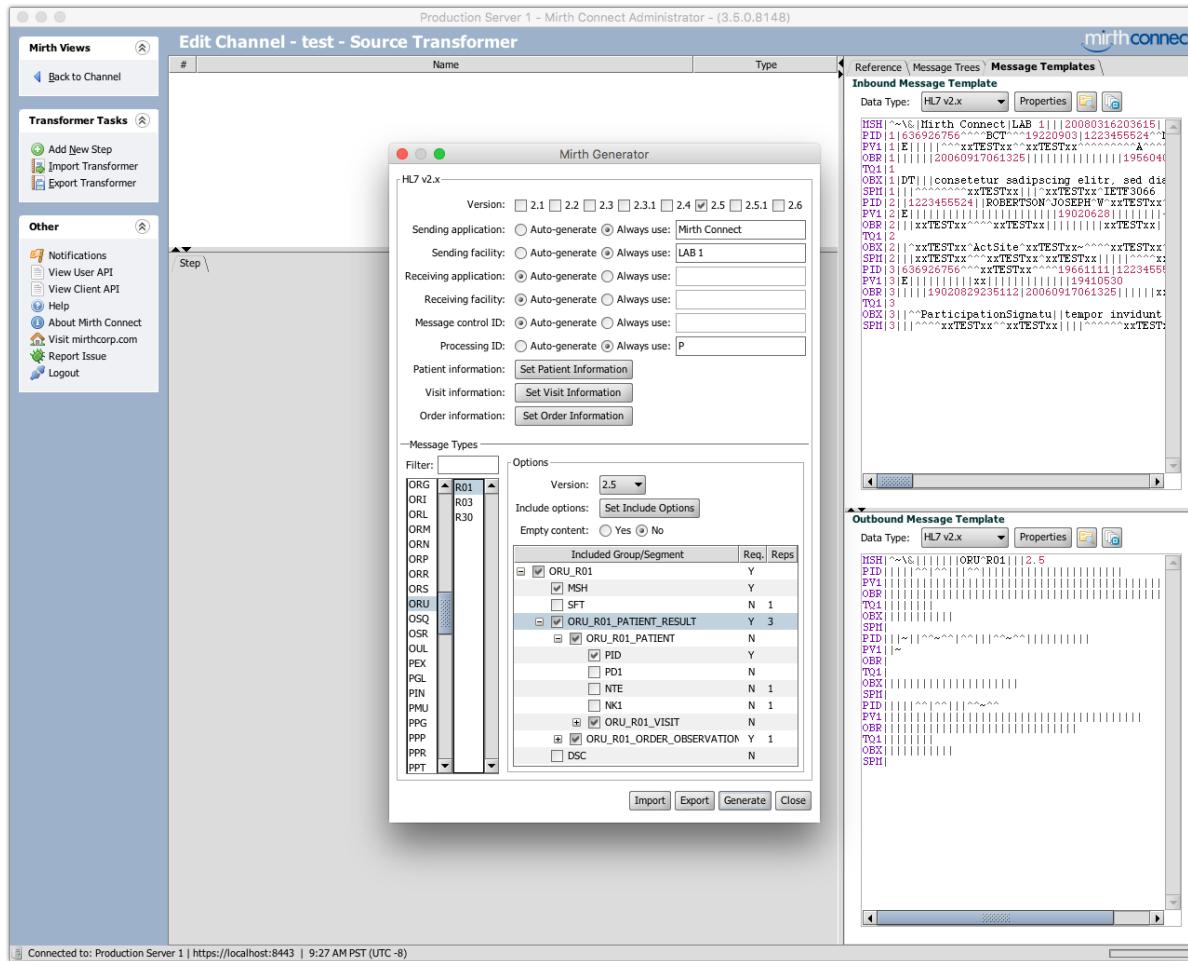
The following attributes are not currently mapped to any user fields in Mirth Connect:

Attribute
userPassword
objectClass
privateKey
userCertificate
keyAlgorithm
privateKeyFormat
publicKeyFormat

A modal dialog box titled "Test Connection" is open, displaying the message: "Successfully connected. Found 4 user accounts." It also contains instructions to review user attribute mappings and map them in the LDAP Attribute Mapping section.

