User Guide for Mirth® Connect by NextGen Healthcare, 4.5

NOTICE: This document contains information that is proprietary to NextGen Healthcare, Inc. and its subsidiaries and affiliates ("Company") and is to be accessed and used solely by those utilizing Company's products and/or services under one or more agreements between you and Company now in force.

This document may not be copied, reproduced, published, displayed, otherwise used, transmitted, or distributed in any form by any means as a whole or in any part, nor may any of the information it contains be used or stored in any information retrieval system or media, or translated into another language without the prior, written consent of Company. Any other use or distribution of the contents of this document, as a whole or in any part, is prohibited.

Although we exercise great care in creating this publication, Company assumes no responsibility for errors or omissions that may appear in this publication and reserves the right to change this publication at any time without notice.

© NXGN Management, LLC. All Rights Reserved.

Our registered trademarks and issued and published patents can be found at https://www.nextgen.com/legal-notice. All other names and marks are the property of their respective owners.

Contents

Chapter 1 Legal Notice 19

Chapter 2 Introduction to Mirth Connect 20
The NextGen Connected Health Solutions Mission 20
About Mirth Connect 20
The Healthcare Interoperability Challenge and Solution 21
Mirth Connect as Open Source Software 22

Chapter 3 Getting Started with Mirth Connect 23
Mirth Connect System Requirements 23
Java Requirements 24
Database Requirements 24
Install Mirth Connect Using the Mirth Connect Installer 25
Install Mirth Connect Using the Command-Line Installer 36
The Mirth Connect Server Manager 44
Service Tab 44
Server Tab 44
Database Tab 46
Info Tab 47
The Web Dashboard 47
Changing the Database Type 49
Backup Current Server Configuration 49
Change Database Settings 50
Editing the Properties File Directly for Windows, Linux, and Mac 52
Restart the Mirth Connect Server 53
Restore Server Configuration 53
Using Java 9 or greater 55
Mirth Connect Administrator Overview 55
Download the Administrator Launcher 55
Launch the Administrator 56
Log On 57
Logging on for the First Time 58

Chapter 4 The Fundamentals of Mirth Connect 60
About Channels and Connectors 60

Channel Components 60
General Channel Properties 61
Source Connector 61
Destination Connectors 62
Channel Scripts 62
Connector Components 63
General Connector Properties 63
Connector-Specific Properties 63
Filter 65
Transformer 65
About Message Data 65
Message Metadata 66
Message Content 67
Message Attachments 68
The Message Processing Lifecycle 68
Source Processing Steps 69
Destination Processing Steps 70
Final Processing Steps 70
Destination Chains 71
About Data Types 72
About Filters 74
The "msg" Object 75
Filter Rule Types 75
About Transformers 76
The "msg" Object 76
The "tmp" Object 76
Response Transformers 76
Transformer Step Types 77

Chapter 5 Administrator Launcher Overview 78

Chapter 6 Mirth Connect Administrator 79
Administrator Layout 79
Working With Tables 80
Monitor Views 84
Dashboard View 84
Dashboard Table 86
Dashboard Table Columns 86
View Messages for a Channel 88
Show or Hide Channel Groups 88
Change How Tags Display 89

Server Log 89
Connection Log 91
Global Maps 92
Global Maps Table Columns 92
Dashboard Tasks 93
Send Message 95
Remove All Messages 96
Clear Statistics 97
Filter By Channel Name or Tag 98
View All Available Tags/Names 98
Auto-Complete Tags/Names 99
Filter By Channel Tags 99
Filter By Channel 101
Filter By Partial Channel Name 101
Filter By Multiple Criteria 102
Clear Filter Criteria 102
Message Browser View 103
Navigation 104
From the Dashboard 104
From the Channels View 105
Metadata Table 106
Add a Column to the Metadata Table 107
Metadata Table Columns 108
Custom Metadata Columns 111
Message Content Tab 111
View Message Content 112
Message Content Types 113
Formatting Messages 114
Mappings Tab 115
View Mappings 116
Mappings Table Columns 116
Errors Tab 116
View Message Errors 117
Error Content Types 118
Attachments Tab 118
Attachment Table Columns 119
View Attachments 119
Text Attachment Viewer 120
Image Attachment Viewer 121
DICOM Attachment Viewer 122
PDF Attachment Viewer 123

Search Messages 124
Message Search Options 125
Advanced Search Filter 127
Message Browser Tasks 130
Import Messages 131
Export Results 133
Remove Results 135
Reprocess Results 136
Reprocess Message 137
Export Attachment 138
Alerts View 139
Navigation 139
Alerts Table 140
Alerts Table Columns 140
Alerts Tasks 140
Events View 141
Navigation 142
Events Table 142
Metadata Table Columns 143
PHI Events 144
Event Attributes Table 145
Searching Events 145
Advanced Search Filter 146
Event Tasks 147
Export All Events 147
Management Views 148
Channels View 148
Navigation 149
Channel Table 150
Channel Table Columns 151
Show or Hide Channel Groups 151
Change How Tags Display 151
Filter By Channel Name or Tag 151
Use Drag and Drop 152
Get the Channel Name/ID 152
Assign Channels to a Group 152
Import Channels/Groups Using Drag-and-Drop 153
Channel Tasks 154
Import Channel 156
Debug Channel 157
Export Channel 157

Group Tasks . . . 158
Edit Group Details . . . 159
Import Group . . . 160
Export Group . . . 160
Users View . . . 160
Navigation . . . 161
Users Table . . . 161
Users Table Columns . . . 162
Users Tasks . . . 163
Add/Update a User Account . . . 163
Settings View . . . 166
Navigation . . . 166
Server Settings Tab . . . 167
General Settings . . . 169
Channel Settings . . . 170
Email Settings . . . 171
Notification Settings . . . 172
Tasks . . . 173
Restore Config . . . 174
Clear All Statistics . . . 174
Administrator Settings Tab . . . 174
System Preferences . . . 175
User Preferences . . . 177
Code Editor Preferences . . . 178
Tasks . . . 179
Tags Settings Tab . . . 179
Tags Table . . . 180
Adding a Tag . . . 181
Removing a Tag . . . 181
Channels Table . . . 181
Indeterminate Check Boxes . . . 182
Configuration Map Settings Tab . . . 182
Table Columns . . . 183
Tasks . . . 183
Database Tasks Settings Tab . . . 184
Database Tasks Table Columns . . . 185
Affected Channels Table . . . 185
Running a Database Task . . . 185
Resources Settings Tab . . . 186
Resources Table Columns . . . 187
Tasks . . . 188

Reload Resource . . . 188
Directory Resource . . . 188
Using Resources in Channels/Connectors . . . 190
Data Pruner Settings Tab . . . 190
Status . . . 191
Schedule . . . 192
Prune Settings . . . 193
Archive Settings . . . 194
Tasks . . . 195
Settings Tasks . . . 196
Extensions View . . . 196
Navigation . . . 197
Installed Connectors Table . . . 198
Installed Connectors Table Columns . . . 199
Installed Plugins Table . . . 199
Installed Plugins Table Columns . . . 200
Install a New Extension . . . 200
Extension Tasks . . . 201
Show Properties . . . 202
Edit Views . . . 203
Edit Channel View . . . 203
Navigation . . . 203
Summary Tab . . . 204
Channel Properties . . . 205
Set Data Types Window . . . 207
Select Data Types . . . 208
Change Data Type Properties . . . 209
Bulk Edit Mode . . . 210
Set Dependencies Window . . . 212
Code Template Libraries . . . 213
Link Code Template Libraries . . . 213
Library Resources . . . 214
Deploy/Start Dependencies . . . 215
Deploy/Start Channels . . . 217
Pause/Stop/Undeploy Channels . . . 218
Attachment Handlers . . . 219
Extraction . . . 220
Reattachment . . . 220
Expanded Replacement Tokens . . . 220
Attachment MIME Types . . . 221
Attachment Handler Properties . . . 221

Entire Message Attachment Handler Properties 221
Regex Attachment Handler Properties . . 222
Regular Expressions Table . . 223
String Replacement Tables . . 224
DICOM Attachment Handler Properties . 224
JavaScript Attachment Handler Properties 224
Scope Variables. 225
Extract Attachments 225
Custom Attachment Handler Properties . 226
Properties 227
Message Storage Settings 227
Message Pruning Settings 229
Custom Metadata Columns. 230
Modifying Custom Metadata Columns 231
Channel Description 231
Source Tab 232
Choose a Source Connector. 233
Listener Settings 233
Polling Settings 234
Interval Schedule Settings. 235
Time Schedule Settings. 236
Cron Schedule Settings. 236
Advanced Settings 237
Source Settings. 238
HTTP Authentication Settings 239
Choose an Authentication Type 240
Basic HTTP Authentication 240
Digest HTTP Authentication 241
JavaScript HTTP Authentication 243
Custom Java Class HTTP Authentication 244
OAuth 2.0 Token Verification. 245
Source Connector Properties 245
Destinations Tab 246
Destination Table. 247
Destination Tasks 248

Destination Settings 249
Advanced Queue Settings 251
Destination Connector Properties 252
Destination Mappings 253
Standard Variables/Templates 253
Scripts Tab 255
Edit Channel Tasks 256
Edit Filter / Transformer Views 257
Navigation 258
Message Templates Tab 260
Edit Data Types 262
Specify a Template 264
Message Trees Tab 265
Filter By Node Name/Description 267
Create a Rule Builder Rule or Mapper Step 268
Create a Message Builder Step 270
Drag-and-Drop Field Values 272
Reference Tab 273
Reference List 274
Available Variables 275
Create New Rules/Steps 276
Rule/Step Table 277
Filter Rule Properties 278
Rule Builder Filter Rule 278
JavaScript Filter Rule 279
External Script Filter Rule 279
Iterator Filter Rule 280
Transformer Step Properties 281
Mapper Transformer Step 281
Message Builder Transformer Step 282
JavaScript Transformer Step 283
External Script Transformer Step 283
XSLT Transformer Step 283
Destination Set Filter Transformer Step 284
Iterator Transformer Step 286
Response Transformers 286
Modifying the Response 287
Common Scenarios 287
Working With Iterators 288
Creating Iterators From Drag-and-Drop 289
The Assign To Iterator Task 291

Remove From Iterators . . . 294
View Generated Script . . . 294
Filter Tasks . . . 295
Import Filter . . . 296
Transformer Tasks . . . 297
Import Transformer . . . 298
Edit Global Scripts View . . . 299
Navigation . . . 299
Edit Global Scripts . . . 300
Tasks . . . 301
Edit Code Templates View . . . 301
Navigation . . . 302
Code Template Library Table . . . 303
Edit Library Panel . . . 304
Link Libraries to Channels . . . 305
Edit Code Template Panel . . . 305
Code Template Contexts . . . 307
Use JSDoc in Code Templates . . . 308
Code Template Tasks . . . 310
Import Code Templates/Libraries . . . 311
Edit Alert View . . . 313
Navigation . . . 314
Alert Error Types and Regex . . . 315
Alert Error Categories . . . 316
Alert Enabled Channels . . . 317
Alert Actions . . . 318
Alert Variables . . . 319
Edit Alert Tasks . . . 319
Other Tasks . . . 320
Notifications . . . 321

Chapter 7 Data Types 323
Inbound Properties . . . 323
Outbound Properties . . . 323
Delimited Text Data Type . . . 324
DICOM Data Type . . . 327
EDI / X12 Data Type . . . 328
HL7 v2.x Data Type . . . 329
HL7 v3.x Data Type . . . 332
JSON Data Type . . . 333
NCPDP Data Type . . . 334

Raw Data Type . . . 335
XML Data Type . . . 336
Batch Processing . . . 337
JavaScript Batch Script . . . 338

Chapter 8 Source Connectors 340
Channel Reader . . . 340
Source Map Variables . . . 341
DICOM Listener . . . 341
Source Map Variables . . . 344
Database Reader . . . 346
Editing Database Drivers . . . 350
File Reader . . . 351
Advanced FTP Options . . . 357
Advanced SFTP Options . . . 357
Advanced SMB Options . . . 359
Advanced Amazon S3 Options . . . 359
Source Map Variables . . . 361
HTTP Listener . . . 362
JMS Listener . . . 366
JMS Connection Templates . . . 369
JavaScript Reader . . . 371
JavaScript Reader Return Values . . . 371
TCP Listener . . . 372
Source Map Variables . . . 376
TCP Transmission Modes . . . 376
Basic TCP Transmission Mode . . . 377
Byte Abbreviations . . . 377
MLLP Transmission Mode . . . 378
Byte Abbreviations . . . 380
Web Service Listener . . . 381

Chapter 9 Destination Connectors 384
Channel Writer . . . 384
Source Map Variables . . . 386
DICOM Sender . . . 386
Database Writer . . . 390
Document Writer . . . 393
File Writer . . . 395
Connector Map Variables . . . 400
HTTP Sender . . . 400

Connector Map Variables . . . 407
JMS Sender . . . 407
JavaScript Writer . . . 410
JavaScript Writer Return Values . . . 411
SMTP Sender . . . 411
TCP Sender . . . 416
Web Service Sender . . . 422

Chapter 10 Mirth Connect and JavaScript 428
About JavaScript . . . 428
Variables . . . 428
Comments . . . 429
Arrays . . . 429
Operators . . . 430
Arithmetic Operators . . . 430
Assignment Operators . . . 431
Comparison Operators . . . 431
Logical Operators . . . 432
Conditional Statements . . . 432
Functions . . . 432
Loops and Iterations . . . 433
for loops . . . 433
for each…in loops . . . 433
while loops . . . 434
do…while loops . . . 434
Exception Handling . . . 435
Using JavaScript in Mirth Connect . . . 435
About E4X . . . 435
Accessing Message Data with E4X . . . 436
Adding Segments to a Message . . . 437
Deleting a Segment . . . 439
Iterating Over Message Segments . . . 439
Iterating Over Repeating Fields . . . 440
Adding a New Repeating Field . . . 440
Message Variables . . . 440
Built-In Code Templates . . . 441
Using Java Classes . . . 441
Regular Expressions . . . 441
Logging with JavaScript . . . 443
Generating a Hash with JavaScript . . . 443
Using the JavaScript Editor . . . 445

Using the Context Menu in the JavaScript Editor 445
Finding/Replacing Code in the JavaScript Editor 446
Folding in the JavaScript Editor 447
Using the Auto-Completion Popup in the JavaScript Editor 448
Remapping Editor Shortcut Keys 449
Variable Maps 449
Connector Map 450
Channel Map 451
Source Map 451
Response Map 452
Global Channel Map 452
Global Map 453
Configuration Map 453
The Variable Map Lookup Sequence 454
Attachment JavaScript Functions 455
Built-In Attachment Functions 455
The AttachmentUtil Class 457
The Attachment Object 458
Examples 459
The User API (Javadoc) 459
Mirth Connect Debugger 461
To Edit the mcserver.vmoptions file 462
Use the Debugger 463
Debugger Window 464
Debugger Menus 465
File Menu 465
Edit Menu 466
Debug Menu 467
Window Menu 467
Coding Area 468

Chapter 11 Velocity Variable Replacement 469
Basic Syntax 469
Conditional Statements 470
For Loops 470

Chapter 12 Mirth Connect Command Line Interface 471
Running the Command Line Interface 471
Using Non-interactive Scripting 472

Chapter 13 Mirth Connect REST API 473

Authentication 476

Chapter 14 Installation Directory 478
Application Data Directory 478
configuration.properties 478
extension.properties 479
keystore.jks 479
server.id 480
temp 480
Configuration Directory 480
dbdrivers.xml File 480
Adding a new entry to dbdrivers.xml 481
log4j2.properties File 482
log4j2-cli.properties File 484
mirth.properties File 485
Split Database Connection Pools 497
Default Supported Cipher Suites 498
New Default Digest Algorithm in Mirth Connect 4.4 499
Update the Digest Algorithm 499
mirth-cli-config.properties File 500
Other Files and Folders 500

Chapter 15 Frequently Asked Questions 502

Chapter 16 Channel Development Best Practices and Tips 509
Channel Performance 509
Adjust the Message Storage slider so that the channel only retains the message data that you will actually need. 510
If large messages are expected, use an Attachment Handler to improve throughput and reduce memory and disk usage. 510
Enable source queuing if an auto-generated acknowledgement is sufficient for the upstream system. 511
Enable destination queuing if you do not need to respond to the originating system from your destination. 511
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 511
Increase the Queue Threads value if the downstream system can accept multiple concurrent connections. 512
Clear the "Wait for previous destination" checkbox, unless you need destinations to process messages one after another. 512
Use the "Destination Set Filter" feature in your source transformer when you

need to route messages to only one (or a subset) of destinations in your channel. . . . 512
Channel Configuration . . . 513
Use tags to categorize your channels. . . . 514
Place any reusable JavaScript code into Code Template functions and organize them into Code Template Libraries. . . . 514
Use Custom Meta Data Columns to increase the performance of searching the message history on a particular message field. . . . 514
Use Deploy/Start Dependencies if one channel depends on another to operate correctly. . . . 515
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. . . . 516
Message Integrity . . . 516
Use hashing to ensure message integrity . . . 517
Other Tips . . . 517
Do not reference Mirth Connect internal Java classes in your JavaScript code. . . . 518

Chapter 17 Security Best Practices 519
Secure Installation and Deployment . . . 519
Installation Directory . . . 519
Networking. . . . 520
Web Server Certificate . . . 521
Connect to Database using SSL/TLS . . . 522
Importing the Database Server Certificate . . . 522
PostgreSQL . . . 523
MySQL. . . . 523
Oracle . . . 524
SQL Server . . . 524
Secure Configuration. . . . 525
Encryption Settings . . . 525
Encrypt Database Password . . . 527
Plain HTTP Main Web Server . . . 527
Default TLS/SSL Settings . . . 527
Default Supported Cipher Suites . . . 528
Cipher Suites Removed From Earlier Versions . . . 529
New Protocol/Cipher Suite Support in Java 11 . . . 531
Password Requirements . . . 532
SSL Manager Extension . . . 533
Operational Procedures . . . 536
Users/Permissions. . . . 536

Auditing . . . 537
Environmental Upkeep . . . 538
Mirth Connect Software . . . 538
Operating System . . . 539
Java . . . 539

Chapter 18 Troubleshooting 540
Logs . . . 540
Configuration . . . 540
Mirth Connect Engine fails to start up . . . 540
Unable to Launch Mirth Connect Administrator . . . 542
Clearing your Java Cache . . . 543
Opening the Java Client Console . . . 546
Out of Memory Errors . . . 549
Manually reset a password in the database . . . 551

Chapter 19 Upgrade Guide 553
Before You Upgrade . . . 553
Upgrade Mirth Connect . . . 554
After the Upgrade . . . 554
Version-Specific Upgrade Instructions . . . 555
4.5.0 Upgrade Notes . . . 555
4.4.2 Upgrade Notes . . . 557
4.4.1 Upgrade Notes . . . 557
4.4.0 Upgrade Notes . . . 558
4.3.0 Upgrade Notes . . . 559
4.2.0 Upgrade Notes . . . 563
4.1.0 Upgrade Notes . . . 564
4.0.0 Upgrade Notes . . . 564
3.12.0 Upgrade Notes . . . 567
3.11.0 Upgrade Notes . . . 568
3.10.0 Upgrade Notes . . . 570
3.9.0 Upgrade Notes . . . 571
3.8.0 Upgrade Notes . . . 571
3.7.0 Upgrade Notes . . . 572
3.5.0 Upgrade Notes . . . 575
3.4.0 Upgrade Notes . . . 576
3.2.0 Upgrade Notes . . . 577
3.1.1 Upgrade Notes . . . 578
3.1.0 Upgrade Notes . . . 578
3.0.2 Upgrade Notes . . . 579

3.0.0 Upgrade Notes 579

Chapter 20 Commercial Support/Extensions 582
Advanced Alerting 583
Advanced Clustering 584
ASTM E1381 Transmission Mode 585
ASTM E1394 Data Type 586
Channel History 587
Cures Certification Support 588
Email Reader 588
Enhancement Bundle 589
FHIR Connector 590
Interoperability Connector Suite 592
LDAP Authorization 594
Message Generator 595
Multi-Factor Authentication 596
Serial Connector 597
SSL Manager 598
Role-Based Access Control 599
NextGen Results CDR Connector 600

Chapter 21 Training 602

Chapter 22 Cures Certification 603
Summary 604
§170.315(b)(10) Electronic Health Information (EHI) Export 605
Viewing Exported Attachments 606
§170.315(d)(1) Authentication, Access Control, Authorization 607
§170.315(d)(2) Auditable Events and Tamper-Resistance, §170.315(d)(3) Audit Report(s), and §170.315(d)(10) Auditing Actions on Health Information 607
§170.315(d)(5) Automatic Access Time-out 609
§170.315(d)(6) Emergency Access 609
§170.315(d)(7) End-User Device Encryption 616
§170.315(d)(8) Integrity 616
§170.315(d)(9) Trusted Connection 617
§170.315(d)(12) Encrypt Authentication Credentials 617
§170.315(d)(13) Multi-Factor Authentication 618
§170.315(g)(4) Quality Management System 619
§170.315(g)(5) Accessibility-Centered Design 619

CHAPTER 1

Legal Notice

NOTICE: This document contains information that is confidential and proprietary to NextGen Healthcare, Inc. and its subsidiaries and affiliates ("Company") and is intended for use solely by Company's authorized clients. This document may not be copied, reproduced, published, displayed, otherwise used, transmitted, or distributed in any form by any means as a whole or in any part, nor may any of the information it contains be used or stored in any information retrieval system or media, translated into another language, or otherwise made available or used by anyone other than the authorized client to whom this document was originally delivered without the prior, written consent of Company.

By retaining or using this document, you represent that you are a client or an authorized representative of a client of Company who is authorized to use this document under one or more agreements between you and Company now in force, and that you will use this document and the information it contains solely as and to the extent those agreements permit. Any other use or distribution of the contents of this document, as a whole or in any part, is prohibited. Although we exercised great care in creating this publication, Company assumes no responsibility for errors or omissions that may appear in this publication and reserves the right to change this publication at any time without notice.

You will not, directly or indirectly, use, or permit any third party to use, any information obtained from or through this document: (a) as a training set for machine learning or the training of an artificial intelligence ("AI"); (b) in connection with machine learning, development and refinement of algorithms, and/or the enhancement and/or improvement of the operation and functionality of your or any third party's products or services; or (c) in any generative artificial intelligence software, tools or technologies, including, but not limited to, any natural language processing, deep learning algorithms, or machine learning models (commonly known as "Generative AI").

© 2024 NXGN Management, LLC. All Rights Reserved.

NextGen is a registered trademark of NXGN Management, LLC. Microsoft and SQL Server are trademarks of the Microsoft group of companies. Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. macOS® is a registered trademark of Apple Inc. Apache®, Apache Tomcat® and Apache Derby™ are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. GlassFish® is a trademark of Eclipse Foundation, Inc. All other names and marks are the property of their respective owners.

CHAPTER 2

Introduction to Mirth Connect

Mirth® Connect is an open source standards-based healthcare integration engine that enables interoperability between two systems by accepting incoming information packets, called messages, and processing them according to rules you provide. It speeds message routing, filtering, and transformation between health-info systems over various messaging protocols (e.g., HL7, X12, EDI, DICOM, XML).

This user guide provides you with an overview of Mirth® Connect and instructions on how to install the software and use it to transfer or share healthcare data with security and ease.

The NextGen Connected Health Solutions Mission

NextGen® Connected Health Solutions help many of the nation's largest, most respected healthcare entities streamline their care-management processes to satisfy the demands of a regulated, competitive healthcare industry. With NextGen® Connected Health Solutions, NextGen Healthcare aims to provide the healthcare community with a secure, efficient, cost-effective means of sharing health information. The natural outcome is a family of applications—which includes 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 Mirth Connect

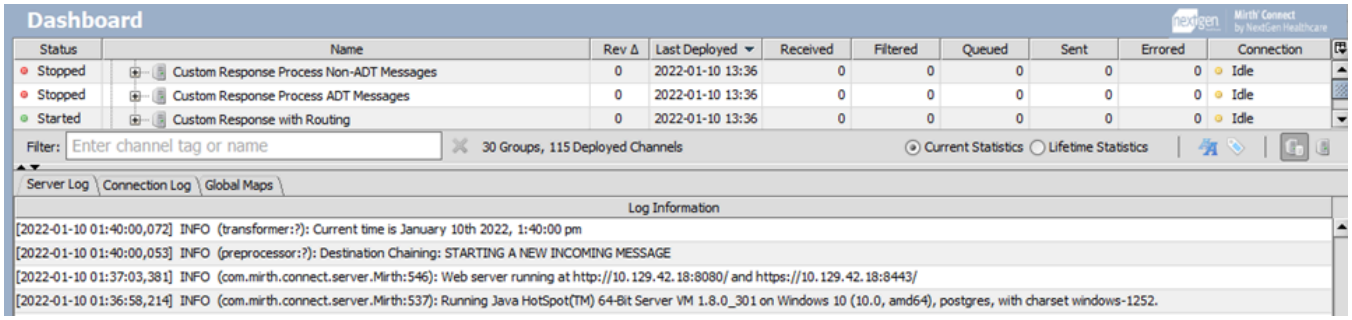
Like an interpreter who translates foreign languages into a language you understand, Mirth® Connect translates message standards into a language your system understands. Whenever a "foreign" system sends you a message, Mirth® Connect with its integration capabilities expedite the following:

- **Filtering**: Mirth® Connect reads message parameters and either passes the message on or stops it on its way to the transformation stage.
- **Transformation**: Mirth® Connect converts the incoming message standard to another standard,

such as from Health Level Seven (HL7) to XML.

- **Extraction**: Mirth® Connect can pull data from and push data to a database.
- **Routing**: Mirth® Connect ensures that messages arrive at their assigned destinations.

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



The Healthcare Interoperability Challenge and Solution

Patient data is most often exchanged through computer systems. For example, a doctor's office sends patient records to a hospital, or 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 transactions less efficient and reliable than they could be. Contributing factors include the following:

- 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 Mirth® Connect's flexibility. A lab's data system sends a Health Level Seven (HL7) message to Mirth® Connect through a Minimal Lower Layer Protocol (MLLP).

Mirth® Connect 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 Mirth® Connect reads patient data from a hospital's electronic medical record (EMR) system. With elements mapped in its own channel, Mirth® Connect generates and sends an HL7 message to a client for outpatient care. Multiple configurations are available depending on how the channel is constructed.

Mirth Connect as Open Source Software

Because Mirth® Connect 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. Mirth® Connect is licensed under Mozilla Public License 2.0. 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

CHAPTER 3

Getting Started with Mirth Connect

If you have a stand-alone instance of Mirth® Connect, an installer is included for Windows®, Linux®, and macOS® systems. This section outlines the system requirements, walks you through the installation instructions, and explains the Mirth® Connect Administrator.

Note: This section mainly pertains to the standalone version of Mirth® Connect. If you have purchased a NextGen® Application Management Console, you can disregard the download and installation procedures.

Mirth Connect System Requirements

The Mirth® Connect server is a fully standalone application that does not require an application server. You do not need to install a container service like Tomcat, Glassfish, etc.

The following requirements apply to any computer running the Mirth® Connect server, Command Line Interface, or Administrator.

Operating System Requirements

- Windows XP or later
- macOS (OS X) 10.7 Lion or later
- Linux

Processor (CPU) Requirements

- 32-bit or 64-bit x86 processor

Note: Apple M1 and M2 processors are currently not fully supported.

Java Requirements

Java requirements for Mirth® Connect include the following:

- Java 8 or later
- Supported JRE/JDK distributions:
    - The official Oracle® JRE/JDK
    - OpenJDK distributions from various organizations

Note: NextGen Healthcare recommends Oracle's OpenJDK distribution.

Note: When using Java 9 or greater, there are some additional Java virtual machine options that need to be set. See Using Java 9 or greater *(on page 55)* for more information.

Database Requirements

The Mirth® Connect Server requires a database for configuration and message storage. For quick deployment, development, and testing, Mirth® Connect already includes an embedded database (Apache Derby™). For production, the latest version of Mirth® Connect supports the following databases:

- PostgreSQL® 8.3+
- MySQL 5.6+
- Oracle® 12c+
- SQL Server® 2005+

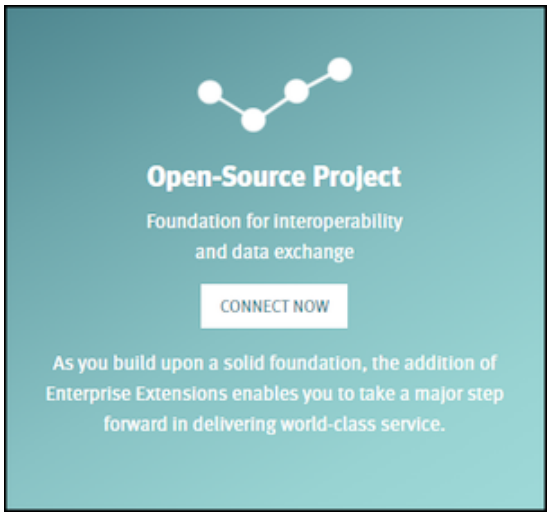
Note: The above database requirements apply only to what is used for the configuration and message storage for the Mirth® Connect server and have no impact on which databases Mirth® Connect can interface with.

Install Mirth Connect Using the Mirth Connect Installer

You can use the Mirth® Connect Installer (for Windows, Linux, and Mac OS X / macOS) to install Mirth® Connect or to upgrade a previous version of the software installed on your system.

You can download the Mirth® Connect Installer from the NextGen Healthcare website.

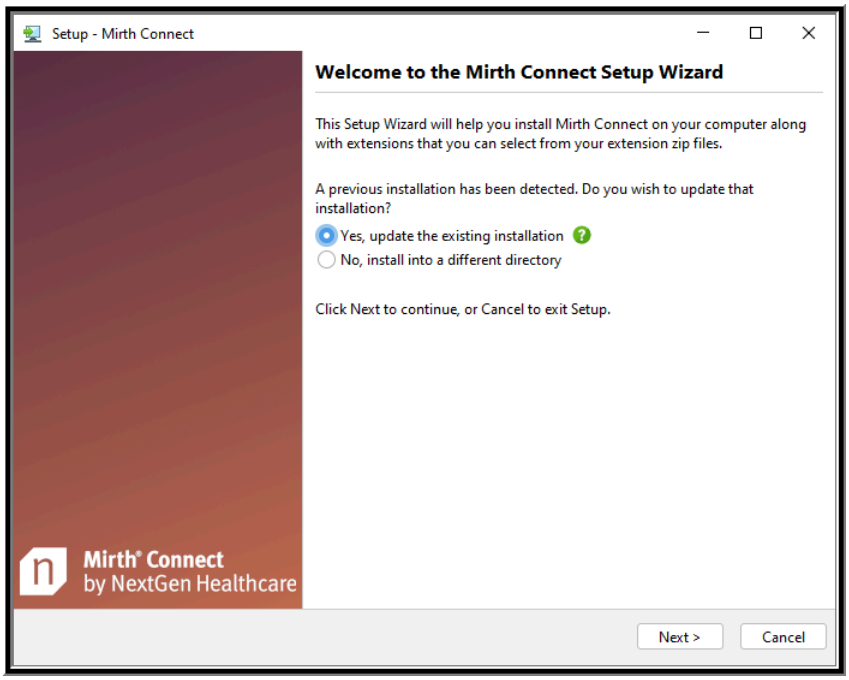
1. Select the following link to access the Mirth® Connect page on the NextGen Healthcare website: https://www.nextgen.com/solutions/interoperability/mirth-integration-engine
2. Scroll to the **Open-Source Project** section and select **CONNECT NOW**. A questionnaire appears.



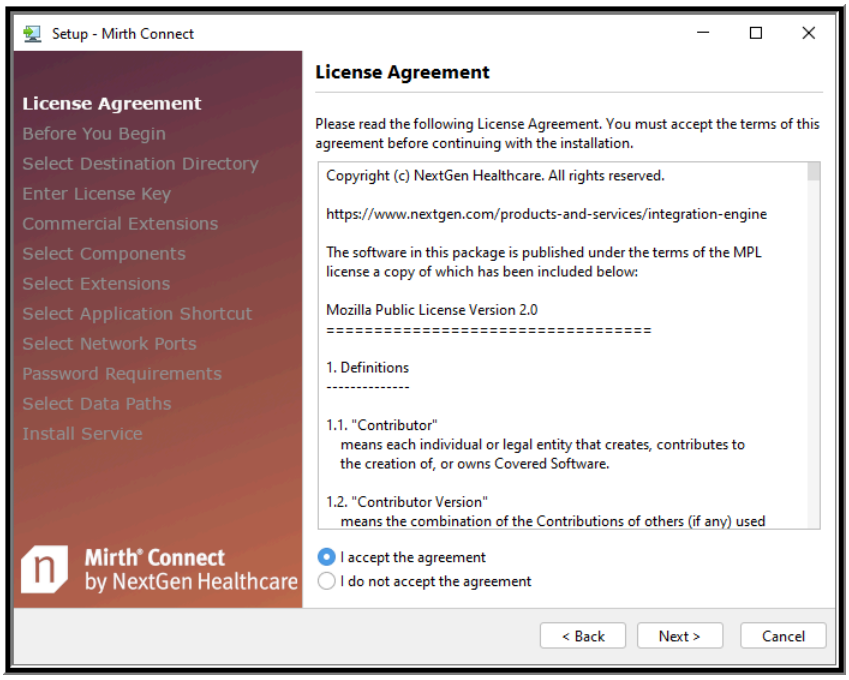
3. Complete the questionnaire, then select **Submit**.
   A download link will be sent to the provided email address.
4. After you receive the emailed link, open the link to access the Downloads page.
5. Review the list of options and select the link to the installer that best matches the system where Mirth® Connect will reside.
   The Mirth® Connect installer downloads.
6. Double-click the downloaded file, then open the Mirth® Connect Installer.

7. On the setup wizard, select **Next >**.

Note: For upgrades, the default installation option is **Yes, update the existing installation**. If you would like to install Mirth® Connect to another location, select **No, install into a different directory"**, then select the new location. Hover your cursor over the green question mark to view the current installation directory.



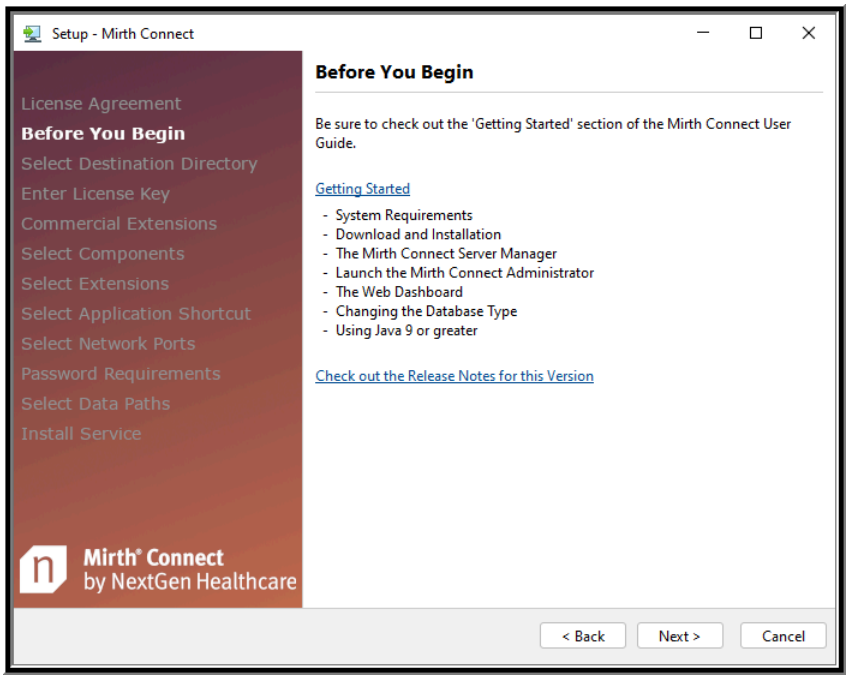
8. On the License Agreement window, read the license agreement, select **I accept the agreement**, then select **Next >**.



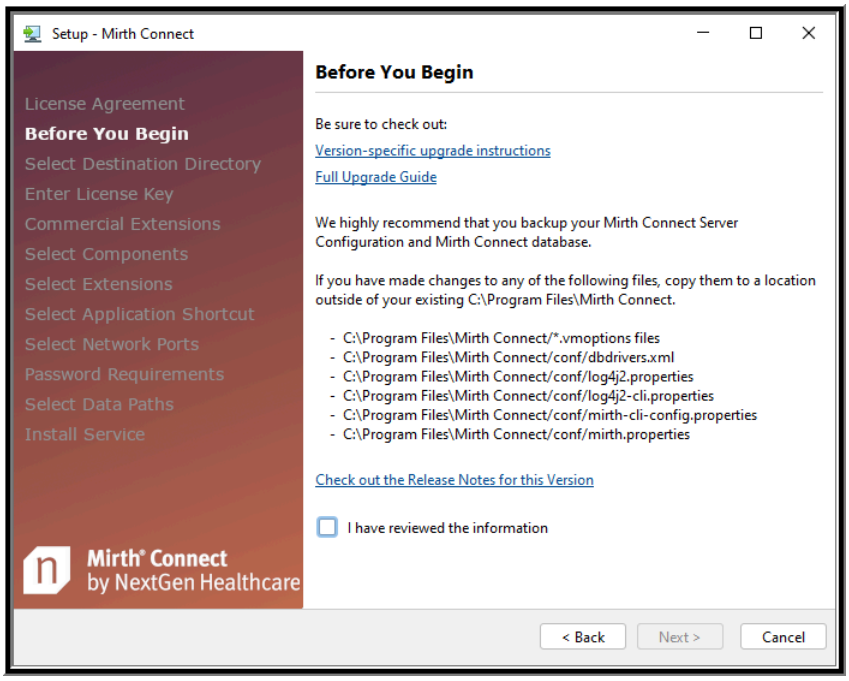
9. On the Before You Begin window, do one of the following:

Choose from:

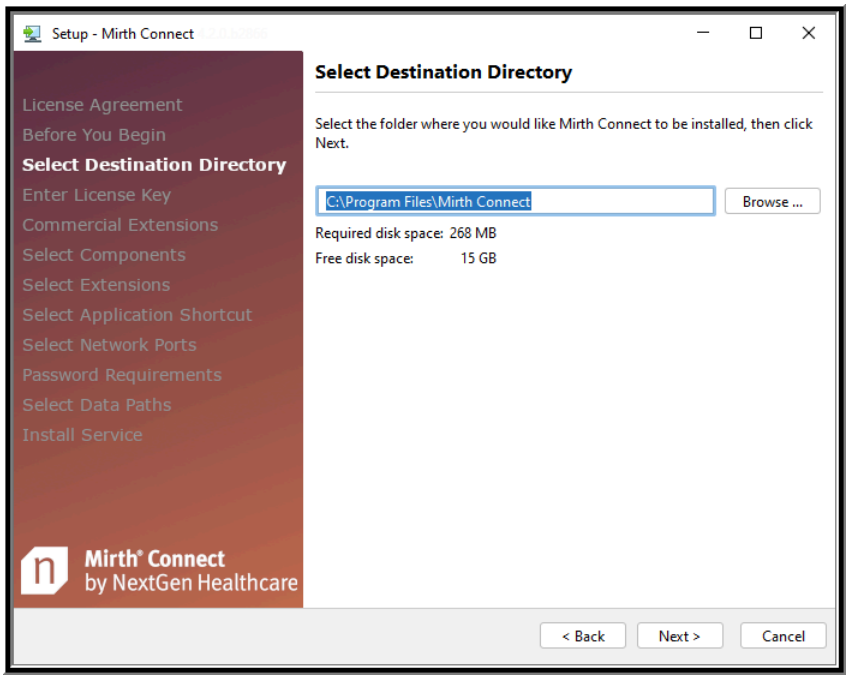
- For a new installation, select **Next**.



- For an upgrade, review the links for upgrading, getting started, and release notes as needed. After you have enough information for an upgrade, select **I have reviewed the information**, and then select **Next**.

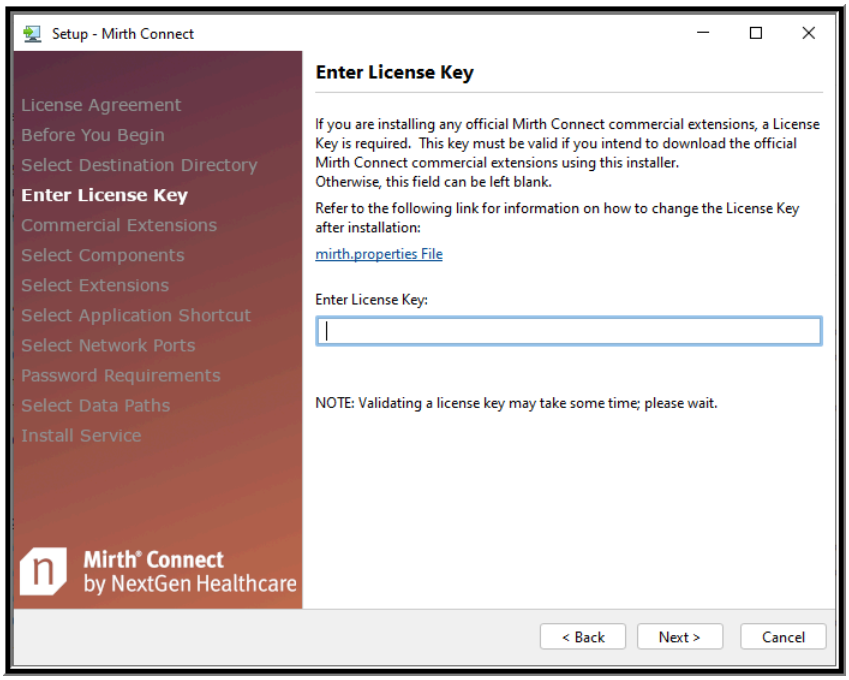


10. On the Select Destination Directory window, select **Browse...**, find and select the folder where you would like to install Mirth® Connect, then select **Next >**.



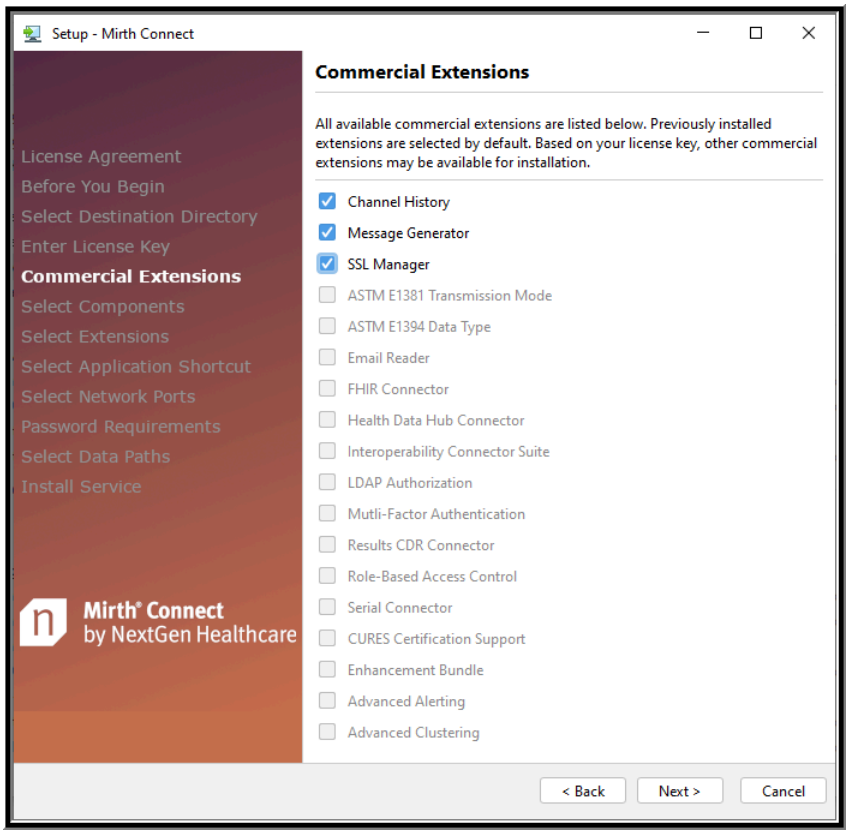
Note: If you are upgrading a previously installed version of Mirth® Connect, this window is skipped and the application is installed in the existing location.

11. On the Enter License Key window, enter the license key, if available, then select **Next >**.

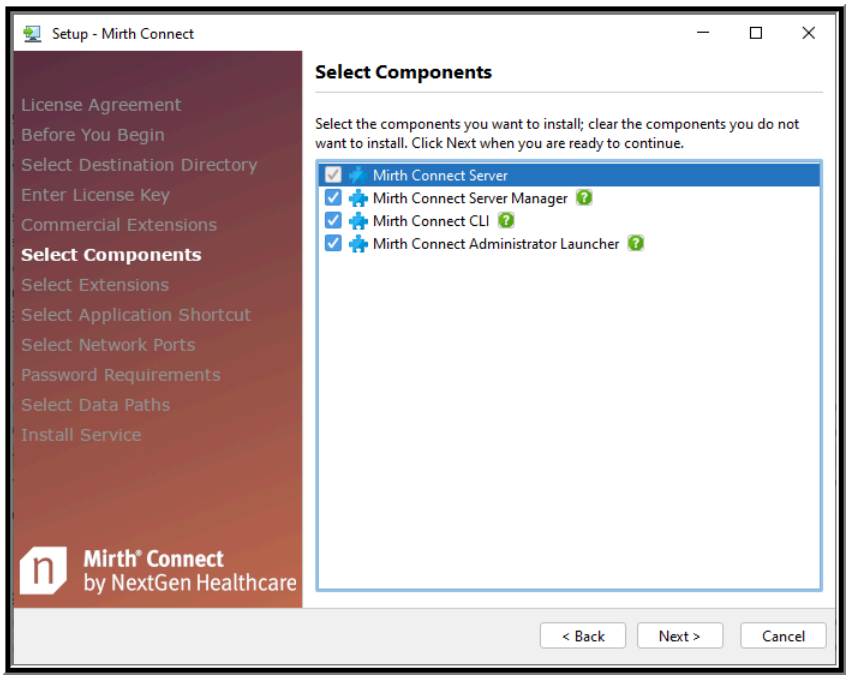


- For upgrades, the existing value from mirth.properties appears as the default.
- If no license key is provided, select **Next>**, the wizard will skip the Commercial Extensions window and continue to the next step.
- If valid license key is provided, select **Next>**. The wizard continues to the Commercial Extensions window to download the available extensions based on the license key.
- If an invalid license key is provided, when you select **Next>**, the window provides you with the reason the key is invalid. You can replace the invalid key or select **Continue with invalid License Key** and select **Next>**.

12. The Commercial Extensions window lists all available commercial extensions in order alphabetically by package (silver, gold, platinum). For new installations, all allowed extensions are pre-selected, by default. For upgrades, previously installed extensions are selected, by default. Based on the license key, other commercial extensions may be available for installation. Select **Next>**.

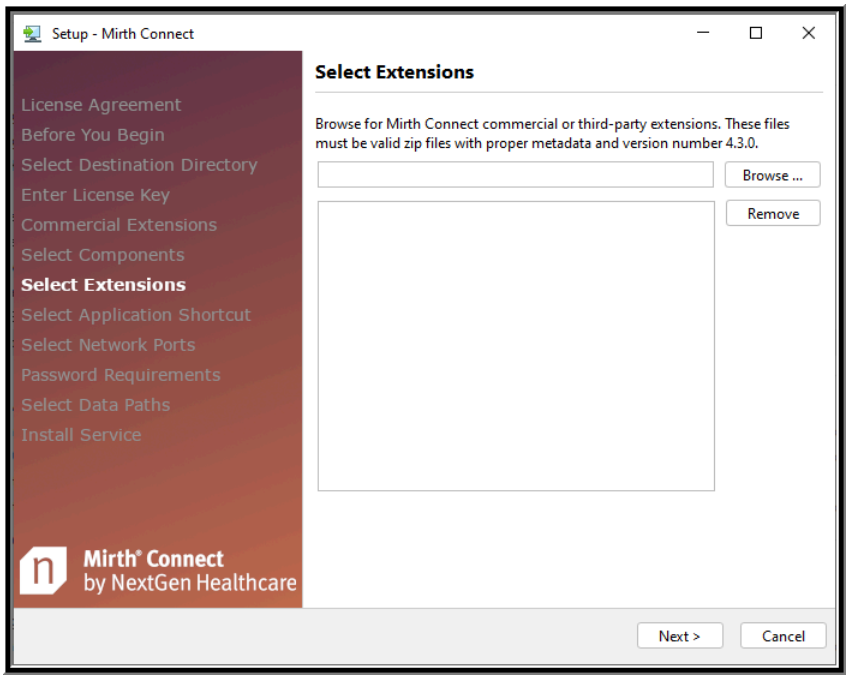


13. On the Select Components window, select the Mirth® Connect components you would like to install, then select **Next >**.

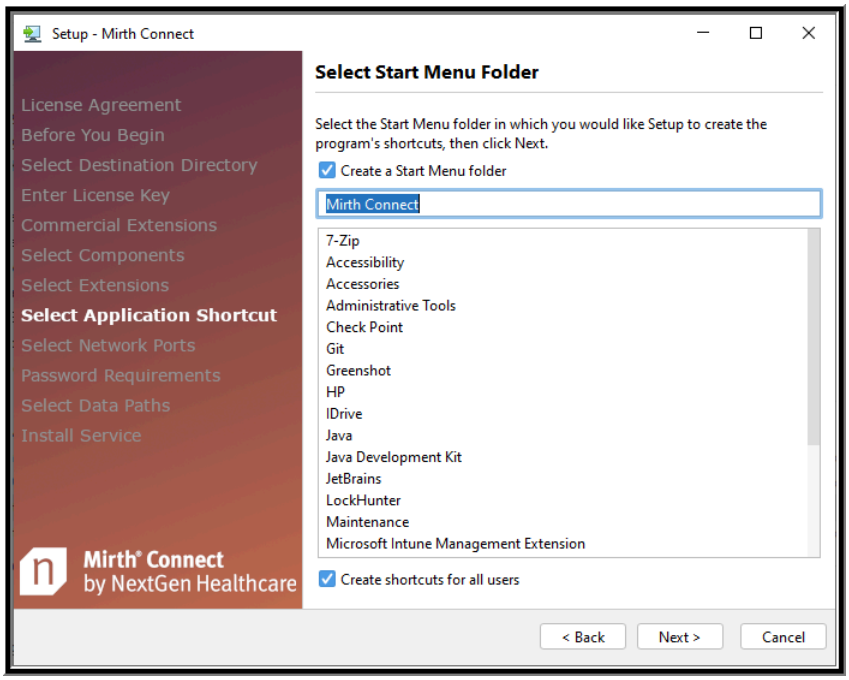


Note: The Mirth® Connect Server component is unavailable because it is not an option. You can select Mirth® Connect Server Manager, Mirth® Connect CLI, or Mirth® Connect Administrator Launcher.

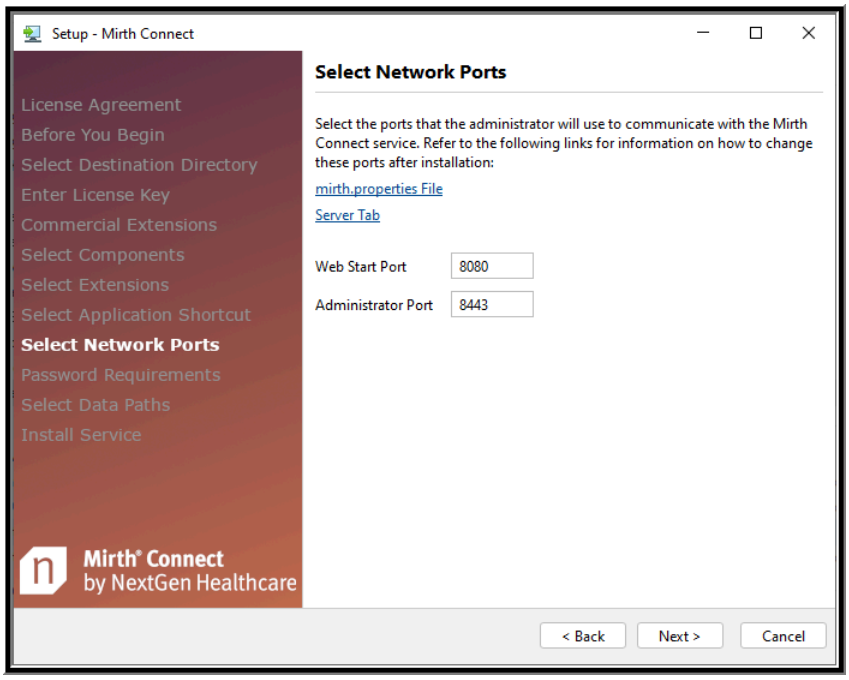
14. On the Select Extensions window, select **Browse...**, find and select the extensions you would like to install, then select **Next >**.



15. On the Select Start Menu Folder window, select **Next >**.



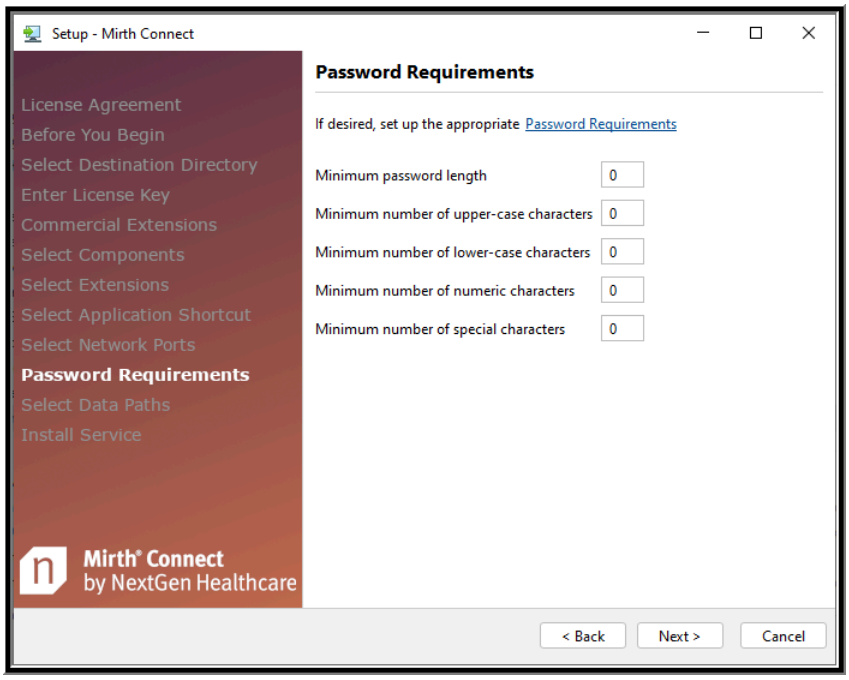
16. On the Select Network Ports window, enter the Web Start and Administrator port values as needed, then select **Next >**.



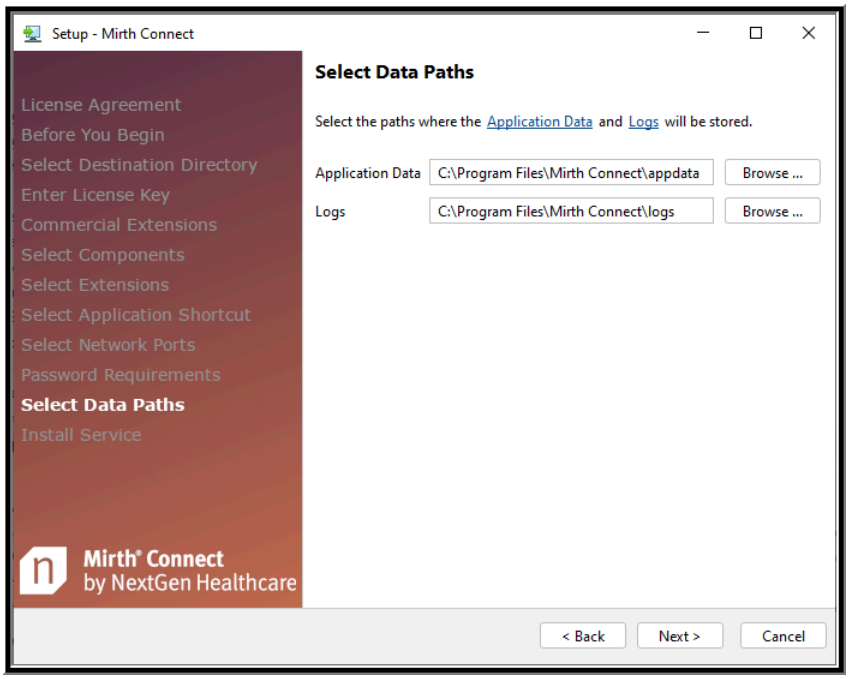
Note: For upgrades, the existing values from mirth.properties appear as the default.

17. On the Password Requirements window, set your password parameters, then select **Next >**.

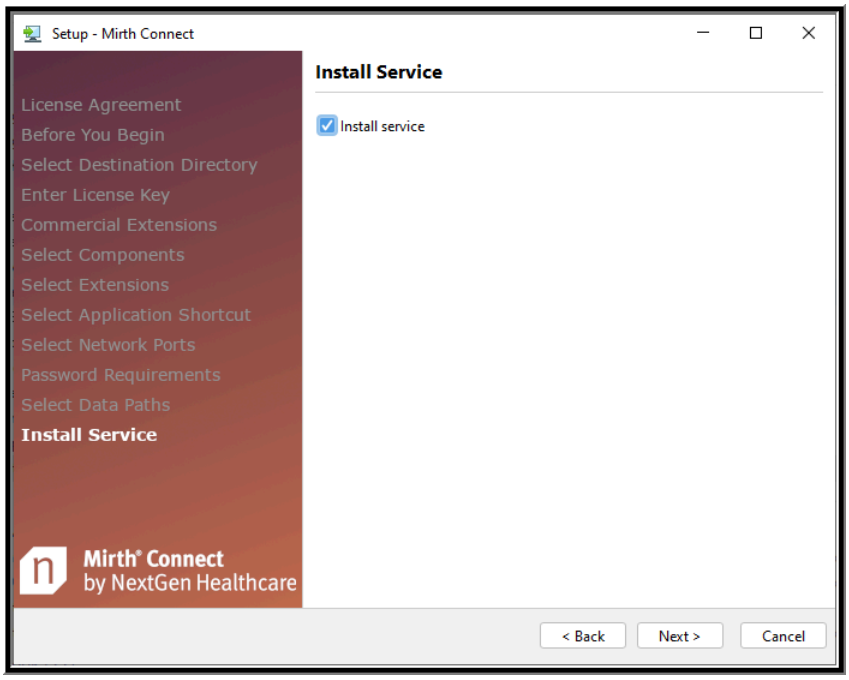
Note: For upgrades, the existing values from mirth.properties appear as the default.



18. On the Select Data Paths window, set the server-settings paths for Application Data and Logs, then select **Next >**.



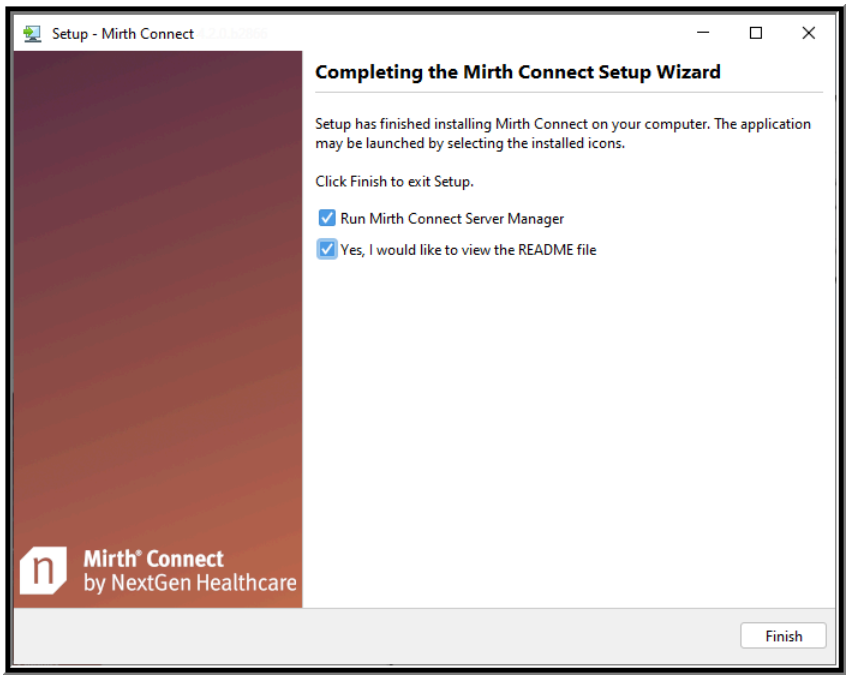
19. On the Install Service window, select **Next >**.



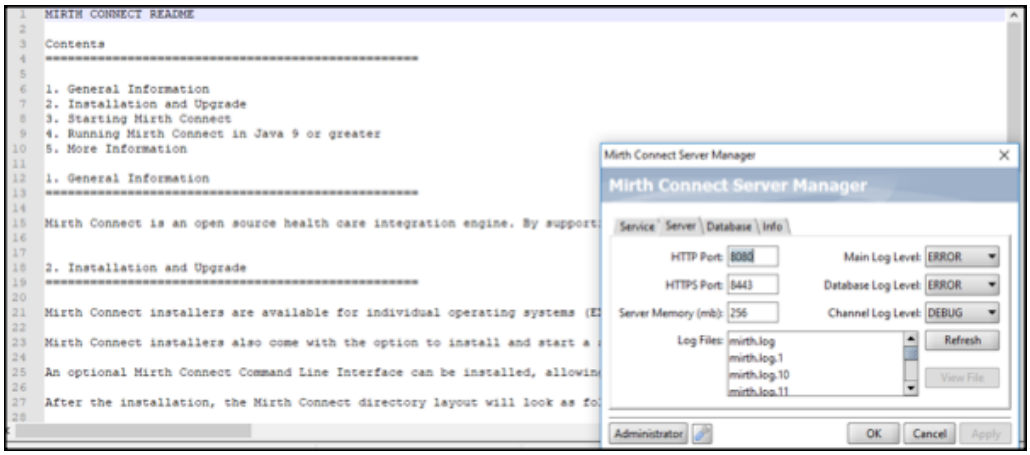
Mirth® Connect begins to install.

After the Mirth® Connect installation completes, the Completing the Mirth Connect Setup Wizard window appears.

20. Select the options that appear on the Completing the Mirth Connect Setup Wizard window as necessary, then select **Finish**.



Depending on the options you selected in the setup wizard, the Mirth® Connect Server Manager, the README file, or both appear.



Install Mirth Connect Using the Command-Line Installer

You can use the command-line installer to install Mirth® Connect on headless Linux environments.

Attention: Some features that are available in the graphical installer are not available from the

command-line installer. The missing features include the following:

- Entering and validating a license key
- Downloading commercial extensions
- Installing extensions
- Running the Mirth® Connect Server Manager

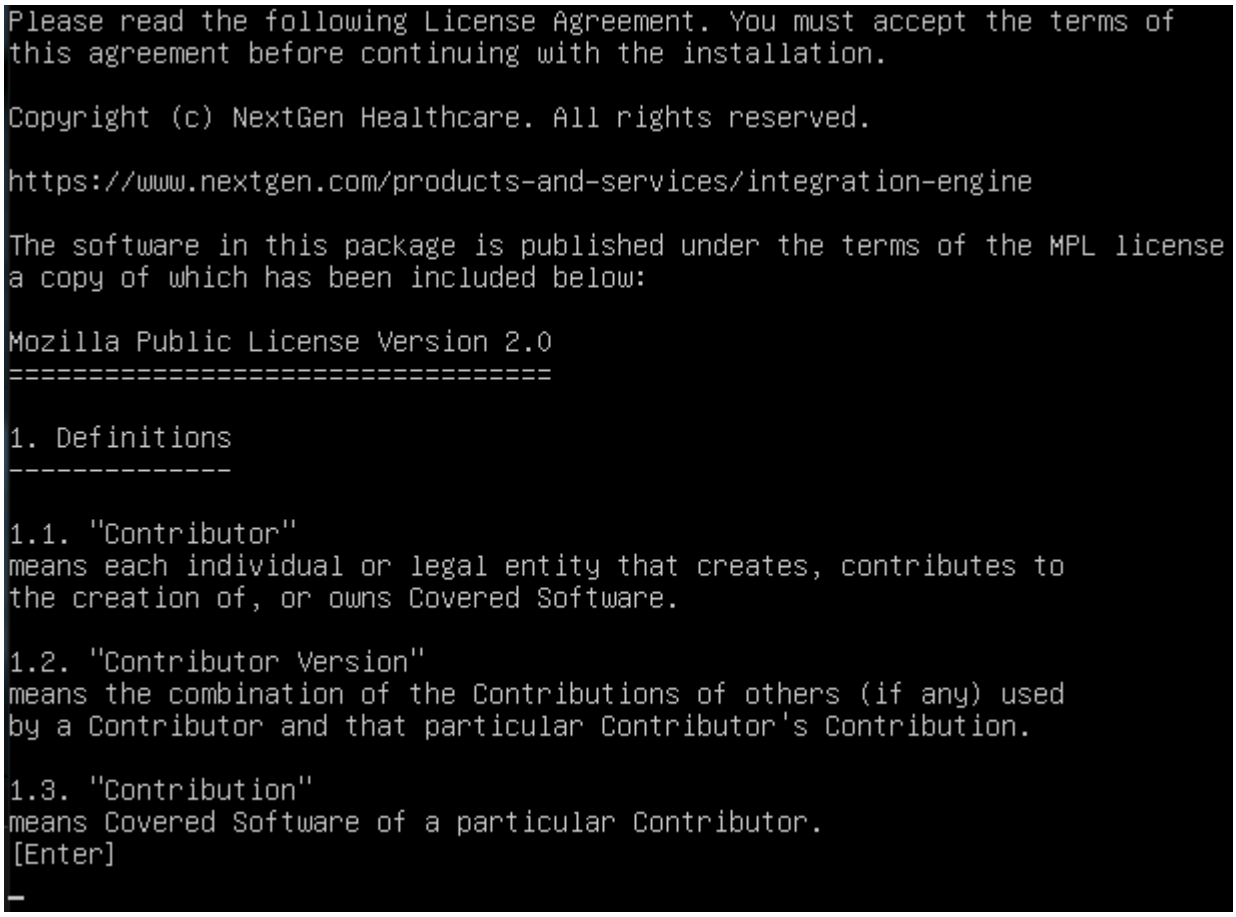
Note: You can use the command-line installer on macOS and non-headless Linux environments by running the `.sh` file (or on Windows by running the `.exe` file) in console mode using the `-c` option. However, NextGen recommends using the graphical installer on those environments.

1. Download the Linux installer file from the NextGen Healthcare website.
2. Make the installer file executable:
   `chmod +x mirthconnect-x.x.x.bxxxx-unix.sh`
3. Run the installer file as a superuser:
   `sudo ./mirthconnect-x.x.x.bxxxx-unix.sh`

   Alternately, to install Mirth® Connect in non-interactive mode using all default options, run the following command:

   `sudo ./mirthconnect-x.x.x.bxxx-unix.sh -q`
4. Enter your Linux user password, if prompted.
   The installer starts and prompts you to continue or cancel.

5. Type `o` or select `Enter` to continue.
   The license agreement appears.



6. Select `Enter` to advance through the text until you've reached the end of the agreement. You are prompted to accept the license agreement.

```
This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this
file, You can obtain one at https://mozilla.org/MPL/2.0/.

[Enter]

If it is not possible or desirable to put the notice in a particular
file, then You may include the notice in a location (such as a LICENSE
file in a relevant directory) where a recipient would be likely to look
for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice
---------------------------------------------------------

This Source Code Form is "Incompatible With Secondary Licenses", as
defined by the Mozilla Public License, v. 2.0.

I accept the agreement
Yes [1], No [2]
```

7. Type `1` to accept.
   The installer displays a link to the “Getting Started” section of the user guide.

```
Before You Begin
Be sure to check out the 'Getting Started' section of the Mirth Connect User
Guide.
[https://docs.nextgen.com/bundle/Mirth_User_Guide_4_3_0/page/connect/connect/topics/c_Getting_Starte
d_mirth_connect_ug.html]
```

8. Press `Enter` to continue.
   The installer displays a link to the latest Mirth® Connect release notes.

```
- System Requirements
- Download and Installation
- The Mirth Connect Server Manager
- Launch the Mirth Connect Administrator
- The Web Dashboard
- Changing the Database Type
- Using Java 9 or greater

Check out the Release Notes for this Version
[https://github.com/nextgenhealthcare/connect/wiki/4.3.0---What's-New]
```

9. Press `Enter` to continue.
   You are prompted to enter the folder where to install Mirth® Connect.

```
Select the folder where you would like Mirth Connect to be installed, then
click Next.
Where should Mirth Connect be installed?
[/usr/local/mirthconnect]
_
```

10. Enter the absolute path of a folder or press `Enter` to accept the default value.
The installer displays information about license keys.

```
Enter License Key
If you are installing any official Mirth Connect commercial extensions, a
License Key is required.  This key must be valid if you intend to download
the official Mirth Connect commercial extensions using this installer.
Otherwise, this field can be left blank.
Refer to the following link for information on how to change the License Key
after installation:
mirth.properties File
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/c_The_mirth_propert
ies_File_connect_ug.html]
_
```

11. Press `Enter` to continue.
You are prompted to enter a license key.

```
Enter License Key:
[]
```

Note: This feature is not available from the command-line installer.

12. Because this feature isn't available from the command-line installer, press `Enter` to continue.

You are asked which components to install. The Mirth® Connect Server is always installed, and you have the option to install the Server Manager, the CLI, and the Administrator Launcher.

```
Which components should be installed?
X: Mirth Connect Server
2: Mirth Connect Server Manager [*2]
3: Mirth Connect CLI [*3]
4: Mirth Connect Administrator Launcher [*4]
(To show the description of a component, please enter one of *2, *3, *4)
Please enter a comma-separated list of the selected values or [Enter] for the default selection:
[2,3,4]
```

13. Follow the on-screen instructions to enter a comma-separated list of values indicating which components to install or press `Enter` to install all components.
You are prompted to select extensions to install.

```
Select Extensions
Browse for Mirth Connect commercial or third-party extensions. These files
must be valid zip files with proper metadata and version number 4.3.0.
[]
```

Note: This feature isn't available from the command-line installer

14. Because this feature isn't available from the command-line installer, press `Enter` twice to continue.
    You are asked whether to create symlinks.

```
Create symlinks?
Yes [y, Enter], No [n]

Select the folder where you would like Mirth Connect to create symlinks, then click Next.
[/usr/local/bin]
```

15. Type `y` or press `Enter` to create them. Otherwise, type `n` to skip creating symlinks.

    Note: If you choose to create symlinks, you will be prompted to select the folder to create them in. Enter the absolute path to the folder or press `Enter` to accept the default value.

    The installer displays information about network ports which includes a link to the appropriate page in the user guide.

```
Select Network Ports
Select the ports that the administrator will use to communicate with the
Mirth Connect service. Refer to the following links for information on how
to change these ports after installation:
mirth.properites File
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/c_The_mirth_propert
ies_File_connect_ug.html]
```

16. Press `Enter` to continue.
    The installer displays a link to the "Server Tab" section of the user guide.

```
Server Tab
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/c_Service_tab_serve
r_manager_mc.html ]
```

17. Press `Enter` to continue.
    You are prompted to enter the Web Start Port that the Mirth® Connect Server should use.

```
Web Start Port
[8080]
```

18. Enter a numeric value or press `Enter` to accept the default value.
    You are prompted for the Administrator Port.

```
Administrator Port
[8443]
```

19. Enter a numeric value or press `Enter` to accept the default value.
    The installer displays a link the “Password Requirements” section of the user guide.

```
Password Requirements
If desired, set up the appropriate password requirements.
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/password_requiremen
ts.html]
```

20. Press `Enter` to continue.
    You are prompted to configure the password requirements.

```
Minimum password length
[0]

Minimum number of upper-case characters
[0]

Minimum number of lower-case characters
[0]

Minimum number of numeric characters
[0]

Minimum number of special characters
[0]
```

21. For each, enter a numeric value or press `Enter` to accept the default.
    The installer displays a link to the “Data Paths” section of the user guide.

```
Select Data Paths
Select the paths where the Application Data and Logs will be stored.
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/c_Application_Data_
Directory_connect_ug.html ]
```

22. Press `Enter` to continue.
    The install displays a link to the “Logs” section of the user guide.

```
[https://docs.nextgen.com/bundle/Mirth_User_Guide_42/page/connect/connect/topics/c_LogsLogs_connect_
ug.html]
```

23. Press `Enter` to continue.
    You are prompted to enter the location to store application data.

```
Application Data
[]
```

24. Enter the absolute path to a folder to continue.
(For example: /opt/mirthconnect/appdata)
You are prompted to enter the location to store logs.

```
Logs
[]
```

25. Enter the absolute path to a folder to continue.
(For example: /opt/mirthconnect/logs)
The installer gives you the option to install the Mirth® Connect Service.

```
Install Service
Install service?
Yes [y, Enter], No [n]
```

26. Type `y` or press `Enter` to install the Mirth® Connect Service, or type `n` to skip installation.
The installer downloads and extracts all necessary files.

```
Extracting files ...

Downloading mirth-administrator-launcher-latest-unix.sh ...
  406.3 MB of 406.3 MB (100.0%) at 24,322 kb/s                    Setup has finished instal
ling Mirth Connect on your computer.
```

If you installed the Mirth® Connect Server Manager, you are prompted to run it.

```
Run Mirth Connect Server Manager?
Yes [y, Enter], No [n]
```

27. Because you are using a command line interface and the Server Manager is a GUI interface, you can skip this step by typing `n`.
You are given the option to view the README file.

```
Yes, I would like to view the README file?
Yes [y, Enter], No [n]
```

28. Type `y` or press `Enter` to view it. Type `n` to skip viewing it.
The installation process is now complete.

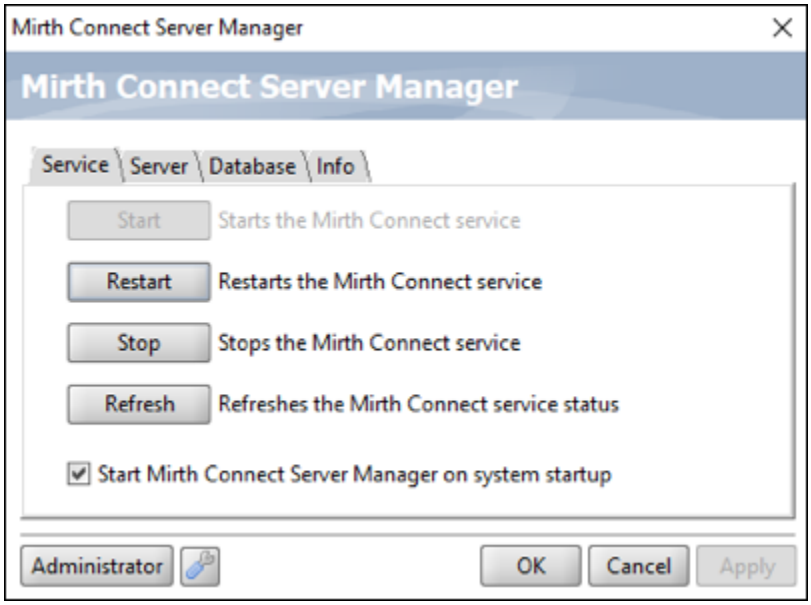
If you chose to install the Mirth® Connect Service, it begins running.

The Mirth Connect Server Manager

The Mirth® Connect Server Manager enables you to configure and access your Mirth® Connect Server. For Windows and Mac OS X version installations, the Mirth® Connect Server Manager application resides in the system tray (Windows) or in the **Applications** > **Mirth Connect** folder (Mac).

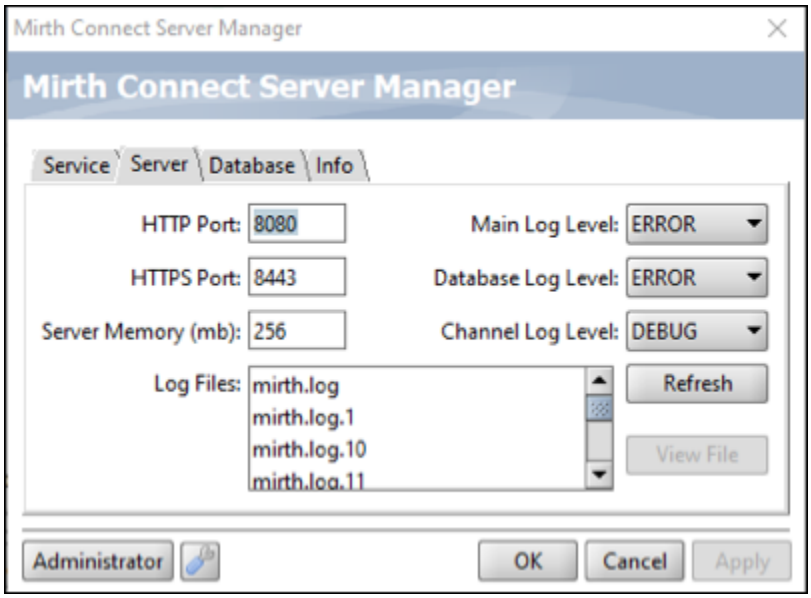
Service Tab

The Service tab provides an easy way to start, restart, stop, and refresh the service, rather than using the task scheduler or command line. It also provides the option to automatically start the Mirth® Connect Server when the system starts.



Server Tab

The Server tab enables you to enter details about your ports, server memory, log levels, and view log files.



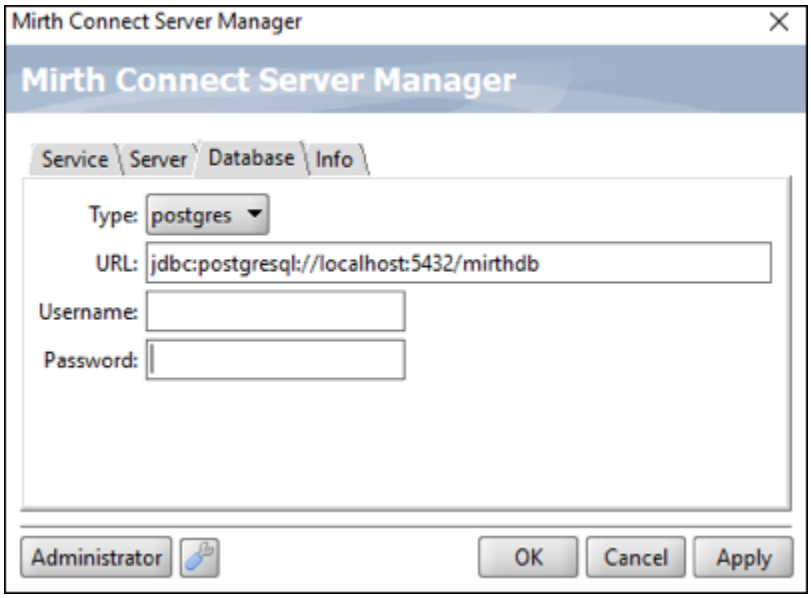
The following fields are available on the Server page.

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 Mirth® Connect 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 Database Log Level Channel Log Level	These fields enables you to select the applicable log level from the drop-down menus. Available options include: • **ERROR** • **WARN** • **INFO** • **DEBUG** • **TRACE** 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 unavailable if a file is not

Function	Description
	selected.
Administrator	(PC only) Opens the **Mirth Connect** login page (inactive on Macs; see Mirth Connect Administrator Overview *(on page 55)*)
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 unavailable unless changes have been made to the page settings.

Database Tab

The Database tab enables you to manage Mirth® Connect's internal database.



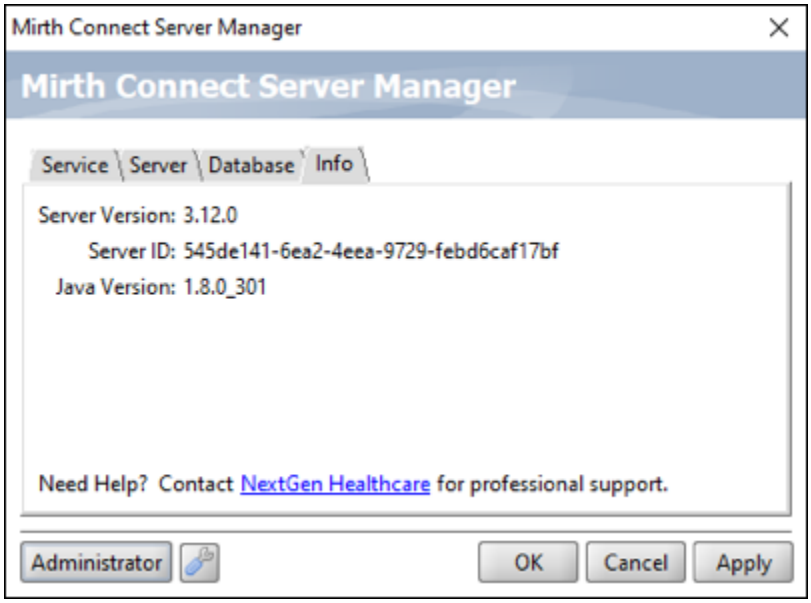
The following fields are available on the Database tab.

Function	Description
Type	Select the database on which Mirth® Connect 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

Warning: Mirth® Connect's default database, Apache Derby, is included only to help you set up quickly. Because it is not a production-level database, NextGen Healthcare strongly recommends that you not use Derby for production.

Info Tab

The Info tab shows the Mirth® Connect server and Java versions as well as the server ID and a link to the NextGen Healthcare website.



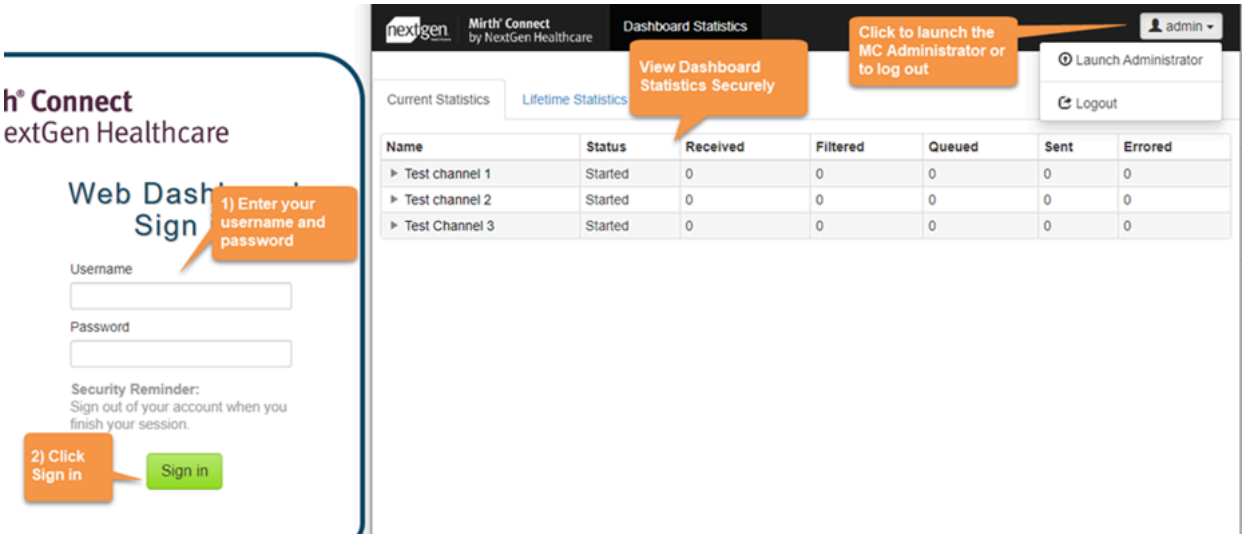
The Web Dashboard

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.

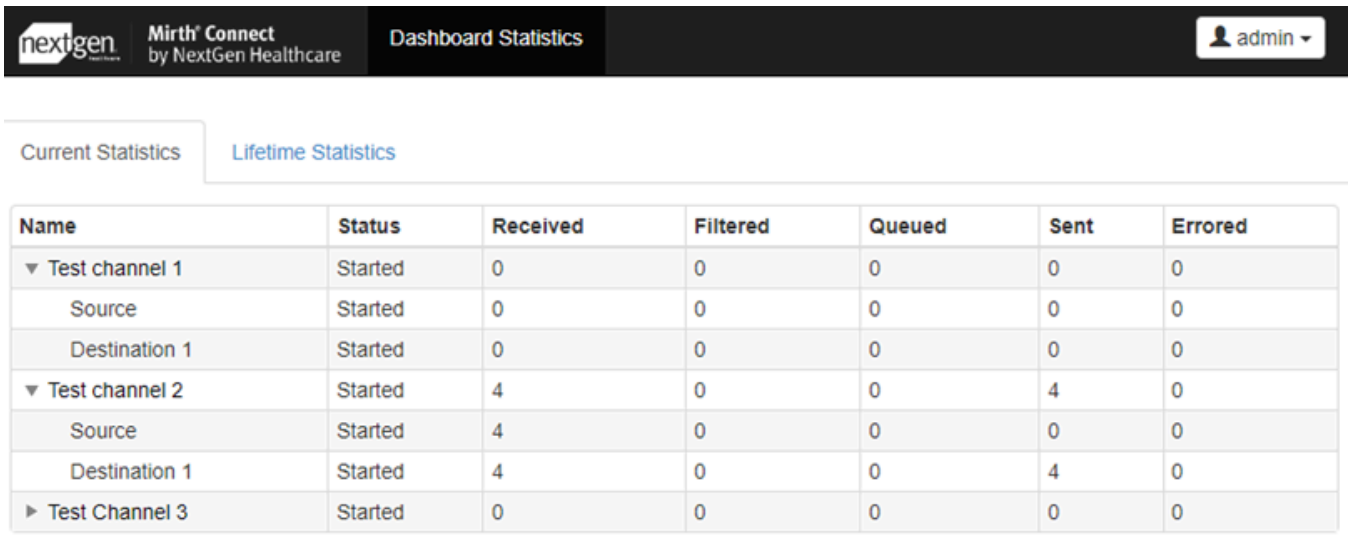
To login to the Web Dashboard:

- In the address field of your browser, type **localhost:** followed by the Web Start Port number. (See the **Server Manager - Server** page; default: **8080**.)

- Select the **Access Secure Site** button for **Web Dashboard Sign in**, the page changes, prompting you to enter a **Username** and **Password**.



After you log on, you can see all of 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.



To launch the Administrator interface from the **Web Dashboard**, select the arrow next to the user avatar in the top-right, and select **Launch Administrator**. You can also logout of the Web Dashboard from this drop-down menu.

Changing the Database Type

By default, Mirth® Connect is configured to connect to an embedded Apache Derby database for quick deployment, development, and testing.

Warning: When using Mirth® Connect in production environments, we recommend changing the underlying database to one of the supported servers:

- PostgreSQL 8.3+
- MySQL 5.0+
- Oracle 10gR2+
- SQL Server 2005+

See Also:

- Connect to Database using SSL/TLS *(on page 522)*

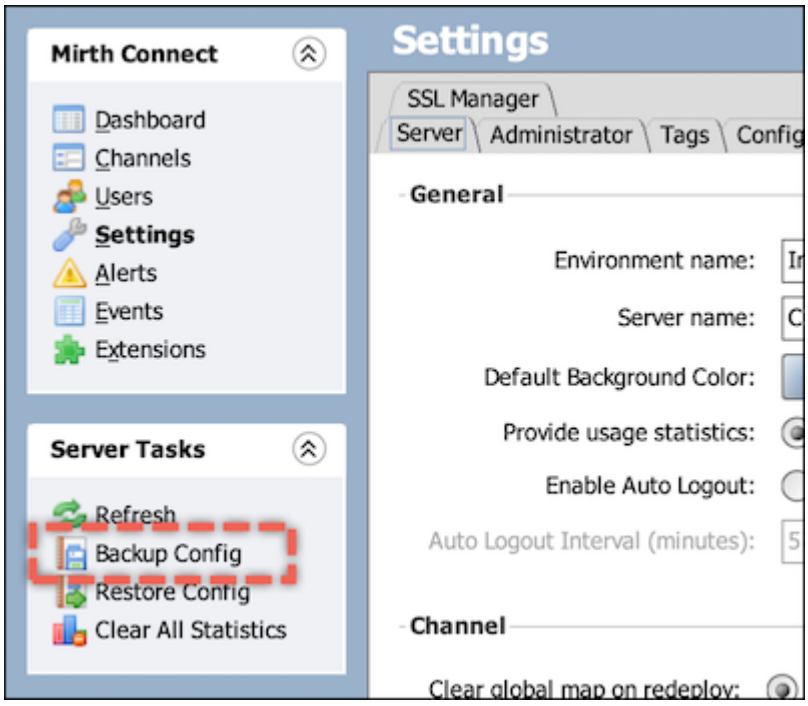
Backup Current Server Configuration

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 Mirth® Connect server and restored into a different server.

Note: If you have just installed Mirth® Connect and you do not yet have any channels or other configurations, you can skip this section.

Use the following steps to backup your server configuration

1. Log into the Mirth Connect Administrator (see Mirth Connect Administrator Overview *(on page 55)*) .
2. In the Mirth Connect pane, select **Settings.**
3. Select the **Server** tab.
4. In the **Server Tasks** pane, select **Backup Config**.



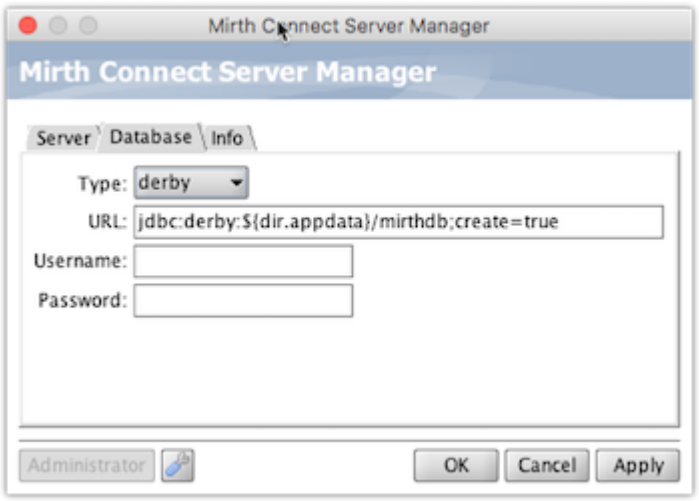
5. Enter a location to save the server configuration XML file, then select **Save**.
6. Review the Information message, then select **OK**.

Warning: The Server Configuration file does not include Users, Events, or Message / Attachment data. To export events, see Searching Events *(on page 145)*. To export message / attachment data, see Message Browser Tasks *(on page 130)*.

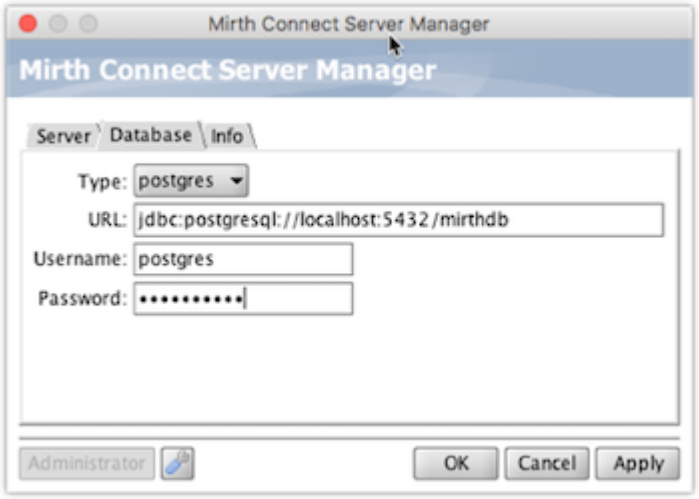
Change Database Settings

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.

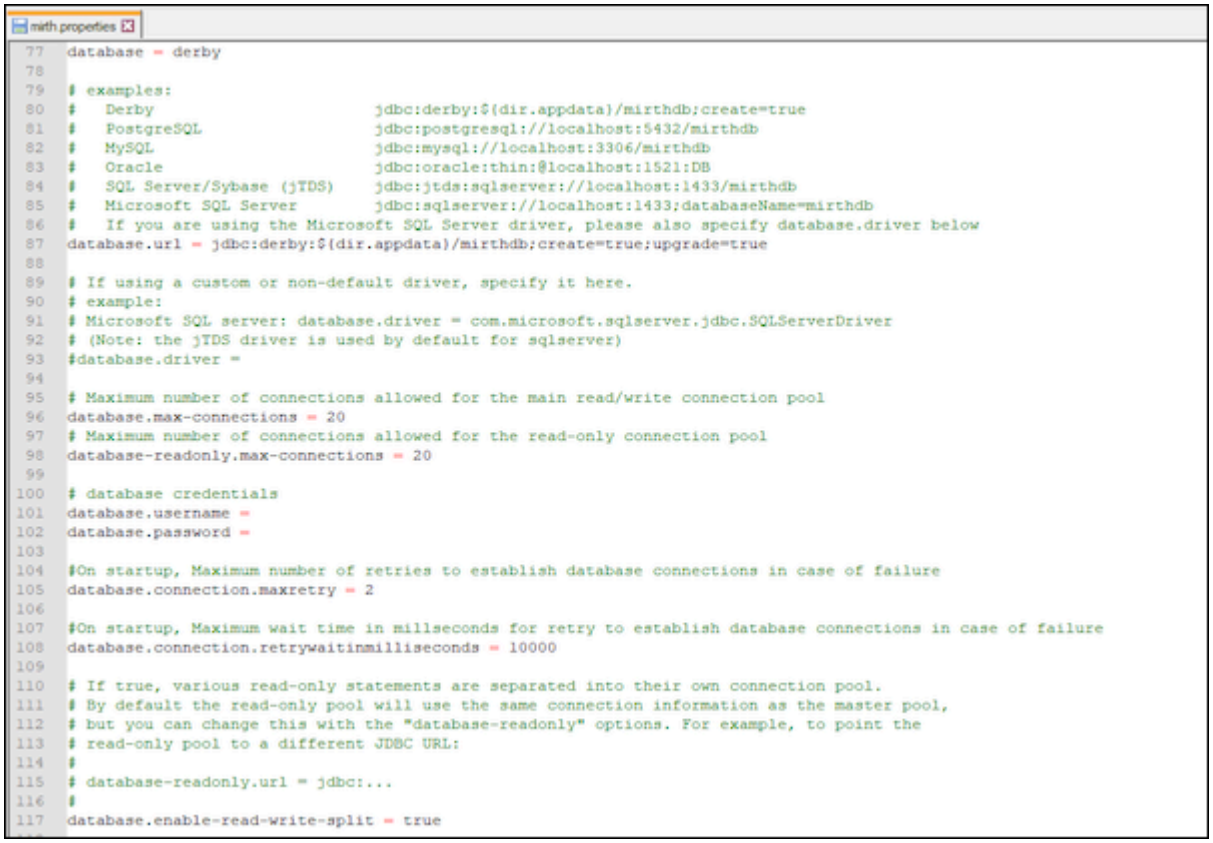


6. Select **Apply**.
7. Select **OK** to exit the Server Manager.

Note: If using sqlserver, the SQL Server/Sybase (jTDS) driver is used by default. If you prefer to use the Microsoft SQL Server driver, you must edit the properties file directly (see Editing the Properties File Directly for Windows, Linux, and Mac *(on page 52)*).

Editing the Properties File Directly for Windows, Linux, and Mac

1. Navigate to the directory where you installed Mirth® Connect.
2. Open the **mirth.properties** file inside the **conf** folder.



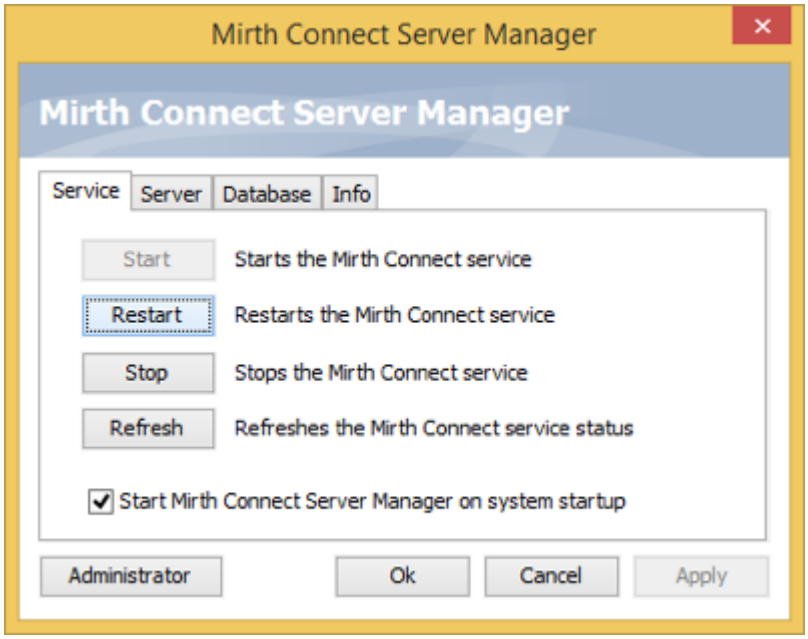
3. Edit the following properties as needed:
    - **database**: The type of database server to connect to (derby, mysql, postgres, oracle, sqlserver)
    - **database.url:** The JDBC URL connection string
        - If you need to connect using SSL/TLS, typically this is where you would add additional options to the URL. See Connect to Database using SSL/TLS *(on page 522)* for additional information.
    - **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

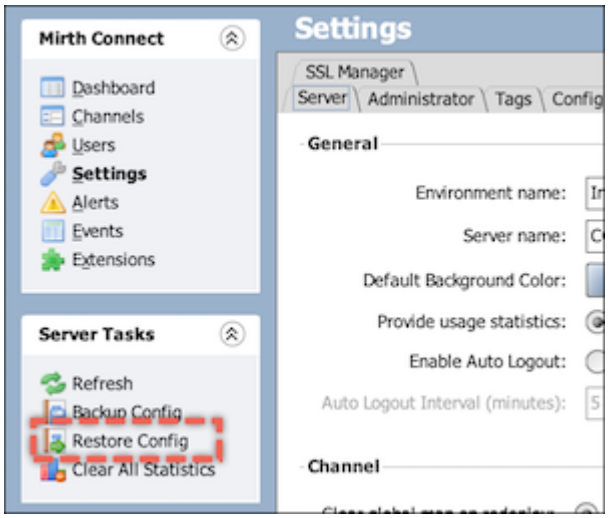
1. If you are using a Windows or Linux operating system and you have Mirth® Connect installed as a service/daemon, open the Server Manager *(on page 44)*.



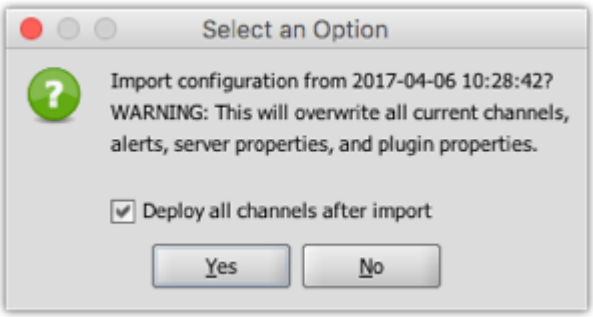
2. Select **Restart** and verify the server has started up successfully.

Restore Server Configuration

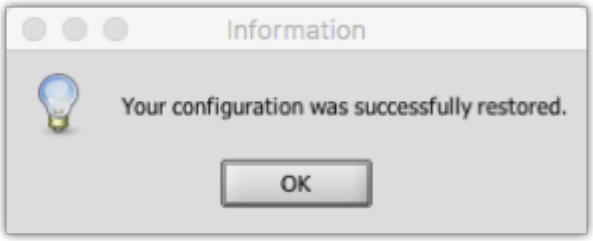
1. Launch the **Mirth Connect Administrator** (see Mirth Connect Administrator Overview *(on page 55)*) .
2. In the **Mirth Connect** pane, select **Settings**.
3. Select the **Server** tab if not already selected.
4. In the **Server Tasks** pane, select **Restore Config**.



5. Select the server configuration XML file you exported earlier, then select **Open**.
6. You will 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. Select **Yes** to begin the restore process.
8. Once the restore process is complete, a dialog appears indicating whether the restore was successfully performed.



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

Using Java 9 or greater

As documented in the Mirth Connect System Requirements *(on page 23)*, Mirth® Connect version 3.7 supports Java 8 at minimum. In order to use Java 9 or greater, you will first need to perform an extra manual step.

In the Installation Directory *(on page 478)*, there should be a **docs** folder. Inside, there is a file called **mcservice-java9+.vmoptions**. This contains some extra JVM options that allow Mirth® Connect 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 Mirth® Connect.

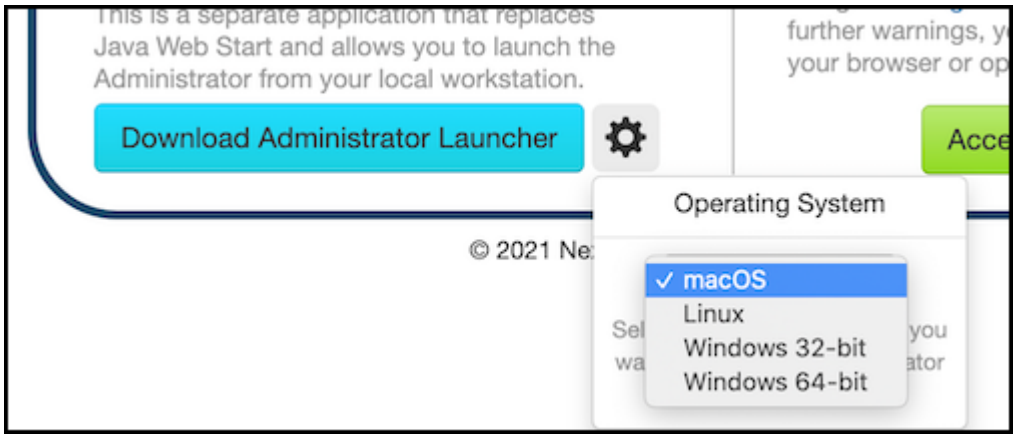
Mirth Connect Administrator Overview

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

Note: You no longer need to use Java Web Start. NextGen Healthcare has released a new Administrator Launcher that you can use instead. See Administrator Launcher Overview *(on page 78)*.

Download 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. Select the cog icon next to the **Download Administrator Launcher** button.



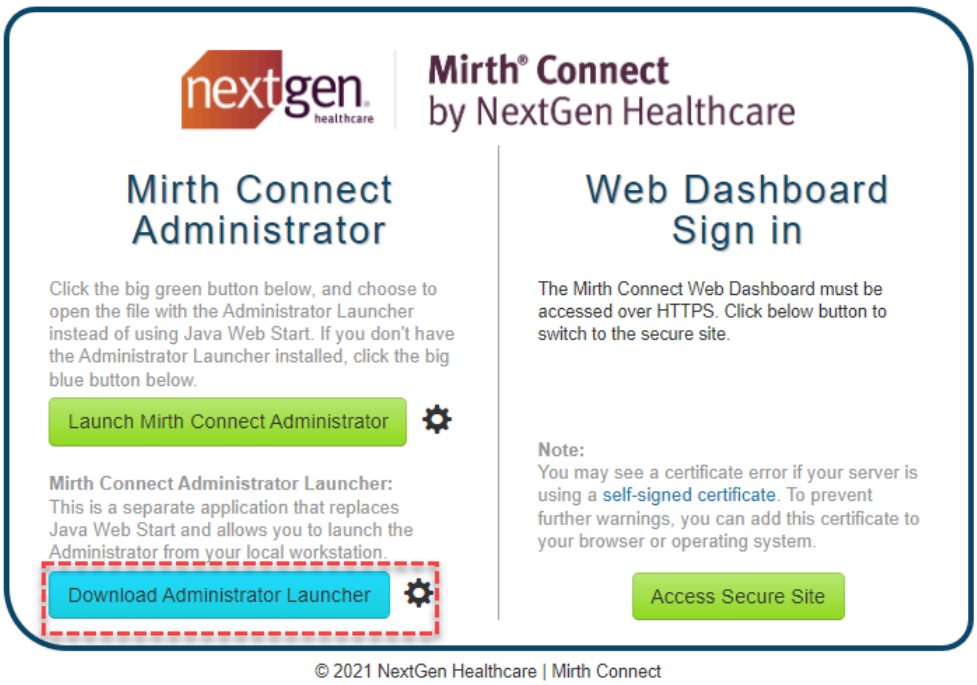
3. Choose the operating system for your local machine that you want to install the Administrator Launcher on.
4. Select **Download Administrator Launcher** to download the installer file.

Launch the Administrator

After you download the Administrator Launcher, you can perform the following steps to launch and log on to the Mirth® Connect Administrator.

- 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.

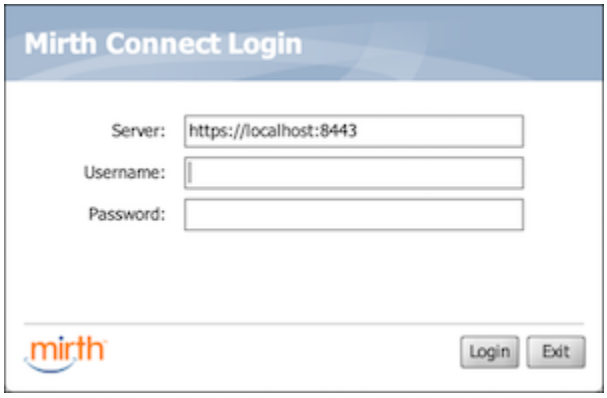
- Using the launch web page:
  - Open a web browser and navigate to your launch page.
    This is usually http://localhost:8080 or https://localhost:8443, though you may need to edit the IP or port depending on your settings.
  - Select **Download Administrator Launcher** to download the installer file.



- The **Access Secure Site** button takes you to the **Web Dashboard Sign In** page.

Log On

When you open the Administrator view, the following window appears:



The **Server** field is pre-filled with the correct IP address and port, however, you can change the address, if needed.

Note: If this is the first time you are logging on to Mirth® Connect, the default username is *admin*

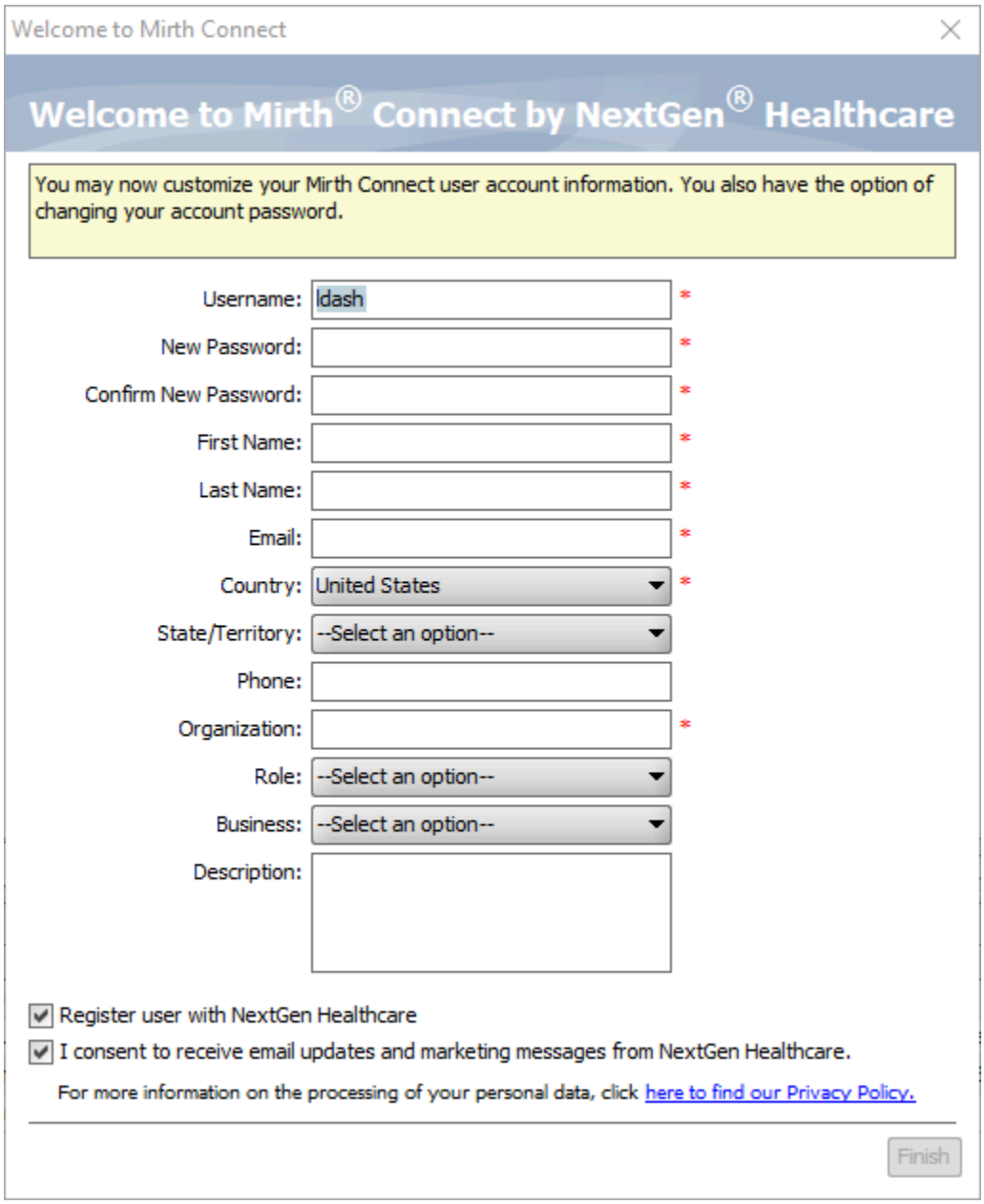
and the default password is *admin*.

1. Enter your username and password in the fields.
2. Select **Login**.

Note: If **Require Login Notification and Consent (Settings > Server)** is set to **Yes**, a window appears displaying the notification text. Select **Accept** to continue. The window closes after 5 minutes of inactivity, returning to the logon window.

Logging on for the First Time

When you log on to Mirth® Connect for the first time, you must set up your user account. During account setup, you can set your username, password, and personal information.

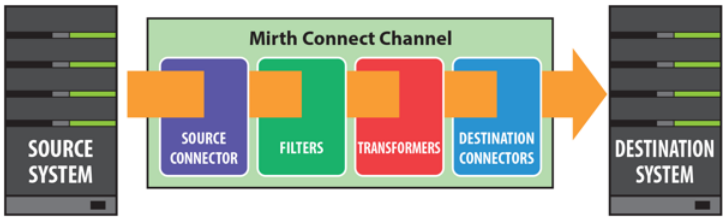


Note: If you are the first person to log on to Mirth® Connect, the "Register User with **NextGen Healthcare** check box does not appear because you are automatically registered as a user.

CHAPTER 4

The Fundamentals of Mirth Connect

As explained in the Introduction to Mirth Connect *(on page 20)* section, Mirth® Connect 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**.



About Channels and Connectors

A Mirth® Connect channel consists of multiple **connectors**. A connector is a piece of a channel that does the job of getting data into Mirth® Connect (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 *(on page 61)*
- Source Connector *(on page 61)*
- Destination Connectors *(on page 62)*

- Channel Scripts *(on page 62)*

General Channel Properties

General Channel properties are configured on the Summary Tab *(on page 204)* within the Edit Channel View *(on page 203)* and include:

- Unique ID, name, and description.
- Links to Code Template Libraries *(on page 213)*: These let the channel know which custom functions are available in specific JavaScript contexts.
- Links to Library Resources *(on page 214)*: These let the channel know which custom Java classes are available in specific connectors and/or JavaScript contexts.
- Deploy/Start Dependencies *(on page 215)*: 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 *(on page 219)*: 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 *(on page 227)*: 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 *(on page 229)*: These determine how long to keep message data around before automatically removing it with the Data Pruner *(on page 190)*. You can also decide to archive data out to a file somewhere before pruning it.
- Custom Metadata Columns *(on page 230)*: These allow you to extract pieces of data from your message and store them in dedicated columns in the internal Mirth® Connect database. You can then view and search on them in the Message Browser *(on page 103)*.

Source Connector

Every channel has exactly one source connector which gets data into Mirth® Connect from an external system. The source connector is configured on the Source Tab *(on page 232)* within the Edit Channel View *(on page 203)*. 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 takes any incoming data and splits it into multiple messages that each proceed discretely through the channel. See Batch Processing *(on page 337)* for additional information.
- 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. This \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 *(on page 246)* within the Edit Channel View *(on page 203)*. 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 *(on page 68)*.)
- 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. See Response Transformers *(on page 76)* for additional information.

Channel Scripts

Channel Scripts can be configured on the Scripts Tab *(on page 255)* within the Edit Channel View *(on page 203)*. 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 Properties *(on page 221)* 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.
- **Post Processor 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 *(on page 63)*
- Connector-Specific Properties *(on page 63)*
- Filter *(on page 65)*
- Transformer *(on page 65)*

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 *(on page 372)* will be different from a Database Writer *(on page 390)* and so on. Here is a list of source and destination connectors supported by Mirth® Connect:

- Source Connectors *(on page 340)*

    - Channel Reader *(on page 340)*
    - DICOM Listener *(on page 341)*
    - Database Reader *(on page 346)*
    - File Reader *(on page 351)*
    - HTTP Listener *(on page 362)*
    - JMS Listener *(on page 366)*
    - JavaScript Reader *(on page 371)*
    - TCP Listener *(on page 372)*
    - Web Service Listener *(on page 381)*

- Destination Connectors *(on page 384)*
    - Channel Writer *(on page 384)*
    - DICOM Sender *(on page 386)*
    - Database Writer *(on page 390)*
    - Document Writer *(on page 393)*
    - File Writer *(on page 395)*
    - HTTP Sender *(on page 400)*
    - JMS Sender *(on page 407)*
    - JavaScript Writer *(on page 410)*
    - SMTP Sender *(on page 411)*
    - TCP Sender *(on page 416)*
    - Web Service Sender *(on page 422)*

Additional connectors are made available as commercial extensions *(on page 582)*:

- Email Reader *(on page 588)*
- Serial Connector *(on page 597)*
- NextGen Results CDR Connector *(on page 600)*

- FHIR Connector *(on page 590)*
- Interoperability Connector Suite *(on page 592)*

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 *(on page 257)* within either the Source Tab *(on page 232)* or Destinations Tab *(on page 246)* within the Edit Channel View *(on page 203)*. See About Filters *(on page 74)* for additional information.

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 *(on page 257)* within either the Source Tab *(on page 232)* or Destinations Tab *(on page 246)* within the Edit Channel View *(on page 203)*. For additional information, see About Transformers *(on page 76)*.

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 *(on page 72)*

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 are part of the message that flow through a single connector. For example if your channel has two destinations, then a single message will have three connector messages associated with it as it is processed through the channel: One source connector, and two destination connectors. These connector messages correspond to the rows of data you see in the Metadata Table *(on page 106)* within the Message Browser *(on page 103)*.

Note that messages do not 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 *(on page 60)* 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. Message Metadata includes:

- **Message ID:** Every message for any given channel has a unique integer ID, the **Message ID.** The Message ID is used to organize data in the message browser and join connector messages together.

- **Connector Metadata ID:** 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 increment for each new destination you add to the channel.