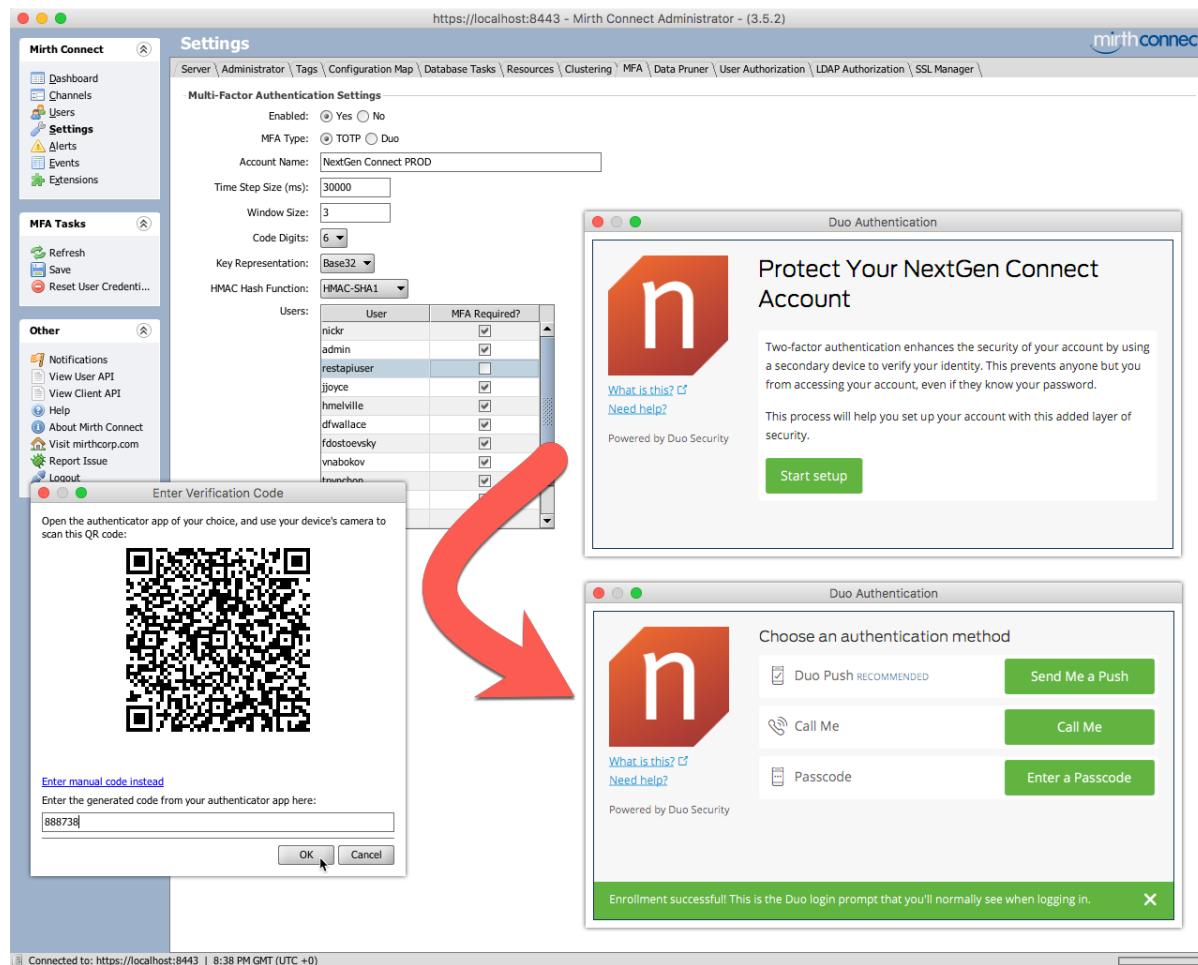
## Message Generator

With the Message Generator, quickly and easily generate HL7 v2.x messages for use as a transformer's inbound or outbound template, for sending to a channel, or for testing. Create messages of any type/trigger and any version (HL7 2.1 through 2.6), with specific options for which segments, fields, and components to include. Where applicable, pseudo data is generated (dates, names, free text, and even random values from the HL7 specification tables), though there are also several options to override parts of the message with your own data. Feel free to [Contact Us](#) if you have any questions.



## Multi-Factor Authentication

This extension provides an extra layer of security for all user accounts. A secondary device such as a phone, tablet, or landline is used to ensure that no one but the actual user can login, even if they know the correct username and password. Both Duo and generic apps such as Google Authenticator / Authy are supported. When using Duo, advanced options such as lockout protection, automatic enrollment, and policy management are available in the Duo administrator dashboard. Feel free to [Contact Us](#) if you have any questions.



## Serial Connector

The Serial Connector allows NextGen Connect Integration Engine to send and receive data over serial communication ports, such as those compliant with the RS-232 standards. Any installed transmission mode may be used in conjunction with the connectors, including a raw serial mode, **MLLP**, or the **ASTM E1381** transmission mode. Options are available to select the port to connect to, the baud rate, the parity, and much more, allowing for fully customizable communication. This extension comes with both a Serial Listener source connector and a Serial Sender destination connector. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the 'Edit Channel - RS-232 Sender' configuration screen in the Mirth Connect Administrator. The left sidebar includes 'Mirth Connect' (Dashboard, Channels, Users, Settings, Alerts, Events, Extensions), 'Channel Tasks' (Save Changes, Validate Connector, New Destination, Delete Destination, Clone Destination, Disable Destination, Edit Filter, Edit Transformer, Edit Response, Import Connector, Export Connector, Export Channel, Deploy Channel), and 'Other' (Notifications, View User API, View Client API, Help, About Mirth Connect, Visit mirthcorp.com, Report Issue, Logout). The main panel displays the 'Edit Channel - RS-232 Sender' configuration with the following details:

- Connector Type:** Serial Sender
- Destination Settings:**
  - Queue Messages:  Always
  - Advanced Queue Settings: Interval 10000 ms
  - Validate Response:  Yes
- Serial Sender Settings:**
  - Transmission Mode:** ASTM E1381
  - ASTM E1381 Settings:**
    - Port: /dev/tty.Bluetooth-Incoming-Port
    - Baud: 9600
    - Data Bits: 8
    - Parity: None
    - Stop Bits: 1
    - Flow Control: None
    - Reconnect Interval (ms): 10000
    - Response Timeout (ms): 5000
    - Data Type:  Text
    - Encoding: Default
    - Template: \${message.encodedData}

A 'Destination Mappings' sidebar on the right lists various message components: Channel ID, Channel Name, Message ID, Raw Data, Transformed Data, Encoded Data, Message Source, Message Type, Message Version, Date, Formatted Date, Timestamp, Unique ID, Original File Name, Count, XML Entity Encoder, XML Pretty Printer, Escape JSON String, JSON Pretty Printer, CDATA Tag, and DICOM Message Raw Data.

## SSL Manager

Use the SSL Manager to quickly enable and configure certificate-based SSL connectivity for socket-based connectors such as the [HTTP Listener / Sender](#), [Web Service Listener / Sender](#), and [FTP Reader / Writer](#). The central settings view allows you to manage and store your certificates in one location. Trusted certificates and advanced SSL settings such as client (two-way) authentication and hostname verification can be applied on a per-connector basis. Feel free to [Contact Us](#) if you have any questions.

Alias	Subject CN	Issuer CN	Valid Until
[Java Truststore]	--	--	--
geotrust global ca	GeoTrust Global CA	--	2018-08-20 21:00:00
google internet authority g2	Geo Internet Authority G2	GeoTrust Global CA	2017-12-31 15:59:59
imap.gmail.com	imap.gmail.com	Google Internet Authority G2	2017-03-02 02:18:00
www.digicert.com	--	--	--
digicert sha2 extended validation server	--	--	--
www.chase.com	--	--	--
symantec class 3 ev ssl ca - g3	--	--	--
www.amazon.com	--	--	--
symantec class 3 secure server ca - g4	--	--	--
www.entrust.com	--	--	--
entrust certification authority - l1m	--	--	--
entrust root certification authority - g2	--	--	--

## User Authorization

User Authorization provides role-based access control to all aspects of the Mirth Connect Administrator. Create new roles with specific permissions to areas such as channel management or message browsing. Assign any number of roles to users. Use this to manage access to sensitive channel and messaging data across your enterprise. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the 'User Authorization' section of the Mirth Connect Administrator interface. The left sidebar includes links for Dashboard, Channels, Users, Settings (which is selected), Alerts, Events, and Extensions. The main content area has a title 'Settings' and a breadcrumb trail: Server \ Administrator \ Tags \ Configuration Map \ Database Tasks \ Resources \ Data Pruner \ User Authorization \ LDAP Authorization \ SSL Manager \. The main table displays users and their assigned roles:

User	Roles	Edit
admin	Administrator	<a href="#">Select Roles...</a>
johnd	Help Desk	<a href="#">Select Roles...</a>
janed	Power Users	<a href="#">Select Roles...</a>

Below the table, the 'Role Settings' section is expanded, showing detailed permissions for the 'Administrator' role. The permissions are organized into categories: Alerts, Channels, Scripts and Templates, Messages, Events, Users, Extensions, Server Configuration and Settings, Tags, and Configuration Map. Most permissions are checked by default, except for some under 'Channels' and 'Events' which are unchecked. At the bottom of the 'Role Settings' panel, there is a note: 'All channels are accessible by this role.' The status bar at the bottom indicates: Connected to: Production Server 1 | https://localhost:8443 | 9:34 AM PST (UTC -8).