- **Status**: 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 is sitting in a queue waiting to be processed, or one or more attempts have been made to process the message, but it has not yet been successfully processed.

- **Timestamps:** Timestamps are used 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.

- **Custom Metadata Columns:** Custom metadata columns are configurable columns located on the Summary Tab *(on page 204)* within the Edit Channel View *(on page 203)*. They allow you create your own columns that show up in the Metadata Table *(on page 106)* and are searchable in the Message Browser *(on page 103)*.

All of this information and more is visible in the Metadata Table *(on page 106)* within the Message Browser *(on page 103)*.

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.

Note: 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.

For additional information on the various content types, see Message Content Types *(on page 111)*, Variable Maps *(on page 449)*, Error Content Types *(on page 116)*.

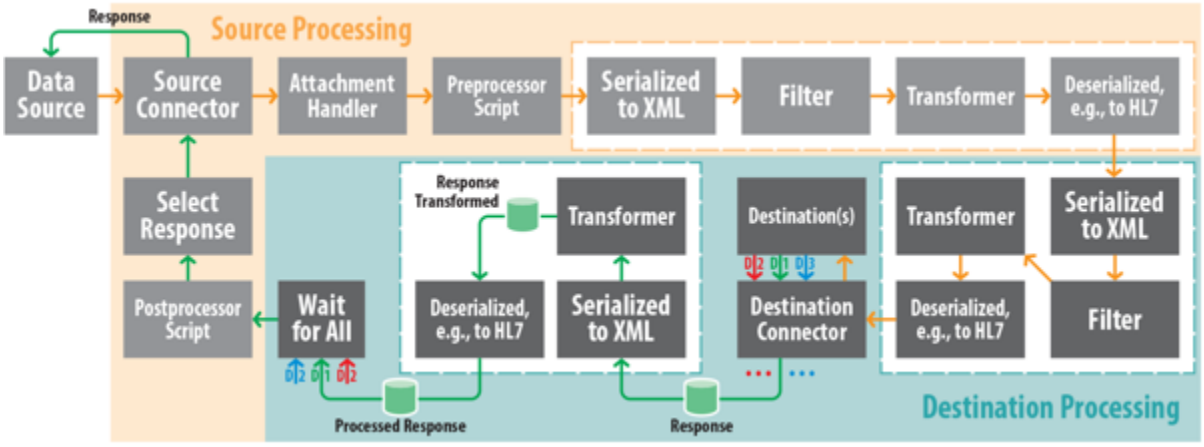
Message Attachments

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 *(on page 219)*.

The Message Processing Lifecycle

As explained in the About Message Data *(on page 65)* section, each source and destination connector has various versions of the message data as it flows through the channel. The **raw** inbound message enters Mirth® Connect, 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 all attachment data.
4. The post-attachment-handler content 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 *(on page 70)*.
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.

    Note: 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 only 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 *(on page 71)*.

Destination Processing Steps

Note: 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 *(on page 70)*.
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.

   Note: If no filter or transformer is configured, there will not be any transformed or encoded data. In that case, the content used from here on will only 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

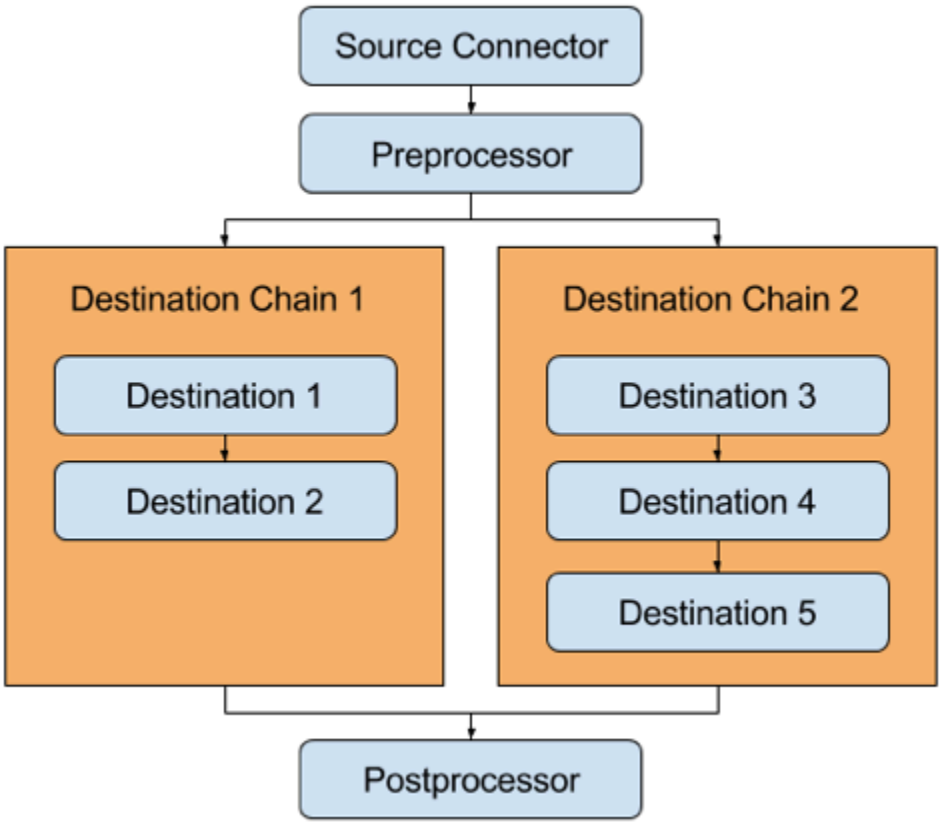
Note: This point is reached when all destination chains have finished processing the message. If

the message was filtered or errored out on the source connector, then flow will immediately jump down to this step.

1. The post processor 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 post processor 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:

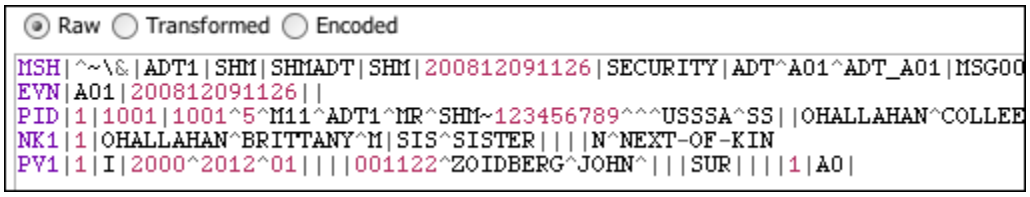


In the above example, there are 5 total destinations. However **Destination 3** does not wait on **Destination 2**, so it marks the beginning of a new chain. If each destination takes 1 second to process, then the overall time it takes the message to process through the channel will **not** be 5 seconds,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

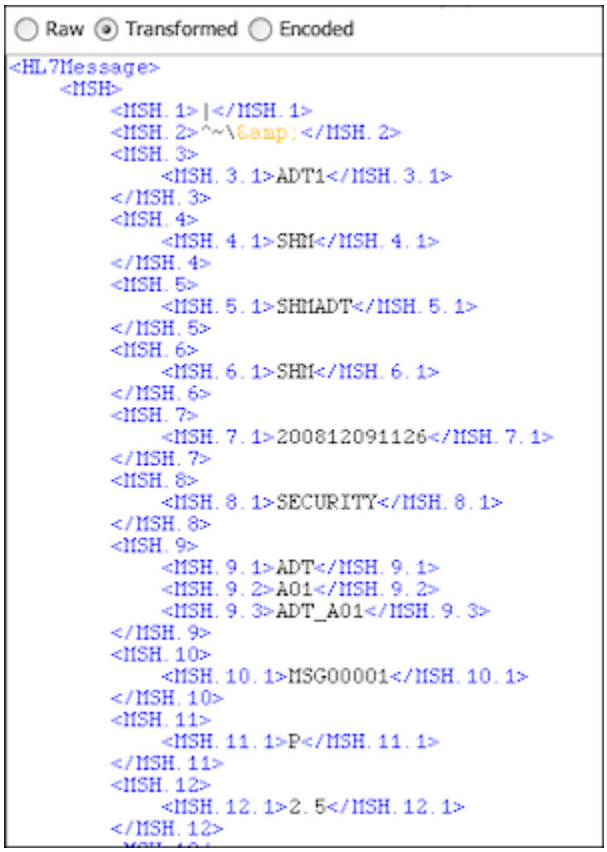
About Data Types

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 *(on page 329)* serializes ER7 data from this:



into this:



In this case, the internal representation is XML, 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 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 a Java String.

For additional information, see Data Types *(on page 323)*. Mirth® Connect supports the following data types:

- Delimited Text Data Type *(on page 324)*
- DICOM Data Type *(on page 327)*
- EDI / X12 Data Type *(on page 328)*
- HL7 v2.x Data Type *(on page 329)*
- HL7 v3.x Data Type *(on page 332)*

- JSON Data Type *(on page 333)*
- NCPDP Data Type *(on page 334)*
- Raw Data Type *(on page 335)*
- XML Data Type *(on page 336)*
- Batch Processing *(on page 337)*
- JavaScript Batch Script *(on page 338)*

The following additional data types are made available as commercial extensions:

- ASTM E1394 Data Type *(on page 586)*
- FHIR Connector *(on page 590)*

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 *(on page 257)* within either the Source Tab *(on page 232)* or Destinations Tab *(on page 246)* within the Edit Channel View *(on page 203)*.

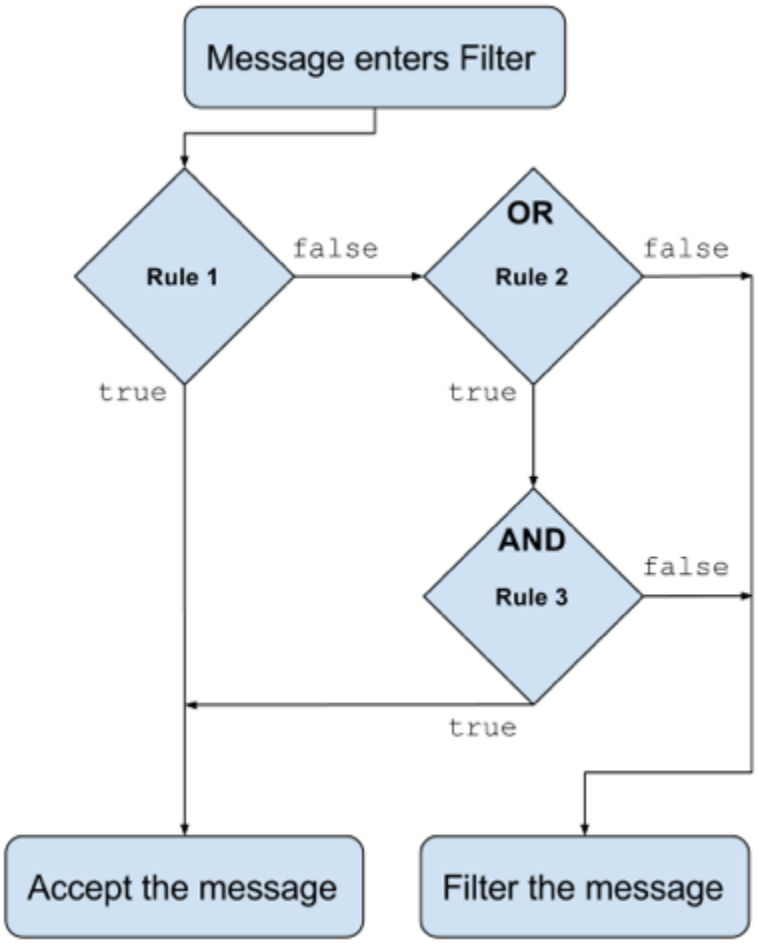
A filter returns **true** or **false**. When the filter returns true, the message is said to have been **accepted**. When the filter returns false, the message is said to have been **filtered**.

- 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 *(on page 72)* 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

Mirth® Connect supports the following filter rule types:

- Rule Builder Filter Rule *(on page 278)*

- JavaScript Filter Rule *(on page 279)*
- External Script Filter Rule *(on page 279)*
- Iterator Filter Rule *(on page 280)*

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 *(on page 257)* within either the Source Tab *(on page 232)* or Destinations Tab *(on page 246)* within the Edit Channel View *(on page 203)*.

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 *(on page 72)*.

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 *(on page 74)*.

The "tmp" Object

This is similar to **msg**, except that it is the internal representation of the outbound template *(on page 260)* configured in your transformer settings *(on page 257)*. 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 *(on page 286)*

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.

Note: Response transformers will only execute if there is an actual response payload to transform. For example if you are using an HTTP Sender *(on page 400)* 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 *(on page 335)*. 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

Mirth® Connect supports the following transformer step types:

- Mapper Transformer Step *(on page 281)*
- Message Builder Transformer Step *(on page 282)*
- JavaScript Transformer Step *(on page 283)*
- External Script Transformer Step *(on page 283)*
- XSLT Transformer Step *(on page 283)*
- Destination Set Filter Transformer Step *(on page 284)*
- Iterator Transformer Step *(on page 286)*

CHAPTER 5

Administrator Launcher Overview

The **Administrator** is the graphical user interface (GUI) that is used to log into a Mirth® 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 regardless of which version of Mirth® Connect you are using or which version of Java it is running on.

For more details about the Administrator Launcher, go to the NextGen Knowledge Center and view the *Installation Guide for Mirth® Connect Administrator Launcher*

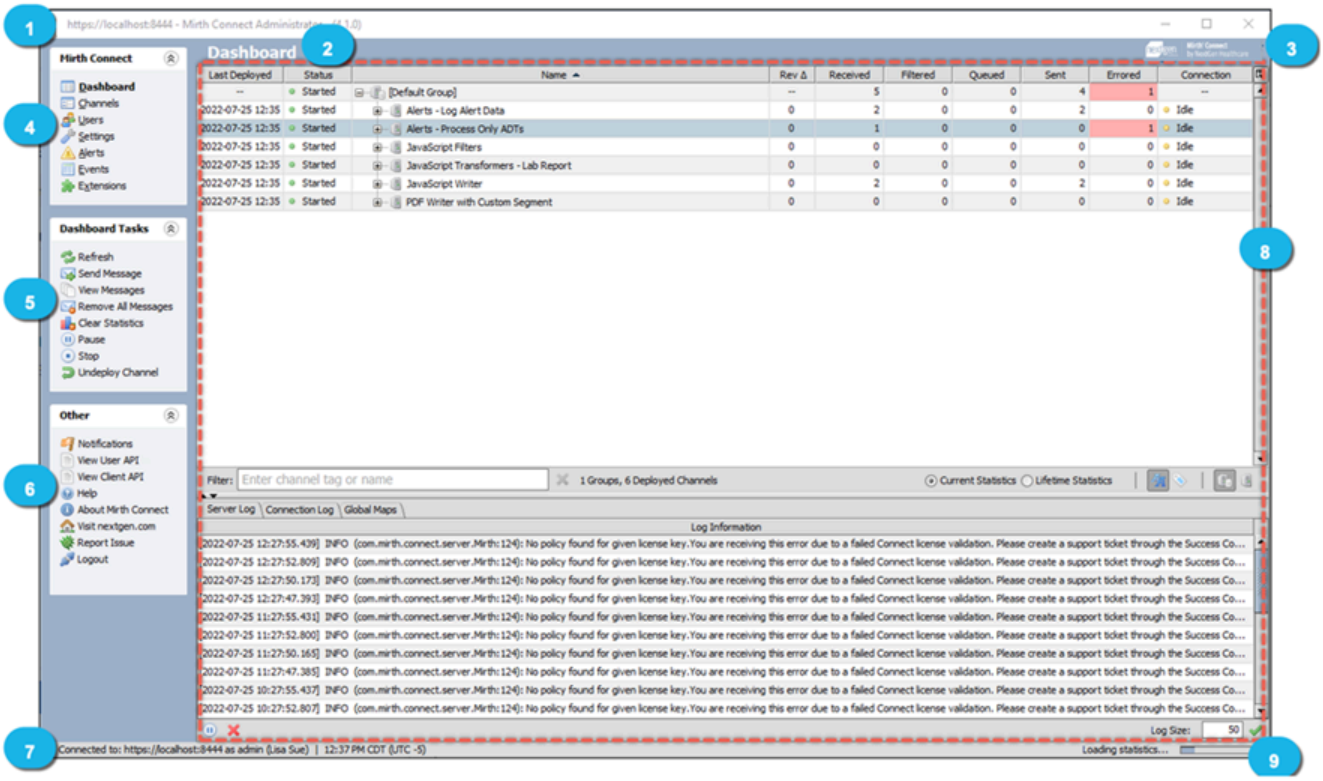
CHAPTER 6

Mirth Connect Administrator

The Mirth® Connect Administrator is the graphical user interface (GUI) that is used to log on to a Mirth® Connect server and create/modify/deploy channels. See Administrator Launcher Overview *(on page 78)* for information on opening Mirth® Connect Administrator. It can be accessed from the main launch page in a browser. For additional information on opening the Administrator, see Mirth Connect Administrator Overview *(on page 55)*.

Administrator Layout

Main view selections and context-specific tasks are located on the left side of the window. On the right side is the currently selected view. After you login to the Administrator, you will first be taken to the Dashboard view.



Item	Description
1	Server name and Mirth® Connect version
2	Current view title
3	Link to Mirth® Connect website
4	Main Administrator views: The highlighted view identifies the current view panel (in this case it is the Dashboard)
5	Context-specific tasks
6	Miscellaneous tasks
7	Server connection status
8	Current view panel (in this case, the Dashboard View)
9	Task progress bar

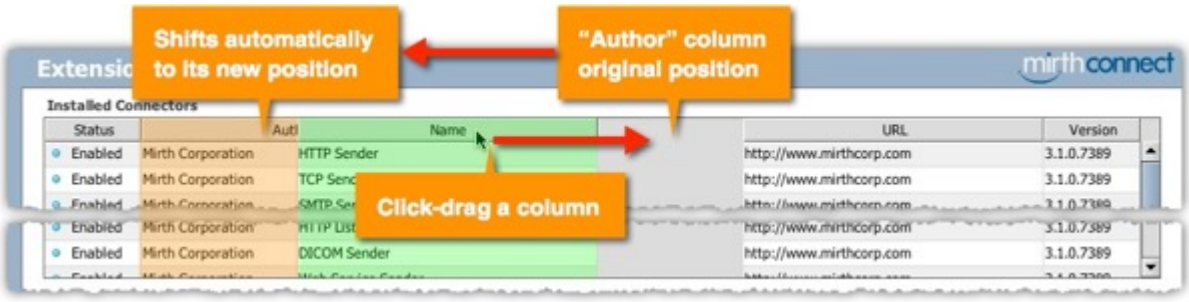
Working With Tables

Many of the tables throughout the Mirth® Connect Administrator share the same general control scheme that enables you to adjust the table layout to make the data easier to work with.

1. To rearrange columns, do either of the following:

   Choose from:

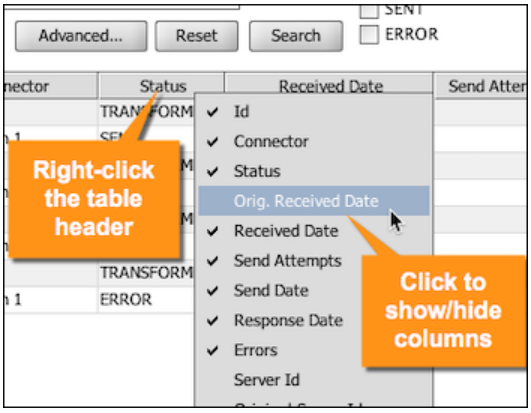
   - Select and hold the header for the column you would like to move.
   - Drag the column to its new location. As you drag a column, the next columns automatically slide in the opposite direction to make room for the column you are moving.



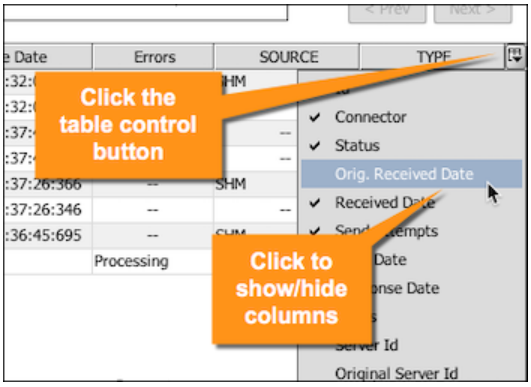
2. To show or hide columns, do either of the following:

   Choose from:

   - Right-click the table header, then select the name of the column you would like to show or hide.



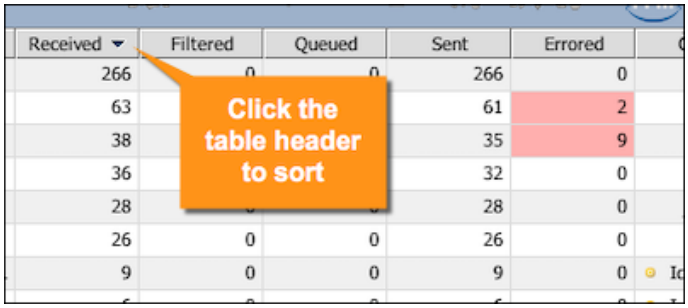
   - Select the table control button in the top-right portion of the table, then select the name of the column you would like to show or hide.



3. To sort column data, select the table header column name to create an ascending sort (

) or descending (

) sort for the column.

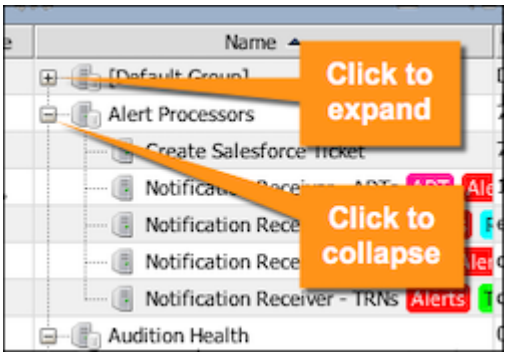


Note: Not all tables support sorting.

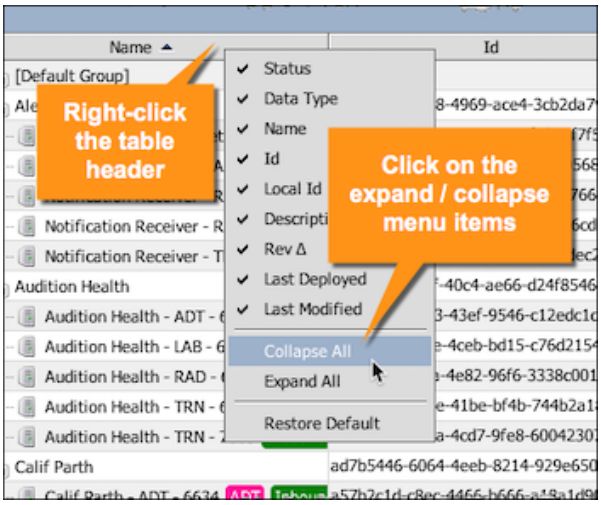
4. To expand or collapse rows, do either of the following:

   Choose from:

   - Select the plus/minus icons next to the row you would like to expand (+) or collapse (-).



   - Right-click the table header and select **Expand All** or **Collapse All** from the menu.

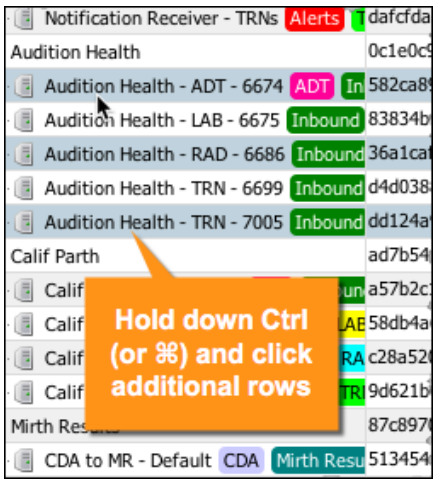


5. To select multiple rows, do either of the following:

   Choose from:

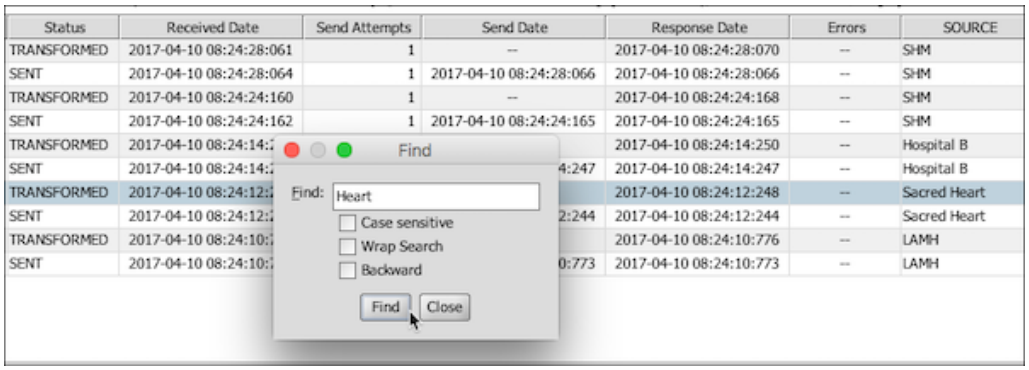
   - To select contiguous rows, select one of the rows in the table, hold down the **Shift** key and select a different row. While keeping the **Shift** key held down, press the **Up/Down** arrow keys to select additional rows.
   - To select non-contiguous rows, hold down the **Ctrl** key (or Apple Command key, ⌘)

and select multiple rows.

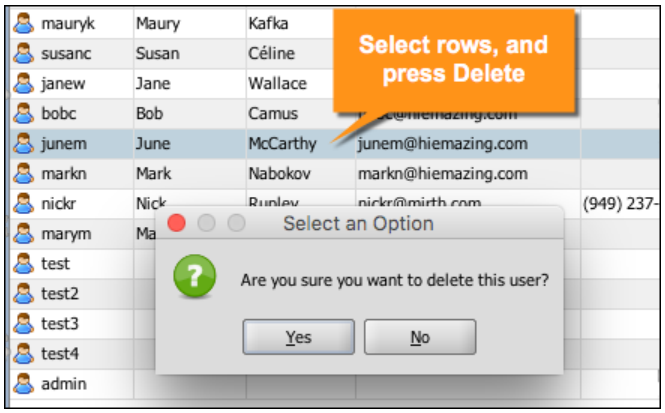


Note: Not all tables support multi-row selection,

6. To search the table, do the following:
   a. Press **Ctrl+F** (or ⌘+**F**) to open the Find window.
   b. Enter the string you would like to find, then select the **Find** button. If a matching row is found, it will be selected and scrolled to.



7. To delete a row, select the row you want to delete and press the **Delete** button:



Note: Not all tables support row deletion.

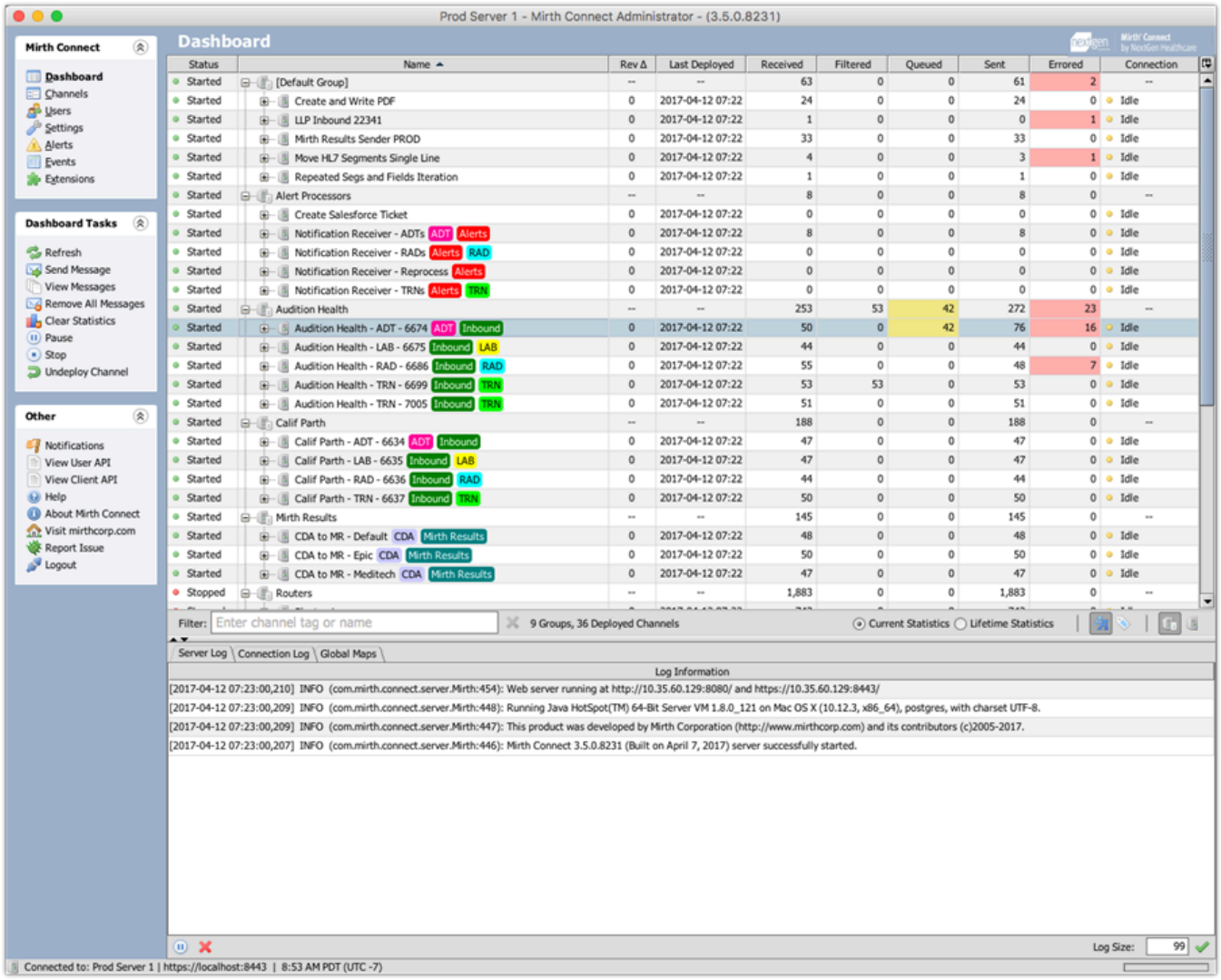
Monitor Views

This section is separated into the following topics:

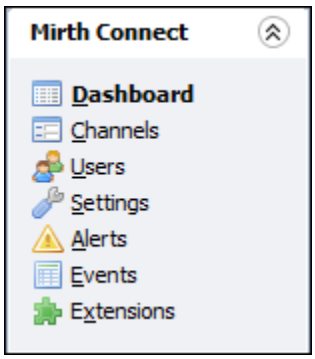
- Dashboard View *(on page 84)*
- Message Browser View *(on page 103)*
- Alerts View *(on page 139)*
- Events View *(on page 141)*

Dashboard View

The Dashboard is the default view you will see after logging on to the Mirth® Connect Administrator. The Dashboard gives you an overall view of all of your deployed channels. It also enables you to start or stop specific channels and connectors.

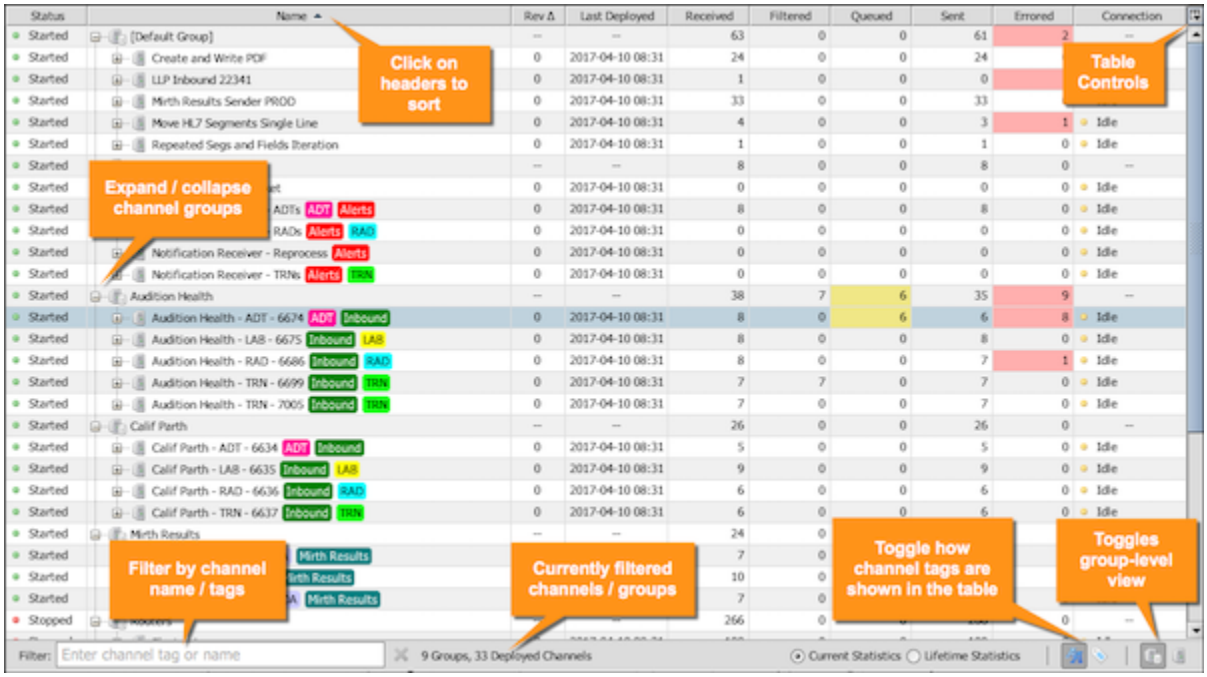


To access the Dashboard view from a different location, select the **Dashboard** link in the Task panel in the upper-left side of the window:



Dashboard Table

This is the main section of the dashboard. The Dashboard Table shows the current status and statistics for your deployed channels. For general information about working with tables in Mirth® Connect, see Working With Tables *(on page 79)*.



Dashboard Table Columns

Column	Description
Status	The status of the deployed channel. Common statuses include: • **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.

Column	Description
	• **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. 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.
Name	The name of the connector, channel, or group. This column also shows any tags associated with a channel.
Rev Δ	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 Δ = 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.
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 of 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	The current activity of the channel / connector. Not all connector types implement this column, so the default value you will see is **Idle**. Common connection statuses include:

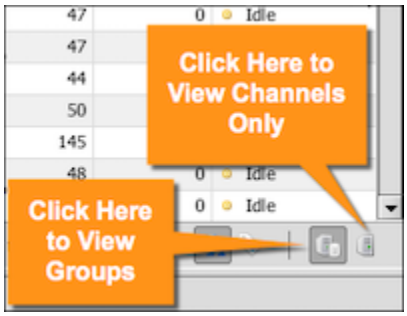
Column	Description
	• **Idle**: The source connector is not currently in the process of receiving a message. • **Reading**: The source connector is currently reading a message into the channel. Generally used by polling source connectors. • **Writing**: The destination connector is currently dispatching 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. **Note:** 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.

View Messages for a Channel

To view messages for a particular channel, select the channel and then select the **View Messages** task, as described in Dashboard Tasks *(on page 93)*. An alternative method is to simply double-click the channel from the dashboard table.

Show or Hide Channel Groups

By default, the channels are 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, select the control icons in the bottom-right side of the table.



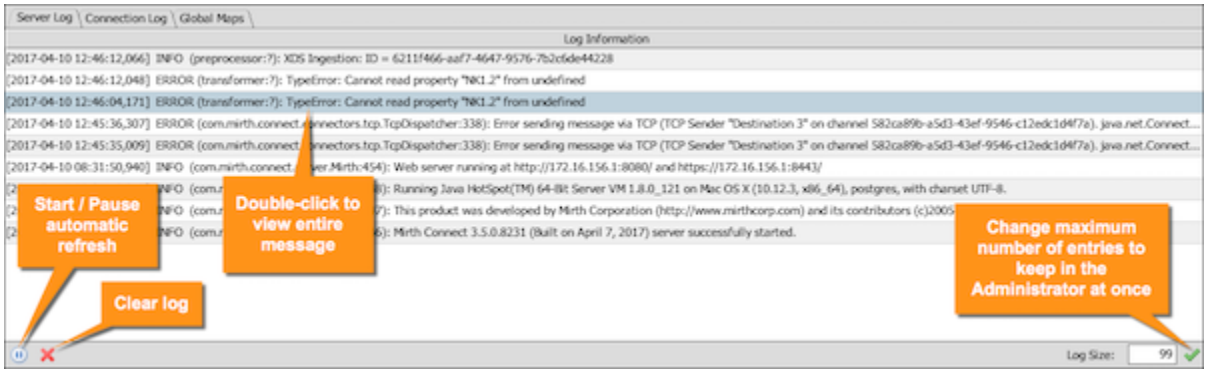
Change How Tags Display

By default, tags appear as bubble-text next to the channel names in the table. To change this to a small icon, select the control icons in the bottom-right side of the table.

To not hide tags in the table, select the currently selected icon again to deselect it.

Server Log

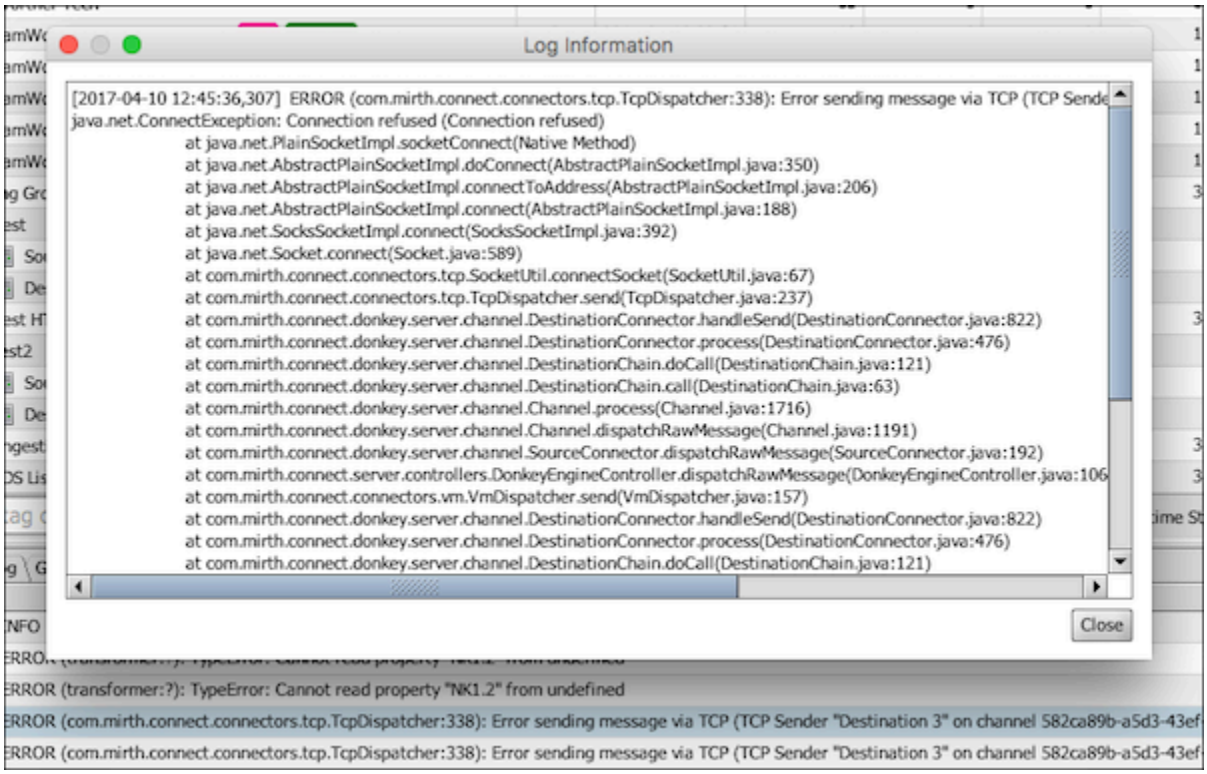
The **Server Log** tab in the Dashboard is a place to view the latest entries that have been written to the server logs. These entries mirror what gets written to the logs folder in the installation directory. For additional information, see Installation Directory *(on page 478)*.



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

- The current time stamp. 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.

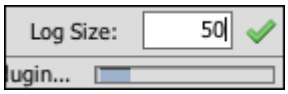
For long messages, the value shown in the table will be truncated. To show the entire message for these entries, double-click the row.



By default a maximum of 99 entries are 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 on the bottom-right side of the window.

- Select the check icon.

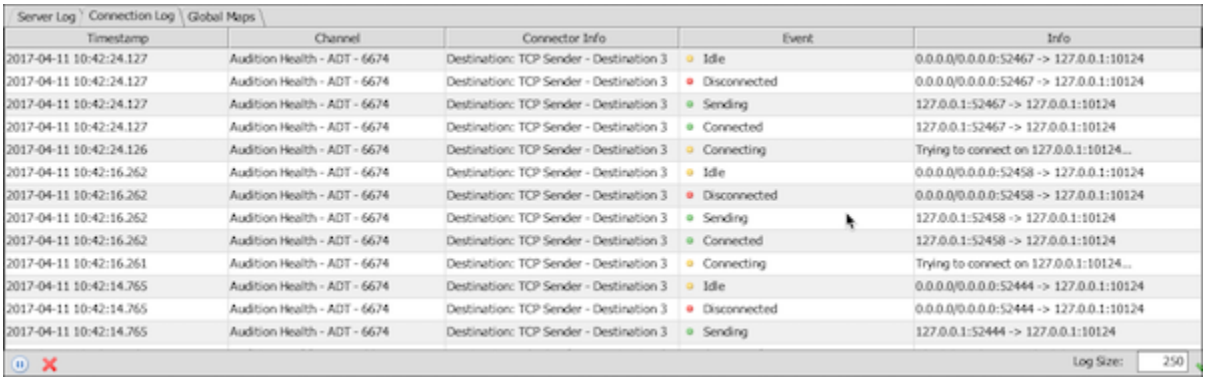


To clear the logs shown in your Administrator session, select the **X** button in the bottom-left.

To start/pause the logs from being updated in your Administrator session, select the **start/pause** button on the bottom-left side of the window. 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.

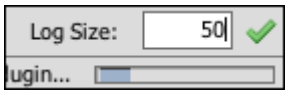


The message shown in the **Info** column depends on the type of channel / connector used. For example, for a TCP Sender destination this will typically show the connected socket information: <

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

By default a maximum of 250 entries are shown at once in the **Connection Log** tab. Newly received entries 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 on the bottom-right side of the window.
- Select the check icon.



To clear the logs shown in your Administrator session, select the **X** button on the bottom-left side of the window.

To start/pause the logs from being updated in your Administrator session, select the **start/pause** button on the bottom-left side of the window. 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 *(on page 449)*.

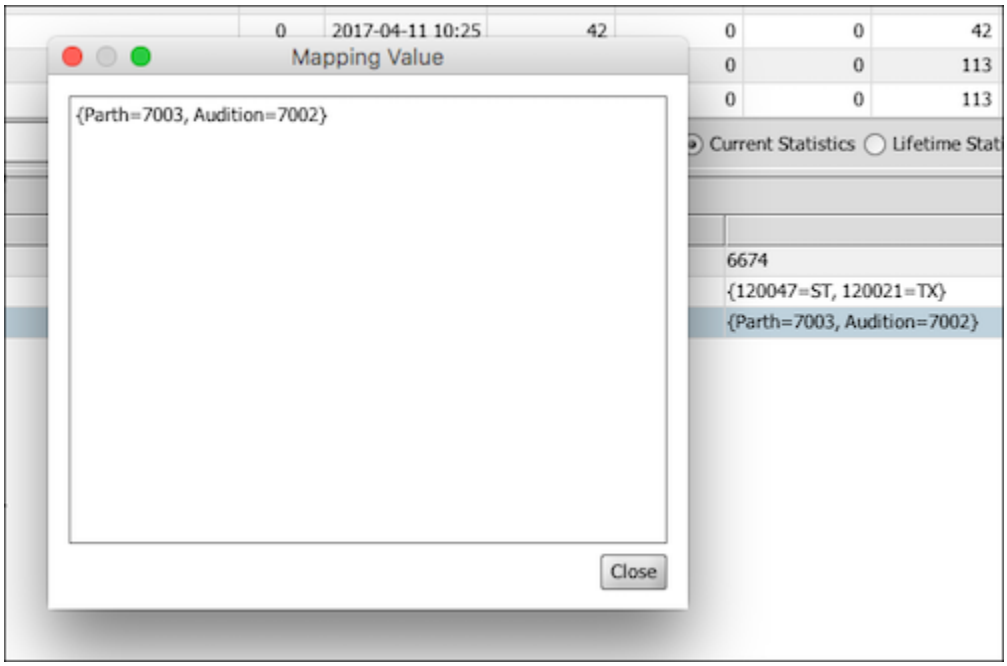
The Global Maps table shows all current global map entries, and all current global channel map entries for the channels that are currently selected in the dashboard table. You can multi-select channels to view global channel map entries across multiple channels at once.

Global Maps Table Columns

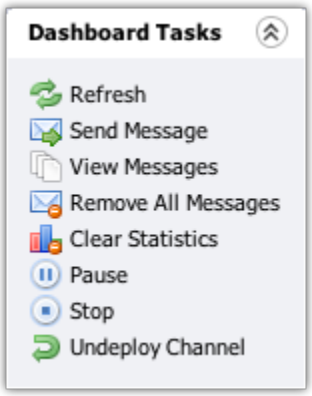
Column	Description
Server ID	Identifies the ID of the server where the global map variable resides.
Channel	This will either be the channel name, or "<Global Map>" if the entry refers to the global map rather than a global channel map.
Key	The string used to uniquely identify the entry within the given map.

Column	Description
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.

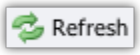
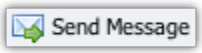
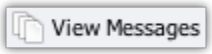
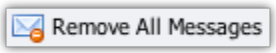
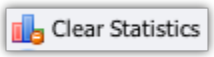
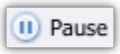
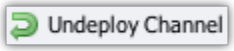
For values with long string representations, the value shown in the table will be truncated. To show the entire message for these entries, double-click the value cell:



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. For additional information, see Send Message *(on page 95)*.
	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. For additional information, see Remove All Messages *(on page 96)*.
	Clear All 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. For additional information, see Clear All Statistics *(on page 174)*
	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) **immediately**. 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 instead will only be visible from the Channels view. When undeploying, the channel will first be gracefully stopped, and any currently processing messages will first be finished. Note that the Channel Dependencies workflow may apply to this task.

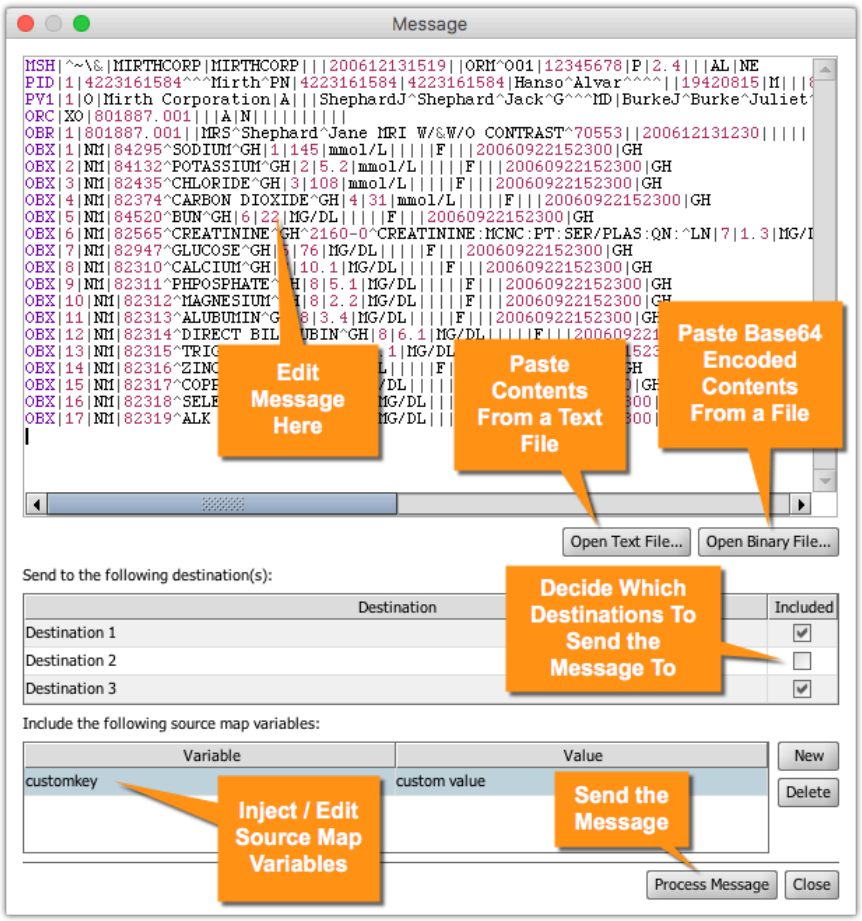
Send Message

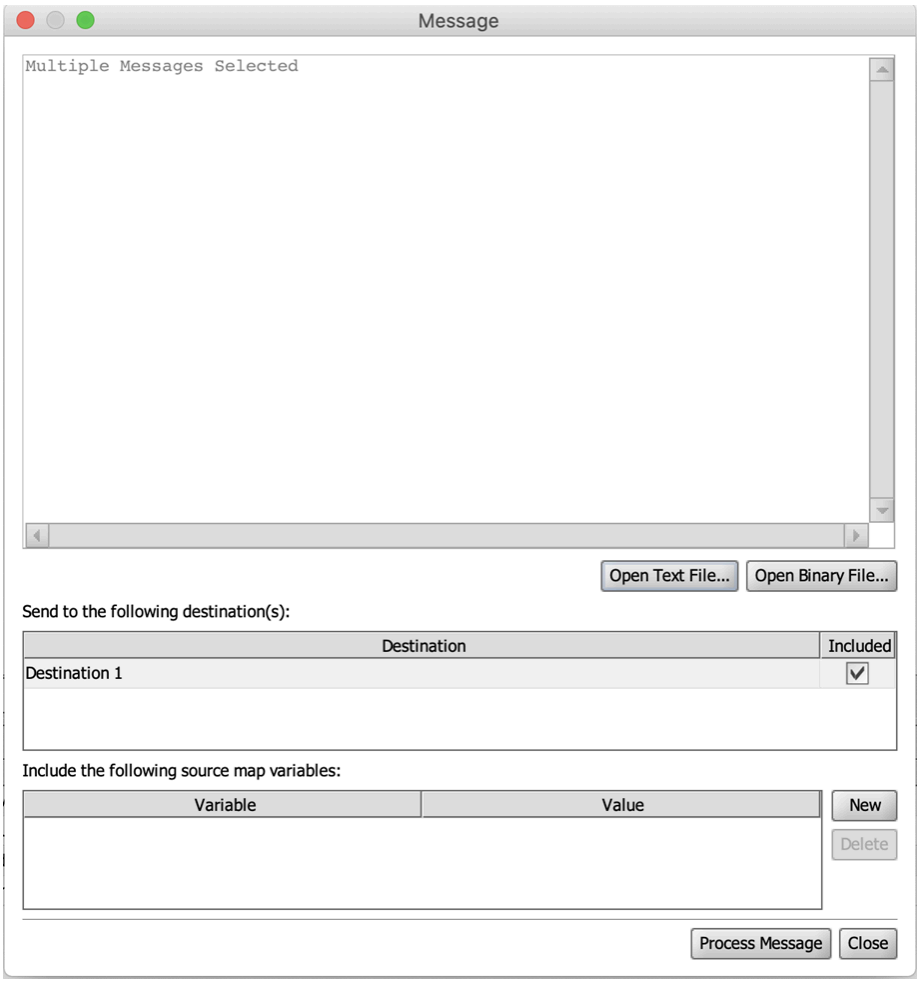
When you select the **Send Message** task, a dialog pops up allowing you to select the message to send. Type in a custom message, or select a file from disk to process. When reading a file, you can open it as a "Text" file which uses the default Java charset encoding. You can also open it as a "Binary" file, which pastes in the Base64-encoded contents of the actual file bytes. If you select multiple file names, the ability to edit is disabled. The text "Multiple Messages Selected" will be displayed in the preview panel. On processing, each selected file will be sent as individual messages.

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.

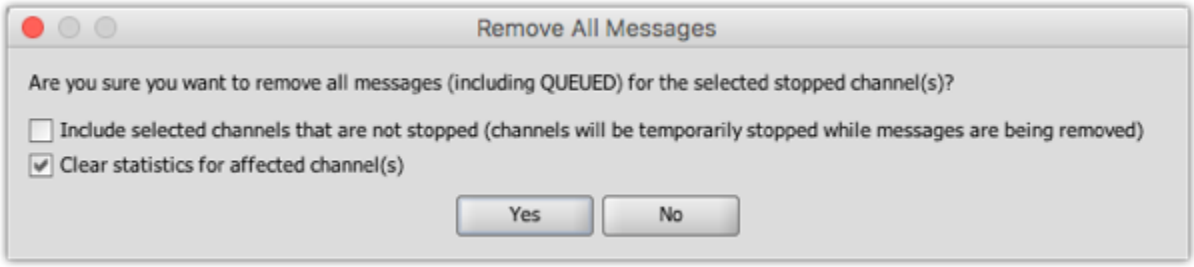
Once you are ready to send the message, select the **Process Message** button.





Remove All Messages

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



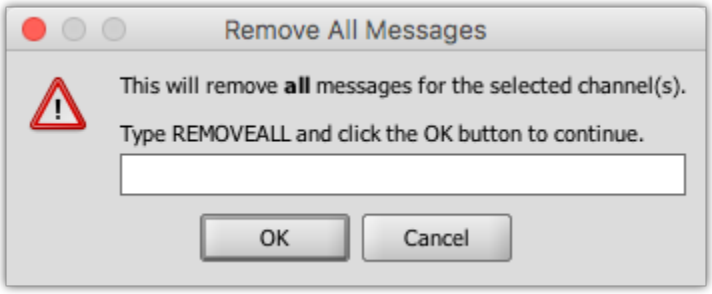
- **Include selected channels that are not stopped (channels will be temporarily stopped while**

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 automatically start 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.

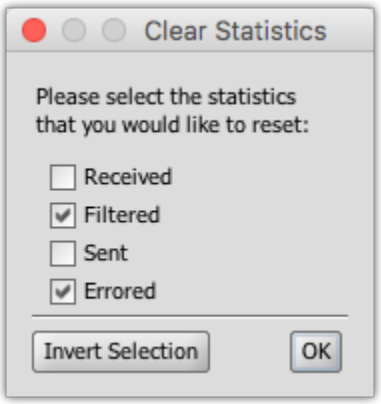
After selecting **Yes** to continue, you will see the following additional dialog:



This is an additional layer of security to ensure that you do not remove message data accidentally. To proceed, you must type in "REMOVEALL" in all capital letters and select **OK**. If this additional layer of security is annoying, you can disable it in the Administrator Settings Tab *(on page 174)*.

Clear Statistics

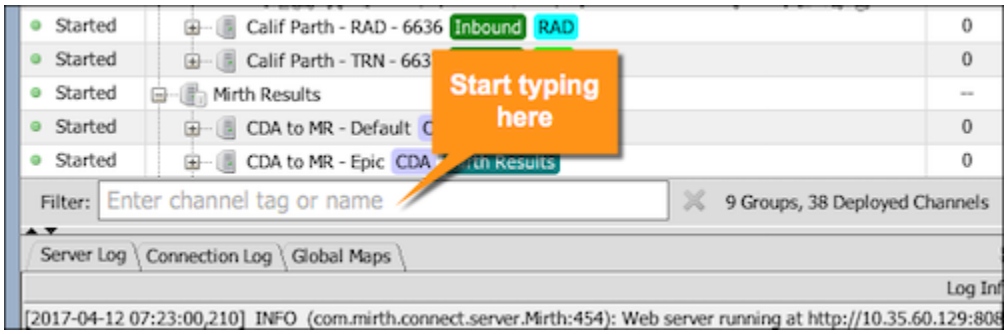
When you select the **Clear Statistics** task, a confirmation dialog pops up. Select the statistics you wish to reset to zero. You can quickly select all values by selecting the **Invert Selection** button. When you are ready, select the **OK** button to proceed.



Note that this only resets current statistics. To reset lifetime statistics, go to the Server Settings Tab *(on page 167)*.

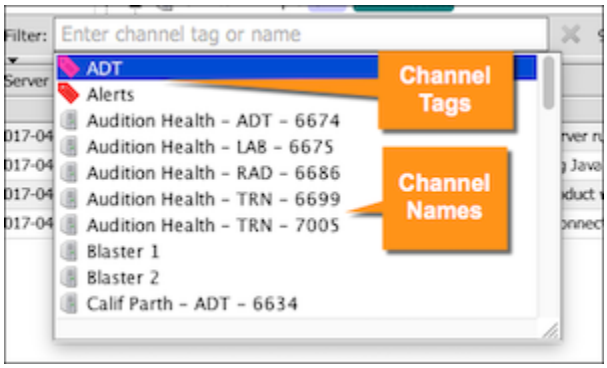
Filter By Channel Name or Tag

The **Filter** field, located in the bottom-left side 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.

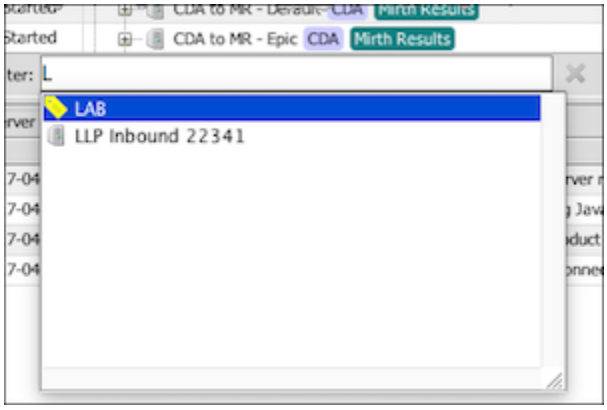


Note:

- 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.



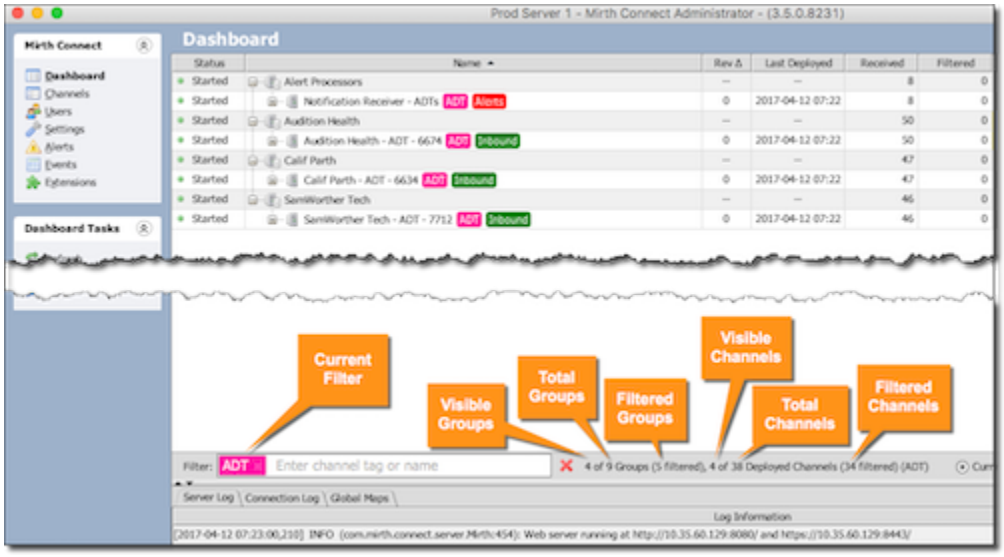
Filter By Channel Tags

To apply a channel tag filter:

- Select a channel tag from the drop-down list, then hit the **Enter** key,

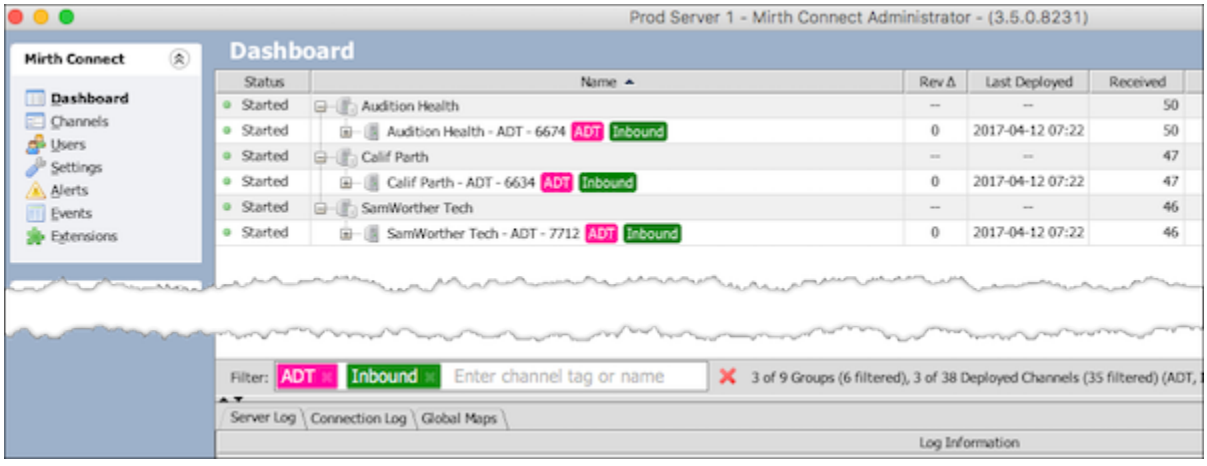
- **OR** double-click a channel tag from the drop-down list,
- **OR** type a tag into the text field, then hit the **Enter** key.

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.



When a filter is present, the status label next to the **Filter** field displays 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.



Filter By Channel

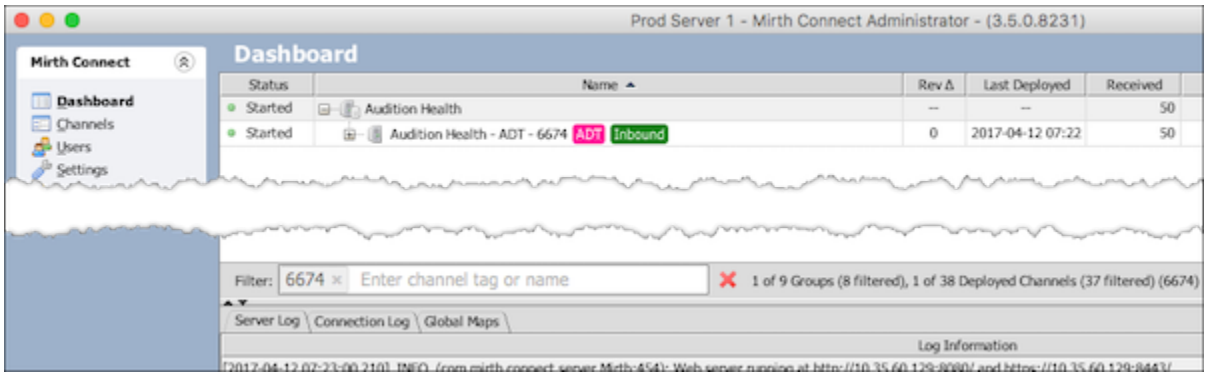
To filter down to a specific channel:

- 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

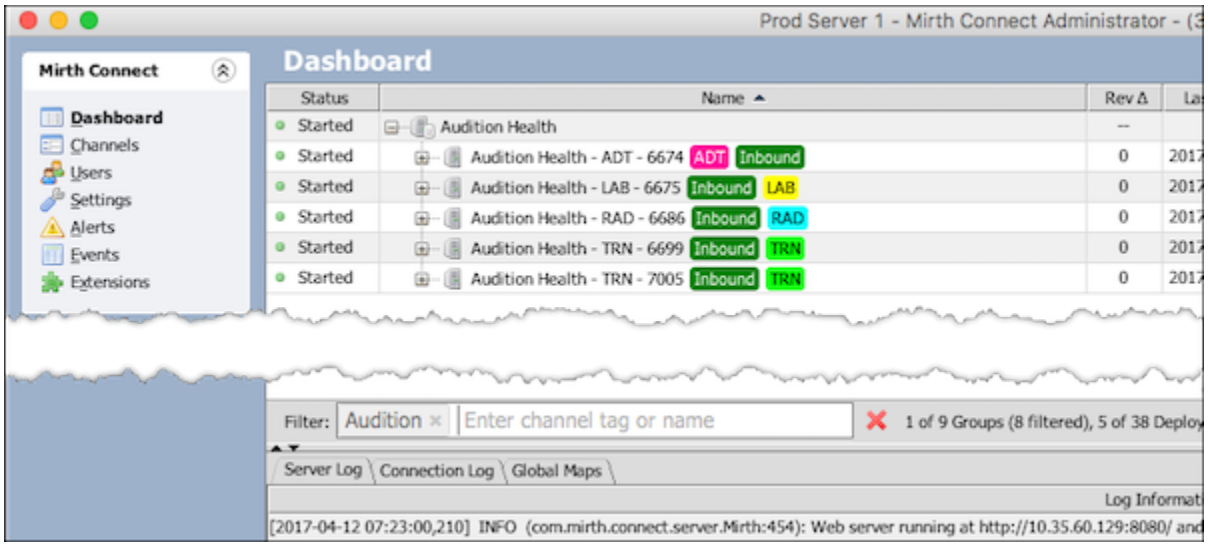
To filter down to all channels that partially match a certain string:

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



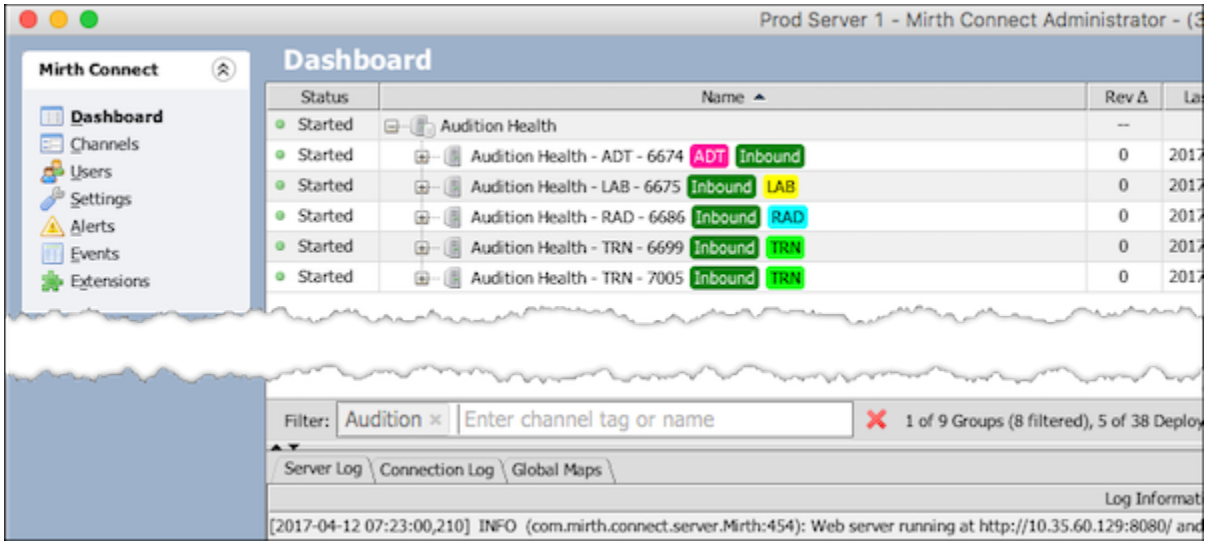
If the string you wish to filter by already matches the beginning of a channel name, then that channel will automatically be selected instead. If you do not want that to happen:

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



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.



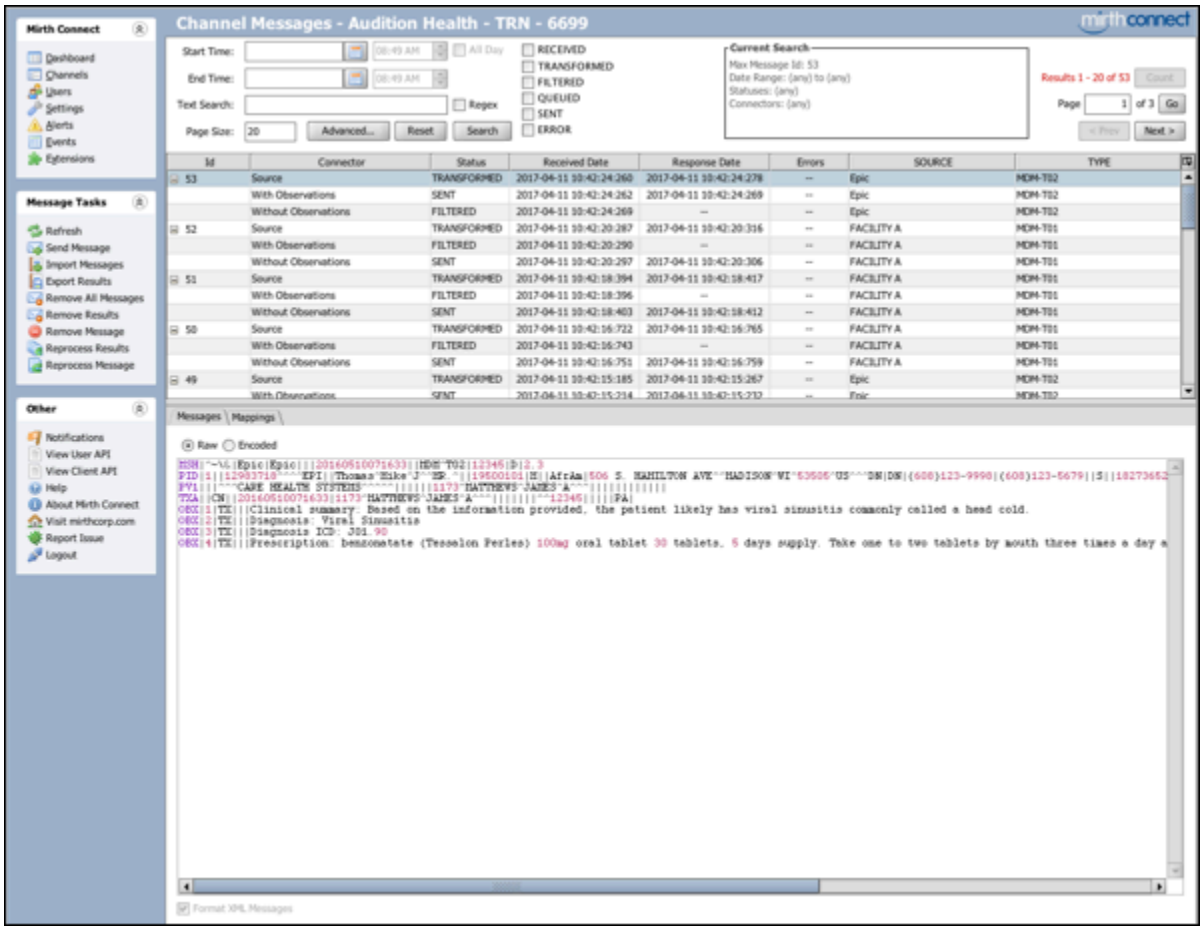
Clear Filter Criteria

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

To clear all filter criteria, select the red **X** button next to the **Filter** field:.

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.

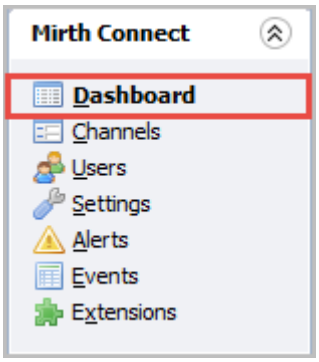


Navigation

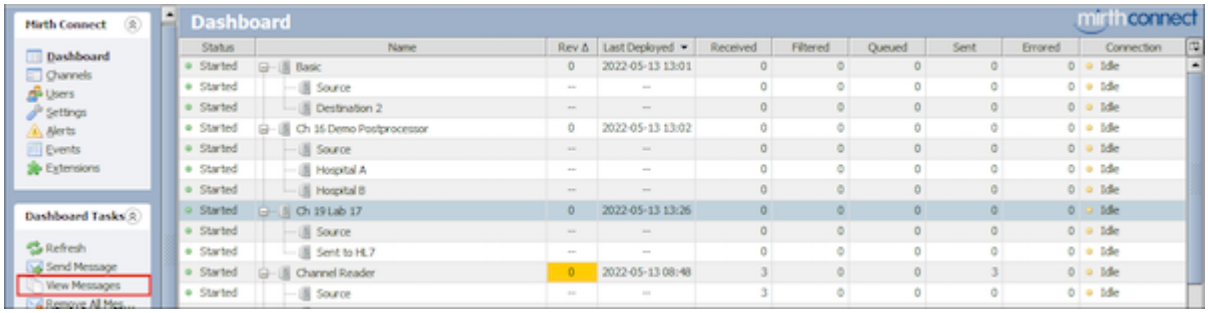
- From the Dashboard *(on page 104)*
- From the Channels View *(on page 105)*

From the Dashboard

1. Select the **Dashboard** link in the **Mirth Connect** task panel on the upper-left side of the window.

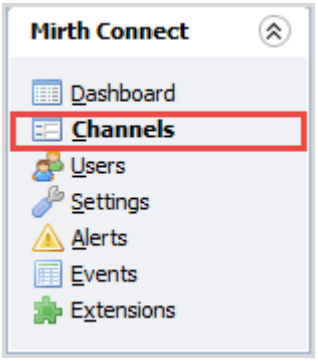


2. Select a channel where the messages you would like to view reside.
3. Select the **View Messages** task in the left task list or double-click the channel in the Dashboard Table *(on page 86)*. For additional information, see Dashboard Table *(on page 86)*:

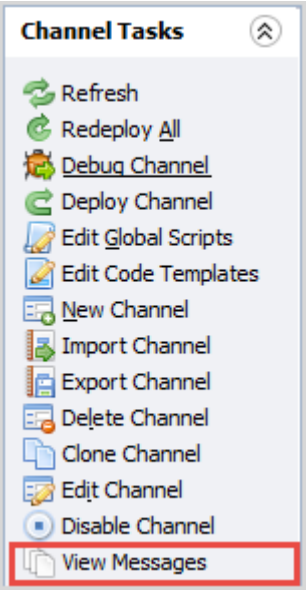


From the Channels View

1. Select the **Channels** link in the **Mirth Connect** task panel on the upper-left side of the window.



2. Select a channel where the messages you would like to view reside.
3. Select the **View Messages** task in the left task list. For additional information, see Channel Tasks *(on page 154)*.

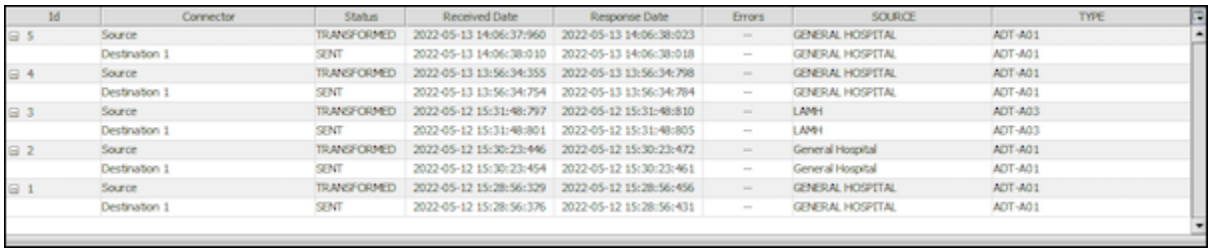


This section is separated into the following topics:

- Metadata Table *(on page 106)*
- Message Content Tab *(on page 111)*
- Mappings Tab *(on page 115)*
- Errors Tab *(on page 116)*
- Attachments Tab *(on page 118)*
- Search Messages *(on page 124)*
- Message Browser Tasks *(on page 130)*

Metadata Table

This table shows metadata (such as ID, status, and 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 (see Search Messages *(on page 124)* for additional information about the Page Size option). When you first enter the Message Browser, by default the latest 20 messages (by ID) appear.



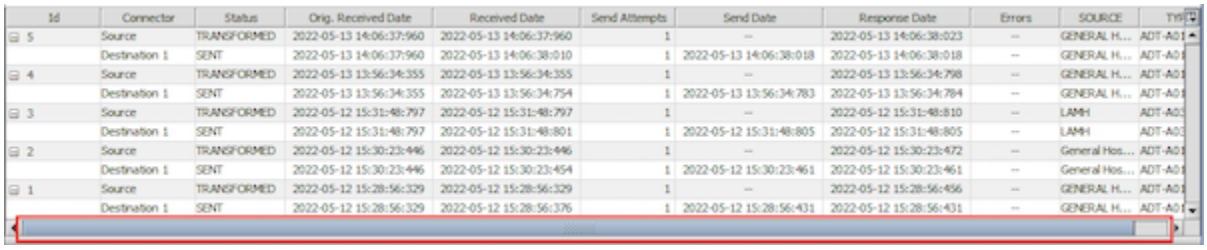
Id	Connector	Status	Received Date	Response Date	Errors	SOURCE	TYPE
5	Source	TRANSFORMED	2022-05-13 14:06:37:960	2022-05-13 14:06:38:023	--	GENERAL HOSPITAL	ADT-A01
	Destination 1	SENT	2022-05-13 14:06:38:010	2022-05-13 14:06:38:018	--	GENERAL HOSPITAL	ADT-A01
4	Source	TRANSFORMED	2022-05-13 13:56:34:355	2022-05-13 13:56:34:798	--	GENERAL HOSPITAL	ADT-A01
	Destination 1	SENT	2022-05-13 13:56:34:754	2022-05-13 13:56:34:784	--	GENERAL HOSPITAL	ADT-A01
3	Source	TRANSFORMED	2022-05-12 15:31:48:797	2022-05-12 15:31:48:810	--	LAMH	ADT-A03
	Destination 1	SENT	2022-05-12 15:31:48:801	2022-05-12 15:31:48:805	--	LAMH	ADT-A03
2	Source	TRANSFORMED	2022-05-12 15:30:23:446	2022-05-12 15:30:23:472	--	General Hospital	ADT-A01
	Destination 1	SENT	2022-05-12 15:30:23:454	2022-05-12 15:30:23:461	--	General Hospital	ADT-A01
1	Source	TRANSFORMED	2022-05-12 15:28:56:329	2022-05-12 15:28:56:456	--	GENERAL HOSPITAL	ADT-A01
	Destination 1	SENT	2022-05-12 15:28:56:376	2022-05-12 15:28:56:431	--	GENERAL HOSPITAL	ADT-A01

Messages are organized by ID, and multiple rows for each **connector message** appear in the table, depending on your search criteria *(on page 124)*.

Add a Column to the Metadata Table

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

Note: The 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 would 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.

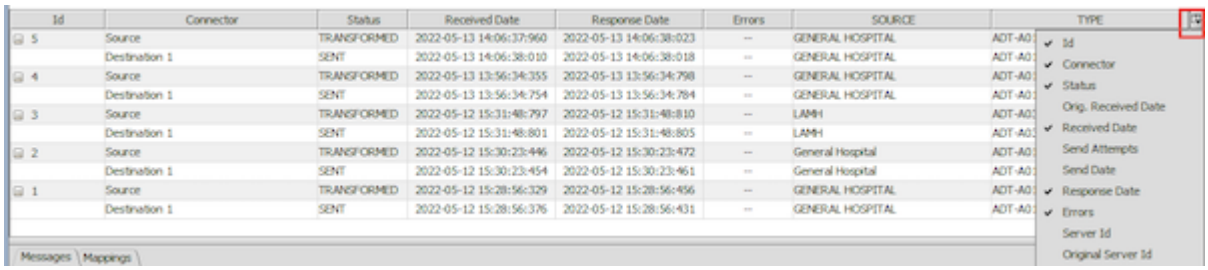


Id	Connector	Status	Orig. Received Date	Received Date	Send Attempts	Send Date	Response Date	Errors	SOURCE	TYP
5	Source	TRANSFORMED	2022-05-13 14:06:37:960	2022-05-13 14:06:37:960	1	--	2022-05-13 14:06:38:023	--	GENERAL H...	ADT-A01
	Destination 1	SENT	2022-05-13 14:06:37:960	2022-05-13 14:06:38:010	1	2022-05-13 14:06:38:018	2022-05-13 14:06:38:018	--	GENERAL H...	ADT-A01
4	Source	TRANSFORMED	2022-05-13 13:56:34:355	2022-05-13 13:56:34:355	1	--	2022-05-13 13:56:34:798	--	GENERAL H...	ADT-A01
	Destination 1	SENT	2022-05-13 13:56:34:355	2022-05-13 13:56:34:754	1	2022-05-13 13:56:34:783	2022-05-13 13:56:34:784	--	GENERAL H...	ADT-A01
3	Source	TRANSFORMED	2022-05-12 15:31:48:797	2022-05-12 15:31:48:797	1	--	2022-05-12 15:31:48:810	--	LAMH	ADT-A03
	Destination 1	SENT	2022-05-12 15:31:48:797	2022-05-12 15:31:48:801	1	2022-05-12 15:31:48:805	2022-05-12 15:31:48:805	--	LAMH	ADT-A03
2	Source	TRANSFORMED	2022-05-12 15:30:23:446	2022-05-12 15:30:23:446	1	--	2022-05-12 15:30:23:472	--	General Hos...	ADT-A01
	Destination 1	SENT	2022-05-12 15:30:23:446	2022-05-12 15:30:23:454	1	2022-05-12 15:30:23:461	2022-05-12 15:30:23:461	--	General Hos...	ADT-A01
1	Source	TRANSFORMED	2022-05-12 15:28:56:329	2022-05-12 15:28:56:329	1	--	2022-05-12 15:28:56:456	--	GENERAL H...	ADT-A01
	Destination 1	SENT	2022-05-12 15:28:56:329	2022-05-12 15:28:56:376	1	2022-05-12 15:28:56:431	2022-05-12 15:28:56:431	--	GENERAL H...	ADT-A01

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.

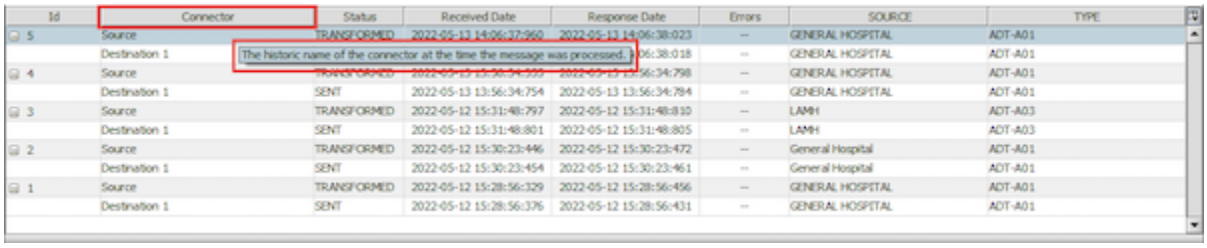
1. On the far-right side of the Message List column header, select the **Column Options** icon.
2. On the **Column Options** menu, select or deselect the column(s) you would like to add or remove from the column header.



Note: 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.



For general information about working with tables in Mirth® Connect, see: Working With Tables *(on page 79)*

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.

Column	Description
Connector	The name of the connector. (e.g. "Source", "Destination 1")
Status	The current status of the connector message. Valid values include: Source connector statuses: • **Received**: If the source queue *(on page 238)* 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

Column	Description
	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	• **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	• **Source Connector**: N/A • **Destination Connector**: The date and time immediately before the most recent send attempt.
Response Date	• **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	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: • **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. • **Post Processor:** An error occurred during the post processor script *(on page 255)*.

Column	Description
	• **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 *(on page 230)* for the current channel also show up in the metadata table. These columns 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 *(on page 167)*.

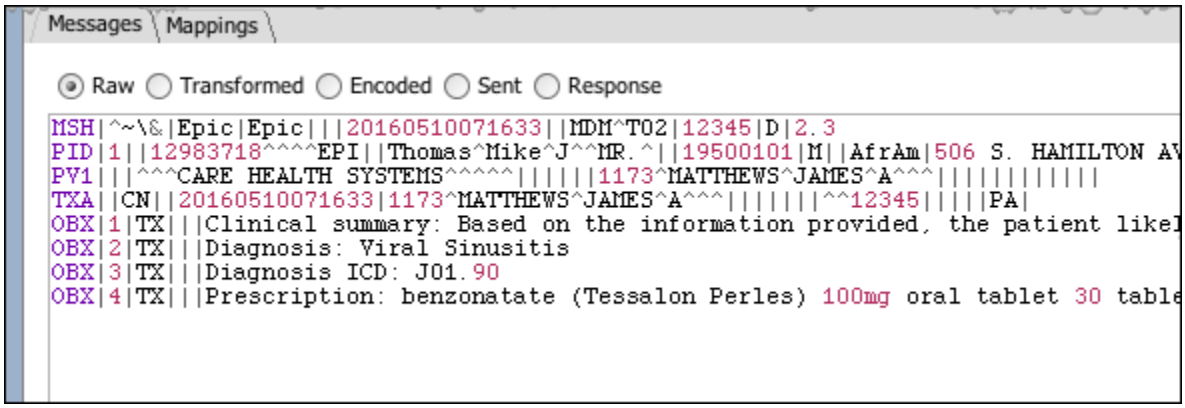
Column	Description
Source	Depends on the inbound data type *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* messages this will usually be the Sending Facility value in MSH.4.1.
Type	Depends on the inbound data type *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* messages this will usually be the Type and Trigger values in MSH.9.1 and MSH.9.2.
Version	Depends on the inbound data type *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* 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 *(on page 167)*.

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

Message Content Tab

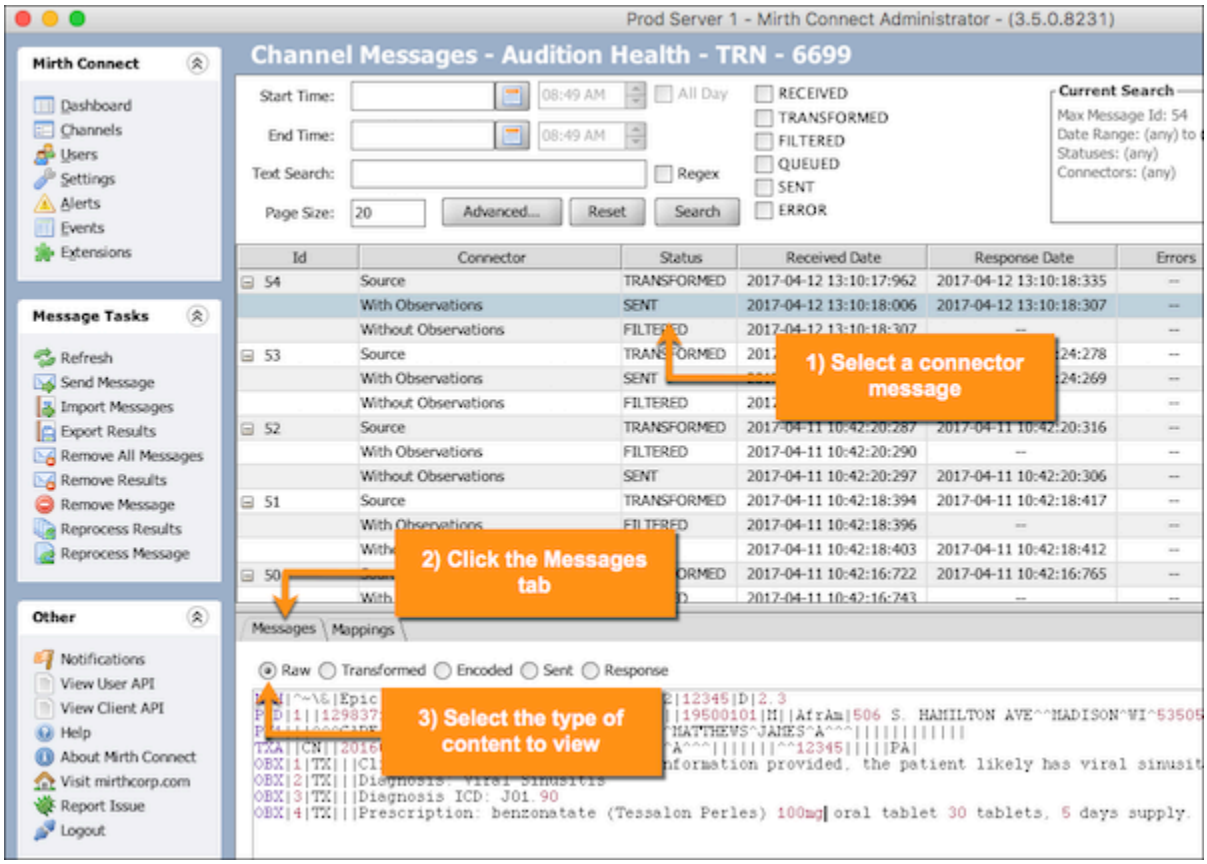
The **Messages** tab in the Message Browser shows the actual content of the message at various states as it

processed through the selected connector message.



View Message Content

1. Select a connector message in the Metadata Table *(on page 106)*.
2. Select the **Messages** tab (located below the Metadata Table).
3. Select the radio button corresponding with the content type you want to view.



Message Content Types

Content Type	Description
Raw	The inbound message as received and stored by the 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

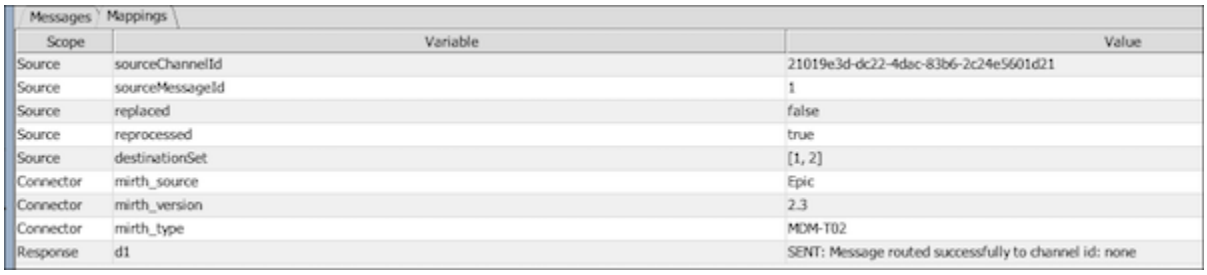
Content Type	Description
	destination connector attempts to dispatch the message.
Response	The Response object returned by the destination connector after a dispatch has been attempted. This will include the response status, the status message, and the actual response payload (if present).
Response Transformed	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 is easier to read. Note that this does not alter the actual message content, only how it is displayed in the Administrator.

Mappings Tab

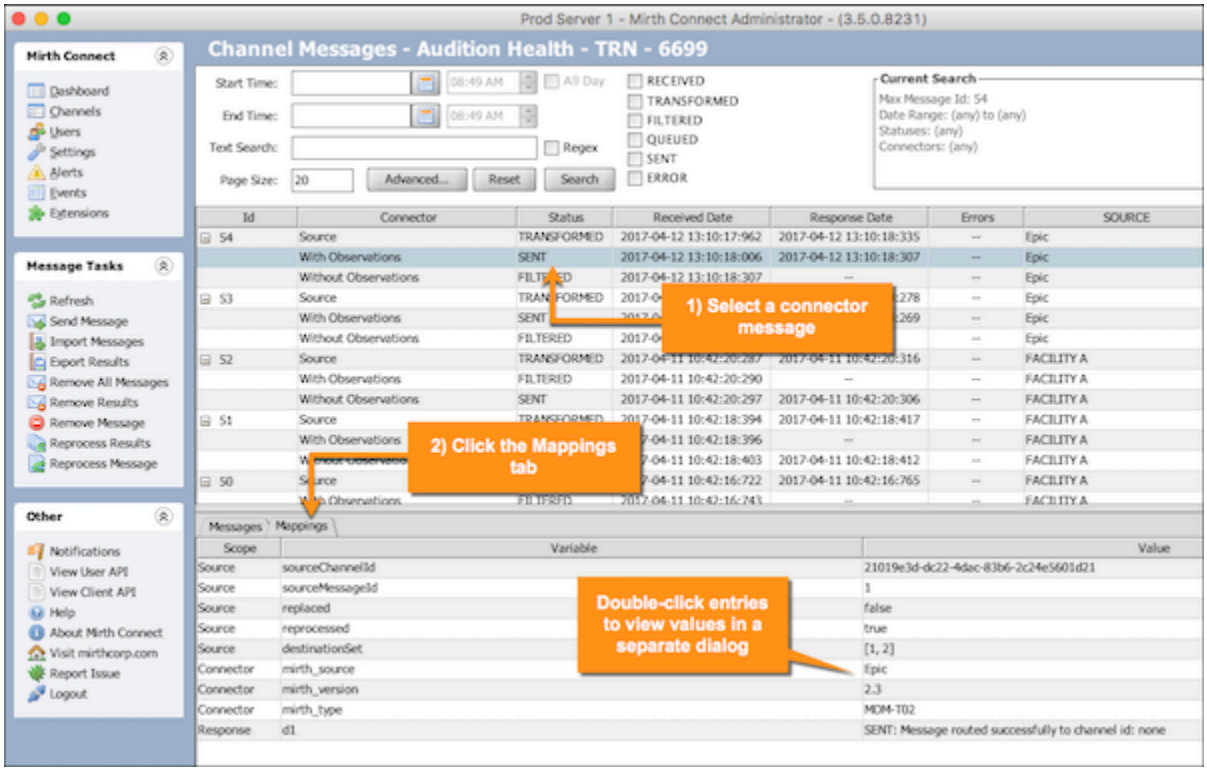
The **Mappings** tab shows all entries stored in variable maps *(on page 449)* for the selected connector message.



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

View Mappings

1. Select a connector message in the Metadata Table *(on page 106)*.
2. Select the **Mappings** tab (located below the Metadata Table).
3. Double-click entries in the table to view the values in a separate dialog.



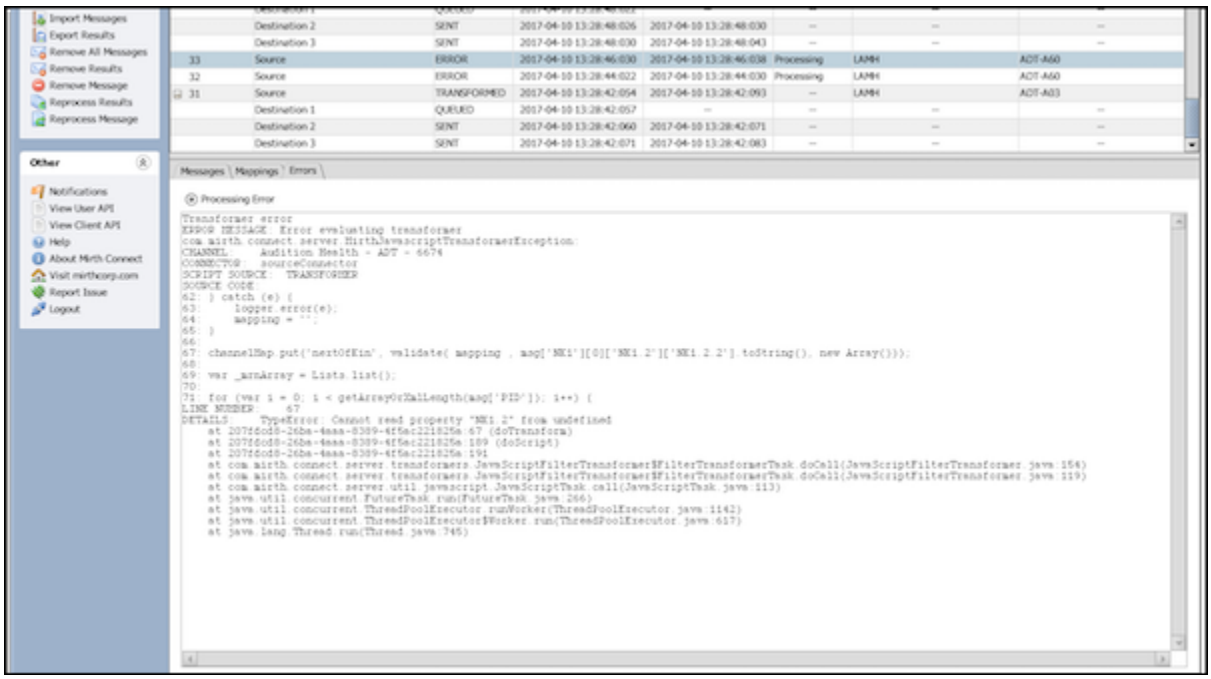
Mappings Table Columns

Column	Description
Scope	The type of map the entry is stored in. For additional information, see Variable Maps *(on page 449)*.
Variable	The unique key identifying the entry within the given map.
Value	The actual value of the entry.

Errors Tab

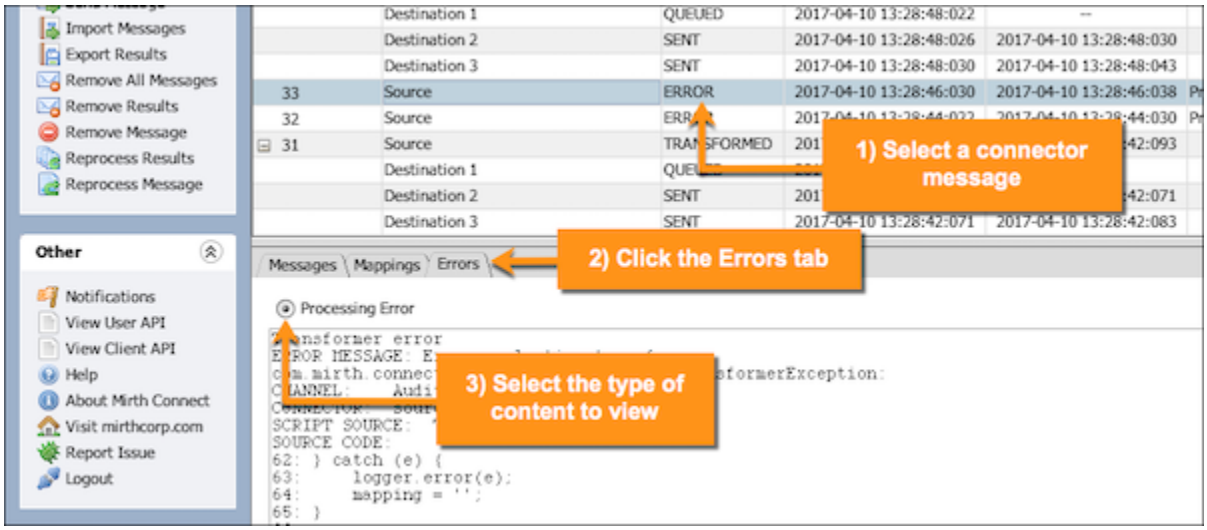
If any error content exists for a connector message, a separate **Errors** tab will be present in the message

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 *(on page 106)*.



View Message Errors

1. Select a connector message in the Metadata Table *(on page 106)*.
2. Select the **Errors** tab (located below the Metadata Table). Note that this tab is only available if the connector message actually has error content.
3. Select the radio button corresponding to the content type you want to view.

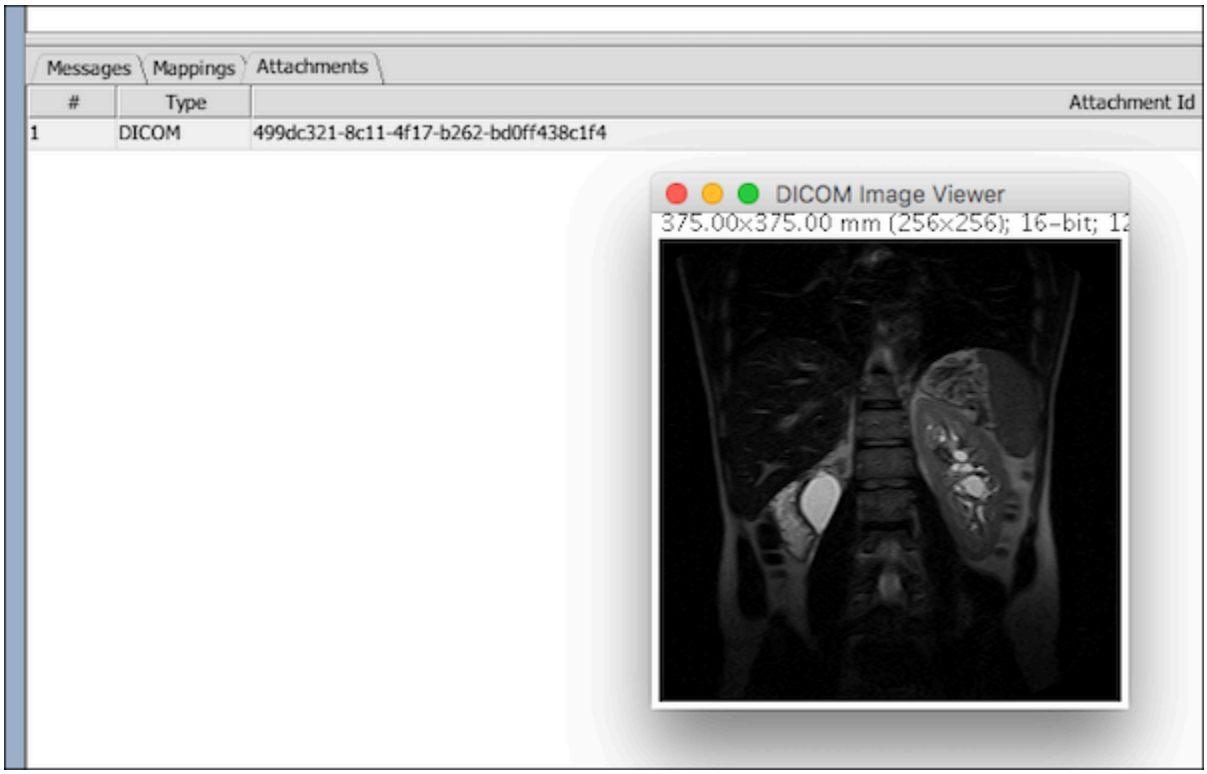


Error Content Types

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.
Post Processor Error	An error occurred during the post processor script.

Attachments Tab

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



Attachment Table Columns

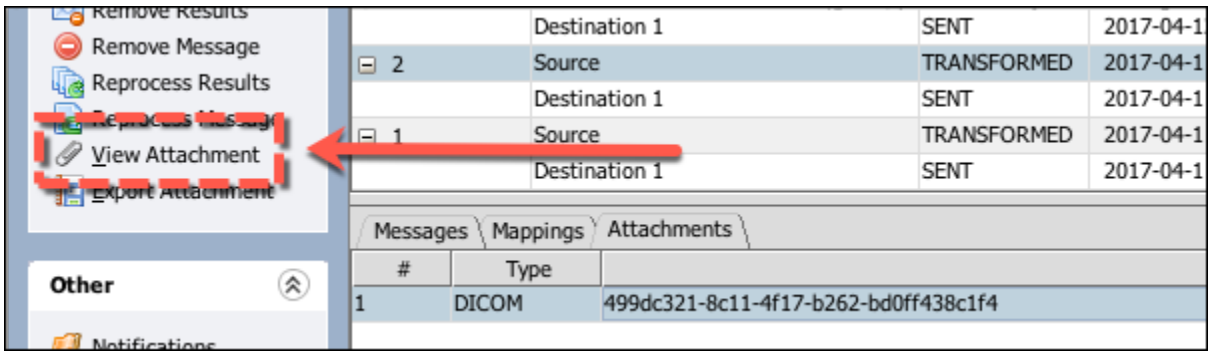
Column	Description
#	The sequence numbers associated with the attachment(s). This usually starts at 1 and increase for every attachment in the table. In some cases for DICOM messages you may see a single row that has a range of numbers in this column, indicating that multiple attachments have been stored 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 will see a comma-separated list of IDs.

View Attachments

To view attachment content:

- Double-click the row in the **Attachments** table,

- **OR**, select the row in the Attachments table, then select 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 *(on page 174)*.

Once you have chosen a viewer, select 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, clear 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.

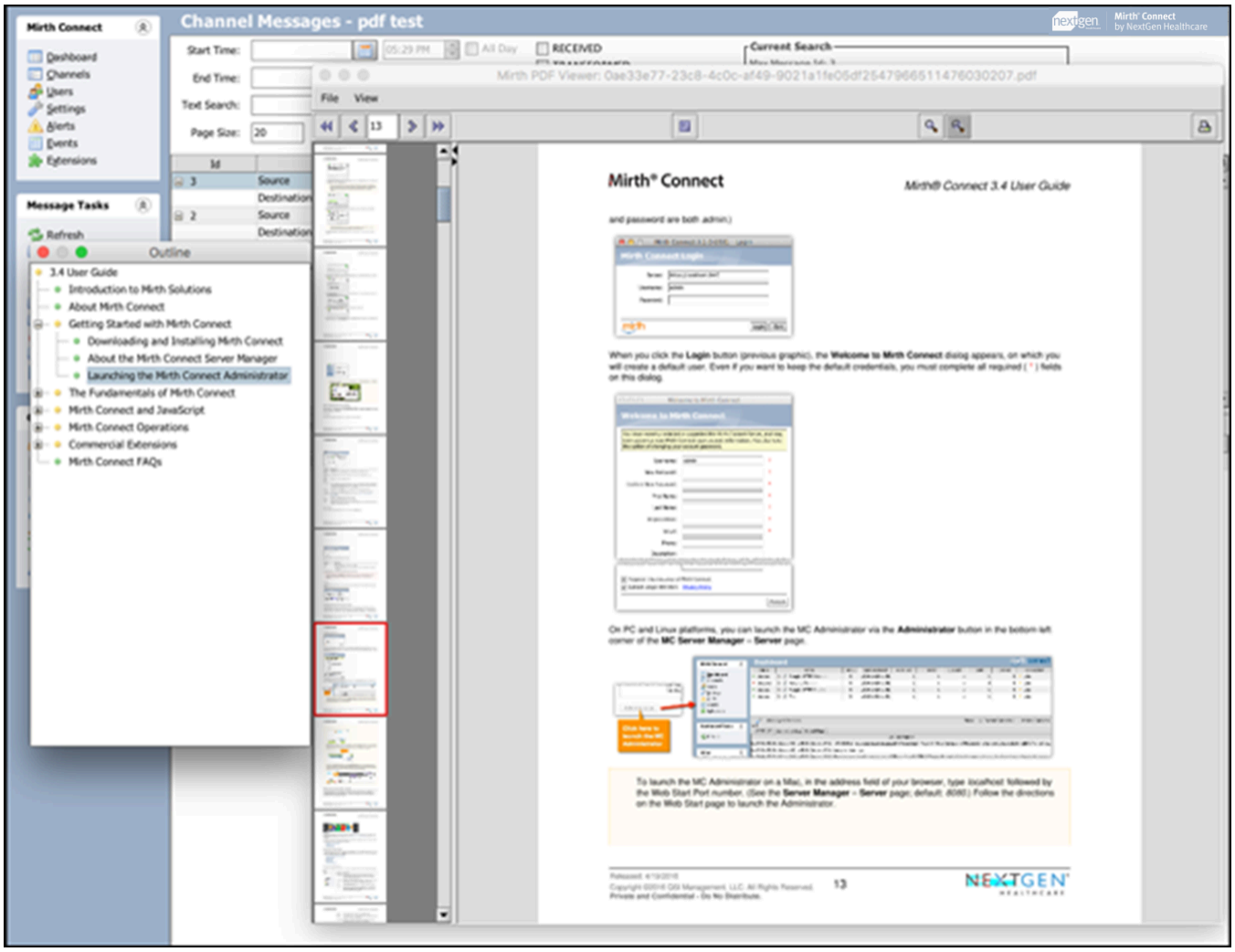
DICOM Attachment Viewer

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.

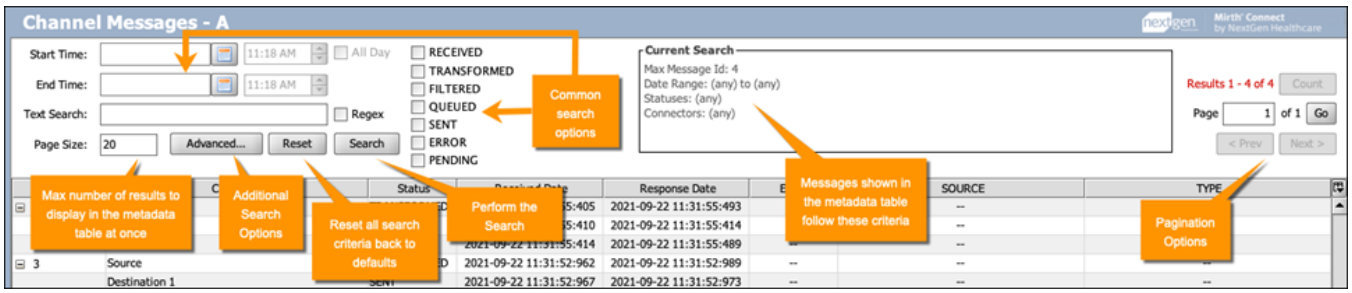
PDF Attachment Viewer

Use the PDF Attachment Viewer for application/pdf attachments. The viewer enables you to switch between pages, print the document, and more. If the PDF has a table of contents included, it is shown in a separate window to the side, enabling you to browse the outline and skip directly to a particular section.



Search Messages

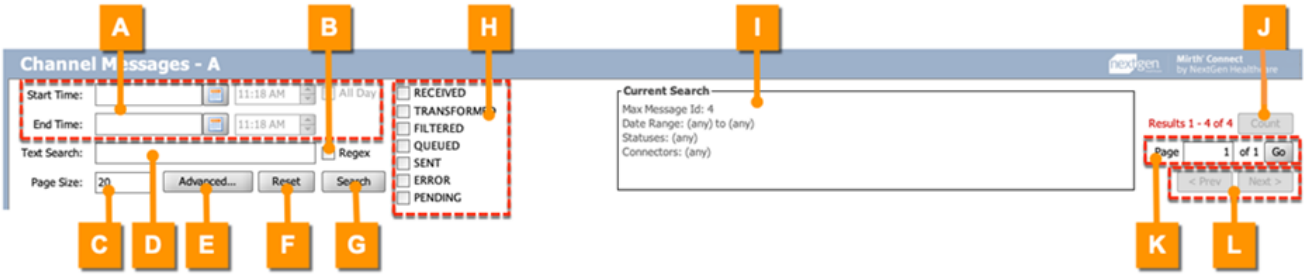
The **Message Browser** supports a wide range of searches, from a general search using a few filters, to finely grained searches using many filters and including the Advanced… window. The Advanced window enables you to enter various IDs, content, and metadata filters to fine-tune your searches



Message Search Options

Because messages have so many distinguishing elements, Mirth® Connect provides numerous filters to help you configure well-targeted searches.

To access these filters, on the page, select a channel, and in the panel, select the **View Messages** task.



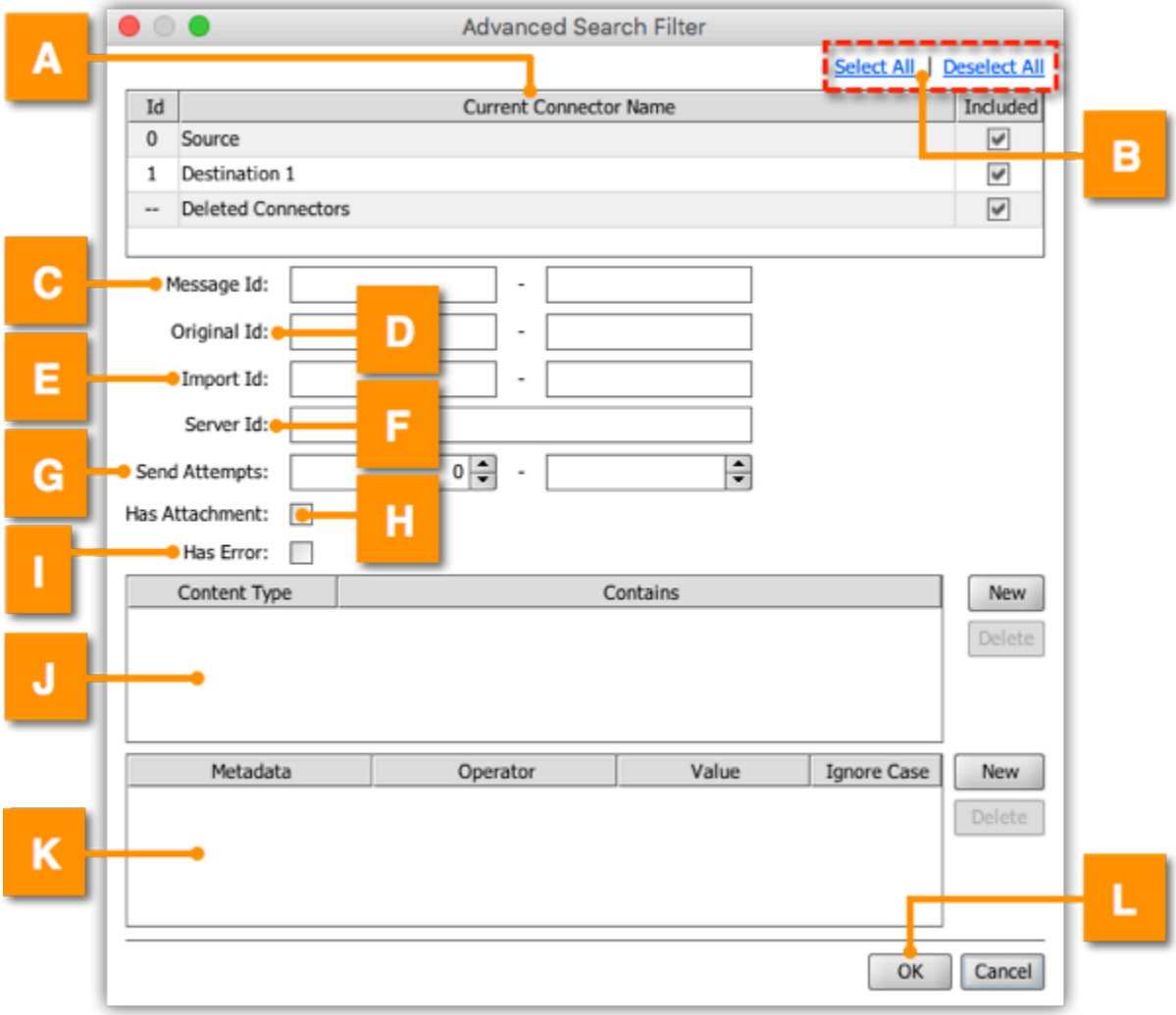
Item	Name	Description
A	Start/End Time	Use the **Calendar** icons to populate these fields with the start/end dates of the desired messages. • Use the **Hour** fields to restrict the search to a range of hours. • Check the **All Day** box to include messages that started/ended at all hours of the day.
B	Regex	Searches all textual message content that matches the regular expression pattern given in the **Text Search** field. **Note:** • Regex matching can be a very costly operation and should be used with caution, especially with a large number of messages. • Message content that is encrypted by this channel is not searchable. • The Regex option is only supported on PostgreSQL, Oracle, and MySQL databases.
C	Page Size	Enter the maximum amount of messages to appear in the Metadata Table at once. The default value is 20, but you can change the default in the Administrator Settings Tab.