## NextGen Results CDR Connector

The NextGen Results CDR Connector provides a scalable and streamlined path to getting data into [NextGen Results CDR](#). Create a resource once, and then select that resource on any Mirth Results Sender destination connector. Supports posting data in a variety of formats, and comes with a built-in connection pool for persistent and highly available processing. Feel free to [Contact Us](#) if you have any questions.

The screenshot shows the Mirth Connect Administrator interface with the following details:

- Header:** Mirth Connect PROD 1 - Mirth Connect Administrator - (3.5.0.8232)
- Left Sidebar (Mirth Connect):**
  - Dashboard
  - Channels
  - Users
  - Settings
  - Alerts
  - Events
  - Extensions
- Left Sidebar (Channel Tasks):**
  - Save Changes
  - Validate Connector
  - New Destination
  - Delete Destination
  - Clone Destination
  - Disable Destination
  - Edit Filter
  - Edit Transformer (3)
  - Edit Response
  - Import Connector
  - Export Connector
  - Export Channel
  - Deploy Channel
- Left Sidebar (Other):**
  - Notifications
  - View User API
  - View Client API
  - Help
  - About Mirth Connect
  - Visit mirthcorp.com
  - Report Issue
  - Logout
- Main Content Area:**

### Edit Channel - CDA to MR

Summary \ Source \ Destinations \ Scripts (1) \

Status	Destination	Id	Connector Type	Chain
Enabled	Post CDA to MR	1	Mirth Results Sender	1

**Connector Type:** Mirth Results Sender  Wait for previous destination

**Destination Settings:**

  - Queue Messages:  Never  On Failure  Always
  - Advanced Queue Settings:  Interval 10000 ms / 10 Threads / Group By mrn
  - Validate Response:  Yes  No
  - Reattach Attachments:  Yes  No

**Mirth Results Sender Settings:**

  - Resource: Mirth Results PROD
  - Data Type:  CDA  HL7 v2.x  CDM
  - Source Id: \${sourceId}
  - Receiver Id: \${receiverId}
  - XDS Ingestion:  Yes  No
  - Clinical Item Key: \${clinicalItemKey}
  - Include Original:  Yes  No
  - Original: \${message.rawData}
  - Document: \${message.encodedData}

**Destination Mappings:**

  - Channel ID
  - Channel Name
  - Message ID
  - Raw Data
  - Transformed Data
  - Encoded Data
  - Message Source
  - Message Type
  - Message Version
  - Date
  - Formatted Date
  - Timestamp
  - Unique ID
  - Original File Name
  - Count
  - XML Entity Encoder
  - XML Pretty Printer
  - Escape JSON String
  - JSON Pretty Printer
  - CDATA Tag
  - DICOM Message Raw Data
  - sourcedId
  - receiverId
  - clinicalItemKey
- Bottom Status Bar:** Connected to: Mirth Connect PROD 1 | https://localhost:7443 | 7:05 PM PDT (UTC -7)

## Training

We offer two levels of [NextGen Connect Certification Training](#):

In **NextGen Connect Fundamentals Certification Training**, we'll cover everything from basic installation to NextGen Connectors. You'll leave with a solid grasp on NextGen Connect Integration Engine. It's designed to be your first NextGen Connect Integration Engine class and certification.

During **NextGen Connect Advanced Certification Training**, for users who have completed the Mirth Fundamentals class, you'll take an even deeper dive into building your own channels and applications with Mirth solutions. This class will dig into topics such as customizing interfaces using Java and JavaScript, in-depth HL7 processing, working with databases, and using HTTP, web services, and JSON. Already certified on NextGen Connect Integration Engine? Take your skills to the next level!

We offer public classes in Costa Mesa, Atlanta, and London. We also offer private classes where we'll come to your location. For more information, go here: <https://www.nextgen.com/Interoperability/Mirth-Solutions/Training>



Don't hesitate to [Contact Us](#) if you have any questions about commercial support, training, or any of the extensions!