Item	Name	Description
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 select the **Search** button, the **Select an Option** window appears, informing you that *Text searching may take a long time, depending on the amount of messages being searched*, then asks *Are you sure you want to proceed?* select **Yes** or **No** as desired. **Note:** You can disable / enable the search confirmation window in the Administrator Settings Tab.
E	Advanced…	Reveals the Advanced Search Filter window consisting of more specific search-filter options. • If any advanced search options have been set, this button appears 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 RECEIVED, TRANSFORMED, FILTERED, QUEUED, SENT, ERROR and/or PENDING 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:** 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.
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 select the **Count** button, enter a value (coinciding with the page range) in this

Item	Name	Description
		field, and select the **Go** button to display the desired page.
L	< Prev / Next >	(Multi-page lists) Select the **< Prev** button to go back one page or the **Next >** button to advance one page.

Advanced Search Filter

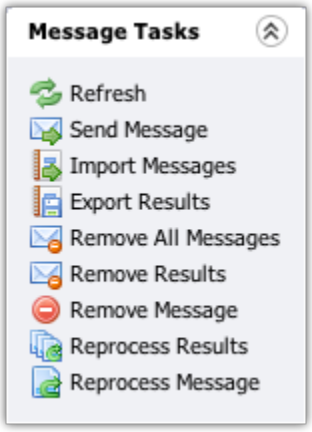
Select the **Advanced...** button in the message browser search pane to view the **Advanced Search Filter**. (See Option E in Message Search Options *(on page 125)*)



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 have metadata IDs greater than zero. • The **Current Connector Name** column shows the current name of each connector. Even if a destination is renamed, the metadata ID remains static, and is used for the search. • The **Deleted Connectors** row refers to historical connector messages associated with destinations that were deleted.
B	Select/Deselect All	Select this checkbox to select/deselect all connectors in the **Connectors Table**.
C	Message Id	Enter 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**) 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.
D	Original Id	
E	Import Id	
F	Server Id	Enter the ID of the server that processed the message.
G	Send Attempts	Enter the min (first field) / max (second field) values to find messages with the number of send attempts in that range.
H	Has Attachment	Check this checkbox to only include messages with one or more attachments in the search.
I	Has Error	Check this checkbox to include only messages with one or more errors *(on page 116)* in the search.
J	Content Search Table	• Select the **New** button to add a search filter, then select the row in the **Content**

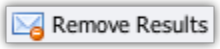
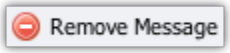
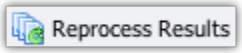
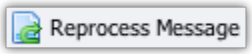
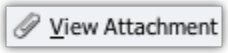
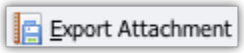
Item	Name	Description
		Type column, and select a menu option. • Double-click the row in the **Contains** column, and enter your search string. When multiple rows are present in this table, only connector messages that match **all** entries will be included in the search. The available content types include message content *(on page 111)*, variable map content *(on page 115)*, and error content *(on page 116)*.
K	Metadata	• Select the **New** button to add a search filter, then select the row in the **Metadata** column, and select a custom-metadata menu option. • Select the row in the **Operator** column, and select a search operator (=, *CONTAINS*, *STARTS/ENDS WITH*), which determines the metadata to search for. • Double-click the row in the **Value** column and enter your search string. • **Ignore Case**: If checked, the search is not case-sensitive. **Note:** When multiple rows are present in this table, only connector messages that match **all** entries will be included in the search.
L	OK / Cancel	Select the **OK** button to save or the **Cancel** button to discard your changes.

Message Browser Tasks



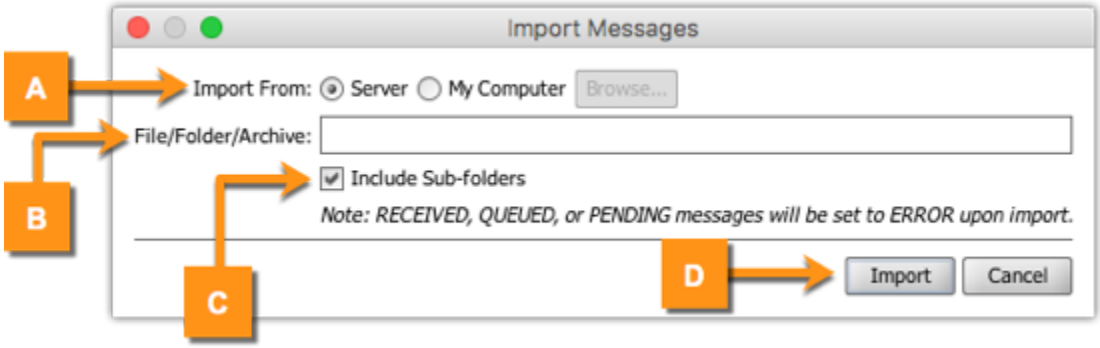
The following context-specific tasks are available from the **Message Browser View**:

Task Icon	Task Name	Description
Refresh	Refresh	Updates messages within the current search page. Note that this is **different** from selecting the **Search** button. The Search operation performs a completely new search without a Max Message Id and resets the count and pagination options. 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.
Send Message	Send Message	Sends a message to the current channel. For additional information, see Send Message *(on page 93)*.
Import Messages	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. For additional information, see Import Messages *(on page 131)*.
Export Results	Export Results	Exports all messages that match the current search criteria. Note that this will include messages on **all** pages, not only the messages in the current page. To see how many messages will be exported, select the Count *(on page 124)* button. For additional information, see Export Results *(on page 133)*.
Remove All Messages	Remove All Messages	Removes **all** messages and attachments stored for the current channel. For additional information, see Remove All Messages *(on page 93)*.

Task Icon	Task Name	Description
	Remove Results	Removes all messages that match the current search criteria. Note that this will include messages on **all** pages, not only the messages in the current page. To see how many messages will be removed, select the Count *(on page 124)* button. For additional information, see Remove Results *(on page 135)*
	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	Reprocesses all messages that match the current search criteria. Note that this will include messages on **all** pages, not only the messages in the current page. To see how many messages will be reprocessed, select the Count *(on page 124)* button. For additional information, see Reprocess Results *(on page 136)*
	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. For additional information, see Reprocess Message *(on page 137)*
	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 *(on page 118)*. For additional information, see Viewing Attachments *(on page 118)*.
	Export Attachment	Exports the attachment data to disk. This task is only visible when an attachment is selected in the Attachments Tab *(on page 118)*. For additional information, see Export Attachment *(on page 138)*

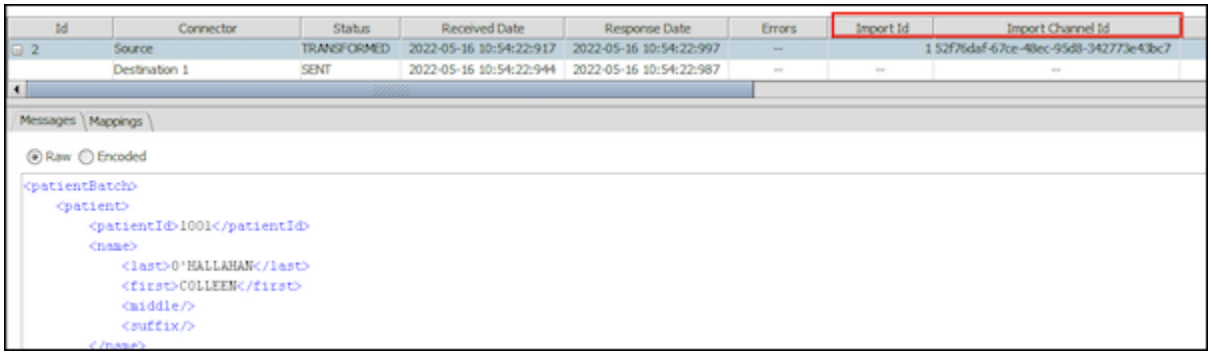
Import Messages

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 are imported from the server file system. • If **My Computer** is chosen, messages from the machine running the Administrator are imported into the server. • Select **Browse** 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 are searched recursively for messages to import.
D	Import	Select **Import** to perform the import operation.

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



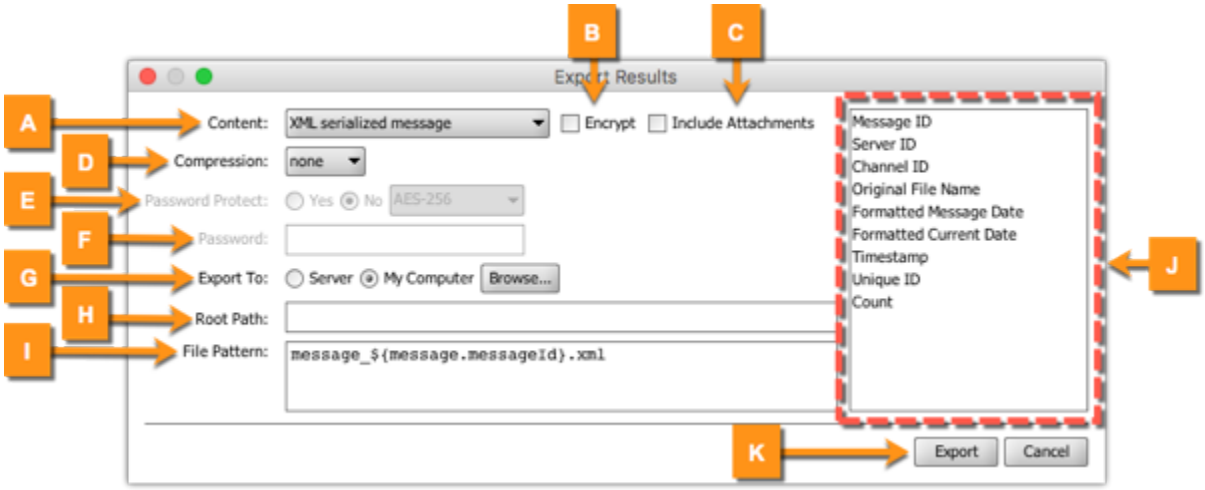
A unique identifier for each imported message appears in the **Import Channel Id** column. This alphanumeric code lets you readily identify messages that have been imported into the channel.

- 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 selected using the **Column Options** icon at the rightmost side of the Message Lists column header. For additional information, see Working With Tables *(on page 80)*.

Export Results

When you select the **Export Results** task, an options window opens. Note that all messages matching the current search criteria are exported, not only the ones shown on the current page.



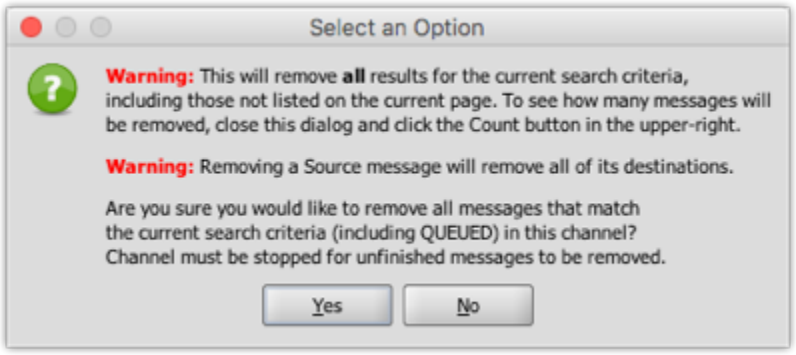
Item	Name	Description
A	Content	The type of content used to export from the messages. Note that an XML serialized message is the only type that can be re-imported, and it includes all content across the message in a single XML file.
B	Encrypt	If checked, the exported message content is encrypted with the server's encryption key.
C	Include Attachments	If checked and the content type is set to XML serialized message, the exported file contains all attachments

Item	Name	Description
		associated with the message.
D	Compression	When compression is enabled, the files and folders created according to the file pattern are made into a compressed file in the root path. The following compression types are supported: • ZIP • tar.gz • tar.bz2
E	Password Protect	This option is only available if ZIP compression is enabled. • Select **No** to create a non-password protected ZIP file. • Select **Yes** to create a password protected ZIP file. 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 is used.
G	Export To	Exports messages directly to the server running Mirth® Connect, or pulls messages from the server and stores them locally on the machine where you are running the Administrator.
H	Root Path	The root path used to store the exported files and folders or compressed file. • If messages are exported directly to the server, relative paths are resolved against the Mirth® Connect Server home directory.
I	File Pattern	The file or folder pattern used to write the exported message files. The file pattern may use variables from the list on the right side of the window.

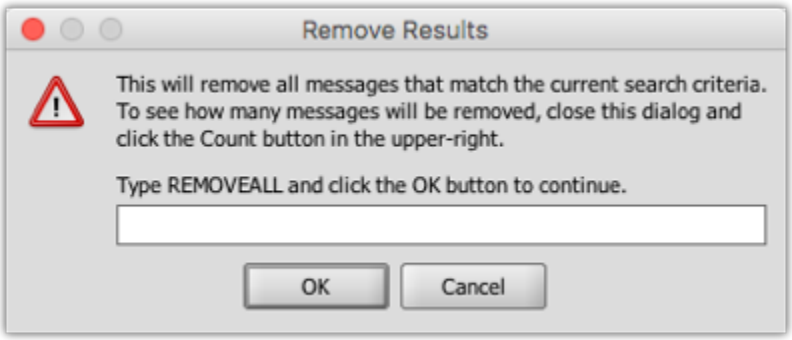
Item	Name	Description
J	Variables	Contains several common variables to aid in loading the File Pattern.
K	Export	Select **Export** to perform the export operation. • All messages matching the current search criteria are exported, not only the ones shown on the current page. • Starting in Mirth® Connect version 4.4.2, when you export channel messages, an additional EXPORTREADME.txt file is also exported. The EXPORTREADME.txt file contains a link to a website with file format specifications.

Remove Results

When you select the **Remove Results** task, a confirmation dialog pops up. Note that **all** messages and associated attachments matching the current search criteria are removed, not only the ones shown on the current page. Removing a Source Connector message also automatically removes all of its associated destination connectors.



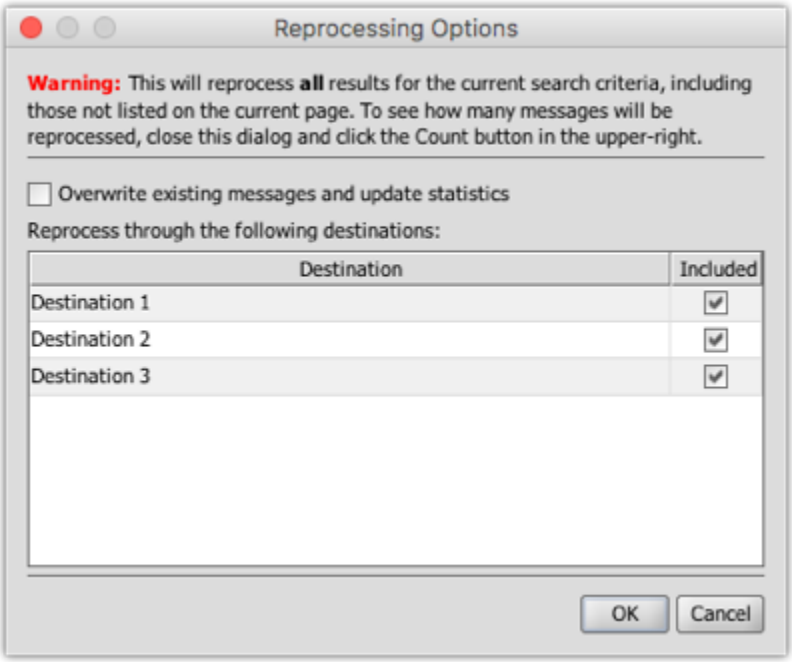
After selecting **Yes** to continue, the following additional dialog displays:



This is an additional layer of security to ensure that you do not remove message data accidentally. To proceed you must type in **REMOVEALL** in all capital letters and select **OK**. Note that, if you find this additional layer of security tedious, it can be disabled in the Administrator Settings Tab *(on page 174)*.

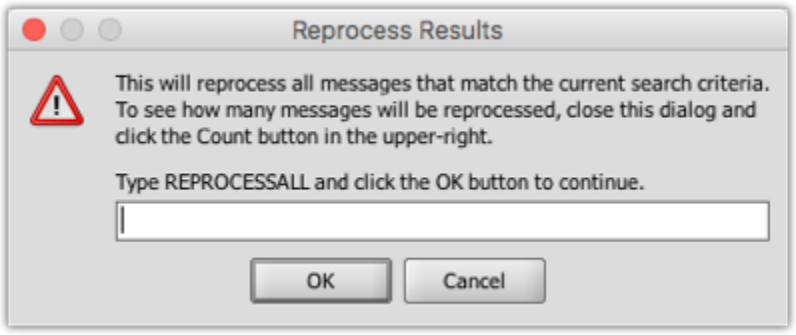
Reprocess Results

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



Other than the additional warning, the options here are the same as in the Reprocess Message *(on page 137)* task.

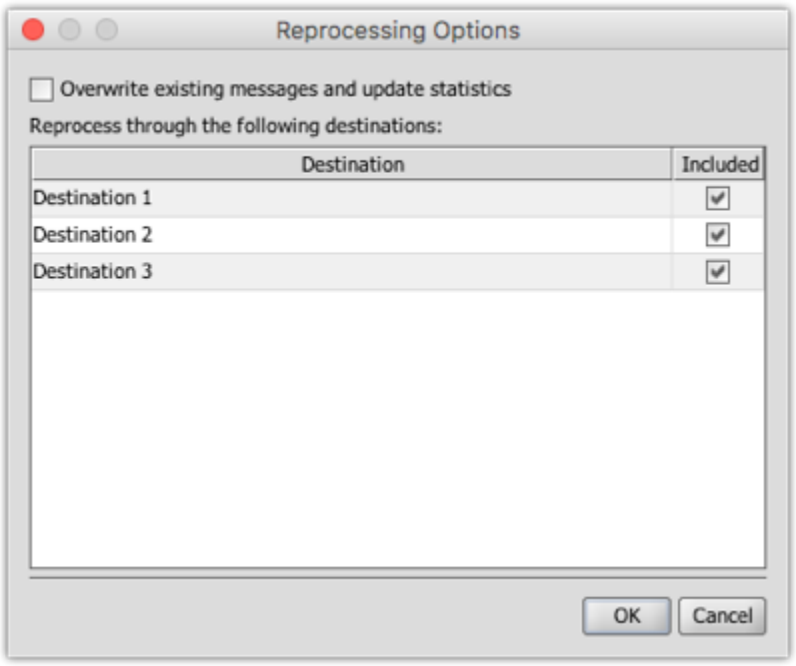
After selecting **Yes** to continue, the following additional dialog displays:



This is an additional layer of security to ensure that you do not reprocess a large amount of messages accidentally. To proceed you must type in **REPROCESSALL** in all capital letters and select **OK**. Note that, if you find this additional layer of security tedious, it can be disabled in the Administrator Settings Tab *(on page 174)*.

Reprocess Message

When you select 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.

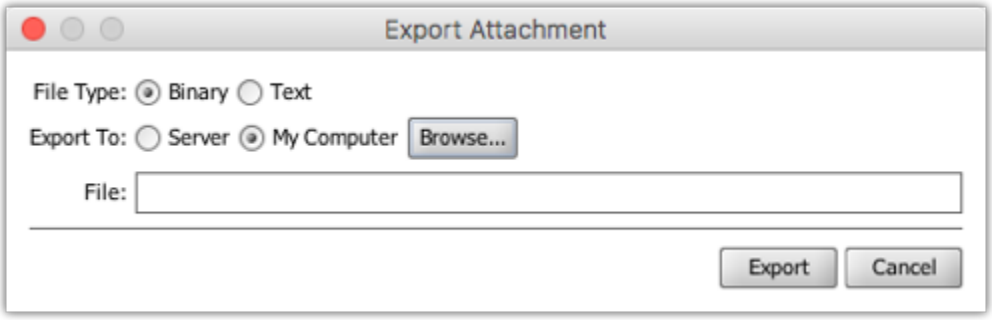


When a message is reprocessed, two new Source Map *(on page 449)* variables are injected into the message:

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

Export Attachment

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

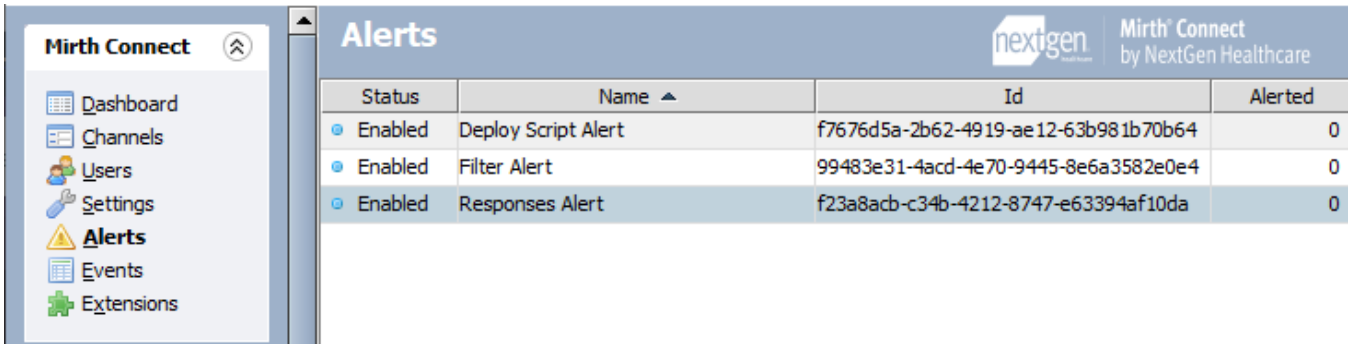


Item	Description
File Type	• Choose **Binary** to write the raw bytes of the attachment out to file. • Choose **Text** to write the Base64 encoded representation of the attachment out to the file.
Export To	• Choose **Server** to export the attachment directly to the machine running the Mirth® Connect 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

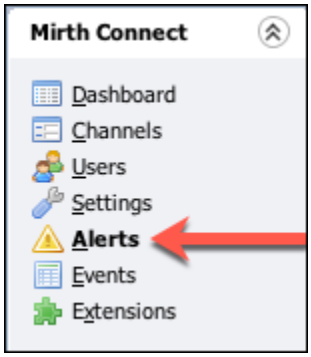
An **alert** is a process that listens for certain types of events and creates triggers based on configurable settings. From these triggers you can take various actions, like dispatching an email to a user or specific address, or sending a message to a channel. Mirth® Connect comes with a built-in, error-based, 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.



Navigation

Select the **Alerts** link in the **Mirth Connect** task panel on the upper-left side of the window.



This section is separated into the following topics:

- Alerts Table *(on page 140)*
- Alerts Tasks *(on page 140)*

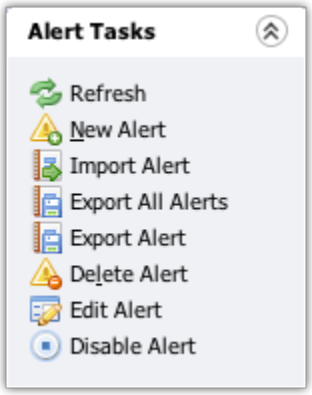
Alerts Table

This is the main section of the Alerts View that shows the current status and action counters for all of your configured alerts. For general information about working with tables in Mirth Connect, see, Working With Tables *(on page 79)*.

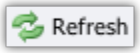
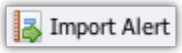
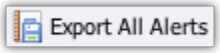
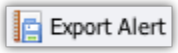
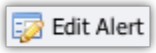
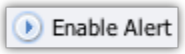
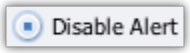
Alerts Table Columns

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

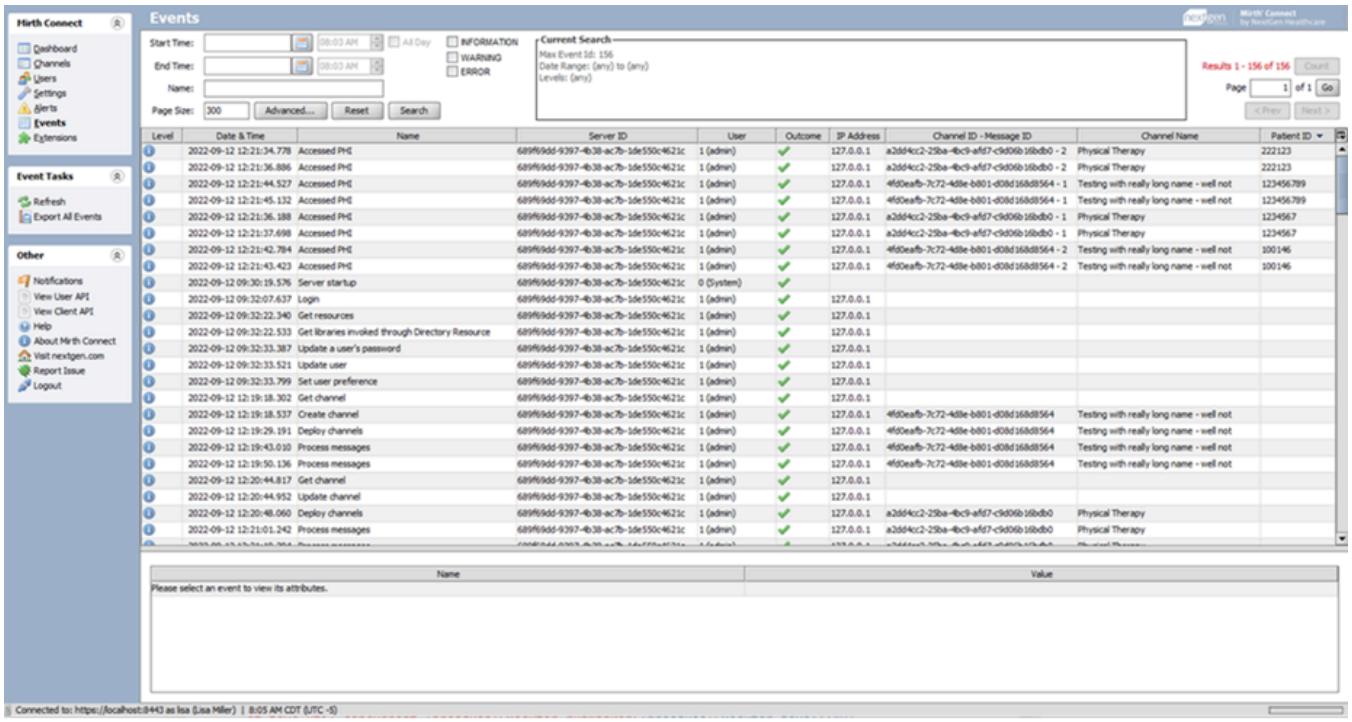


The following context-specific tasks are available from the Alerts View:

Task Icon	Task Name	Description
	Refresh	Updates the Alerts Table. Note that the Alerts view automatically refreshes at an interval defined in the Administrator Settings *(on page 174)*.
	New Alert	Creates a new alert and enters the Edit Alert View *(on page 313)*.
	Import Alert	Imports an alert from an XML file on disk.
	Export All Alerts	Exports all alerts to separate XML files on disk.
	Export Alert	Exports a single alert to an XML file on disk.
	Delete Alert	Removes the alert from the server.
	Edit Alert	Enters the Edit Alert View *(on page 313)* for the selected alert.
	Enable Alert	Activates the alert so that it will begin receiving and acting upon events.
	Disable Alert	Deactivates the alert so that it will stop receiving and acting upon events.

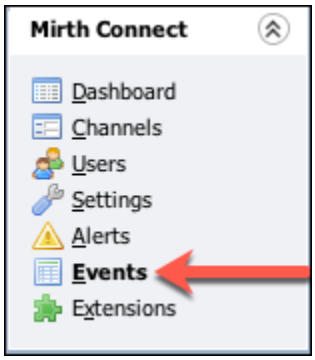
Events View

The Events browser enables you to view and search all user and system events that have occurred on your server. This includes any user actions like logging on and modifying or deploying channels. It also includes system events like startup and shutdown, and the Data Pruner.



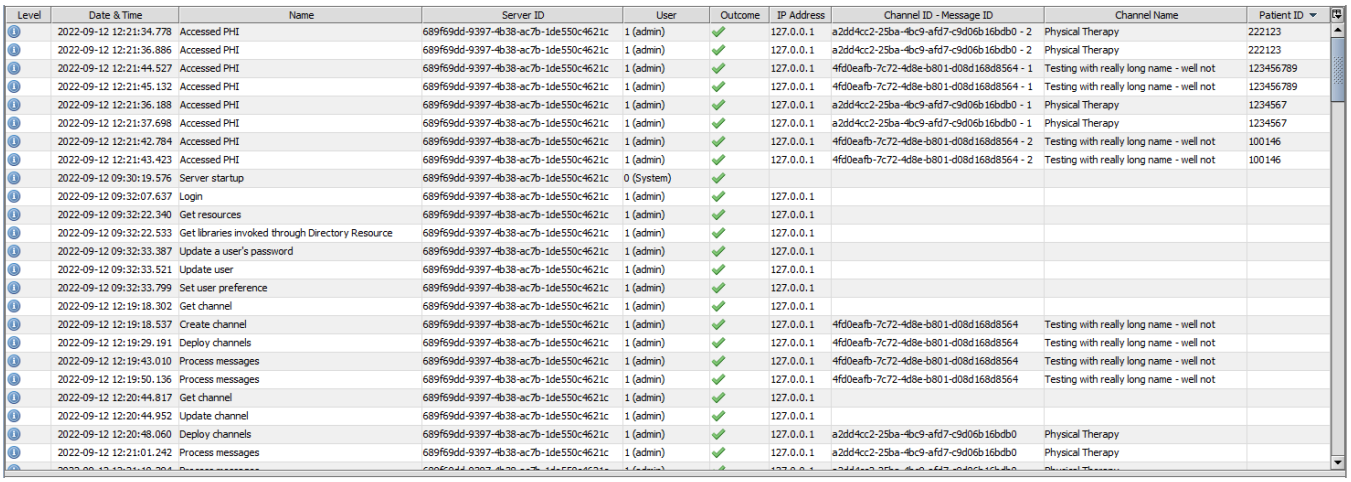
Navigation

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



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 *(on page 145)*) . When you first enter the Event Browser, by default the latest 100 events appear. For general information about working with tables in Mirth® Connect, see Working With Tables *(on page 80)*.



Level	Date & Time	Name	Server ID	User	Outcome	IP Address	Channel ID - Message ID	Channel Name	Patient ID
	2022-09-12 12:21:34.778	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0 - 2	Physical Therapy	222123
	2022-09-12 12:21:36.886	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0 - 2	Physical Therapy	222123
	2022-09-12 12:21:44.527	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564 - 1	Testing with really long name - well not	123456789
	2022-09-12 12:21:45.132	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564 - 1	Testing with really long name - well not	123456789
	2022-09-12 12:21:36.188	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0 - 1	Physical Therapy	1234567
	2022-09-12 12:21:37.698	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0 - 1	Physical Therapy	1234567
	2022-09-12 12:21:42.784	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564 - 2	Testing with really long name - well not	100146
	2022-09-12 12:21:43.423	Accessed PHI	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564 - 2	Testing with really long name - well not	100146
	2022-09-12 09:30:19.576	Server startup	689f69dd-9397-4b38-ac7b-1de550c4621c	0 (System)	✔				
	2022-09-12 09:32:07.637	Login	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 09:32:22.340	Get resources	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 09:32:22.533	Get libraries invoked through Directory Resource	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 09:32:33.387	Update a user's password	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 09:32:33.521	Update user	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 09:32:33.799	Set user preference	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 12:19:18.302	Get channel	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 12:19:18.537	Create channel	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564	Testing with really long name - well not	
	2022-09-12 12:19:29.191	Deploy channels	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564	Testing with really long name - well not	
	2022-09-12 12:19:43.010	Process messages	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564	Testing with really long name - well not	
	2022-09-12 12:19:50.136	Process messages	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	4fd0eafb-7c72-4d8e-b801-d08d168d8564	Testing with really long name - well not	
	2022-09-12 12:20:44.817	Get channel	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 12:20:44.952	Update channel	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1			
	2022-09-12 12:20:48.060	Deploy channels	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0	Physical Therapy	
	2022-09-12 12:21:01.242	Process messages	689f69dd-9397-4b38-ac7b-1de550c4621c	1 (admin)	✔	127.0.0.1	a2dd4cc2-25ba-4bc9-afd7-c9d06b16bdb0	Physical Therapy	

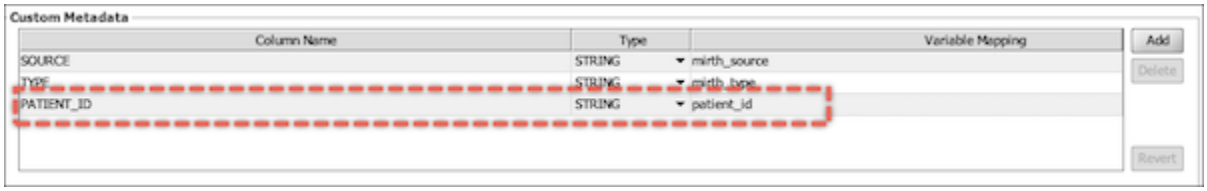
Metadata Table Columns

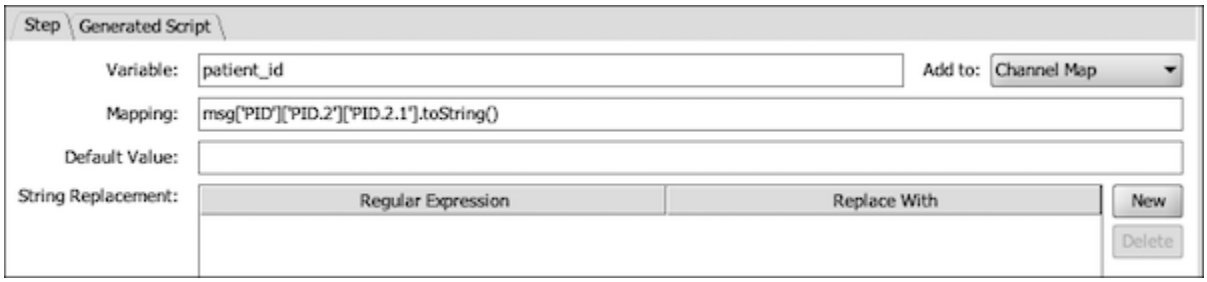
Column	Description
ID	The unique ID of the event in the database. This column is hidden by default. For additional information, see Working With Tables *(on page 80)*.
Level	An icon indicating the event's severity level (**Information**, **Warning**, or **Error**) .
Date & Time	The date and time the 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 Address	For user requests, the IP address of the originating client.
Channel ID - Message ID	If the event occurs in a specific channel, the channel ID will be logged. If the event occurs due to a specific message, a dash with the message ID for

Column	Description
	that channel will be appended to the channel ID. If the event didn't occur in a specific channel, it will be blank.
Channel Name	If the event occurs in a specific channel, the channel name will be logged with the event. If the event didn't occur in a specific channel, it will be blank.
Patient ID	A special column for **Accessed PHI** events that indicates the ID of a patient whose PHI (Protected Health Information) has been accessed from the Message Browser. See PHI Events *(on page 144)* for more information.

PHI Events

There are two PHI-related events that can exist in the Events table. For these events to appear, you must configure at least one channel with a custom metadata column *(on page 230)* named **PATIENT_ID** and populate that metadata column within the channel. For example, a channel that receives HL7v2 messages can have a transformer step that populates **PATIENT_ID** with the value of **PID.2.1**.

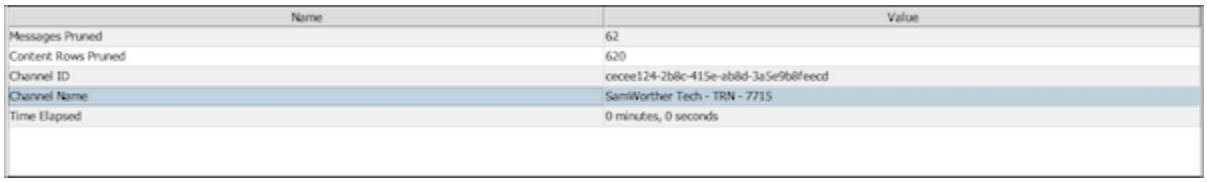




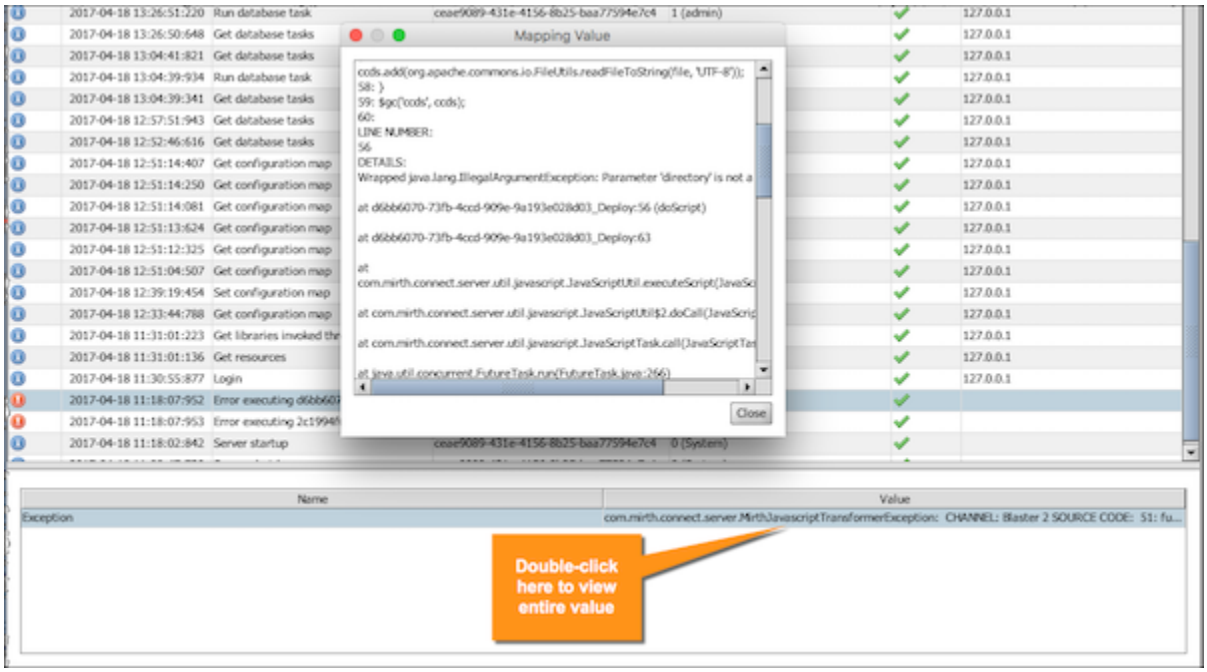
Once you have configured a channel with this information, whenever a user queries that channel's messages from the Message Browser or the REST API, a **Queried PHI** event is created. Similarly, whenever a user views one of that channel's messages that has a **PATIENT_ID**, an **Accessed PHI** event is created.

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.



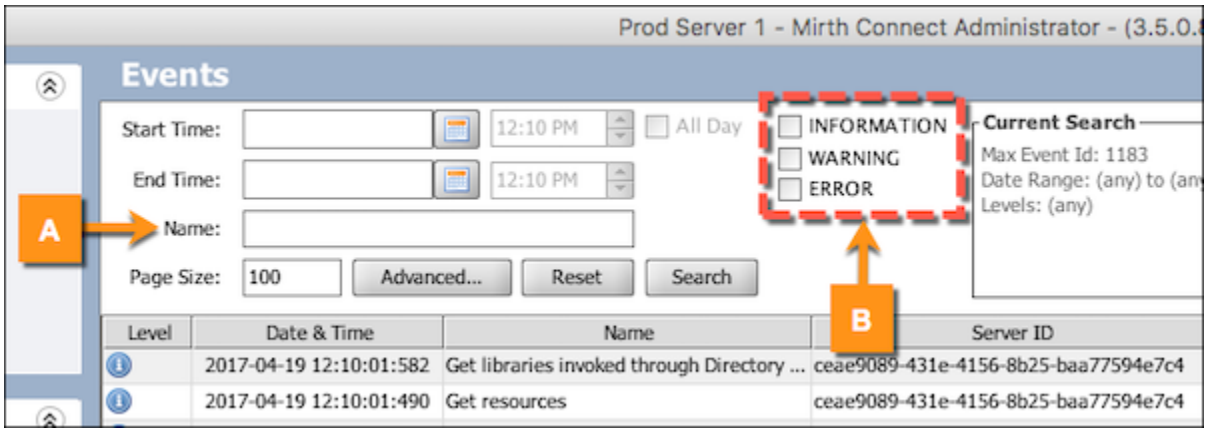
If an attribute value is too long and is truncated in the table, you can double-click the table row to display the value in a separate dialog:



Searching Events

The **Event Browser** search capabilities are very similar to the Message Browser. For information on many of these shared components, see Search Messages *(on page 124)*.

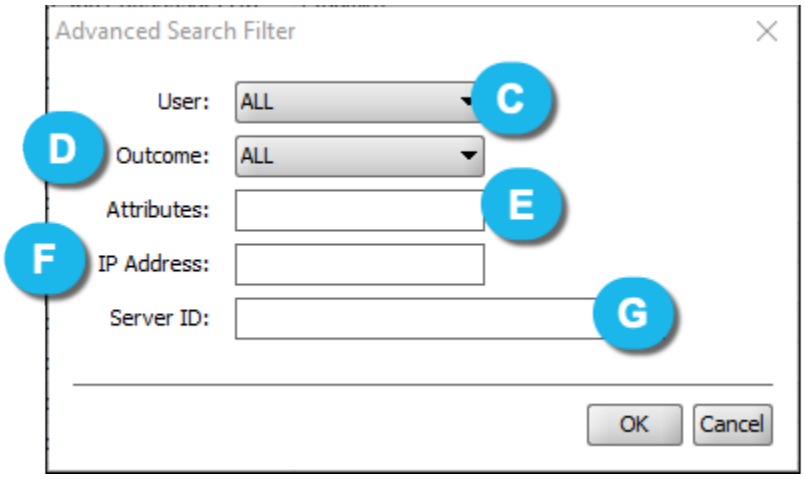
The following search options are unique to the Event Browser:



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 **INFORMATION**, **WARNING**, and/or **ERROR** levels.

Advanced Search Filter

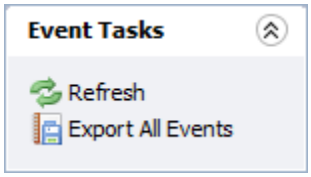
Select on the **Advanced..** button in the event browser search pane to open the Advanced Search Filter.



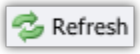
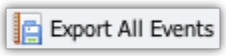
Item	Name	Description
C	User	The user associated with the event. The **System** user is reserved for events where no user is associated, such as startup / shutdown or the Data Pruner.

Item	Name	Description
D	Outcome	For user requests and server tasks, Outcome indicates whether the action was successful or not.
E	Attribute	To search for a specific attribute, such as patient_id.
F	IP Address	For user requests, the IP address of the originating client.
G	Server ID	The unique ID of the server from which the event was logged.

Event Tasks

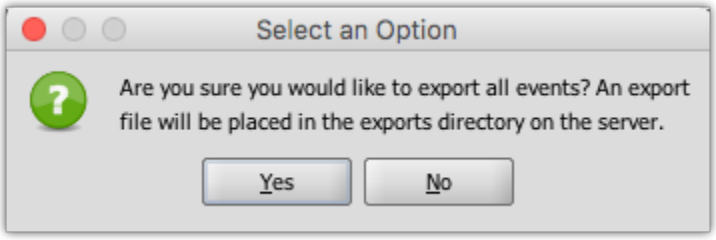


The following context-specific tasks are available from the Events View:

Task Icon	Task Name	Description
	Refresh	Updates events within the current search page. **Note:** Note that this is **different** from selecting the **Search** button. The Search operation performs a completely new search without a Max Event Id and resets the count and pagination options. 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.
	Export All Events	Exports all events to a file in the Application Data Directory *(on page 478)*. For additional information, see Export All Events *(on page 147)*

Export All Events

When you select this task, a confirmation dialog appears:



Select **Yes** to continue. A comma-separated value (CSV) file will be written into an "exports" folder inside your Application Data Directory *(on page 478)*.

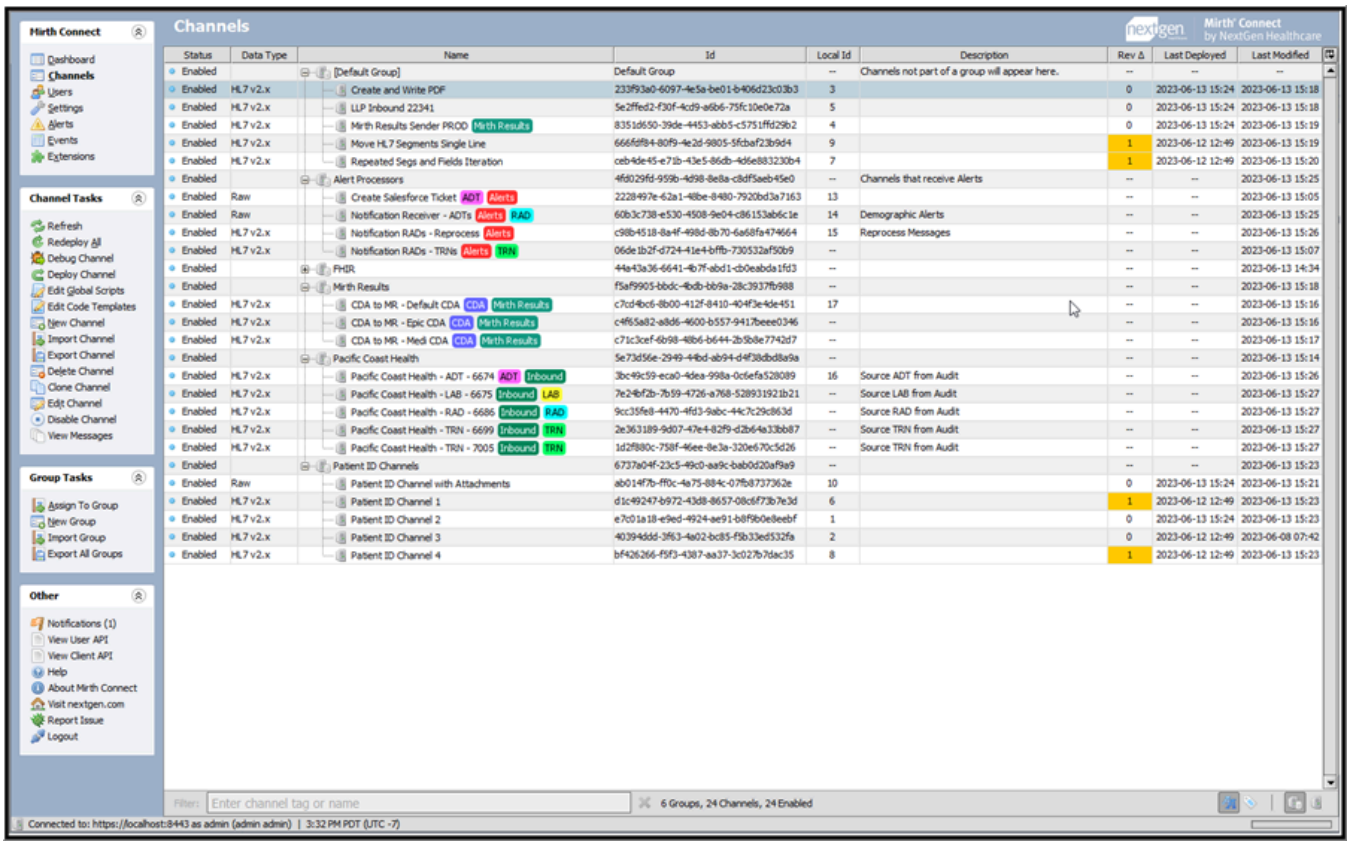
Management Views

This section is separated into the following topics:

- Channels View *(on page 148)*
- Users View *(on page 160)*
- Settings View *(on page 166)*
- Extensions View *(on page 196)*

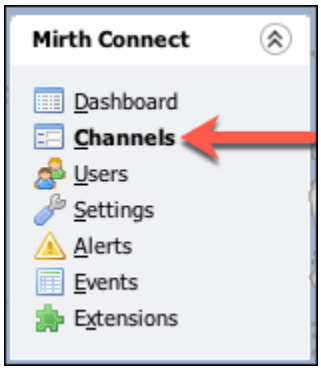
Channels View

The Channels View is the main management window for all channels configured on your Mirth® Connect 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.



Navigation

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



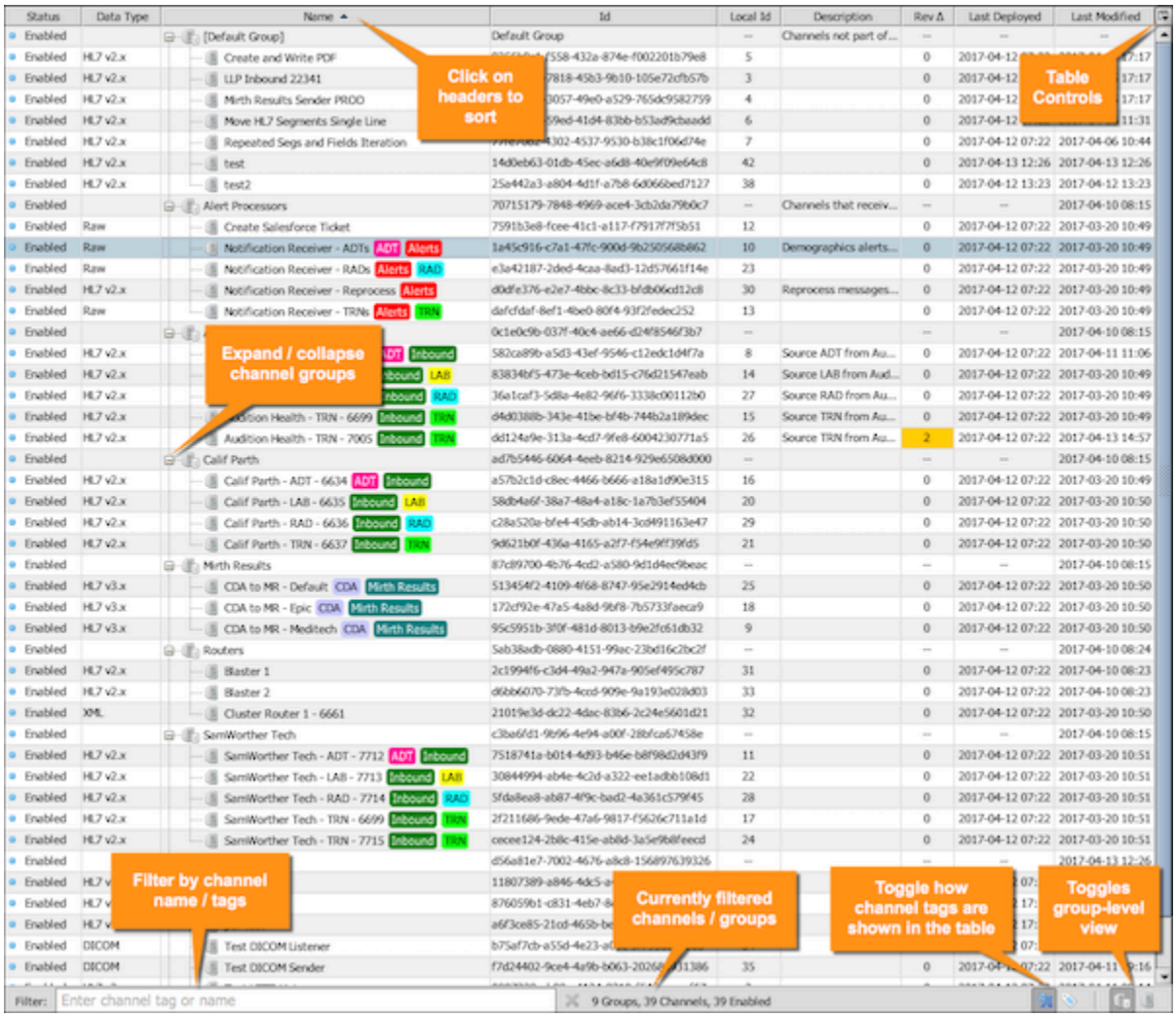
This section is separated into the following topics:

- Channel Table *(on page 150)*

- Channel Tasks *(on page 154)*
- Group Tasks *(on page 158)*

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 Mirth Connect, see Working With Tables *(on page 79)*.



Channel Table Columns

Column	Description
Status	Indicates whether the channel is enabled or disabled. A channel may only be deployed once it is enabled. When the Mirth Connect server starts up, all enabled channels are automatically deployed.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.
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 **not** 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 Δ	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 Δ** = Channel Revision - Deployed Revision
Last Deployed	The date and time this channel was last deployed. This value will be highlighted if it is within the last two minutes.
Last Modified	The date and 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 *(on page 86)*.

Change How Tags Display

For information on how to change how tags display in the channel table, see Change How Tags Are Displayed *(on page 86)*.

Filter By Channel Name or Tag

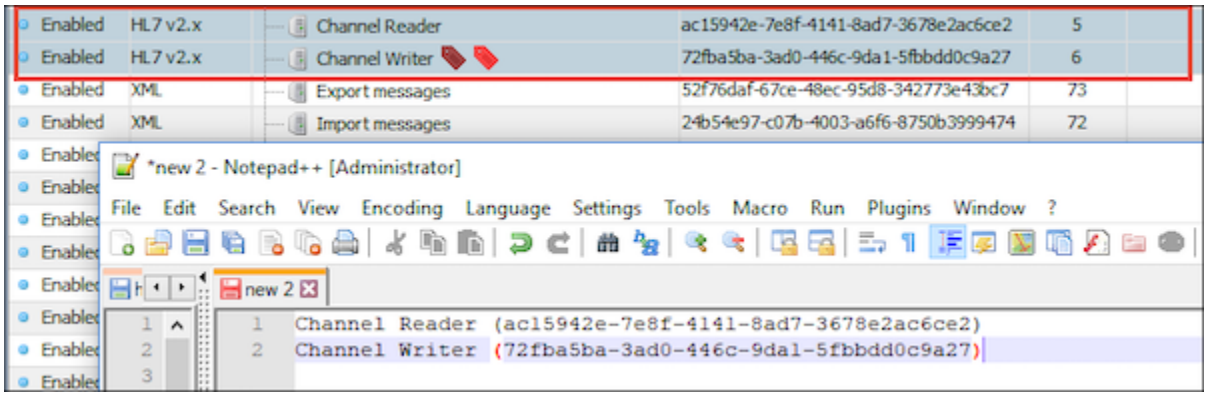
For information on how to filter the channel table down to specific channels based on names or tags, see

Filter By Channel Name or Tag *(on page 98)*.

Use Drag and Drop

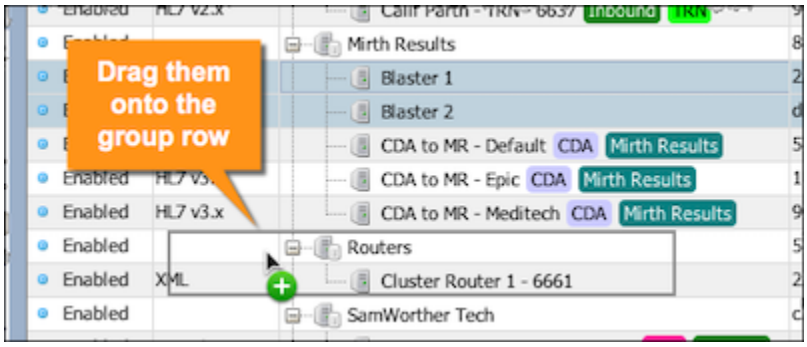
Get the Channel Name/ID

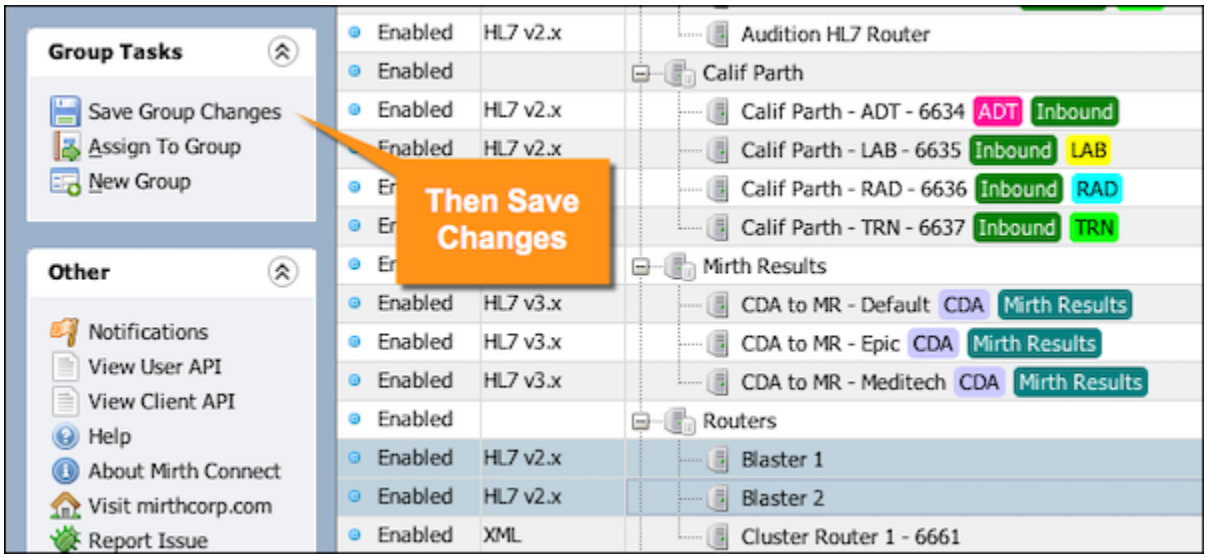
You can quickly copy channel names/IDs by selecting rows in the channel table and dragging them into the text editor of your choice.



Assign Channels to a Group

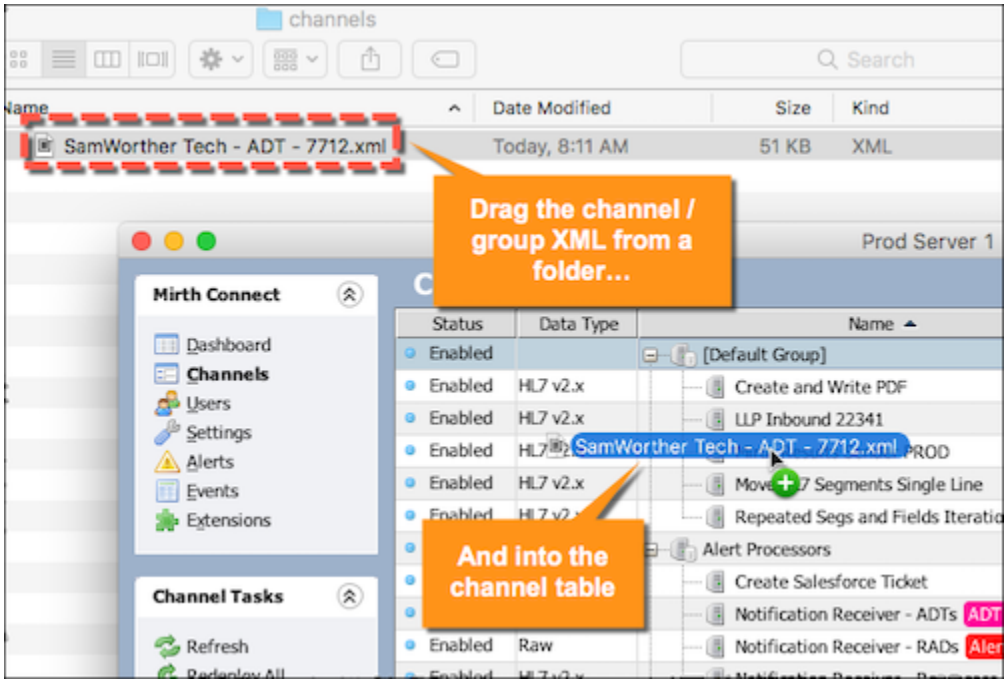
To move channels to a specific group, select the channel rows in the table and drag them onto the group row:





Import Channels/Groups Using Drag-and-Drop

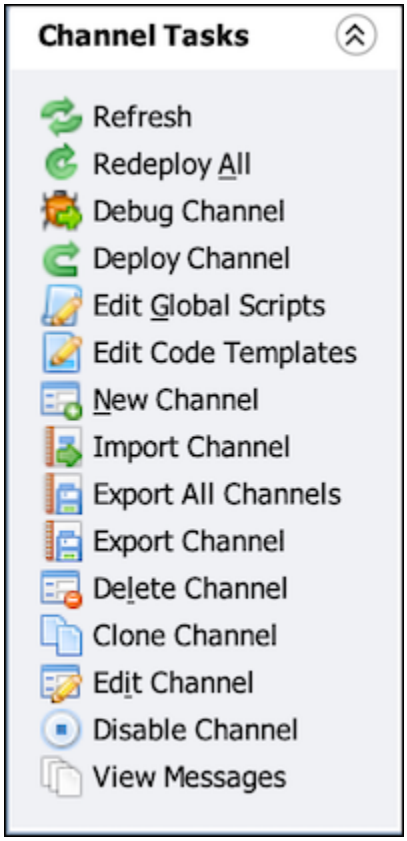
Importing channels/groups may be done using the corresponding **Import** task (for additional information see Channel Tasks *(on page 154)* and Group Tasks *(on page 158)*) , or by simply dragging the XML files you wish to import from a folder into the channel table directly:



For channel files, the Import Channel *(on page 154)* workflow applies. For group files, the Import Group *(on page 158)* 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	Refresh	Updates the channel table.
Redeploy All	Redeploy All	Undeploys all channels, then deploys all currently enabled channels.
Debug Channel	Debug Channel	Displays the Debug Channel *(on page 157)* dialog that allows you to deploy the currently selected channel while debugging specific scripts within the channel.

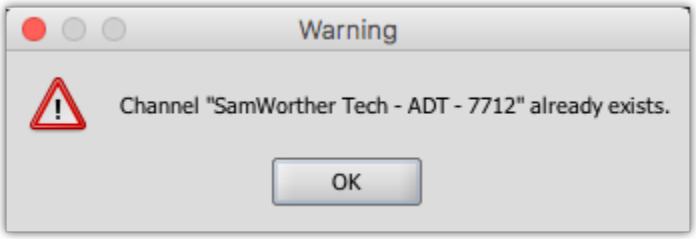
Task Icon	Task Name	Description
	Deploy Channel	Deploys the currently selected channel(s). If a group is selected, deploys all channels underneath that group. Note that the Channel Dependencies *(on page 215)* workflow may apply to this task.
	Edit Global Scripts	Enters the Edit Global Scripts View *(on page 299)*.
	Edit Code Templates	Enters the Edit Channel View *(on page 203)*.
	New Channel	Creates a newchannel and enters the Edit Channel View *(on page 203)*. The channel is **not yet saved**.
	Import Channel	Imports a channel from an XML file and enters the Edit Channel View *(on page 203)*. The channel is **not yet saved**. For additional information, see Import Channel *(on page 156)*.
	Export All Channels	Exports all of the channels to their own respective XML files. For additional information, see Export Channel *(on page 157)*.
	Export Channel	Exports the selected channel(s) to their own respective XML files. For additional information, see Export Channel *(on page 157)*.
	Delete Channel	Removes the selected channel(s) from the server. **All message / attachment data is also deleted.**
	Clone Channel	Copies the channel and enters the Edit Channel View *(on page 203)*. The channel is **not yet saved**. Note that you will be prompted to give the channel a new unique name first.
	Edit Channel	Enters the Edit Channel View *(on page 203)* for the selected channel. This can also be accessed by double-clicking the channel row.
	Enable Channel	Mark this channel as ready to be deployed. Enabled channels are automatically deployed when the server starts up or when the **Redeploy All** task is performed.
	Disable Channel	Marks this channel as not ready to be deployed. The channel may not be deployed until it is re-enabled. However, if the channel was *already* deployed, then disabling it will not automatically undeploy it.

Task Icon	Task Name	Description
View Messages	View Messages	Enters the Message Browser View *(on page 103)* for the selected channel.
View History	View History	Opens the Channel History *(on page 587)*dialog, which allows you to view the history of the selected channel. This task is only available when the Channel History extension has been installed.

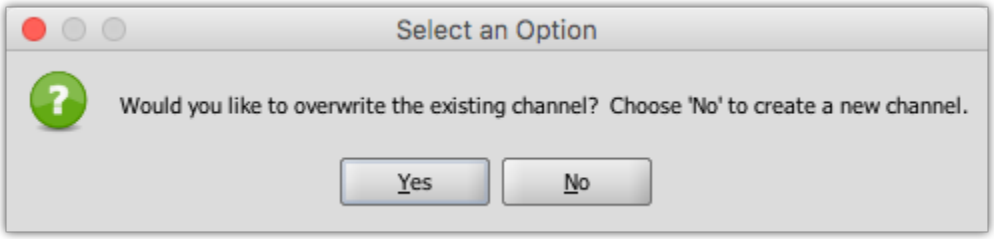
Import Channel

When importing channels, you have the option to include all code template libraries *(on page 301)* 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 *(on page 310)*

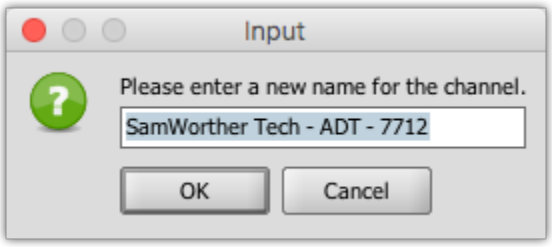
If the channel you are importing has the same name as an existing channel on the server, a warning dialog will be shown:



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

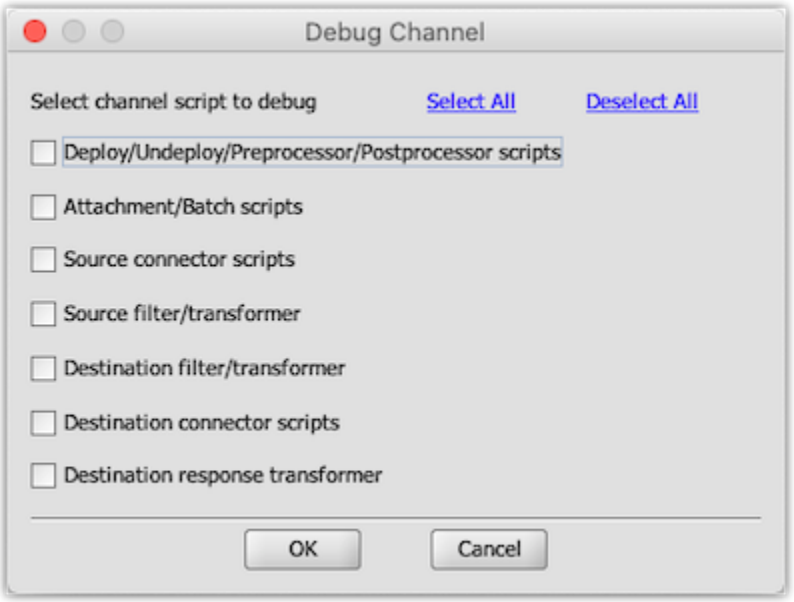


If **No** is chosen, you will then be prompted to enter a new unique name for the imported channel:



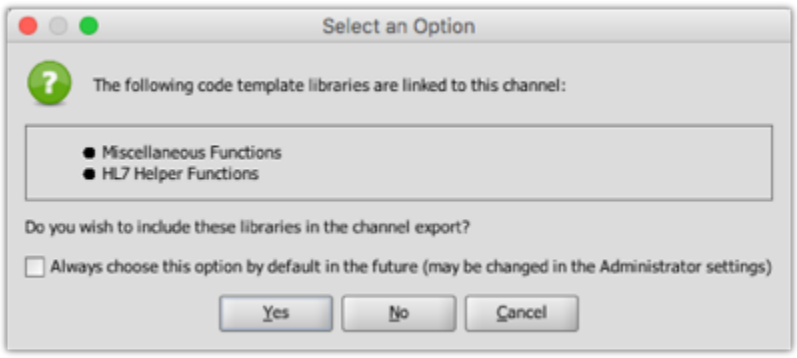
Debug Channel

The **Debug Channel** dialog allows you to deploy the currently selected channel while debugging specific scripts within the channel. Select any number of scripts you would like to debug, and then select **OK** to deploy the channel and begin debugging.



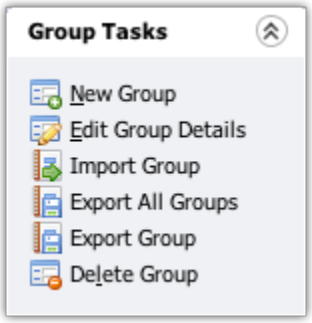
Export Channel

When exporting channels, you have the option to include all code template libraries *(on page 301)* associated with the channel as well.



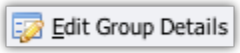
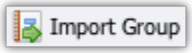
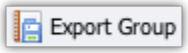
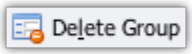
- If **Yes** is chosen, the code template libraries shown in the dialog will be included and embedded into the channel export XML
- If you want to automatically make the same choice (Yes or No) for all future exports, you can check the "**Always choose** "check box. You can change this again later in the Administrator Settings Tab *(on page 174)*.

Group Tasks



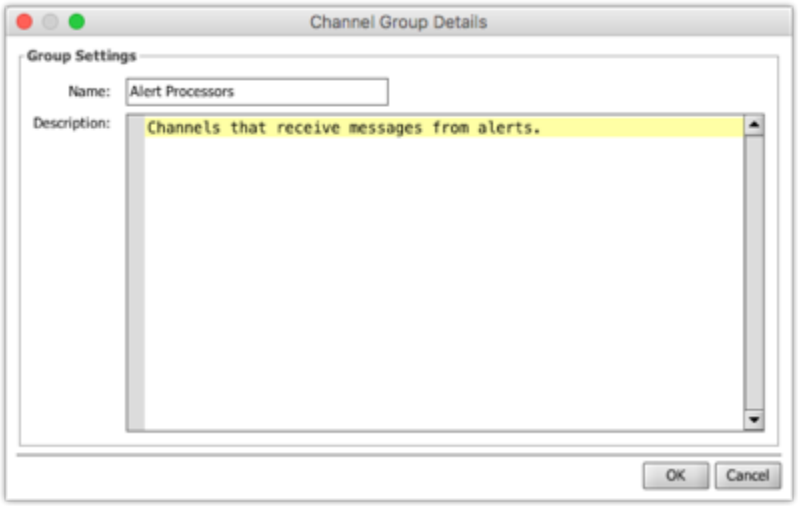
The following context-specific tasks for channel groups are available from the Channels View.

Task Icon	Task Name	Description
Save Group Changes	Save Group Changes	Saves all changes made to channel groups and refreshes the Channels view.
Assign To Group	Assign To Group	Moves the selected channel(s) into a specific group.
New Group	New Group	Creates a new channel group and adds it to the table. The user is prompted for the group name and

Task Icon	Task Name	Description
		description as described in the Edit Group Details *(on page 159)* task.This task appears after the channels view fully loads.
	Edit Group Details	Allows the user to edit the group name and description. This can also be accessed by double-clicking the group row. For additional information, see Edit Group Details *(on page 159)*.
	Import Group	Imports a channel group from an XML file. For additional information, see Import Group *(on page 160)*.
	Export All Groups	Exports all channel groups in the table to their respective XML files. For additional information, see Export Group *(on page 160)*.
	Export Group	Exports the selected channel group(s) to their respective XML files. For additional information, see Export Group *(on page 160)*.
	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:

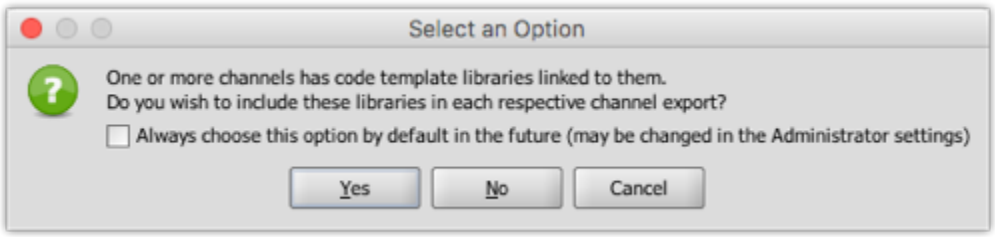


Import Group

The import group workflow follows the Import Channel *(on page 154)* workflow for each channel in the group.

Export Group

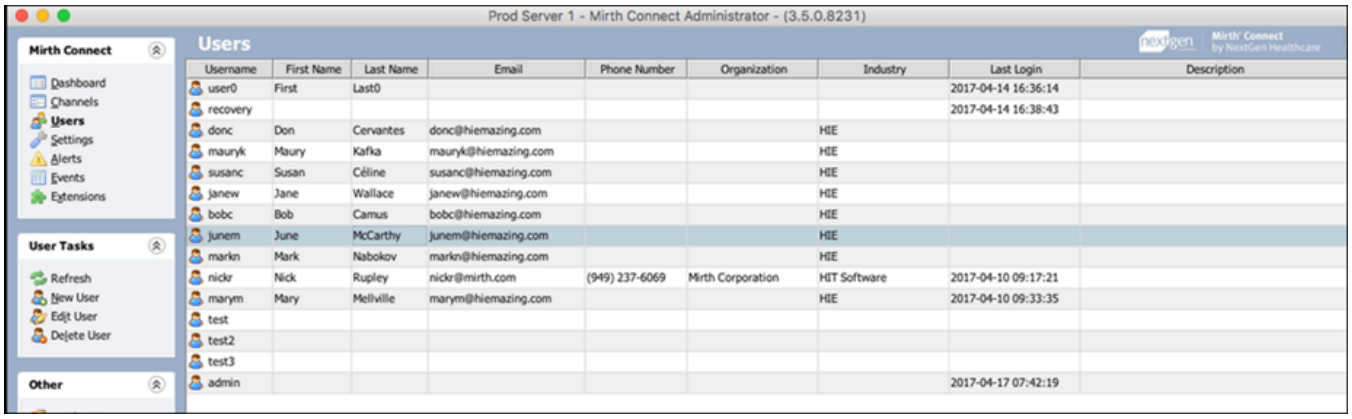
Similar to exporting channels *(on page 154)*, when exporting channel groups you have the option to include all associated code template libraries *(on page 301)*:



- 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 *(on page 174)*.

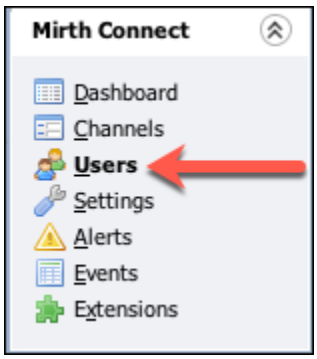
Users View

The Users view is the main management screen for all users that have access to log on to Mirth® Connect, whether you are using the Administrator, the Command Line Interface, or the REST API. In this view you can add / remove / edit users. Role-Based Access Control enables you to restrict access to specific operations to specific user roles.



Navigation

Select the **Users** link in the Mirth® Connect task pane on the top-left side of the window

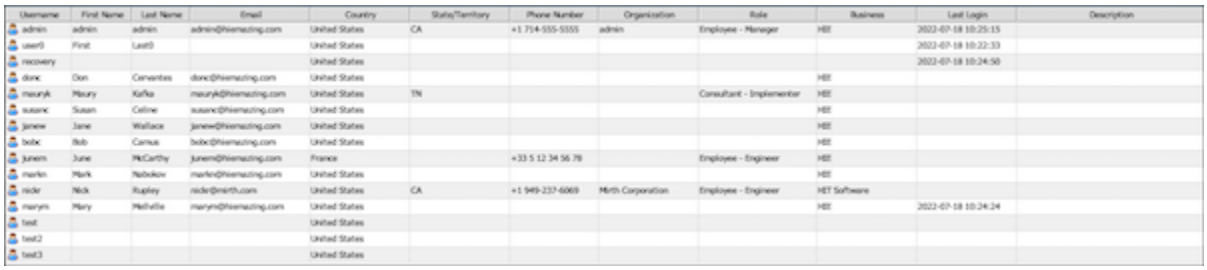


This section is separated into the following topics:

- Users Table *(on page 161)*
- Users Tasks *(on page 163)*

Users Table

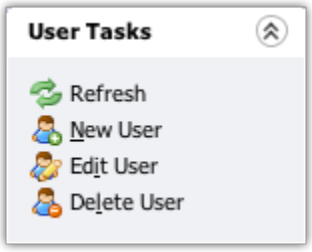
This is the main section of the **Users** view which shows details about all of the users configured on your server. For general information about working with tables in Mirth® Connect, see Working With Tables *(on page 79)*.



Users Table Columns

Column	Description
Username	The unique name the user uses to login.
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 is used.
Country	The user's country of residence.
State/Territory	For residence of the United States, the state or territory where the user lives.
Phone Number	The user's phone number.
Organization	The user's organization.
Role	The role of the user at their organization (such as C-Suite, Employee - Engineer, Independent Contractor).
Business	The user's work field / industry (such as 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

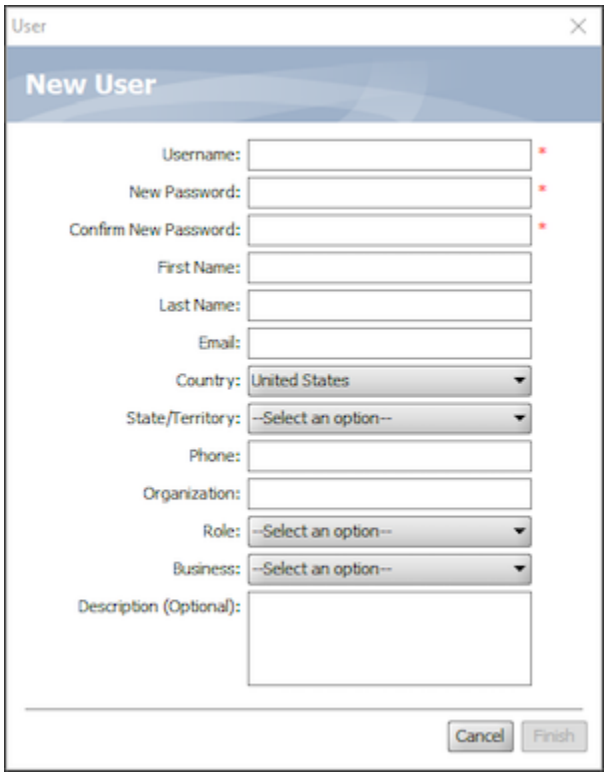


The following context-specific tasks are available from the Users View:

Task Icon	Task Name	Description
Refresh	Refresh	Updates the users table.
New User	New User *(on page 163)*	Creates a new user and adds it to the table.
Edit User	Edit User *(on page 163)*	Edits the details of an existing user.
Delete User	Delete User	Removes the user from the server.

Add/Update a User Account

When creating a new user or editing an existing user, a panel similar to the one provided below appears.



New User Dialog

Sample Edit User Dialog

In general the fields available on the **New User** and **Edit User** dialogs are the same as the columns displayed in the Users Table *(on page 161)*.

- If you are creating a new user: When creating a new user, the **Username**, **New Password** and **Confirm New Password** fields are required.

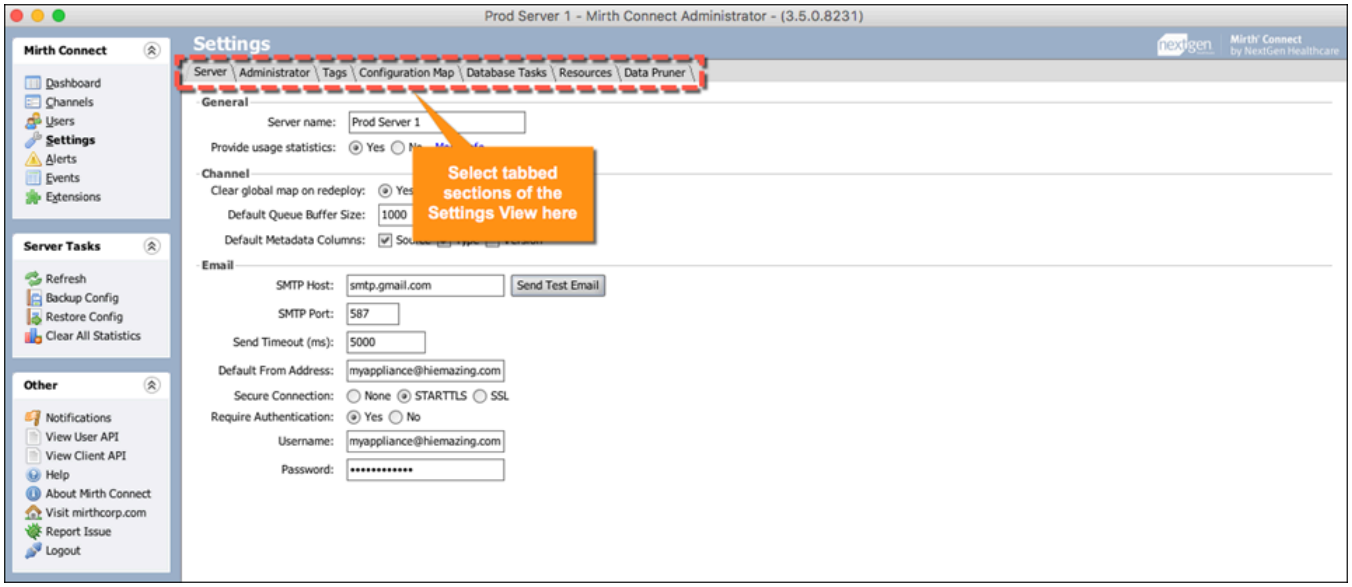
  Note:
  - The value in both password fields must match exactly.
  - The **Username** can be changed on this dialog, however, the new username must be unique. A username that currently exists in the Users Table cannot be reused.

- If you are editing existing user information: When editing an existing user, the **Username** field is required.

Note: The **Username** can be changed on this dialog, however, the new username must be unique. A username that currently exists in the Users Table cannot be reused.

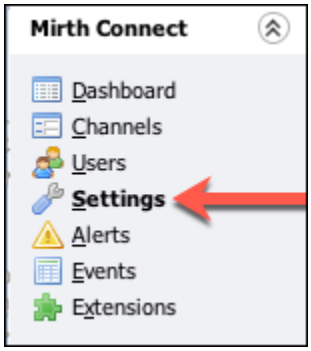
Settings View

The Settings view encompasses a wide variety of management settings and tasks for your Mirth® Connect server. Some of the settings are channel-specific (such as channel tags), and others are more general (such as default email settings).



Navigation

Select the **Settings** link in the Mirth® Connect task pane on the top-left: side of the window.

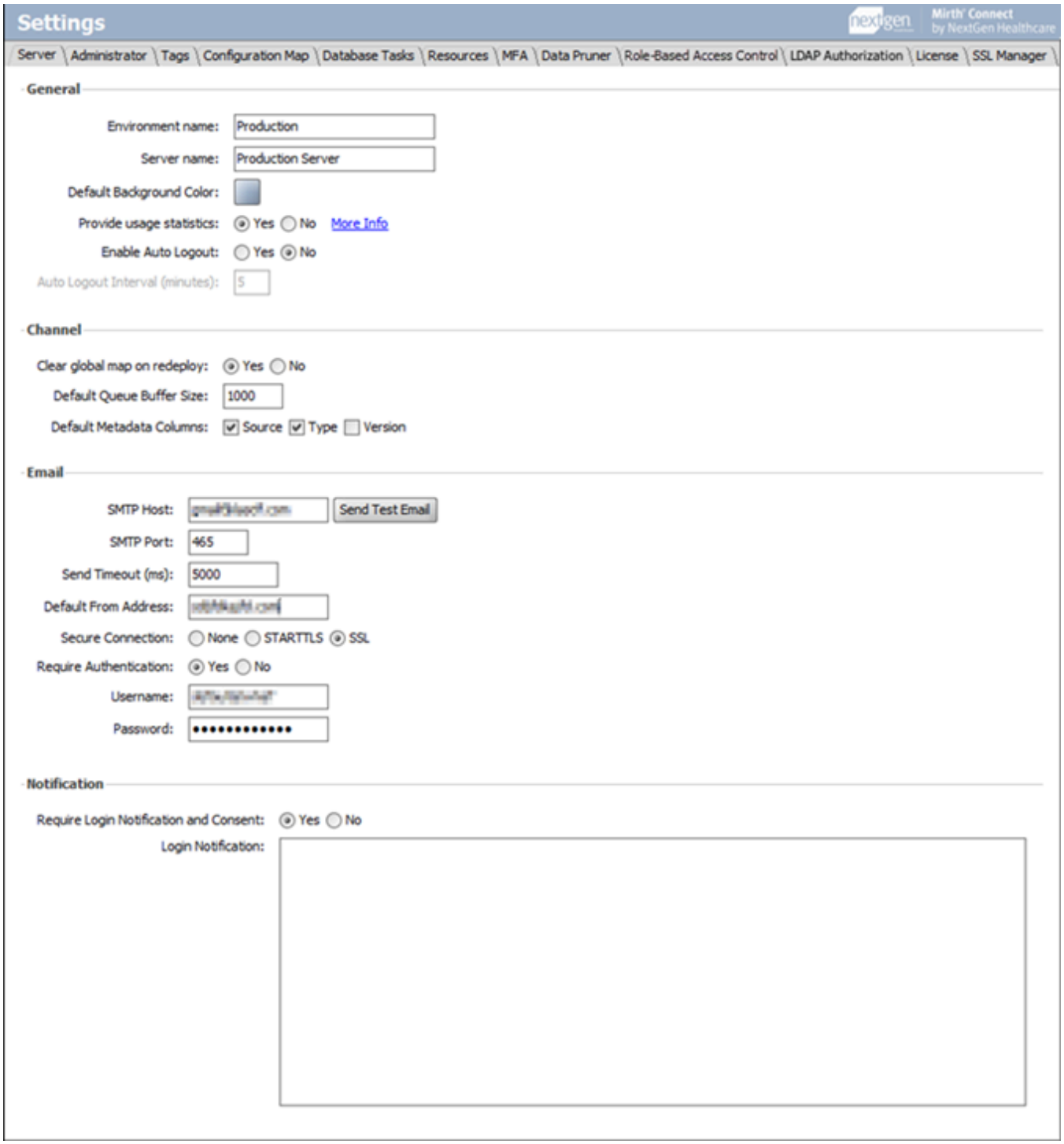


This section is separated into the following topics:

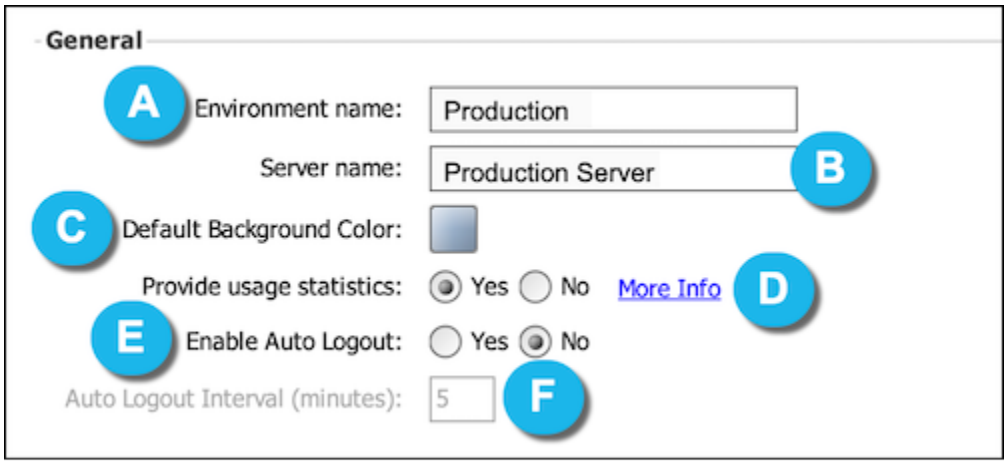
- Server Settings Tab *(on page 167)*
- Administrator Settings Tab *(on page 174)*
- Tags Settings Tab *(on page 179)*
- Configuration Map Settings Tab *(on page 182)*
- Database Tasks Settings Tab *(on page 184)*
- Resources Settings Tab *(on page 186)*
- Data Pruner Settings Tab *(on page 190)*
- Settings Tasks *(on page 196)*

Server Settings Tab

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


Settings
Server
Administrator
Tags
Configuration Map
Database Tasks
Resources
MFA
Data Pruner
Role-Based Access Control
LDAP Authorization
License
SSL Manager
General
Environment name:
Production
Server name:
Production Server
Default Background Color:
Provide usage statistics:
Yes
No
More Info
Enable Auto Logout:
Yes
No
Auto Logout Interval (minutes):
5
Channel
Clear global map on redeploy:
Yes
No
Default Queue Buffer Size:
1000
Default Metadata Columns:
Source
Type
Version
Email
SMTP Host:
Send Test Email
SMTP Port:
465
Send Timeout (ms):
5000
Default From Address:
Secure Connection:
None
STARTTLS
SSL
Require Authentication:
Yes
No
Username:
Password:
Notification
Require Login Notification and Consent:
Yes
No
Login Notification:

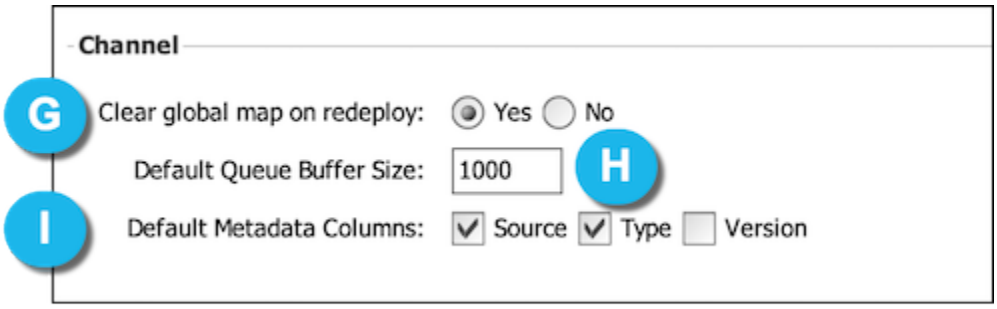
General Settings



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 that appears 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 for this server. Users can override this with their own custom background color in the Administrator Settings Tab *(on page 174)*.
D	Provide usage statistics	Statistics help NextGen Healthcare determine which connectors or areas of Mirth Connect are most widely used. Statistics **do not** contain PHI or channel/script implementation. For additional information, see NextGen Legal Notices. • Select **Yes** to send usage statistics to NextGen Healthcare. • Select **No** to not share usage statistics.
E	Enable Auto Logout	Determines whether a user is automatically logged out of the Mirth® Connect Administrator due to inactivity.

Item	Name	Description
		• Select **Yes** to automatically log out of the Mirth® Connect Administrator due to inactivity as defined in the **Auto Logout Interval** field. • Select **No** to remain logged into the Mirth® Connect Administrator. **No** is the default for this field.
F	Auto Logout Interval (minutes)	The time in minutes that will trigger an inactive user to be automatically logged out of the Mirth® Connect Administrator. The default time is 5 minutes. Valid values are 1 through 60, inclusive.

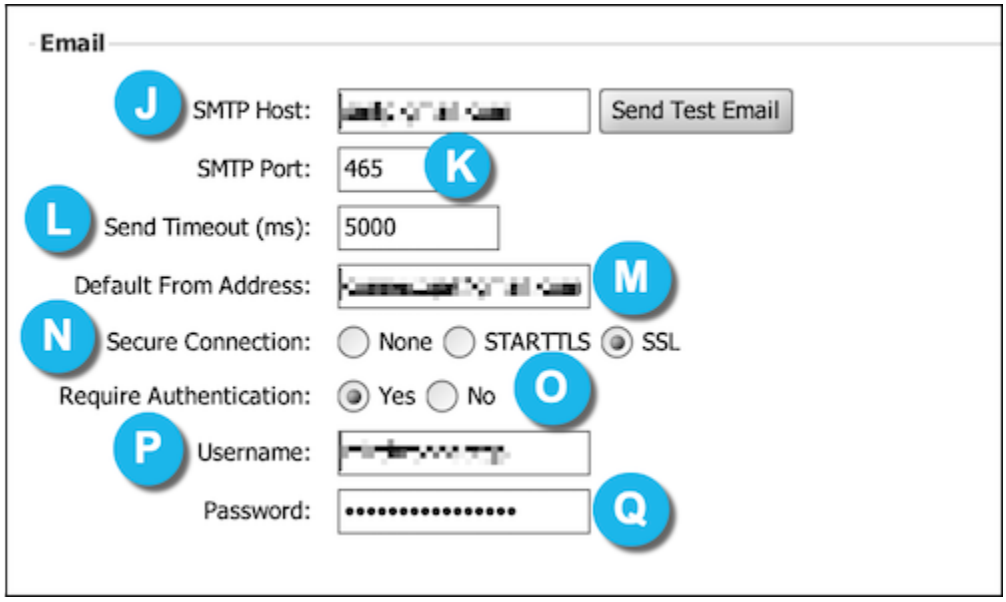
Channel Settings



Item	Name	Description
G	Clear global map on redeploy	• Select **Yes** to clear the global map when redeploying all channels. **Yes** is the default for this field. • Select **No** to keep global map information when redeploying all channels.
H	Default Queue Buffer Size	The default source/destination queue buffer size to use for new channels. The default is **1000** (messages).
I	Default Metadata Columns	Determines which custom metadata columns is added by default when a user creates a new channel. Options include:

Item	Name	Description
		• Source • Type • Version The user can also remove these column on the channel's **Summary** tab. For additional information, see Custom Metadata Columns *(on page 106)*.

Email Settings



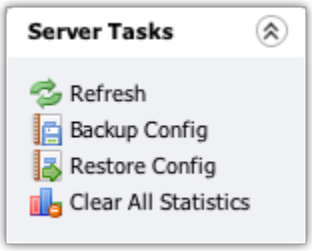
Item	Name	Description
J	SMTP Host	• The SMTP (Simple Mail Transfer Protocol) host field is used as a global e-mail setting for actions such as alerts. Enter the email you would like to use to receive alerts in the SMTP Host field. • Select the **Send Test Email** button to send a simple test e-mail to the **Default From Address** email address. • The SMTP Host, SMTP Port and Default

Item	Name	Description
		From Address fields are required and must contain valid information to use send a test email.
K	SMTP Port	SMTP port used for global email settings (such as alerts)
L	Send Timeout (ms)	SMTP socket connection timeout in milliseconds. The default value is **5000** (ms) or 5 secs.
M	Default From Address	Default address used in the **From** field for emails. A test email will also send to this address.
N	Secure Connection	Determines whether to use the following setting when sending emails: • **None**: no encryption • **STARTTLS**: Explicit TLS • **SSL**: Implicit TLS
O	Require Authentication	Determines whether authentication is required when sending emails from within Mirth® Connect. • Select **Yes** to require authentication when sending emails. • Select **No** if authentication is not required when sending emails.
P	Username	Enter the username used to authenticate the email.
Q	Password	Enter the password used to authenticate the email.

Notification Settings

Item	Name	Description
R	Require Login Notification and Consent	Determines whether a notification and consent message appears upon login. The default is **Yes**.
S	Login Notification	Enter the text that appears in the Notification and Consent message, based on your organizations needs. This message appears when the **Required Login Notification and Consent** field is set to **Yes**.

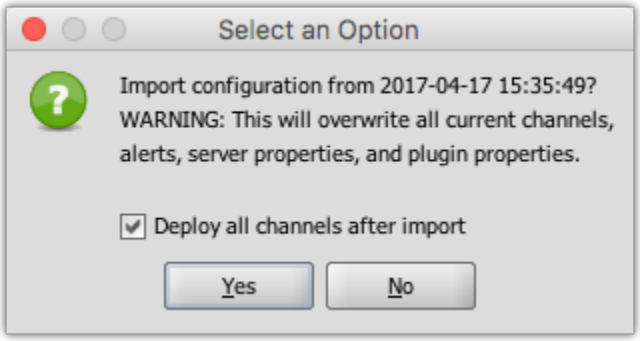
Tasks



Task Icon	Task Name	Description
Backup Config	Backup Config	Exports a snapshot of your current server settings, including all channels, alerts, scripts, and other properties. This configuration may be backed up from one Mirth® Connect server and restored into a different server. **Warning:** The Server Configuration file does not include Users, Events, or Message / Attachment data. To export events, see Event Tasks - Export All Events *(on page 147)*. To export message / attachment data, see Message Browser Tasks - Export Results *(on page 130)*.
Restore Config	Restore Config *(on page 174)*	Overwrites all server settings, including all channels, alerts, scripts, and other properties, with the ones in a given XML file.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.
Clear All Statistics	Clear All Statistics *(on page 174)*	Resets the current and lifetime message statistics for **all** channels on your server.

Restore Config

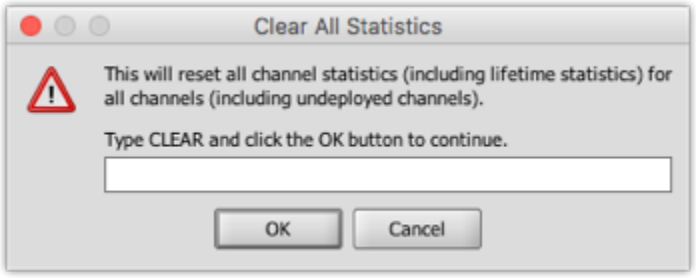
After selecting a server configuration XML file to restore, you will be presented with the following dialog:



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

Clear All Statistics

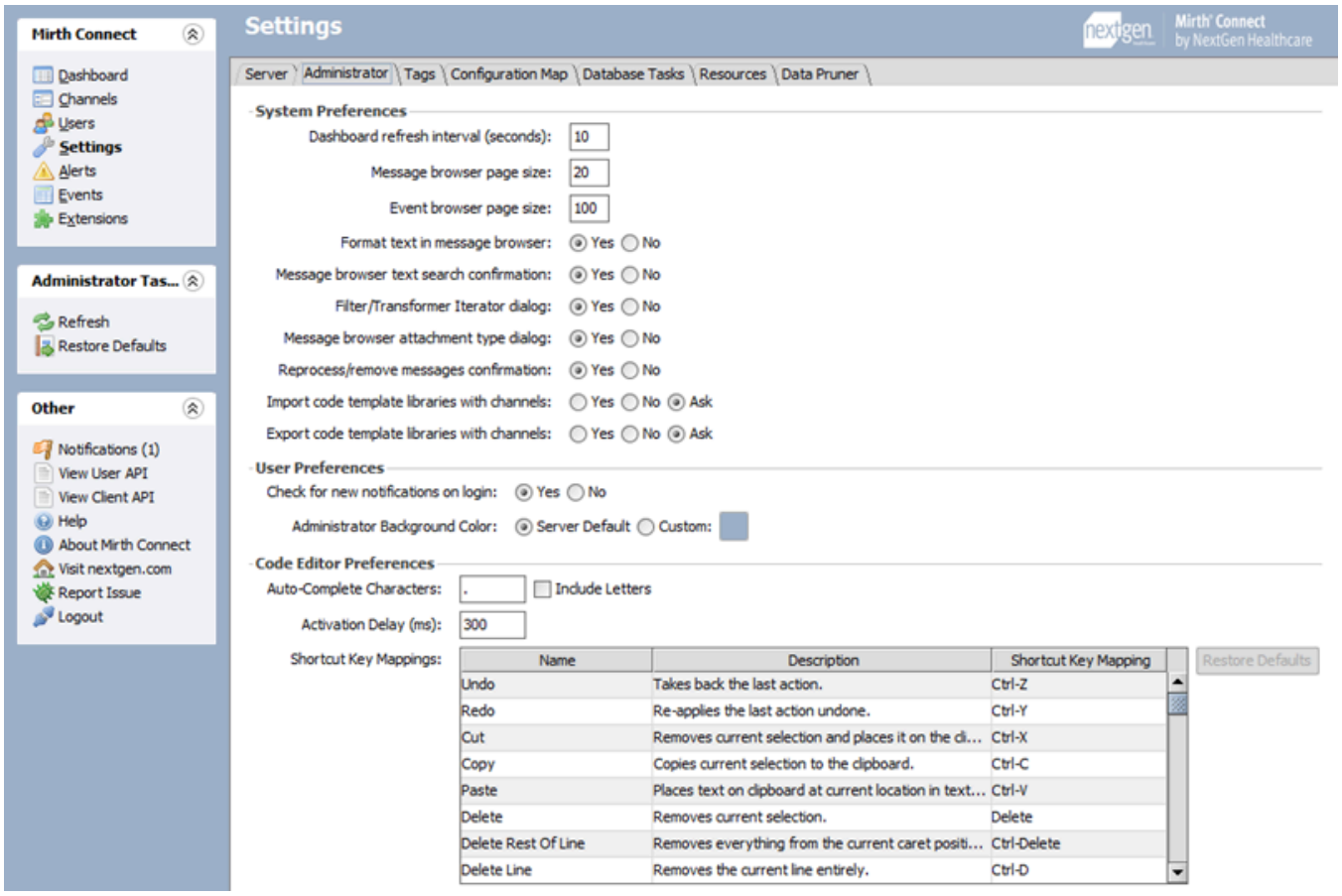
After selecting this task, a confirmation dialog appears:



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

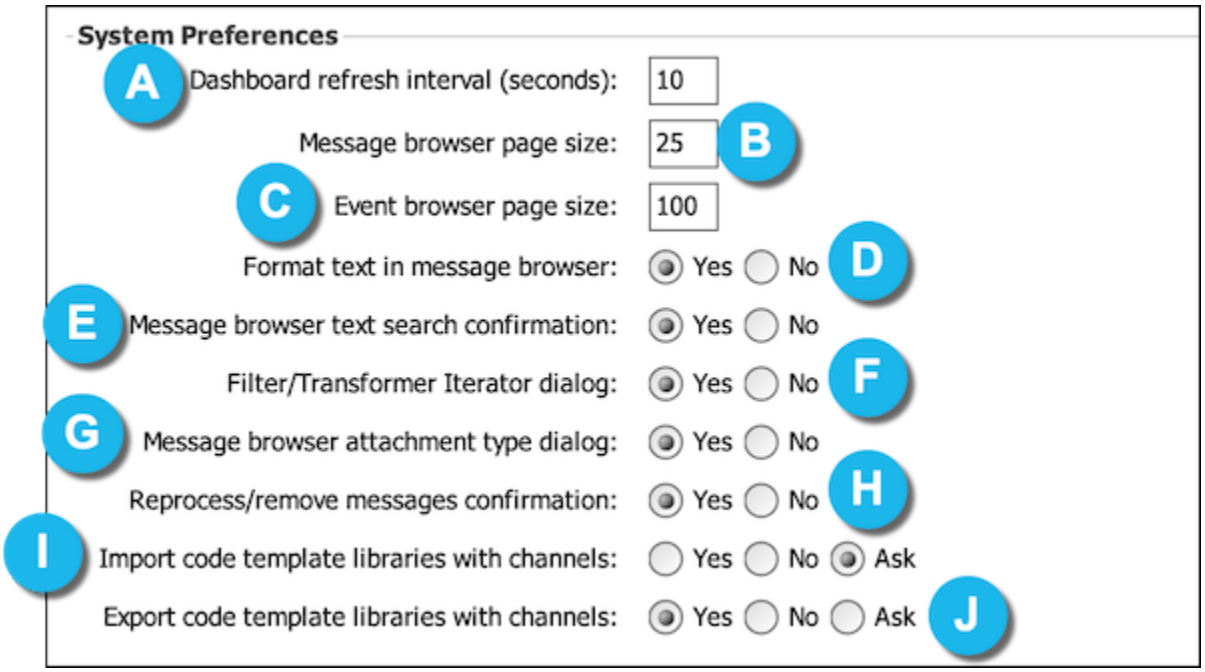
Administrator Settings Tab

The Administrator Settings 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 log on to.



System Preferences

The following image shows the System Preferences options available on the Administrator Settings tab. It is followed by a table that provides descriptions for the System Preferences options.



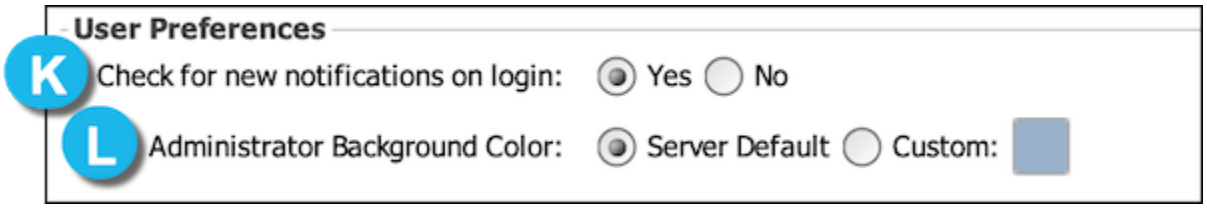
Item	Name	Description
A	Dashboard refresh interval (seconds)	The interval, in seconds, at which the Dashboard and Alerts views are refreshed. Reduce this for faster updates and increase it for servers with more channels or when you have high latency between the client and server.
B	Message browser page 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	Shows 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 number of messages being searched.
F	Filter/Transformer Iterator dialog	Shows a confirmation dialog in the filter/transformer views when dragging and dropping elements from the message tree, asking users whether to use an Iterator.
G	Message browser attachment type dialog	Shows a selection dialog in the message browser when viewing attachments to enable the user to select a specific attachment viewer. • If **Yes** is selected, the dialog enables the user to

Item	Name	Description
		select a specific attachment viewer. • If **No** is selected, the attachment viewer is automatically chosen from the MIME type.
H	Reprocess/remove messages confirmation	Shows a confirmation dialog in the message browser when reprocessing or removing multiple messages that forces the user to type **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 import the code templates libraries with channels. • Select **No** to never import code templates libraries with channels. • Select **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 export code templates libraries with channels. • Select **No** to never export code templates libraries with channels. • Select **Ask** to prompt the user each time.

User Preferences

The following image shows the User Preferences section of the Administrator Settings tab, followed by descriptions of the items displayed.

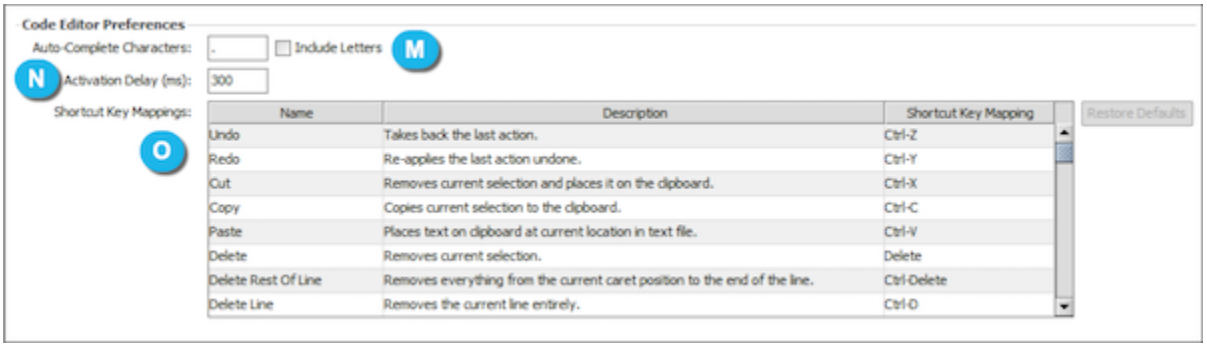
Note: Settings in this section are specific to the user currently logged on the current server.



Item	Name	Description
K	Check for new notifications on login	Checks for notifications from NextGen (such as announcements, available updates) relevant to this version of Mirth Connect whenever the user logs on.
L	Administrator Background Color	• If **Server Default** is selected, the background color chosen on the Server Settings Tab *(on page 167)* is used. • If **Custom** is selected, the color displayed in the box next to the checkbox is used instead.

Code Editor Preferences

The following image shows the options available in the Code Editor Preferences section of the Administrator Settings tab, followed by descriptions for the items.

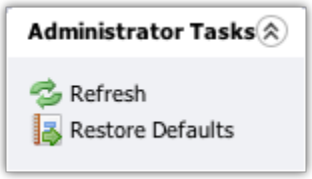


Item	Name	Description
M	Auto-Complete Characters	The auto-completion popup is triggered after any of these characters are typed. • If the **Include Letters** check box is checked, auto-completion is also triggered after any letter (a-z) is typed.

Item	Name	Description
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 restores all shortcut key mappings back to the default settings.

Tasks

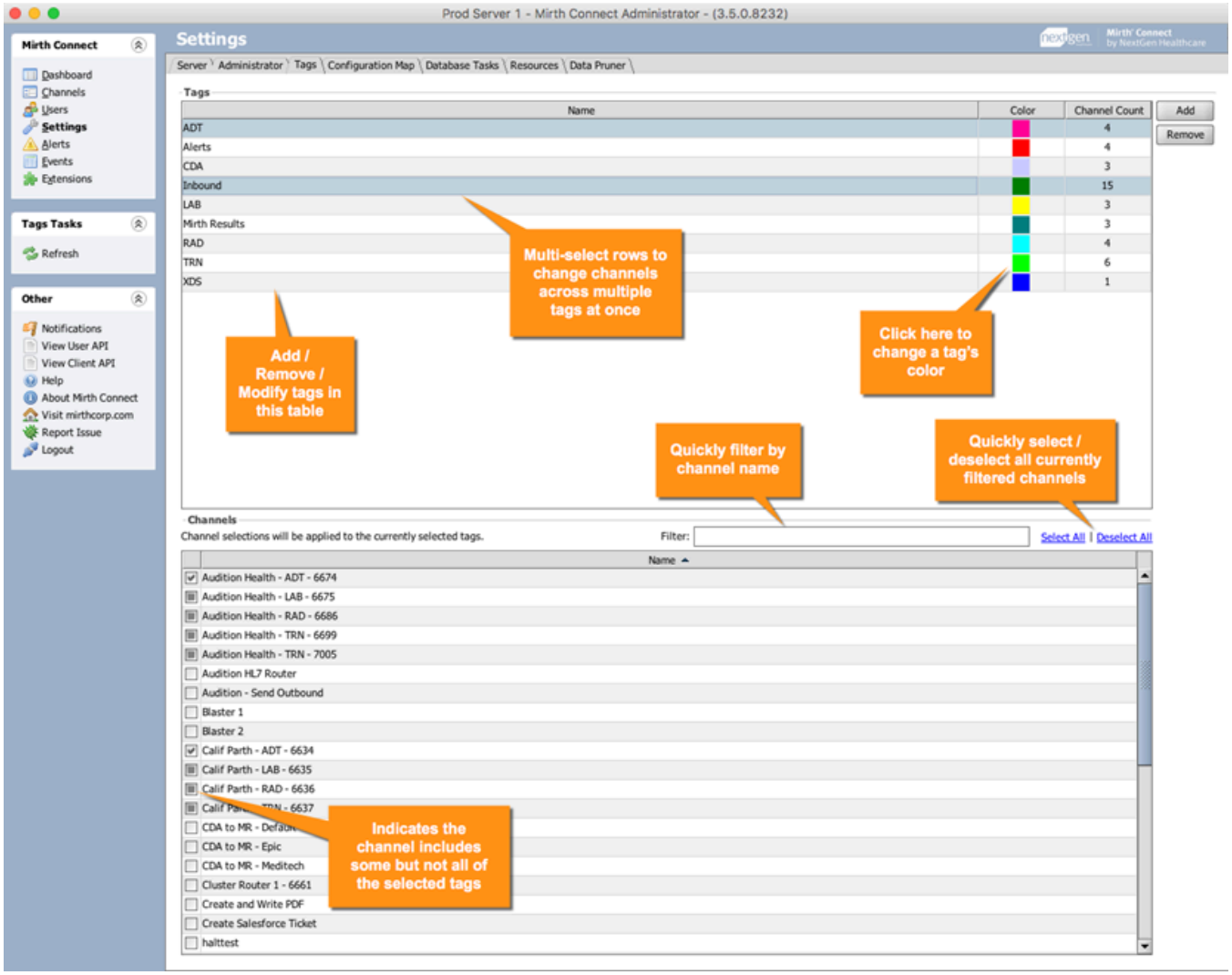
The following image shows the Administrator Tasks panel, followed by a description of its items.



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 and colors, and easily include multiple tags across multiple channels with a few actions.



Tags Table

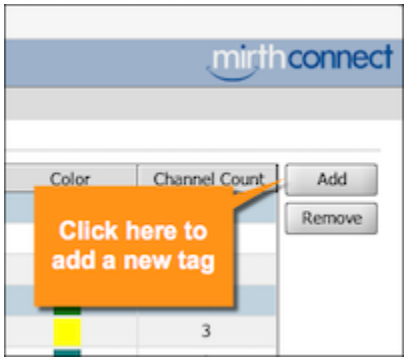
This is the table located in the top half of the **Tags** tab. The table has the following columns:

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. Select 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 *(on page 79)*.

Adding a Tag

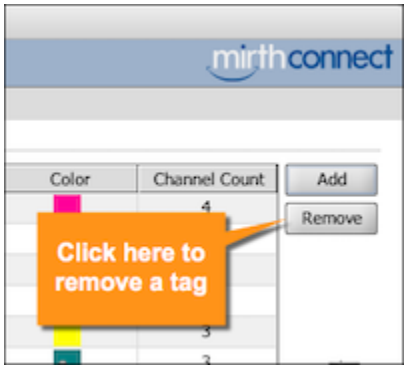
To add a new tag, select the **Add** button on the top-right side of the tags table.



A new row will be added to the table. Double-click the **Name** column to change the name, and select 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 select the **Remove** button at the top-right of the tags table.



Channels Table

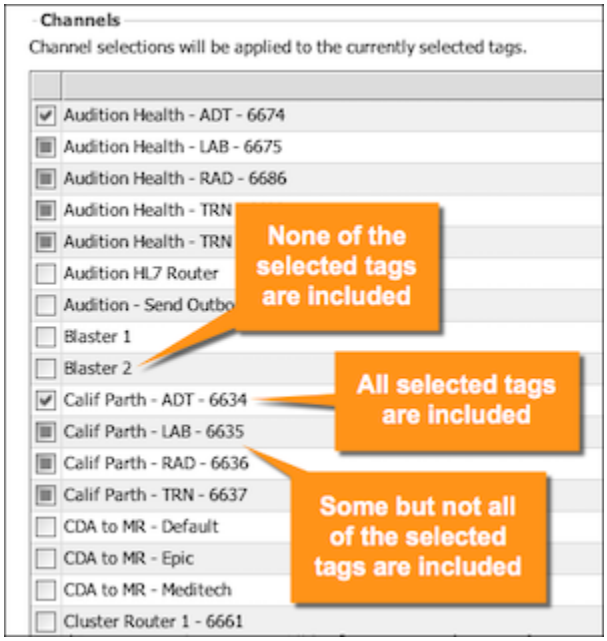
This table shows a list of all channels currently configured on your server. Select one of the column headers at the top of the table to sort by selection status or channel name. Select the check boxes to include the selected tags on specific channels.

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 and the channel in question is included on **only some** of the selected tags.



If you select the check box while it is 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. The fact that it is stored in the Application Data Directory as a flat file makes it unique. It is intentionally not included in server configuration exports. Because of this, you can use variables from this map in your channel properties, and the same channel can be used across multiple Mirth® Connect servers without having to edit the channel for each server.

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

The Configuration Map tab enables you to edit the entries in the configuration map for the current server. Double-click any of the cells in the table to edit the values. Select the **Add** / **Remove** buttons on

the top-right side of the window to add or remove entries from the table.

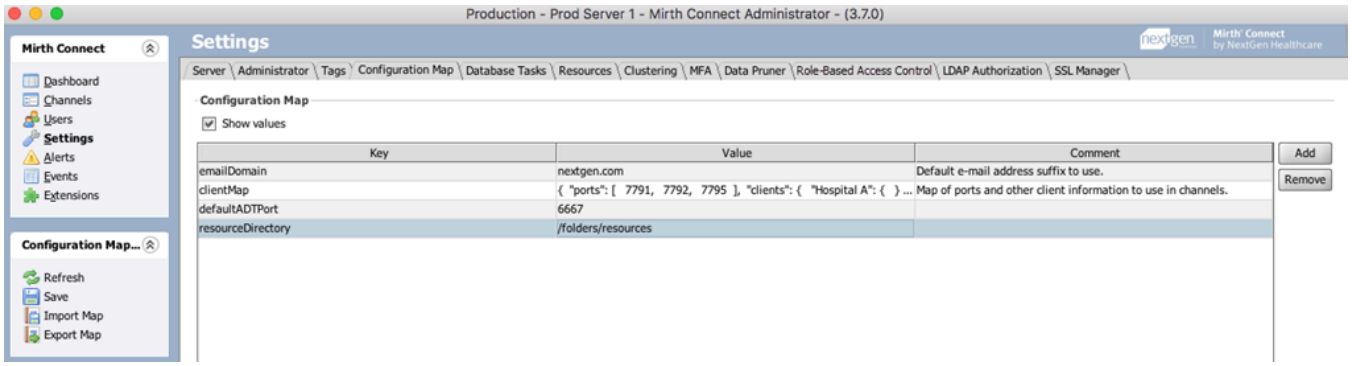
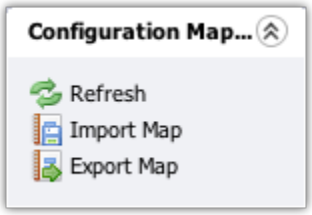
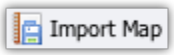


Table Columns

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

Tasks

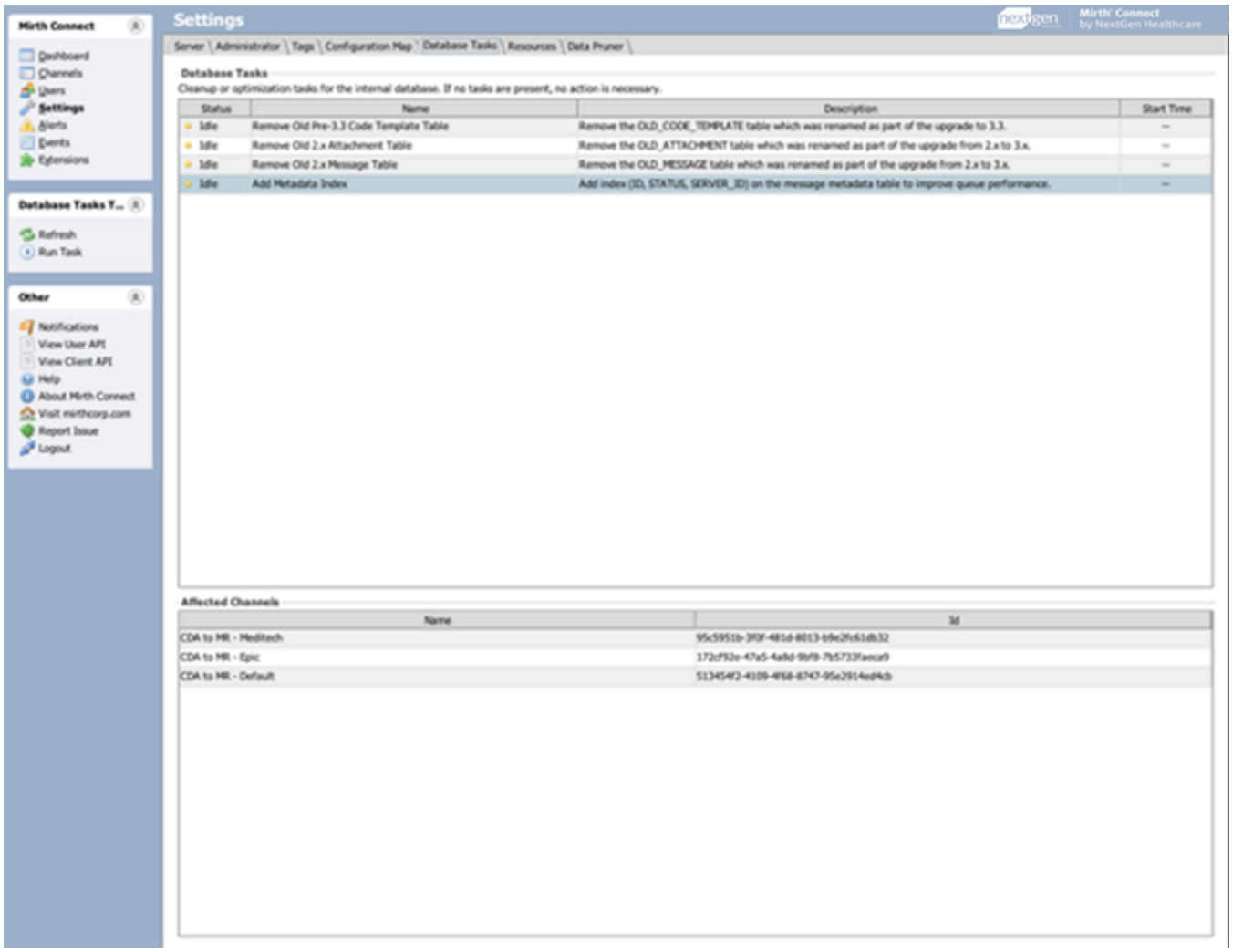


Task Icon	Task Name	Description
	Import Map	Overwrites all entries in the table with the ones in the given properties file.
	Export Map	Exports all entries in the table to a properties file.

Database Tasks Settings Tab

Occasionally in new versions of Mirth® Connect certain things change in the underlying database but depending on the implications those changes are not always automatically performed when you upgrade. 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.

The Database Tasks Settings tab shows you all cleanup or optimization tasks for the internal Mirth® Connect database. If no tasks show up in the table, then that means your database is up-to-date.



Database Tasks Table Columns

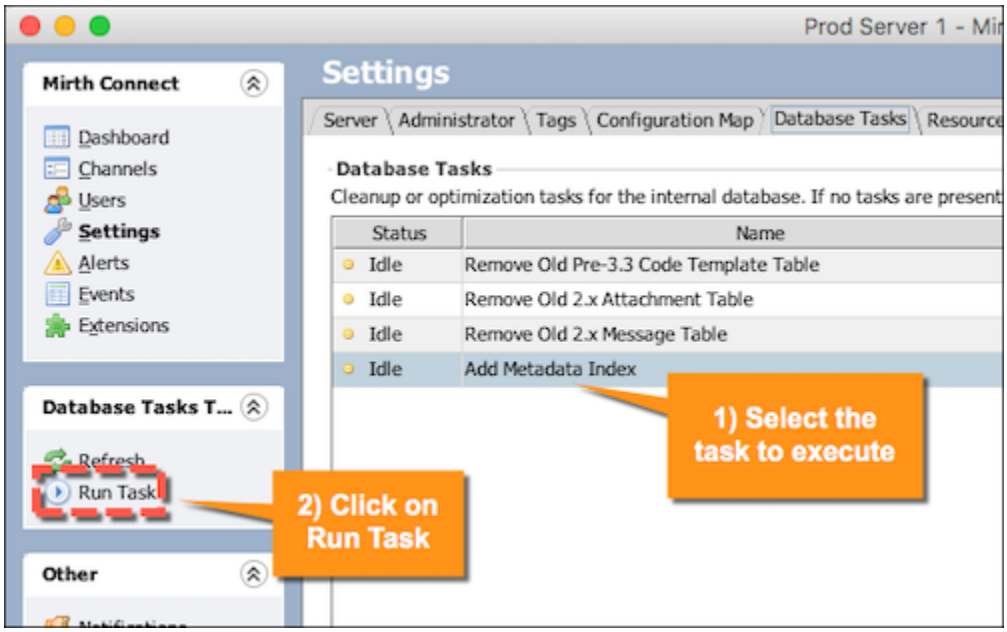
Column	Description
Status	• **Idle**: The task is available to run. • **Running**: The task is in the middle of running.
Name	The name of the task.
Description	A short description of the task.
Start Time	If the task is currently running, this will show the time it started.

Affected Channels Table

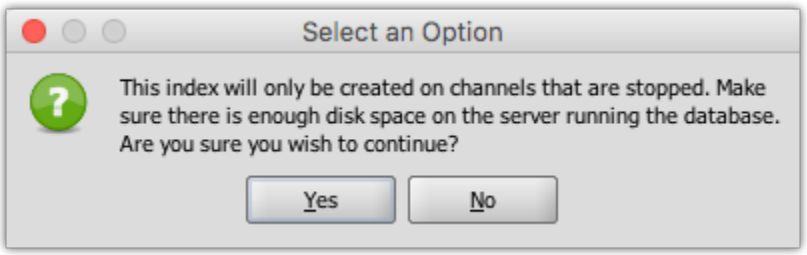
The **Affected Channels** table only applies to database tasks that are performed on specific channel tables. It will show all channels that are affected by the selected task if it is executed. Note that some tasks require the channels to be **Stopped** before the task is actually performed.

Running a Database Task

To run a database task, select the task row in the table, then select the **Run Task** link in the left-side of the window.

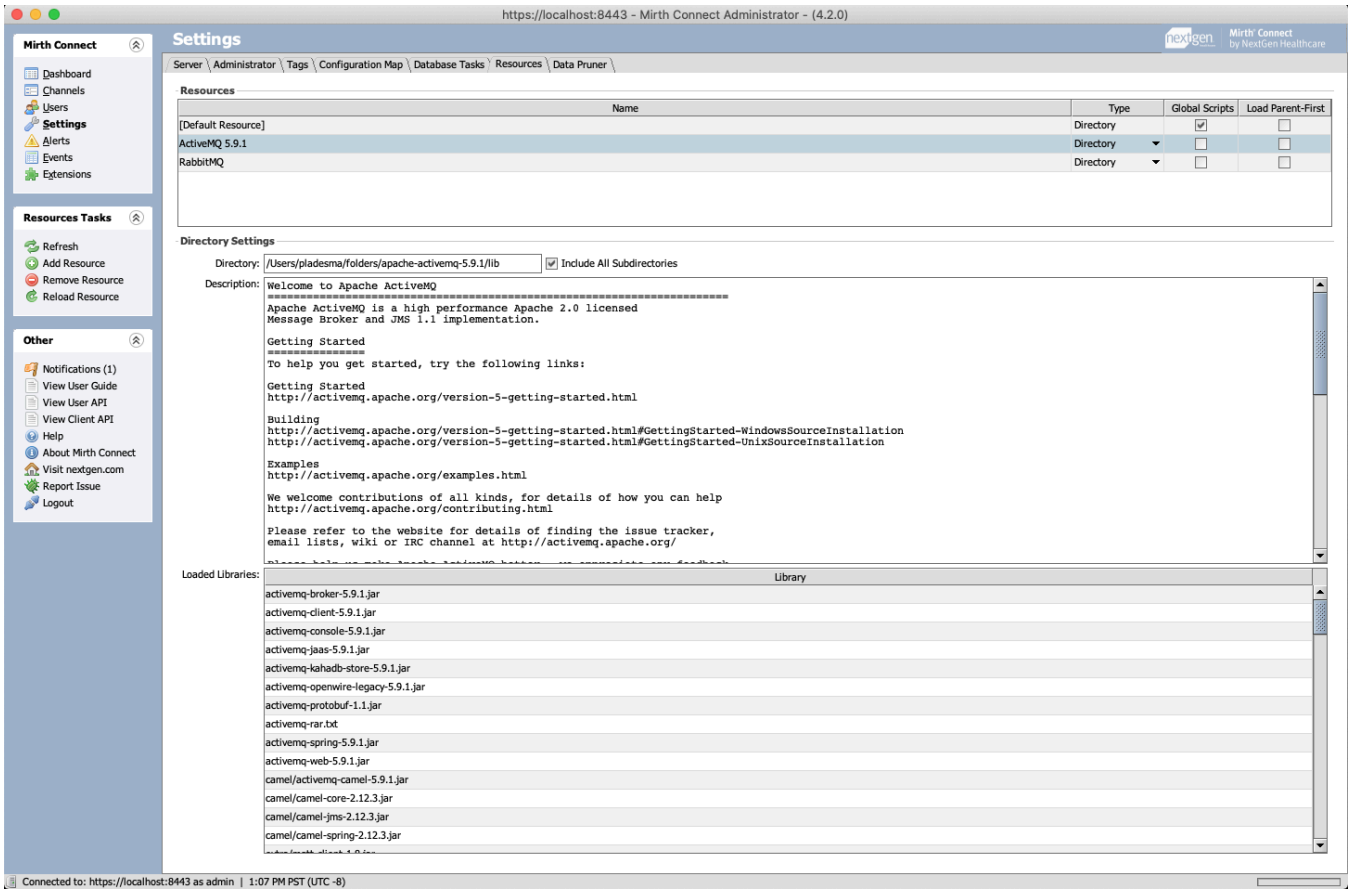


Depending on the task, a confirmation dialog may appear.



Resources Settings Tab

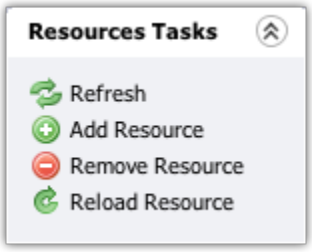
Resources are shared services that can be used in specific channels and connectors or in other places throughout the Mirth® Connect server. They may include custom Java libraries to use within scripts or services to handle outbound connections.

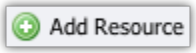
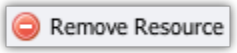
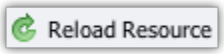


Resources Table Columns

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 selected, any libraries included with the resource will automatically be made available for use in the Global Scripts *(on page 299)*.
Load Parent-First	If selected, classes already included in the overall server classpath will not be able to be overwritten. Classes will attempt to be loaded from the parent ClassLoader first. Also, if this resource is included on a channel with other resources that have this option disabled, you will still not be able to overwrite classes in the parent classpath.

Tasks

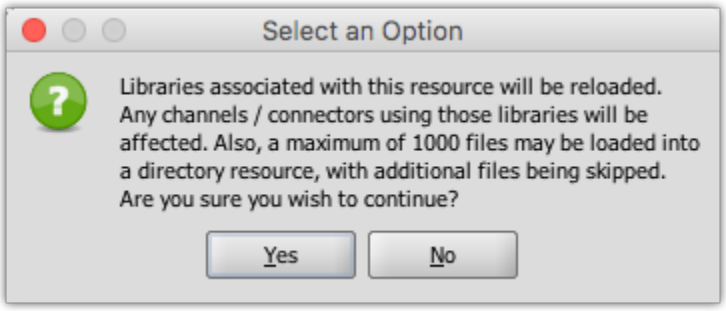


Task Icon	Task Name	Description
	Add Resource	Adds a resource to the table.
	Remove Resource	Removes the selected resource from the table.
	Reload Resource	Reloads the selected resource on the server.For additional information, see Reload Resource *(on page 188)*.

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.

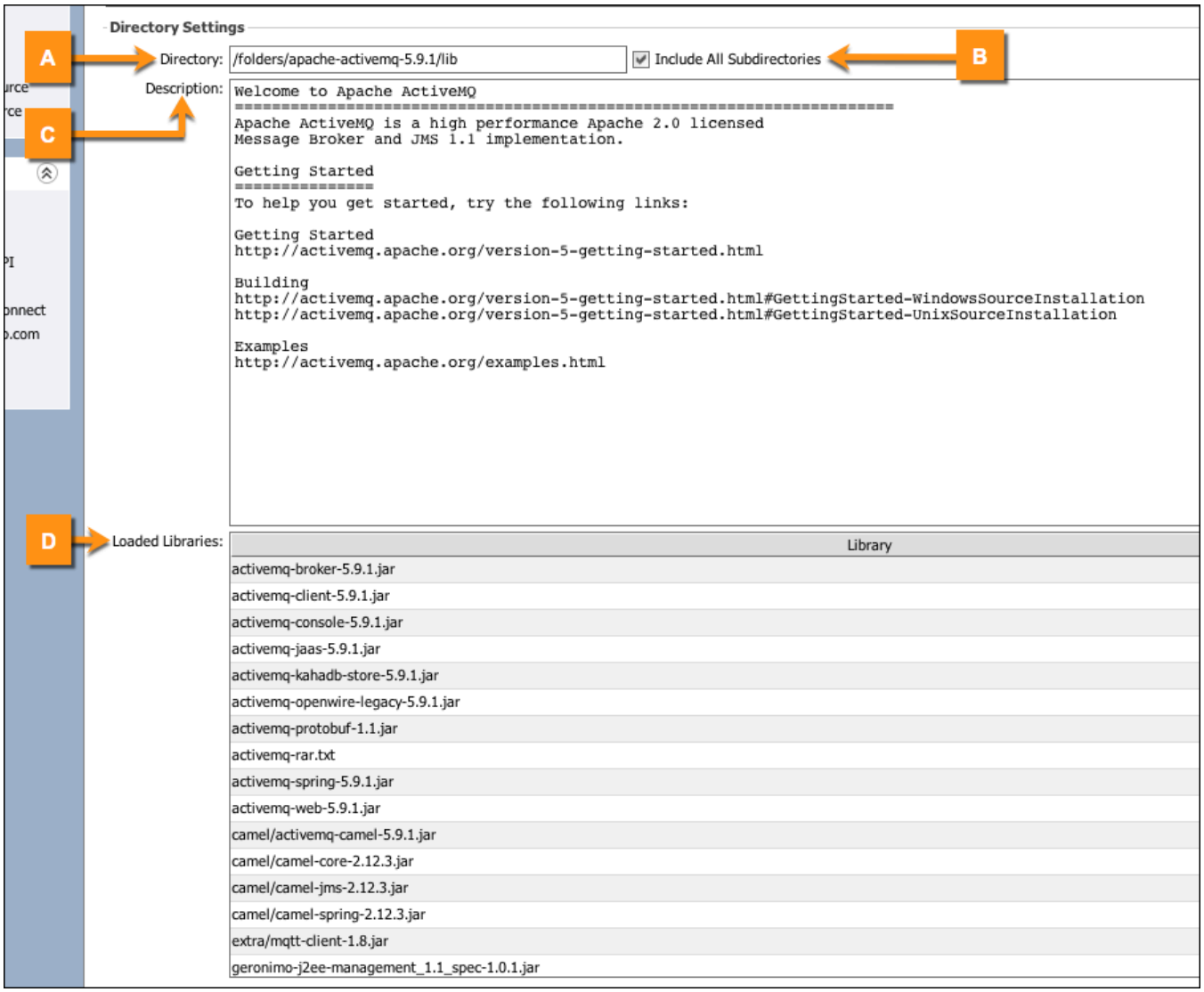
When the task is selected, a confirmation message appears.



Directory Resource

The directory resource enables 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

Item	Name	Description
		and connectors. If you have saved the resource but do not see any libraries in this table yet, try using the **Refresh** task on the left side of the window.

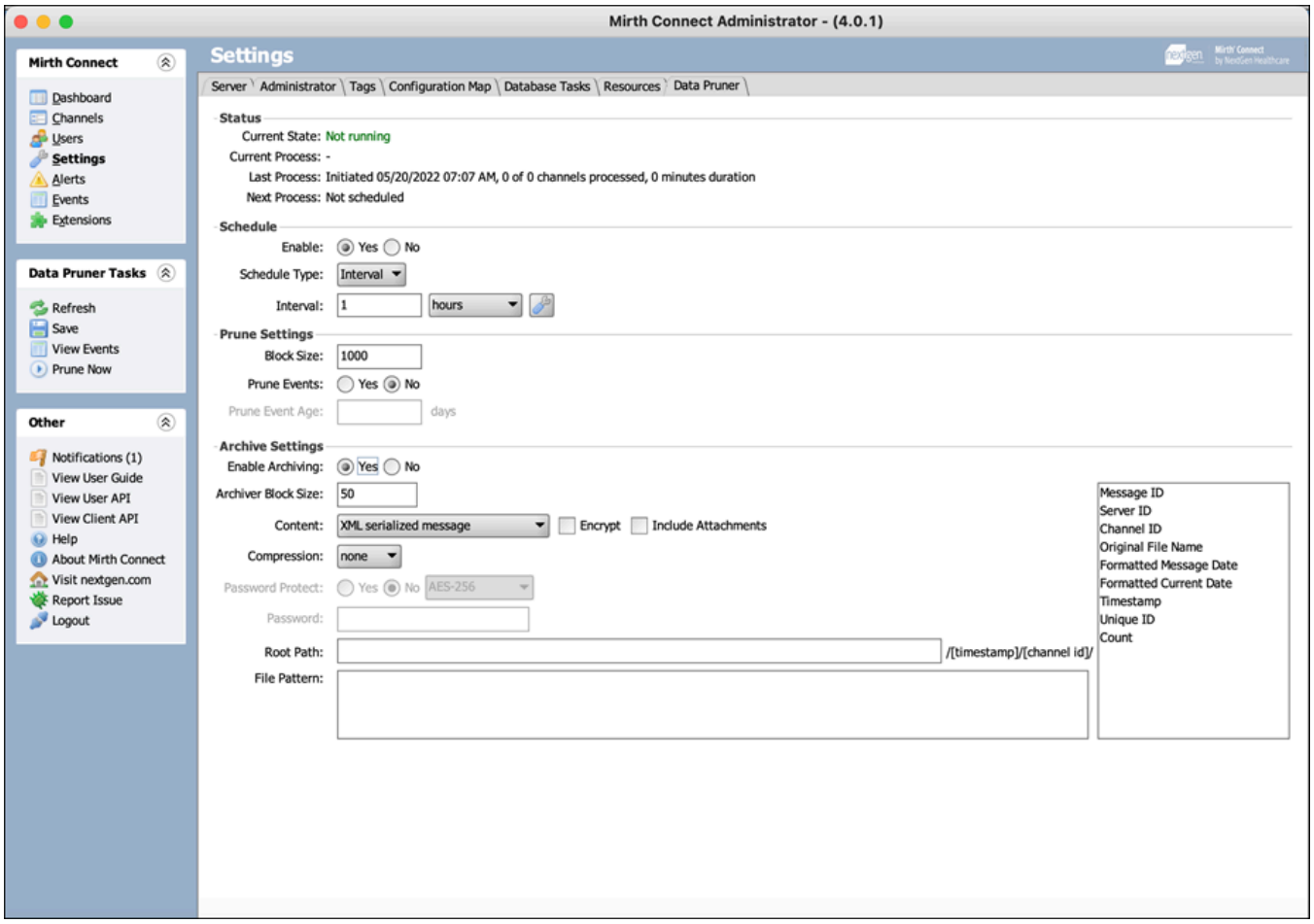
Using Resources in Channels/Connectors

After you add 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 *(on page 214)*.

Data Pruner Settings Tab

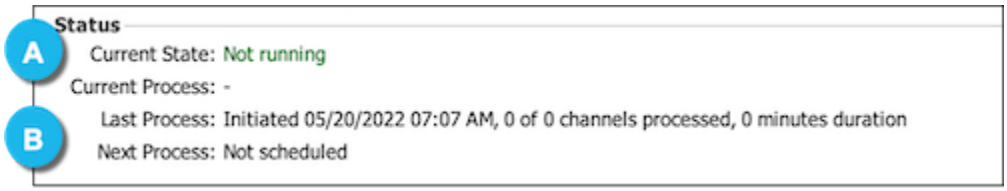
The Data Pruner Settings tab enables you to configure the global data pruner/archiver settings. In addition to seeing the current status of the pruner, you can set a specific schedule for it, and decide whether message data should be archived out to disk before being pruned.

Note: Message/attachment data for channels will not be pruned unless the corresponding pruning settings on the channel are enabled.



Status

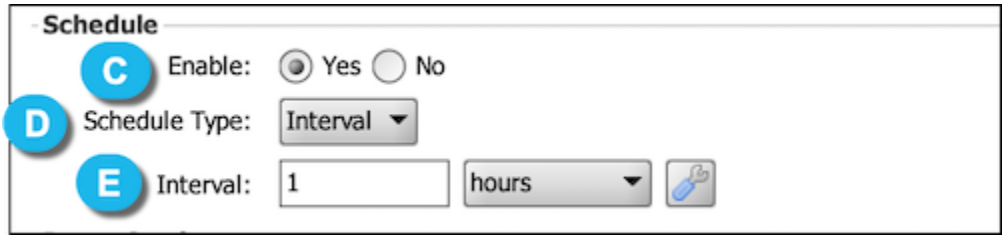
Select the **Refresh** task to update the status.



Item	Name	Description
A	Current State	Indicates whether the Data Pruner is currently running or not.
B	Processes	This area identifies the Current, Last, and Next Process for the pruning.

Item	Name	Description
		• **Current Process**: Identifies the start time of the currently executing pruning job, the number of channels that have been processed, and how long the job has been running. • **Last Process**: Identifies the start time of the last executed pruning job, the number of channels that were processed, and how long the job took to run. • **Next Process**: Indicates the date and time of the next scheduled pruning job.

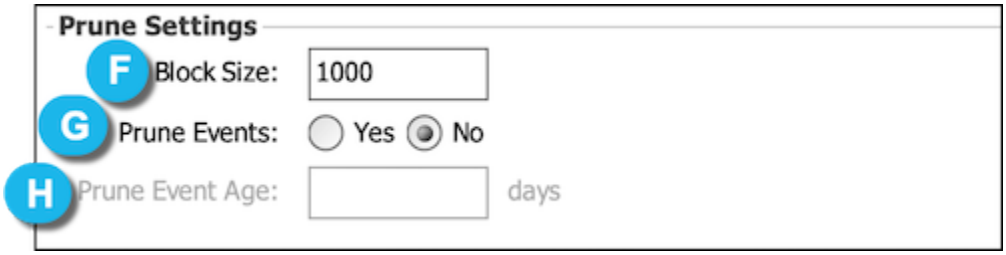
Schedule



Item	Name	Description
C	Enable	Identifies whether the Data Pruner will run on a recurring schedule. • Select **Yes** if to enable the Data Pruner. When Yes is selected, the schedule set up in the **Schedule Type** and **Interval** fields is executed. • Select **No** to execute execute the the Data Pruner manually. When No is selected, you can manually execute Data Pruning from the **Data Pruner Tasks** menu using the **Prune Now** task.
D	Schedule Type	Select the type of schedule to use for the pruner from the pull-down menu. Available schedule types include:: • Interval • Time • Cron For additional information about schedule types, see Polling Settings

Item	Name	Description
		(on page 234).
E	• Interval (Shown) • Time • Cron	This area changes depending on the Schedule Type selected. For details on each Schedule Type, see Polling Settings *(on page 234)*.

Prune Settings



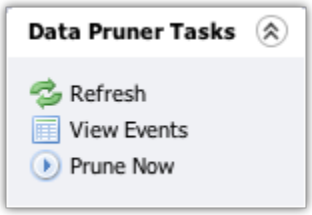
Item	Name	Description
F	Block Size	The number of messages that will be pruned at a time. This value must be between 50 and 10,000. The recommended value for most servers is 1,000. Larger values use more memory but typically will cause the pruner to perform more quickly.
G	Prune Events	• Select **Yes** to prune event records older than the number of days set in **Prune Event Age** field. • Select **No** to leave event records unpruned.
H	Prune Event Age	Enter the number of days that event records will be kept before pruning them. When **Yes** is selected in the **Prune Events** field, event records older than the number of days set in this field are pruned.

Archive Settings

Item	Name	Description
I	Enable Archiving	Determines whether message / attachment data is saved to disk before being pruned. • Select **Yes** to save message / attachment data to disk prior to being pruned. • Select **No** to perform a prune without saving message / attachment data.
J	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.
K	Content	The type of content that is exported from the messages. **Note: 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. • **Encrypt**: Check **Encrypt** to encrypt the exported message content with the server's encryption key. • **Include Attachments**: Check **Include Attachments** to include all attachments associated with the message in your exported file. Note that the content type must be set to **XML serialized message** to perform this action.
L	Compression	When compression is enabled, the files/folders created according to the **File Pattern** are put into a compressed file in the Root Path. Select one of the following supported compression types:

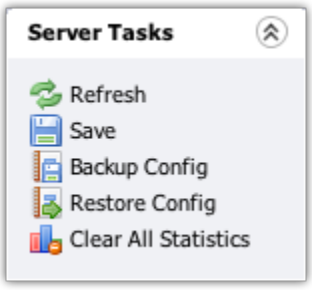
Item	Name	Description
		• None: This selection will not apply compress to the created files/folders. • ZIP • tar.gz • tar.bz2
M	Password Protect	This option is only available if ZIP compression is selected. • Select **Yes** to protect the resulting ZIP file with the password entered in the **Password** field. When selected, you can also choose what encryption algorithm to use when password-protecting files. The following algorithms are supported: Standard, AES-128, AES-256. • Select **No** to create an unprotected .ZIP file.
N	Password	Enter the password to be used If password protection is selected in the **Password Protect** field.
O	Root Path	Enter the root path where you store the exported files/folder or compressed file. Relative paths are resolved against the Mirth® Connect Server home directory.
P	File Pattern	Enter the file/folder pattern where the exported message files are written. Variables from the Variables list can be used in the pattern.
Q	Variables	Contains several common variables you can use to populate the File Pattern. For additional information, see Velocity Variable Replacement *(on page 469)*.

Tasks



Task Icon	Task Name	Description
View Events	View Events	View events specific to the Data Pruner. When selected this leaves the Settings View *(on page 166)* and enters the Events View *(on page 141)*.
Prune Now	Prune Now	Manually starts a pruning / archiving job.
Stop Pruner	Stop Pruner	Stops the currently running pruning job prematurely.

Settings Tasks



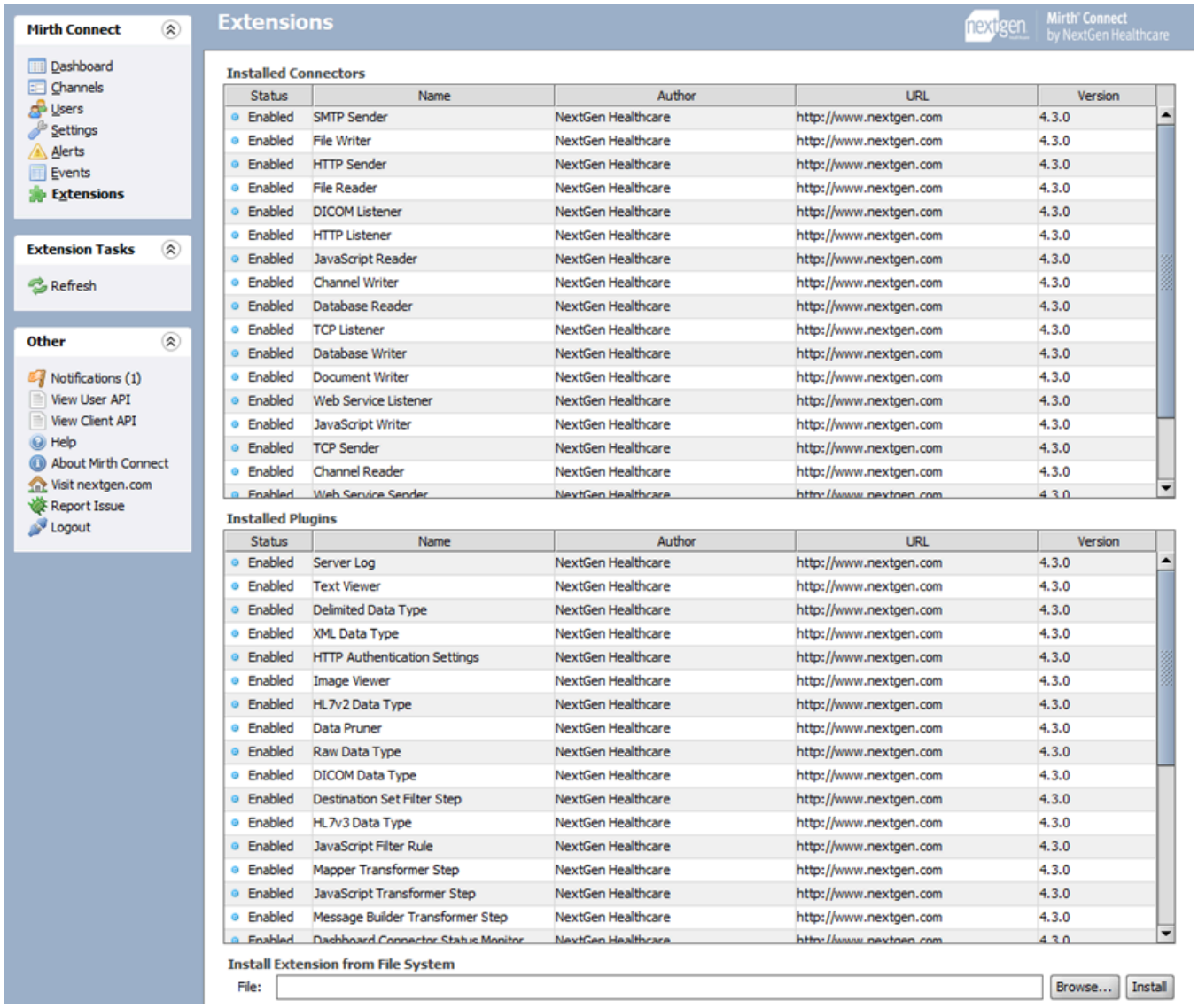
The following context-specific tasks are available from the **Settings View** and are common across most of the tabs:

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

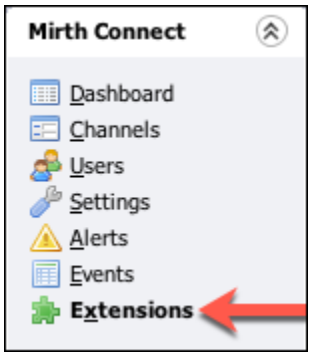
The **Extensions** view allows you to manage all plugins and connectors installed on your Mirth® Connect server. This is where you go to install any Commercial Support/Extensions *(on page 582)*. From the Extensions view you can install / uninstall / enable / disable extensions, and view properties

such as the build number or description.



Navigation

Select the **Extensions** link in the Mirth® Connect Task pane on the top-left side of the window.

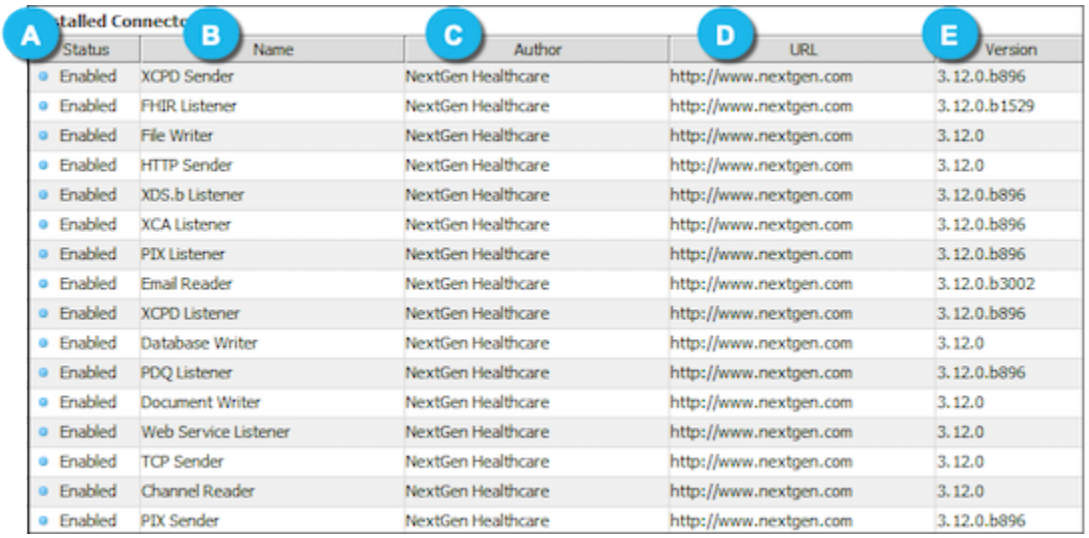


This section is separated into the following topics:

- Installed Connectors Table *(on page 198)*
- Installed Plugins Table *(on page 199)*
- Install a New Extension *(on page 200)*
- Extension Tasks *(on page 201)*

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 Mirth® Connect 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 Mirth® Connect, see Working With Tables *(on page 79)*.

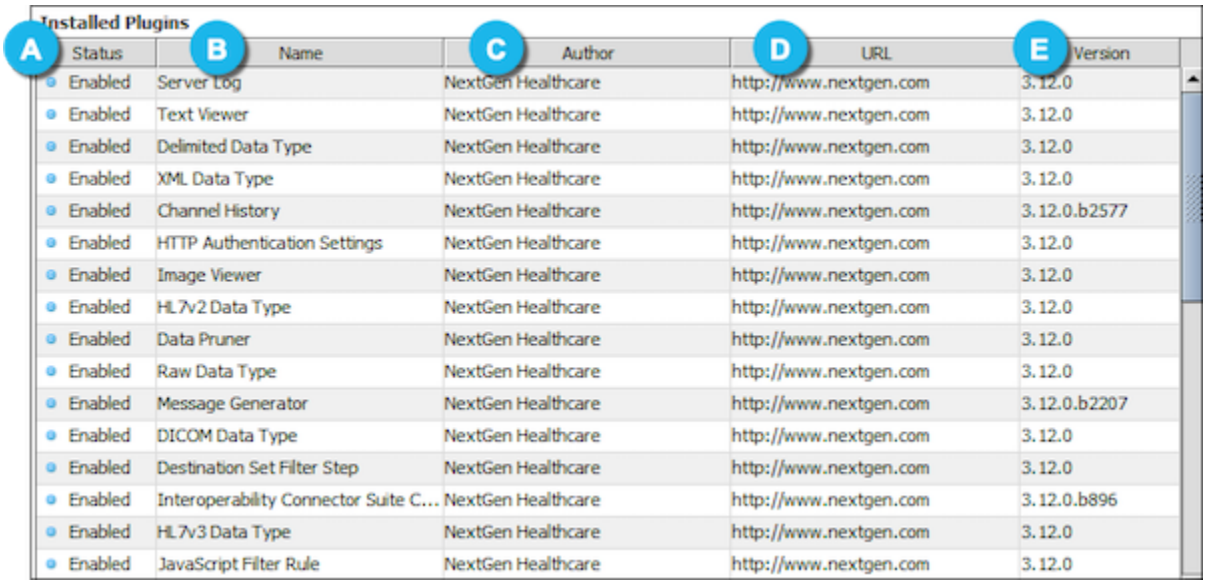


Installed Connectors Table Columns

	Column	Description
A	Status	Indicates whether the connector is enabled or disabled.
B	Name	The name of the connector.
C	Author	The development author of the connector. This will be **NextGen Healthcare** for all official extensions.
D	URL	A link to the author website.
E	Version	The version of the connector installed. This will usually be the same as the NextGen Healthcare 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 Mirth® Connect by NextGen Healthcare 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 Mirth® Connect by NextGen Healthcare, see Working With Tables *(on page 79)*.



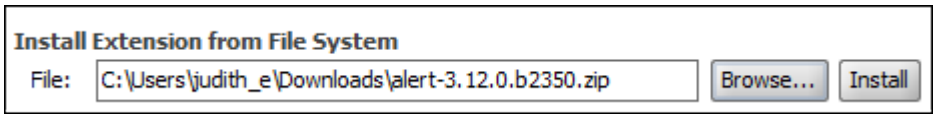
Installed Plugins Table Columns

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

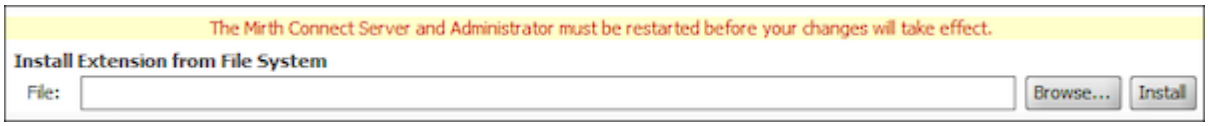
Install a New Extension

Extensions are packaged into ZIP files. To install an extension in Mirth® Connect:

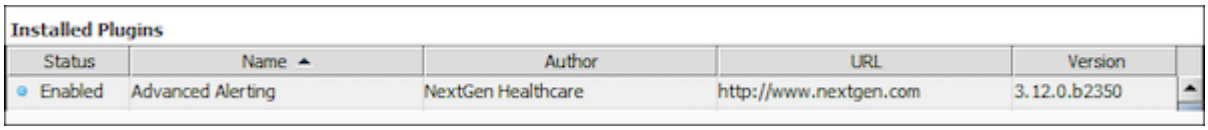
1. Navigate to the **Install Extension from File System** section at the bottom of the Extensions view.
2. Select the **Browse...** button, then select the ZIP file of the extension you would like to install.



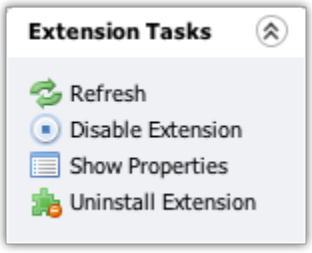
3. Select the **Install** button. If the extension ZIP file is valid, you will see a notification asking you to restart the server:



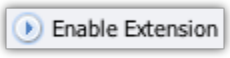
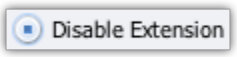
4. Restart the Mirth® Connect server and launch the Mirth® Connect Administrator *(on page 55)* again.
5. You will see the new extension listed in either the Installed Connectors Table *(on page 198)* or Installed Plugins Table *(on page 199)*, depending on the type of extension you installed.

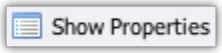
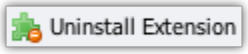


Extension Tasks



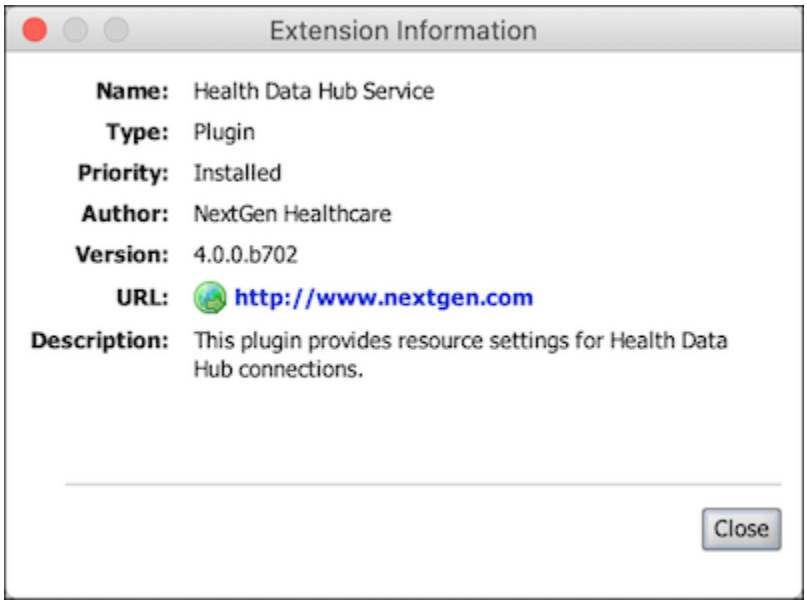
The following context-specific tasks are available from the Extensions View:

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 Mirth® Connect installation. This requires a restart of the server.
	Disable Extension	Disables the extension so that it can no longer be used in your Mirth® Connect installation. This requires a restart of the server.

Task Icon	Task Name	Description
		Warning: 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 *(on page 150)* as "invalid" if a connector or data type it was using has been disabled.
	Show Properties	Displays properties for the currently selected connector or plugin. For additional information, see Show Properties *(on page 202)*
	Uninstall Extension	Uninstalls the extension completely from your Mirth® Connect instance. This requires a restart of the server. **Warning:** 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 *(on page 150)* as "invalid" if a connector or data type it was using has been uninstalled.

Show Properties

Selecting the Show Properties task displays a dialog with more information about the extension.



Edit Views

This section is separated into the following topics:

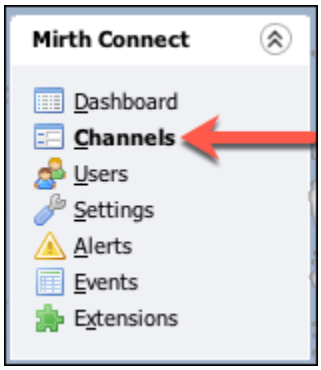
- Edit Channel View *(on page 203)*
- Edit Filter / Transformer Views *(on page 257)*
- Edit Global Scripts View *(on page 299)*
- Edit Code Templates View *(on page 301)*
- Edit Alert View *(on page 313)*

Edit Channel View

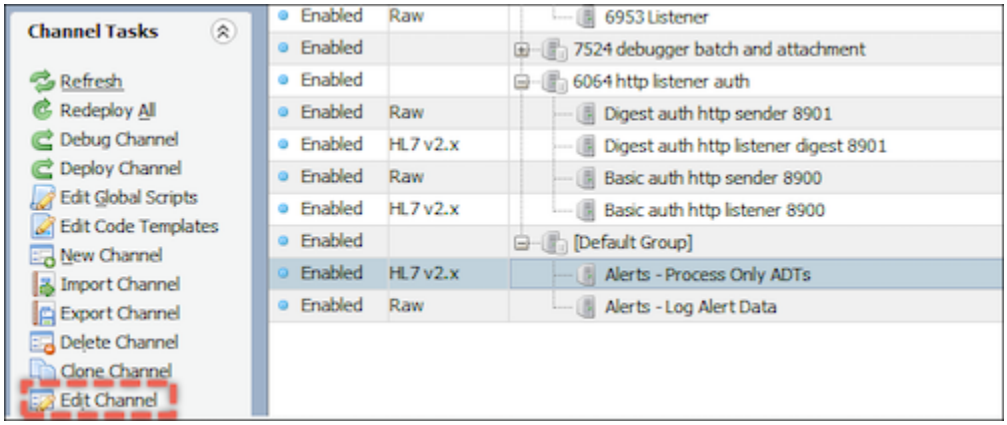
The **Edit Channel** view is where channels are configured. General settings and other properties not specific to a connector are configured on the Summary Tab *(on page 204)*. Connector-specific settings are configured on the Source *(on page 232)* and Destinations *(on page 246)* tabs. Channel-level scripts not specific to connectors are configured on the Scripts Tab *(on page 255)*.

Navigation

Select the **Channels** link in the Mirth® Connect task pane at the top-left to enter the Channels View *(on page 148)*.



In the Channel Table *(on page 150)*, s select the channel you would like to edit, then select the **Edit Channel** task to the left.



You can also double-click the channel row in the table to access the Edit Channel

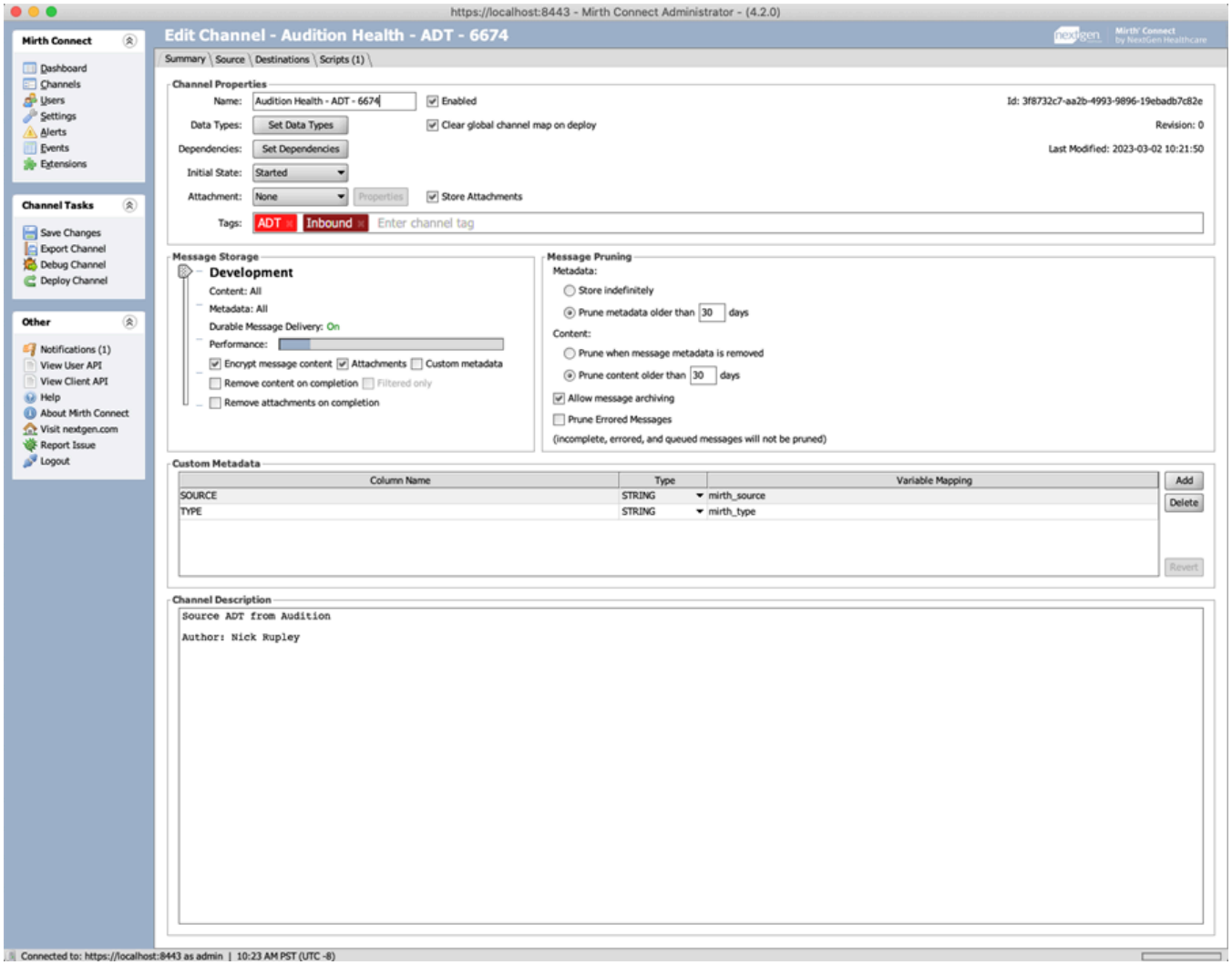
This section contains the following topics:

- Source Tab *(on page 232)*
- Edit Channel View *(on page 203)*
- Destinations Tab *(on page 246)*
- Scripts Tab *(on page 255)*
- Edit Channel Tasks *(on page 256)*

Summary Tab

The Summary tab is what you first see when entering the Edit Channel view. It is where general settings

not specifically tied to connectors are configured, like the channel name, initial state, storage settings, and description.



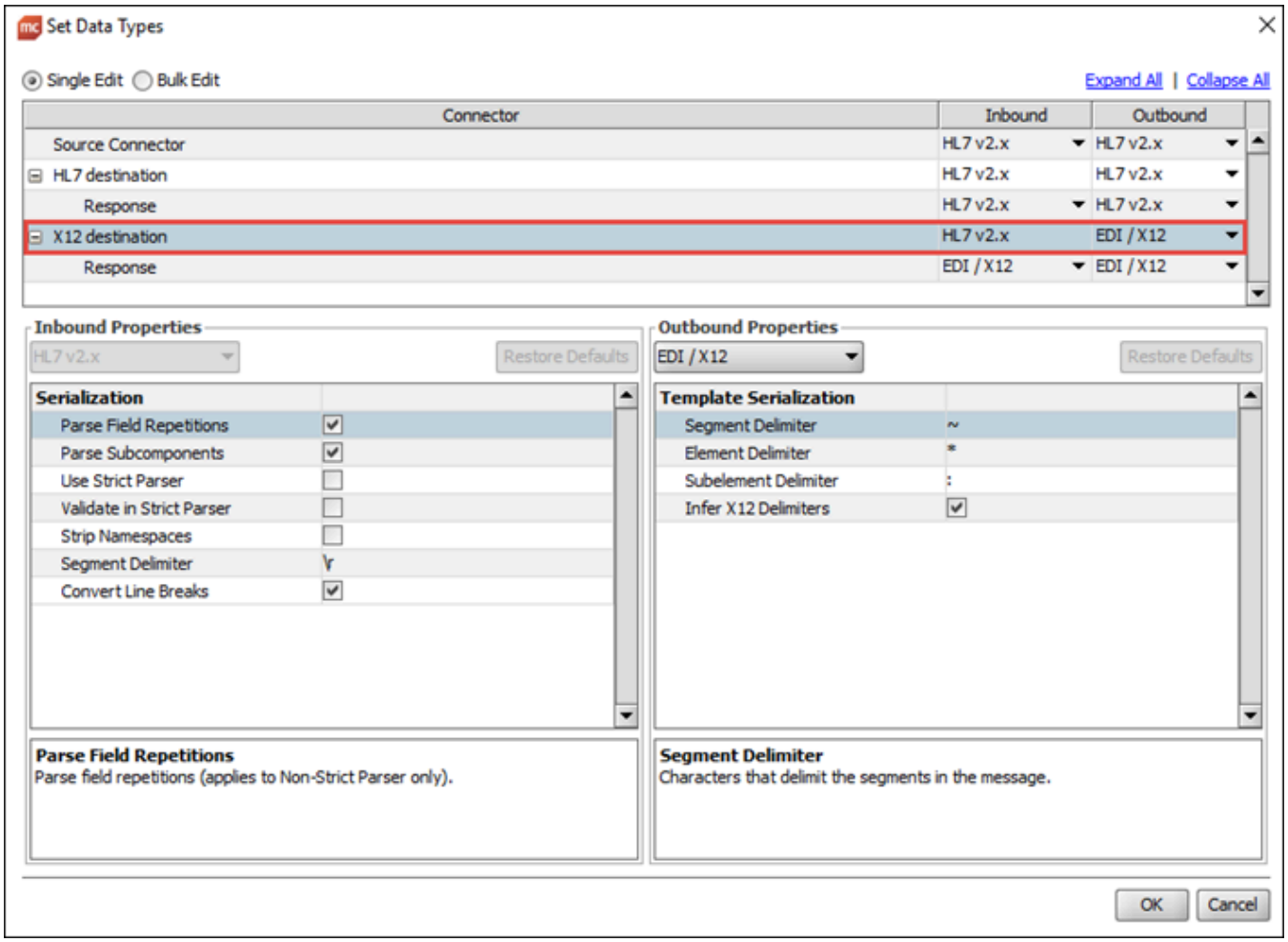
Channel Properties

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 Mirth® Connect by NextGen Healthcare server starts up. If there is an error in your channel configuration, this will automatically be unchecked.
C	Set Data Types Dialog	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 *(on page 207)*.
D	Clear global channel map on deploy	If checked, the global channel map *(on page 449)* for this channel will be cleared out whenever the channel is redeployed.
E	Set Dependencies	Configures Code Template Libraries *(on page 213)*, Library Resources *(on page 214)*, and Deploy/Start Dependencies *(on page 215)* for this channel. For more information, see Set Dependencies Window *(on page 212)*.
F	Initial State	Determines what state the channel should be in after it is deployed. • **Started**: The entire channel (source and destination connectors) are started. • **Paused**: Only the destination connectors is started. The channel does not automatically receive new messages, but existing queued messages will still dispatch outbound. • **Stopped**: The channel is stopped. No messages are received or dispatched outbound.
G	Attachment	Determines how the channel extracts and stores attachments. Each attachment handler does this in a different way. Select the handler to use from the drop-down menu, then select the **Properties** button to edit the configuration for that handler. For additional information, see Attachment Handlers *(on page 219)*.

Item	Name	Description
H	Store Attachments	• If checked, any attachments that are extracted are stored in the database and available for reattachment. • If unchecked, attachments can 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. You can also type a new tag name and hit **Enter** to create a new tag. For additional information, see Filter By Channel Name or Tag *(on page 98)* When new tags are created here, the color is chosen at random. This may be changed later in the tag management view (see Tags Settings Tab *(on page 179)* for additional information).
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 automatically increments 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 Window

This is the main management window for all data types configured across your entire channel. You can see at a glance and easily configure the data formats you are expecting to receive, the formats you are converting the data into for each outbound destination, and the response formats you are receiving from external systems. You can also configure various properties for each data type, which determine how your data can be used inside of filters and transformers and elsewhere.

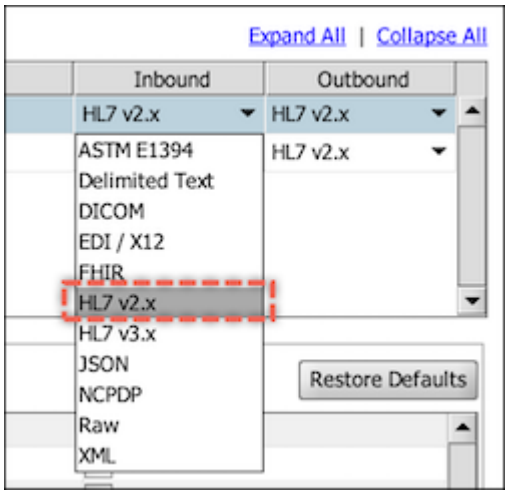


Select Data Types

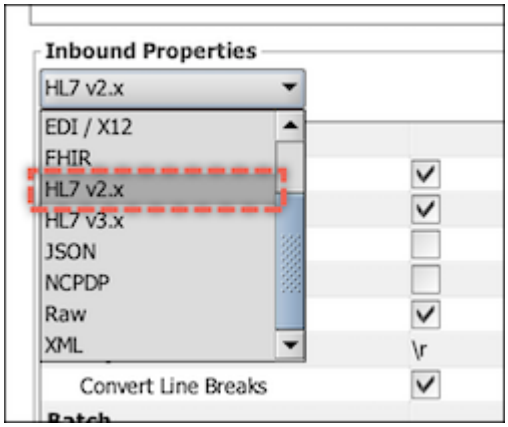
The table at the top of the dialog shows all currently configured connectors for your channel. Since response transformers have their own inbound / outbound data types, for destination connectors you can expand the node in the tree and view the Response row underneath a particular destination.

For a new channel, all data types are set to HL7 v2.x *(on page 329)* by default. This can be changed in two ways:

- The Connector table has Inbound and Outbound columns with drop-down menus. Select the drop-down arrow inside of the cell to make a selection:



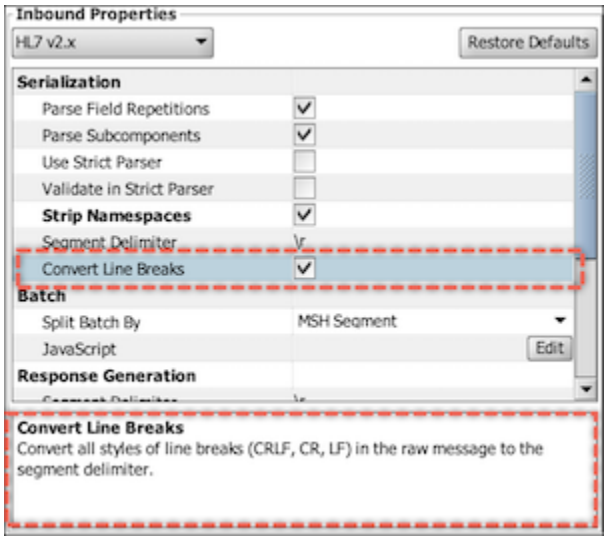
- Use the drop-down menu in the **Inbound Properties** or **Outbound Properties** sections:



Note: 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.

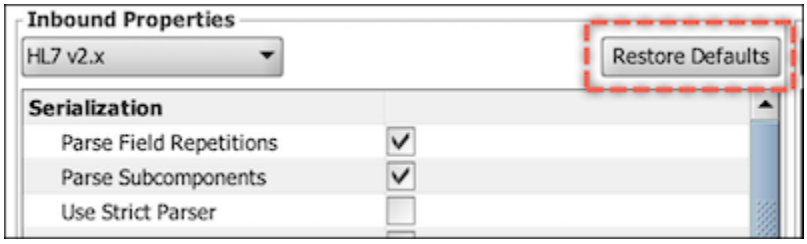
Change Data Type Properties

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. Select 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 is being used as inbound or outbound, the categories of properties shown in the table vary. For additional information on data type properties and options for specific data types, see Data Types *(on page 323)*.

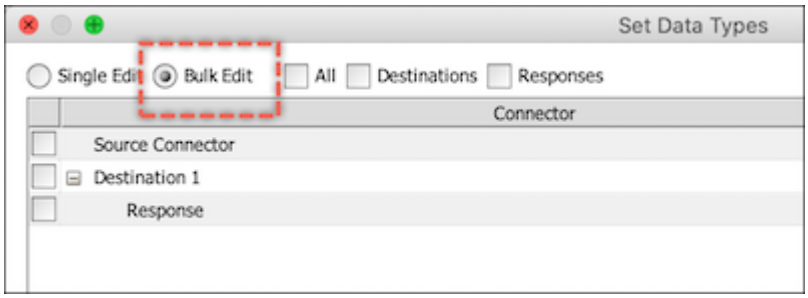
Any properties you have changed or that are not equal to the default value are shown in **bold**. To revert all changes back to the defaults for the selected data type, select the **Restore Defaults** button above the **Properties** tables:



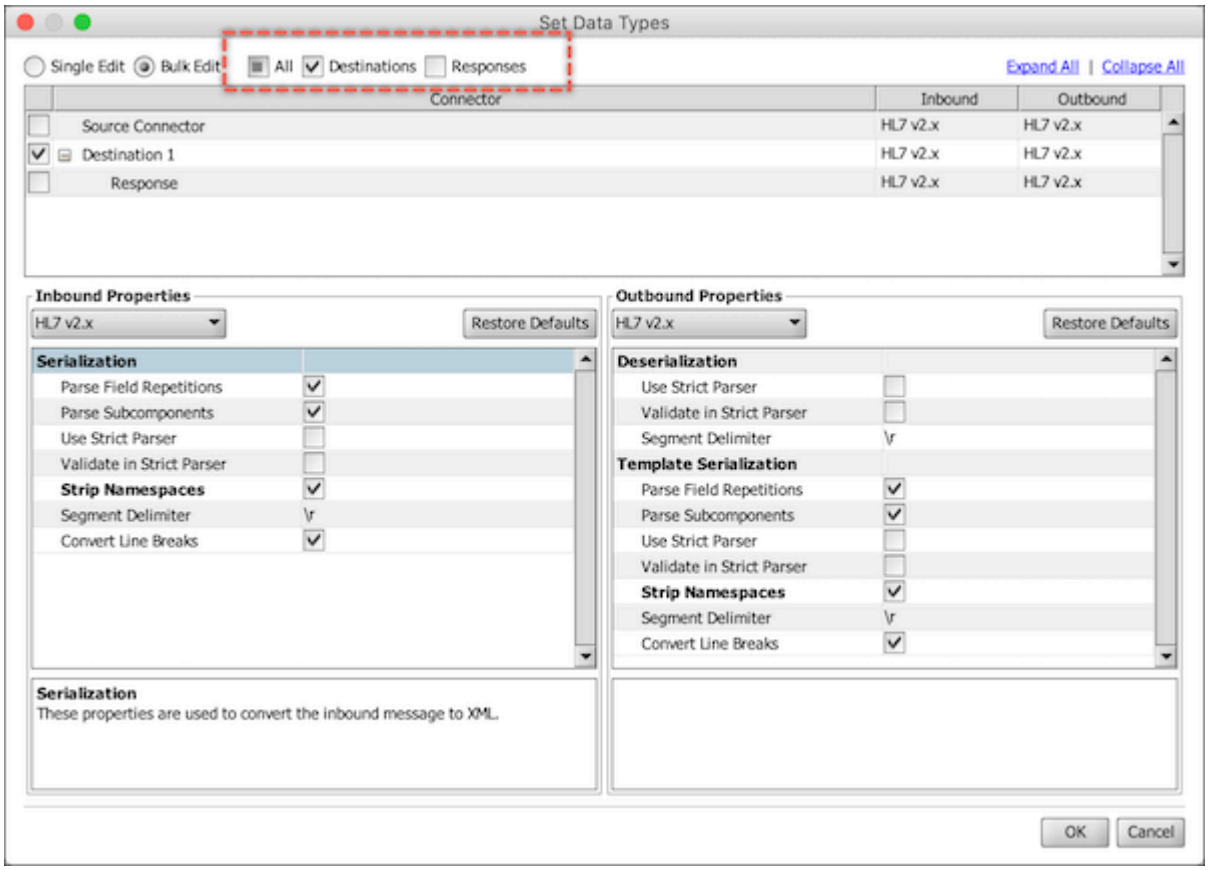
Bulk Edit Mode

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. Select the **Bulk Edit** button on the top-left side 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 have selected at least one, the Inbound / Outbound Properties sections below will become activated.



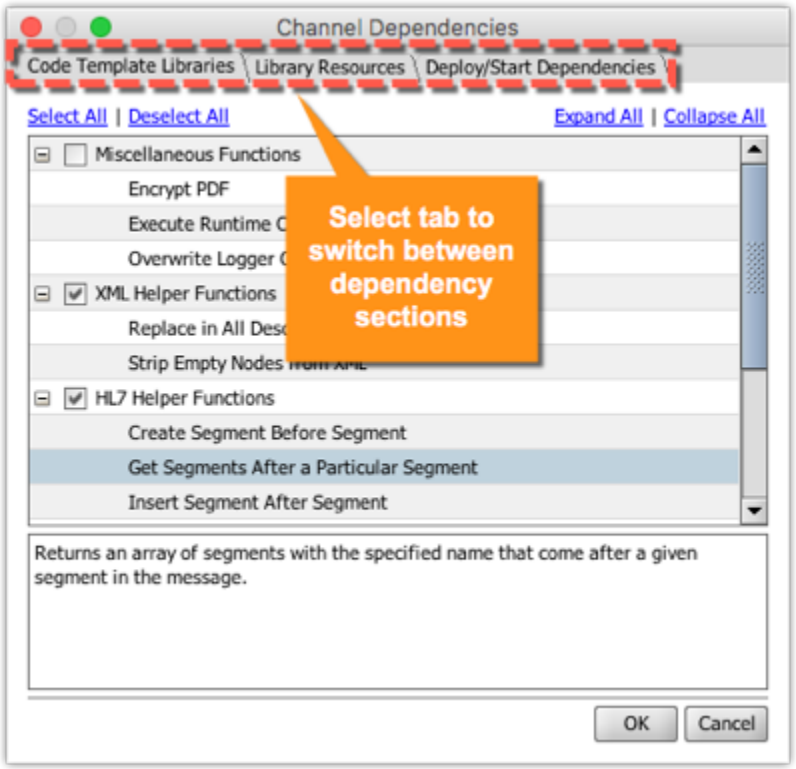
- For either the **Inbound** or **Outbound** Properties section, if the data type is the same across all of the currently selected rows, then you have the option to edit properties. Any changes you make while in this mode will be applied across *all* of the currently selected connector / response rows.

- 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.

4. The **All**, **Destinations**, and **Responses** check boxes at the top are convenient ways to select multiple types of rows at once.
5. Select **OK** to save your changes.

Set Dependencies Window

The Set Dependencies window enables you to configure external dependencies for your channel and connectors. This may include Code Template Libraries or Library Resources you want to use inside your channel, or even other channels that should be labeled as dependents/dependencies of this channel.



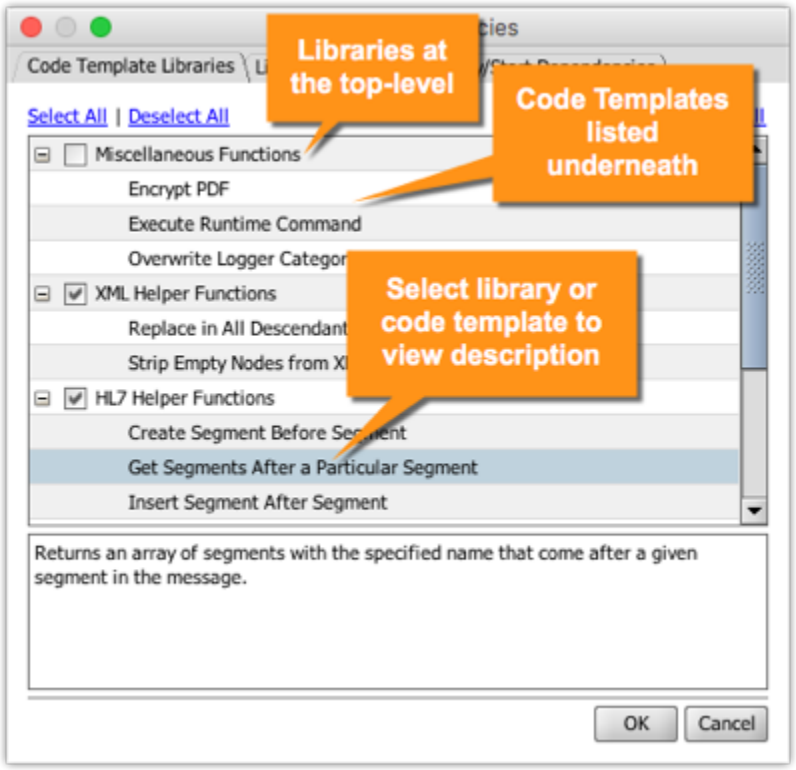
The Set Dependencies window is separated into the following sections:

- Code Template Libraries
- Library Resources

- Deploy/Start Dependencies

Code Template Libraries

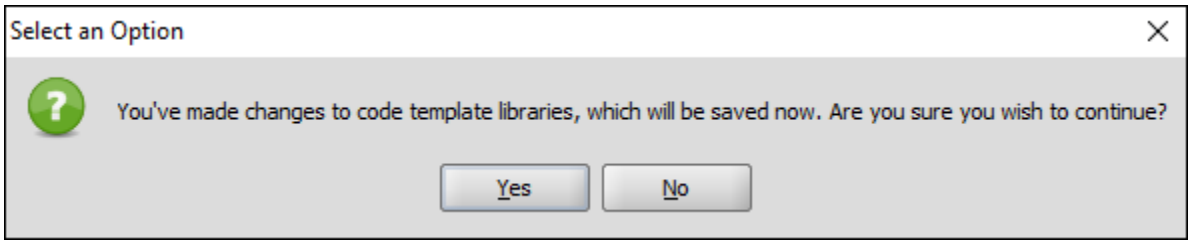
This tab allows you to link code template libraries to the current channel you are editing. Note that this accomplishes the same thing as including channels on the Edit Library Panel *(on page 304)* within the Edit Code Templates View *(on page 301)*. It is included here as well within the Edit Channel View *(on page 203)* for convenience.



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

Link Code Template Libraries

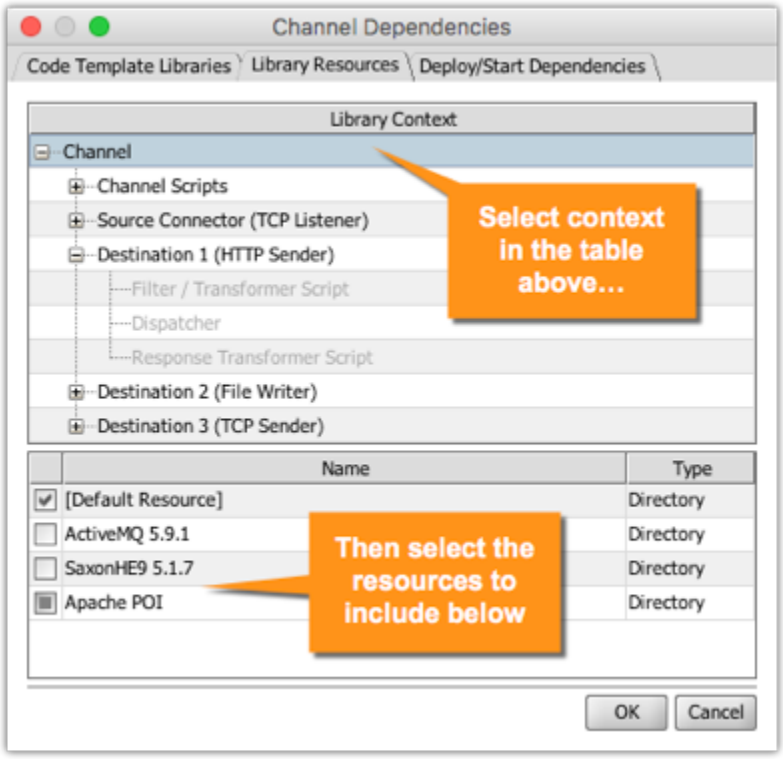
To include a library of code templates on the current channel, check the check box next to the library name, then select the **OK** button. A confirmation dialog is shown:



Because the 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. Select **Yes** to accept the changes and exit the dialog.

Library Resources

Resources *(on page 186)* are shared services that can be used in specific channels / connectors or in other places throughout the Mirth® Connect server. They may include custom Java libraries to use within scripts, or services to handle outbound connections. The **Library Resources** tab allows you to include resources throughout your entire channel, or in more specific contexts such as the channel scripts or certain connectors.



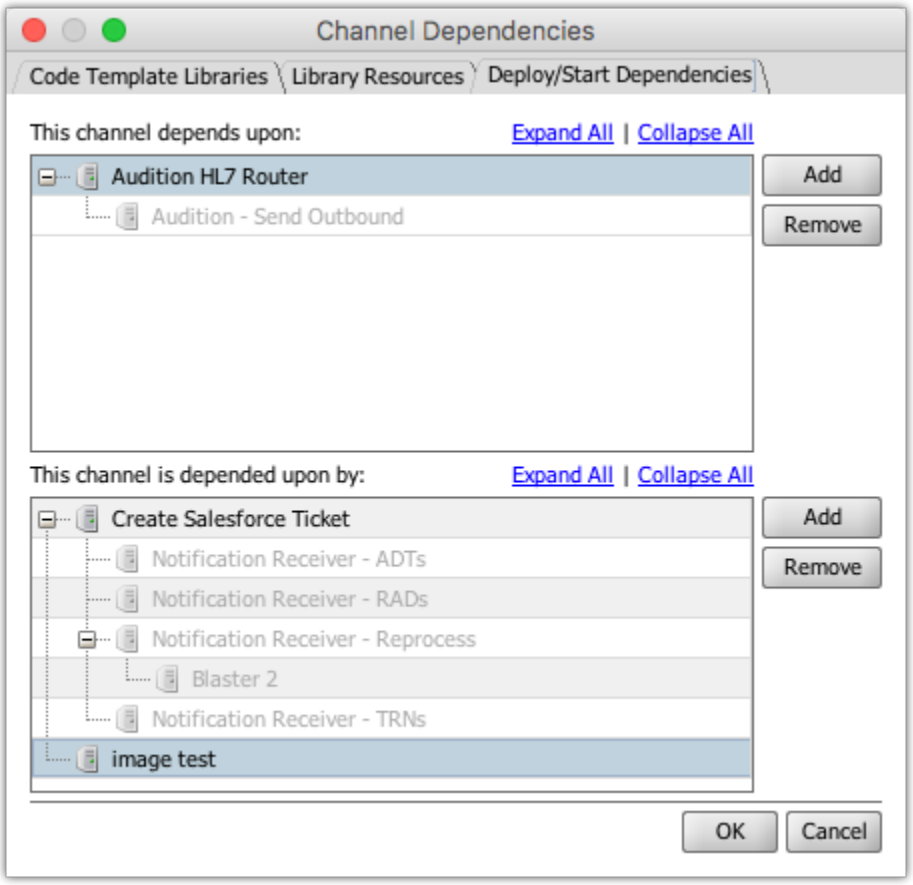
The **Library Context** tree in the top section of the tab allows you to select a specific context, or the

Channel root-level node which represents **all** 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 are using a Database Reader *(on page 346)* 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 are using a JavaScript Writer *(on page 410)* 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**.

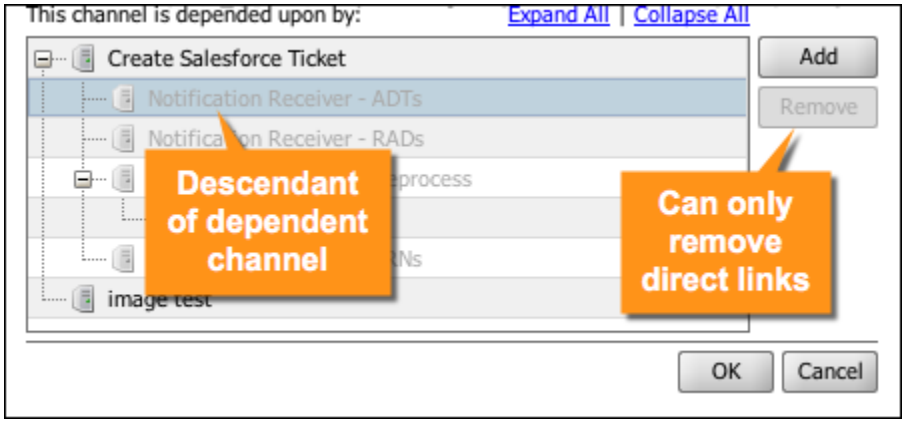


The tree in the top-half of the dialog shows **dependencies of** the current channel. In other words, channels that show up in the top tree are those that the current channel depends upon. By expanding nodes in this tree, you can see descendant dependencies, or "dependencies of the dependencies".

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".

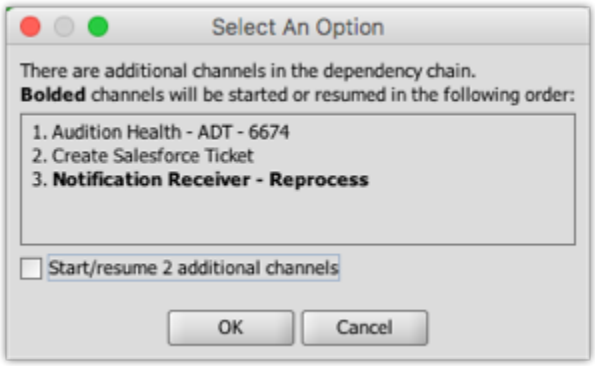
Select the **Add** / **Remove** buttons next to either tree to add or remove direct links.

Note: 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:



Deploy/Start Channels

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

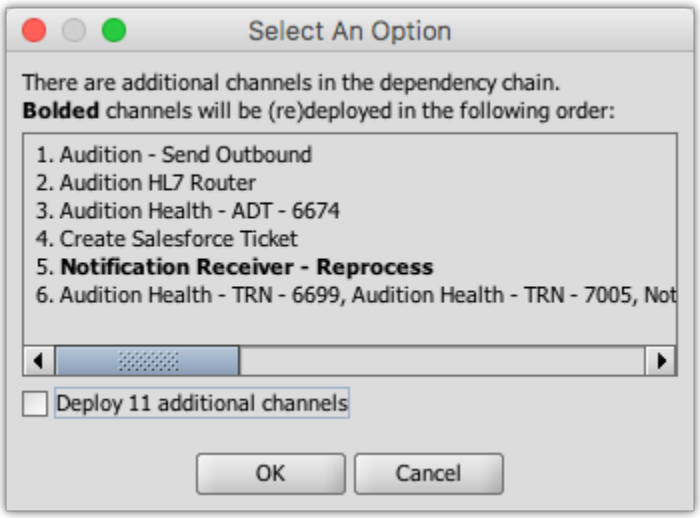


By default only the channels you have selected will be **bolded** in the dialog, meaning that only they will be started. If you want to start the other channels in the dependency chain as well, check the " **Start/ resume # additional channels** " check box before selecting the **OK** button. Upon pressing **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) are 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:

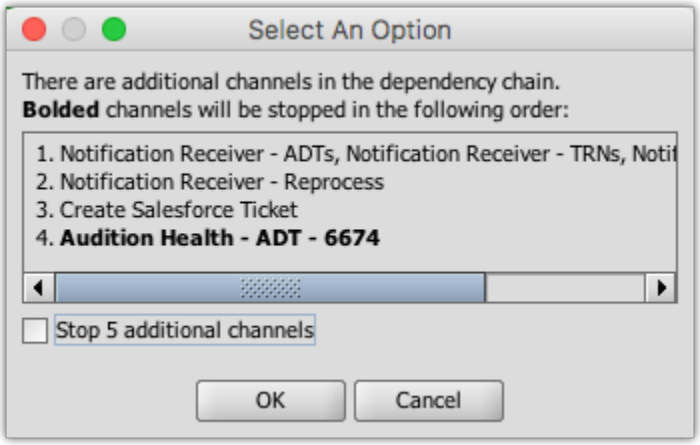


This is because deploying a channel may entail **undeploying** any selected channels first, and then **redeploying** them. So when (re)deploying a particular channel, this sequence is followed:

- 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.

Pause/Stop/Undeploy 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 have 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 selecting the **OK** button. Upon selecting **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 *(on page 205)* section of the Summary Tab *(on page 204)* within the Edit Channel View *(on page 203)*.

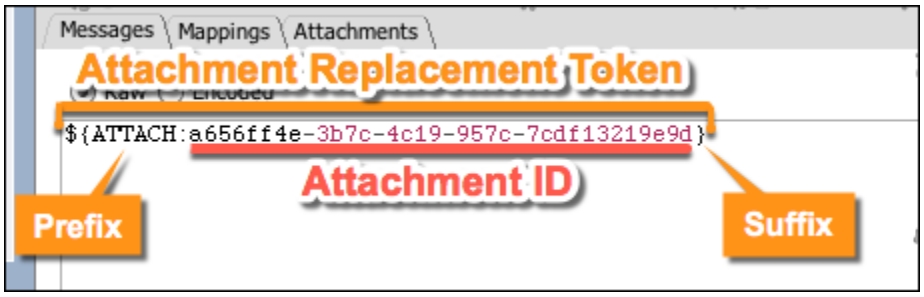
Note: See Attachment JavaScript Functions *(on page 455)* for more information.

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 select the **Properties** button to configure the handler.

Note: 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:



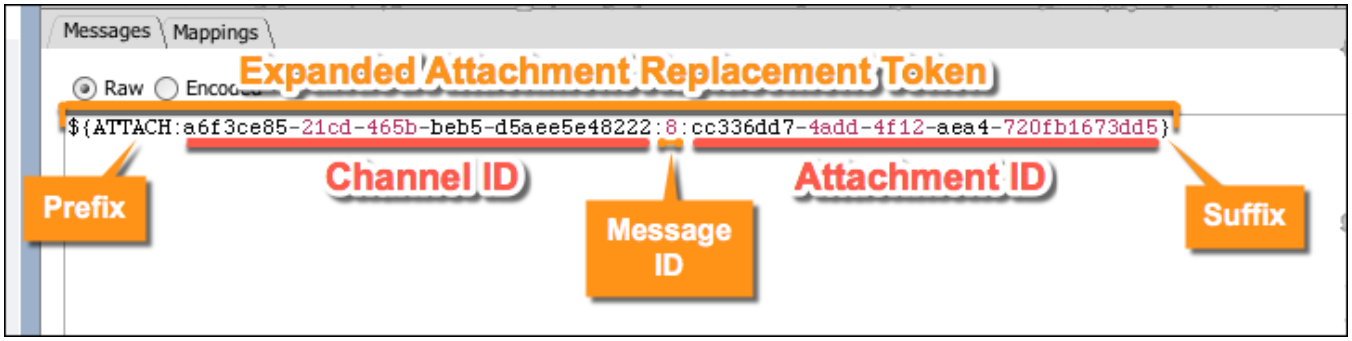
This value tells the destination connector where in your message the attachment should be re-inserted right before dispatching to an outbound system. If multiple attachments were extracted for a message, then there will be multiple replacement tokens in the raw data.

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 *(on page 249)*.

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 *(on page 249)*, 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 *(on page 118)* section, there are four types of attachment viewers in the Message Browser *(on page 103)*: Text, Image, DICOM, and PDF. The **type s** 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 are 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 is displayed in the message browser.

Attachment Handler Properties

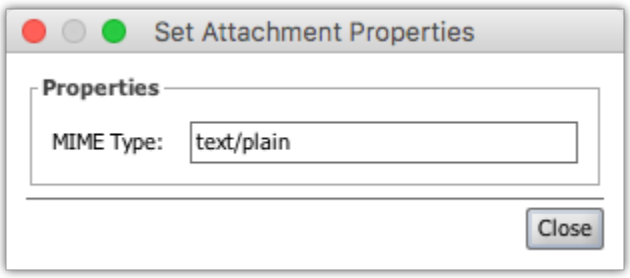
The following attachment handlers are supported:

- Entire Message Attachment Handler Properties *(on page 221)*
- Regex Attachment Handler Properties *(on page 222)*
- DICOM Attachment Handler Properties *(on page 224)*
- JavaScript Attachment Handler Properties *(on page 224)*
- Custom Attachment Handler Properties *(on page 226)*

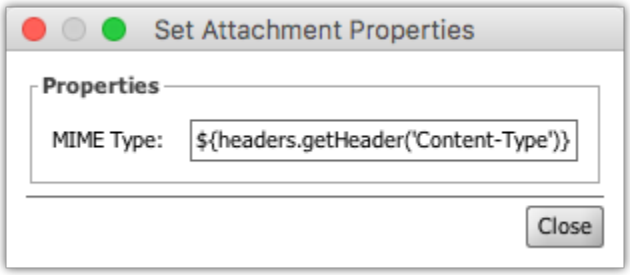
Entire Message Attachment Handler Properties

This attachment handler takes the entire incoming message data and stores it as a single attachment. The

Raw message data afterwards is the attachment token.

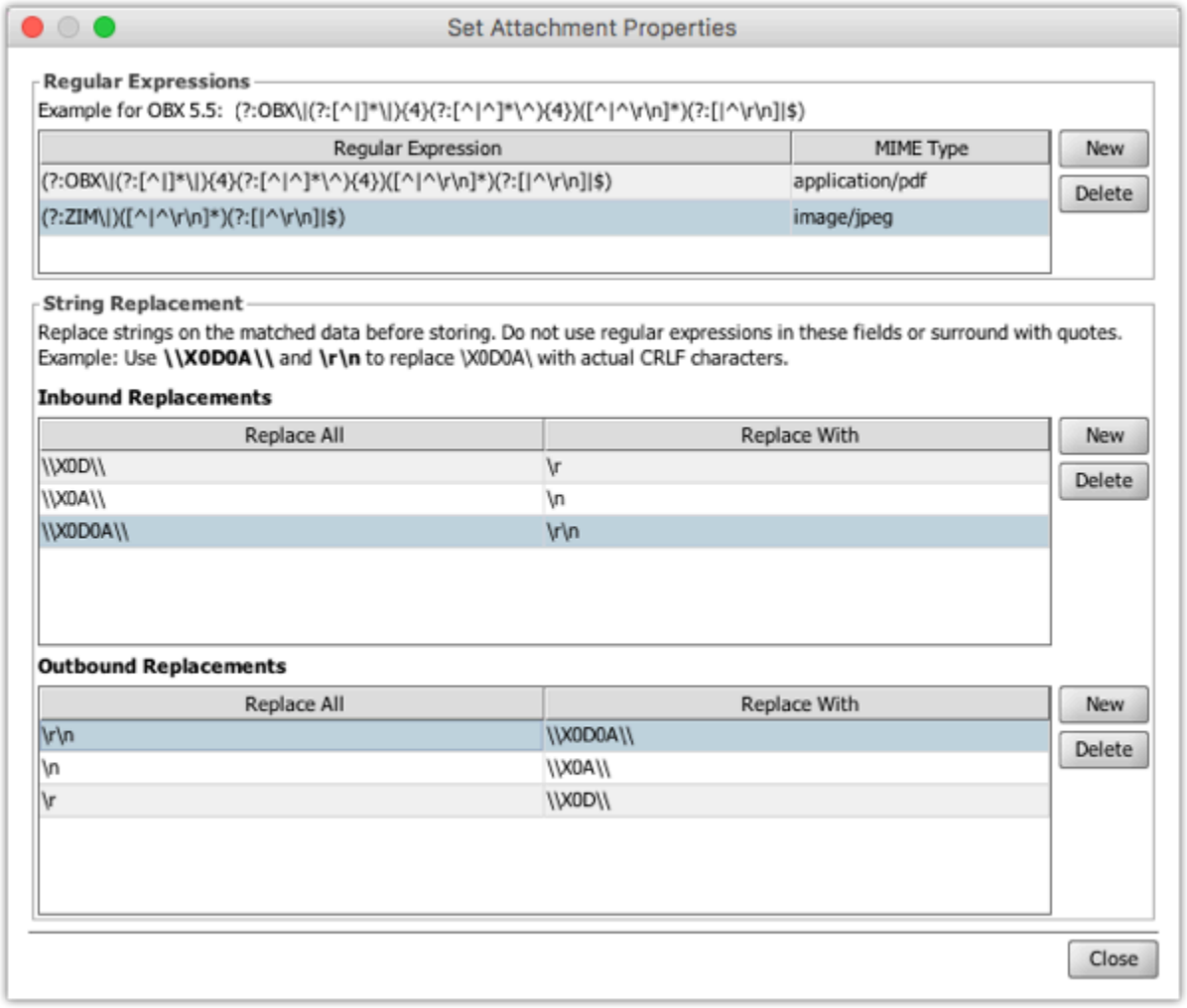


The handler has a single property, **MIME Type**, which specifies what type of attachment data you expect to receive. You can use a specific value like "text/plain", or you can use Velocity replacement *(on page 469)* to inject source map variables. For example if you are using an HTTP Listener *(on page 362)*, 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 then 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]|$)`

Select 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 *(on page 469)*, 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 is 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.

Note: Standard Java String escape characters apply here. For example to replace a backslash, you will 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 automatically takes 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 *(on page 327)* is used in a filter / transformer, it will automatically serialize this Base64 data into an XML message containing all header / tag data.

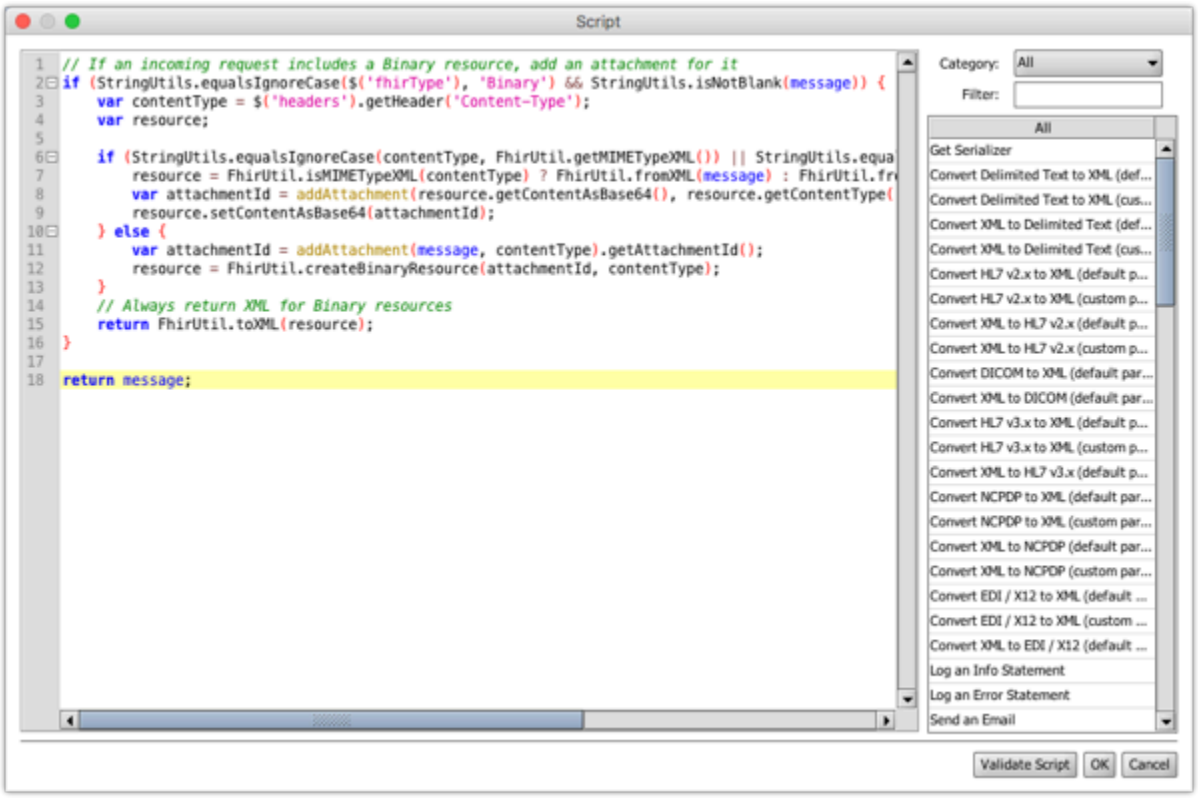
Because there is no attachment replacement token, to reattach DICOM messages on the destination connector side, a special token is used:

- **${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 Mirth® Connect by NextGen Healthcare, see Mirth Connect and JavaScript *(on page 428)* and Attachment JavaScript Functions *(on page 455)*.



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 *(on page 449)*.

Extract Attachments

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

The resulting Attachment object contains the ID you need to inject back into the message. For more information, look at the User API *(on page 459)*. 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.

addAttachment(data, type)

Creates and stores an attachment for the current message.

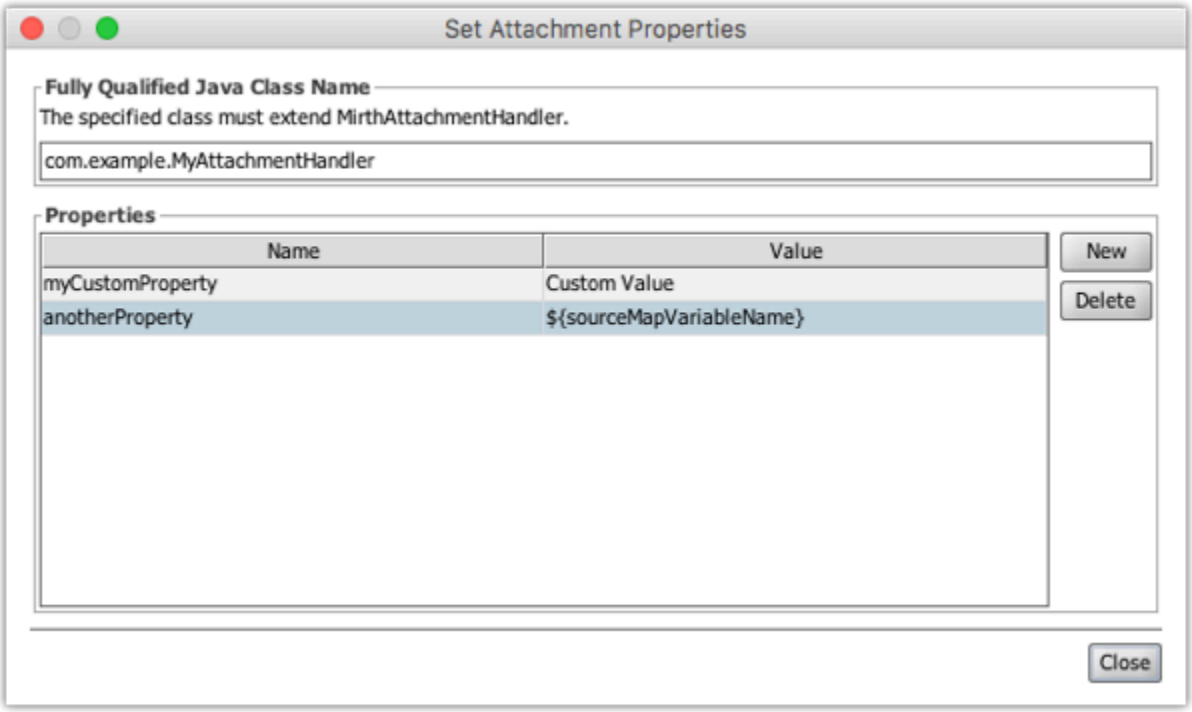
Table 1. Parameters

Name	Type	Description
data	String or Byte array	The actual attachment data to insert. Man be either a String or a byte array. If a string is used, it is 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 *(on page 219)*.
<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 The User API (Javadoc) *(on page 459)*. 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.

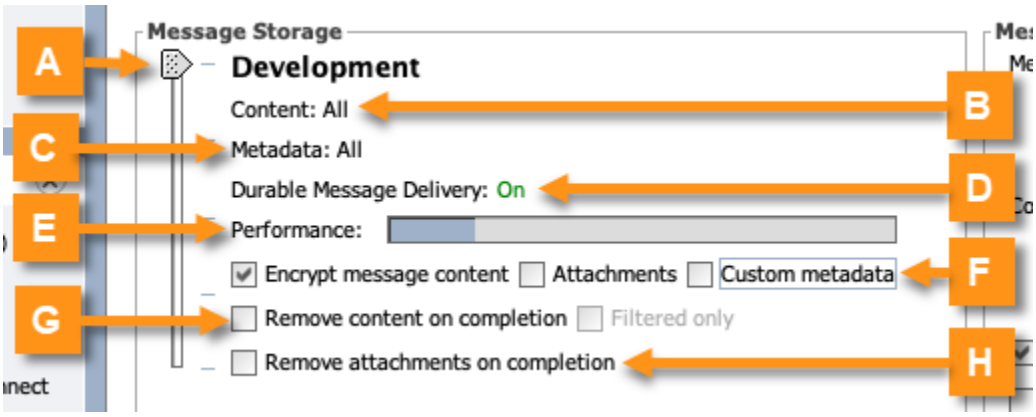


Properties

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 *(on page 204)* enables 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.

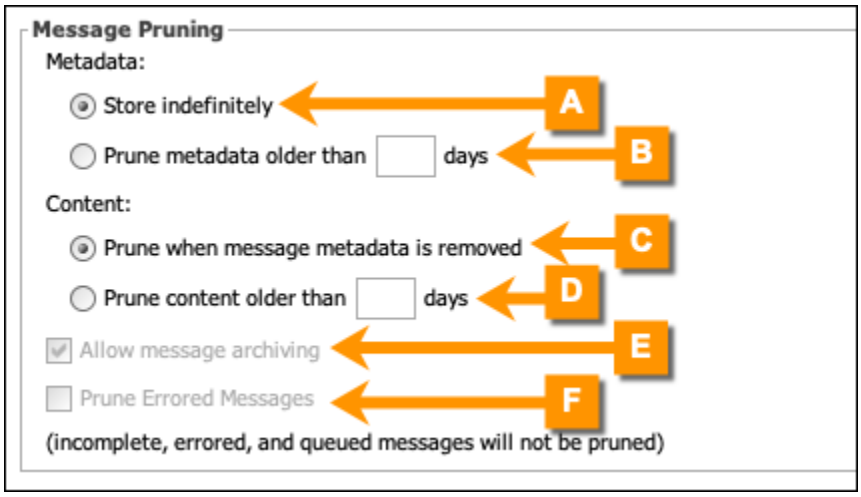


Item	Name	Description
A	Storage Slider	Use this slider bar to change how much data to store as messages process through the channel. The options are: • **Development:** All data will be stored. • **Production :** Everything except Processed Raw, Transformed, Response Transformed, and Processed Response 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 *(on page 65)*) . 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 is halted, or if the server suddenly goes down for any reason.

Item	Name	Description
		• 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. • **Attachments**: Encrypt message attachments that are stored in the database. Attachments that are stored while this option is enabled will still be viewable in the message browser. • **Custom metadata**: Encrypt custom metadata columns that are stored in the database. Custom metadata values that are stored while this option is enabled will still be viewable in the message browser, but the metadata will not be searchable. This will only apply to STRING type custom metadata columns.
G	Remove content on completion	Removes message content after 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	Removes message attachments after the message has completed processing. Not applicable for messages that are errored or queued.

Message Pruning Settings

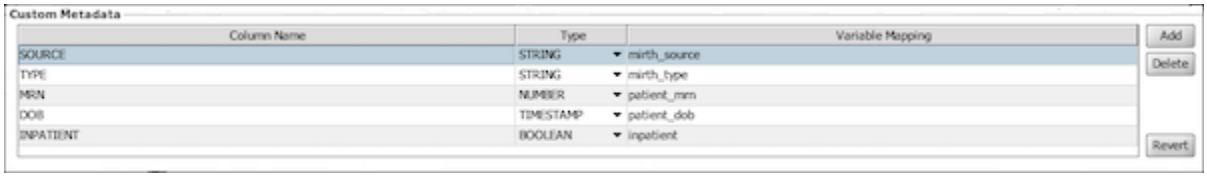
These settings allow you to decide when to prune and archive message data (if at all). Note that pruning does not happen automatically unless you enable the scheduler in the Data Pruner Settings Tab *(on page 190)*. For more information about message data in general, see About Message Data *(on page 65)*.



Item	Name	Description
A	Store indefinitely	When selected, message metadata are never pruned.
B	Prune metadata	When selected, this field determines how long to retain message metadata before it is pruned.
C	Prune when metadata is removed	When selected, message content follows the same pruning options set in the Prune metadata field.
D	Prune content	When selected, the field here determines how long to retain message content before it is pruned.
E	Allow message archiving	When checked and the data pruner and archiver are enabled, messages in this channel are archived first before they are pruned.
F	Prune Errored Messages	When checked and the data pruner is enabled, errored messages in this channel are pruned.

Custom Metadata Columns

Custom metadata columns are user-created, searchable columns that appear in the Metadata Table *(on page 106)* in the Message Browser *(on page 103)*. As messages process through a channel, these columns are populated from the Variable Maps *(on page 449)* using a configurable map key.



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	The type of column to create. • **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	The key used to look up values in all available variable maps for each connector. As explained in the Variable Maps *(on page 449)* section, the following maps (if available) will be checked in this order: • Response • Connector • Channel • Source • Global Channel • Global • Configuration

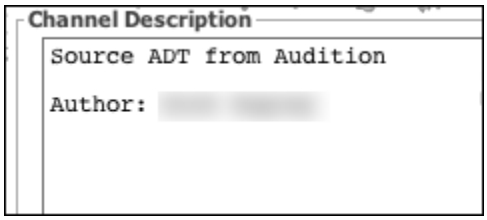
Modifying Custom Metadata Columns

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**.

Warning: 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.

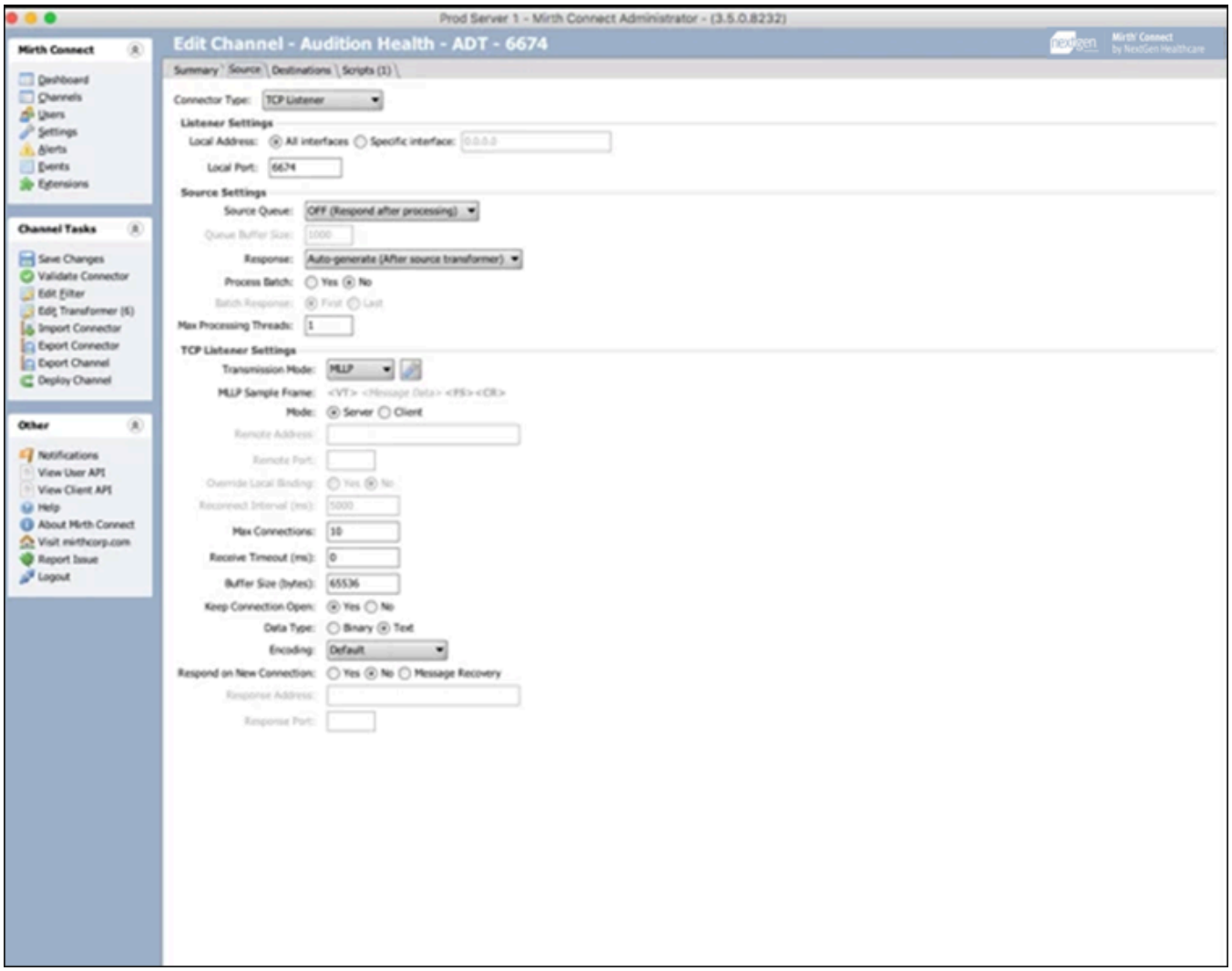
Channel Description

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.



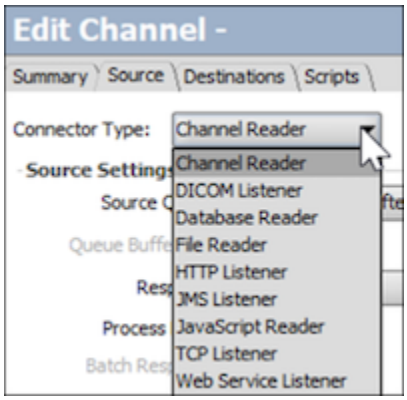
Source Tab

The Source tab 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:

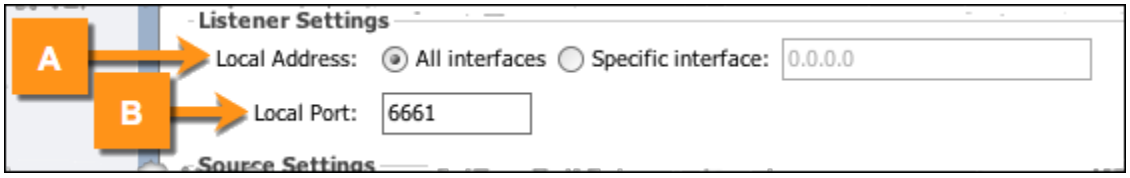


After selecting a source connector type, the settings shown here are broken up into different categories depending on the type of connector chosen:

- Listener Settings *(on page 233)*
- Polling Settings *(on page 234)*
- Source Settings *(on page 238)*
- HTTP Authentication Settings *(on page 239)*
- Source Connector Properties *(on page 245)*

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 *(on page 341)*, HTTP Listener *(on page 362)*, TCP Listener *(on page 372)*, and Web Service Listener *(on page 381)*.

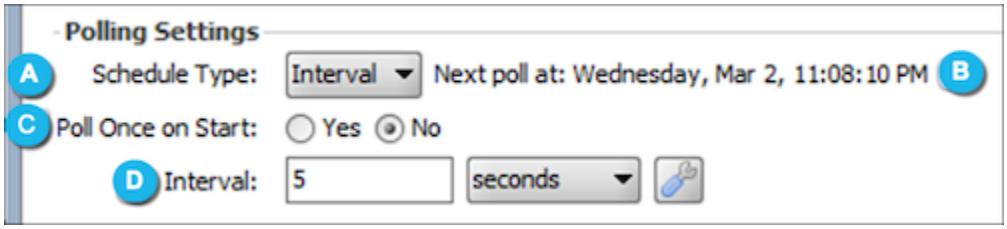


Item	Name	Description
A	Local Address	If **All interfaces** is selected, the connector will listen on all interfaces, using address 0.0.0.0. If **Specific interface** is

Item	Name	Description
		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.

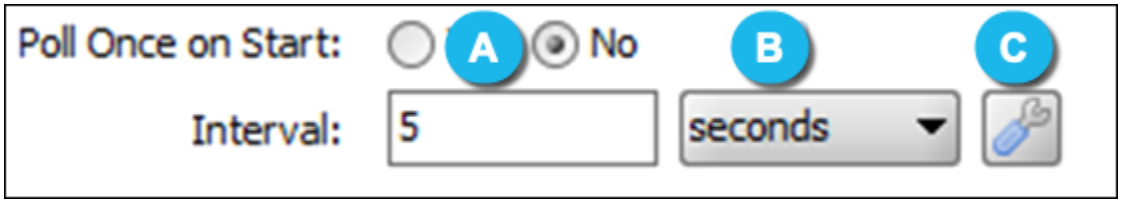
Polling Settings

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: • 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: • Milliseconds • Seconds • Minutes • Hours
C	Advanced	Enables configuring day-of-week / day-of-month settings and time ranges.

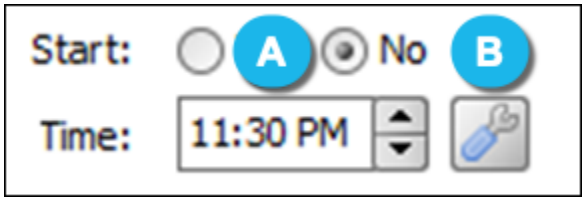
Note: Only one polling window is ever active at any given time for a given source connector. For example, if you have a channel set to poll every 5 seconds, but a single message takes 6 seconds to process, a new polling job will not begin while the first one is still working. In this example the "effective"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

Note: 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 leave the advanced settings as their defaults (All Day range), and choose 1 hour for your interval.

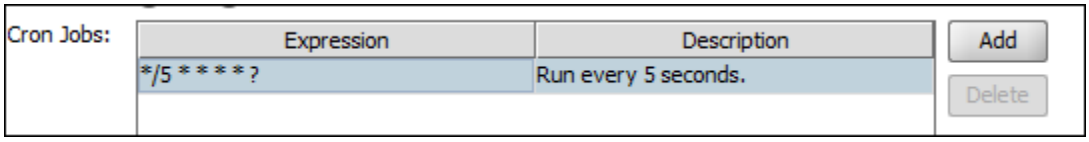
Example: To poll every hour on the 30 minute mark (12:30, 1:30, etc.), leave the interval as 1 hour, and set the Active Time Range to 12:30 AM - 11:30 PM.

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 *(on page 237)*.

Cron Schedule Settings



This schedule type gives you full control to enable advanced, nuanced polling schedules using Cron-like expressions. Select 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	, - * /

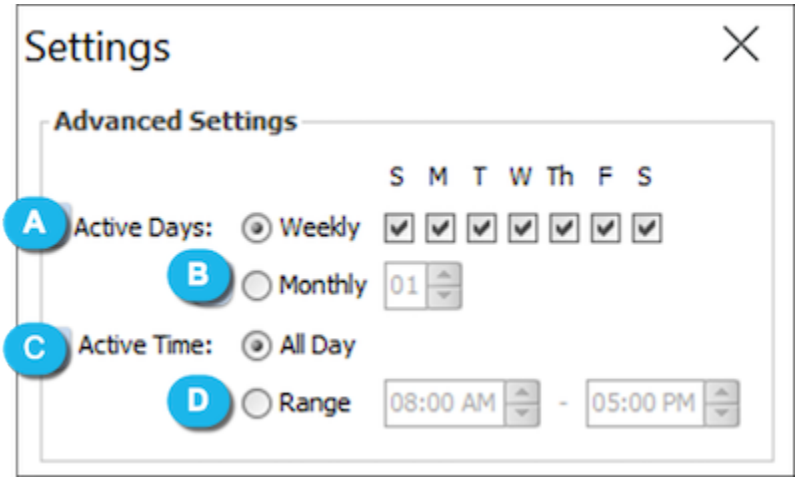
Special Characters

*	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.

Note: 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

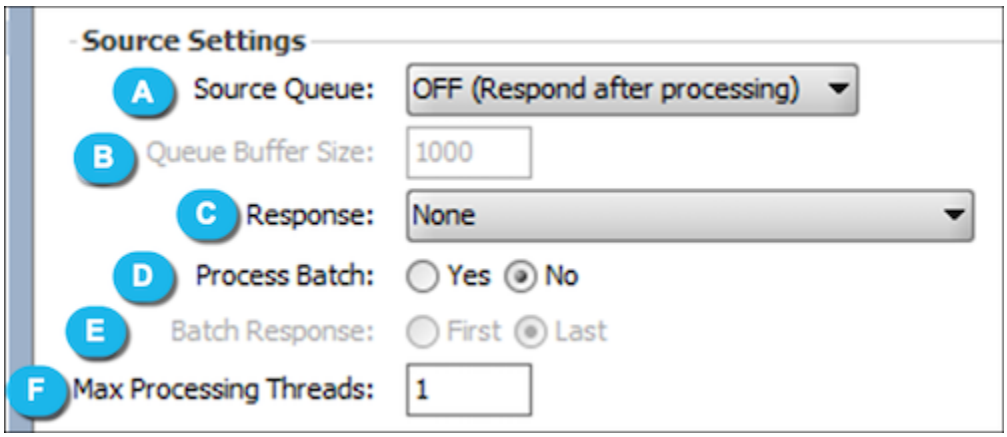


Item	Name	Description
A	Active Days: Weekly	Select this option to only poll on the days of the week that are checked to the right.
B	Active Days: Monthly	Select this option to only poll on the specified day of the month.

Item	Name	Description
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.

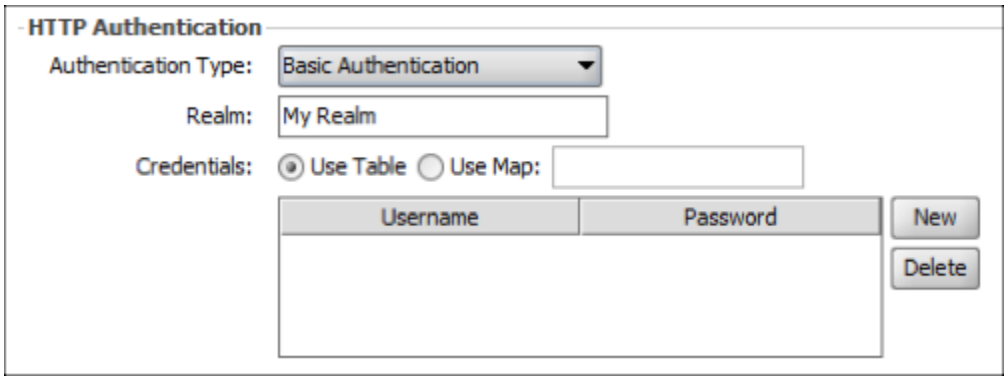


Item	Name	Description
A	Source Queue	This determines whether the source queue is enabled. It also determines **when** the selected response will be sent back to the originating system. • **OFF (Respond after processing):** Selecting **OFF** will process the message before sending the response. In this 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.

Item	Name	Description
C	Response	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 do not want to send a response back at all.Select **Auto-generate** to send a response generated by the inbound data type using the raw message: • **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 **ERROR**, **QUEUED**, **SENT**, **FILTERED**.
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. For additional information see Batch Processing *(on page 337)*
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	The maximum number of messages that can be processed through the channel simultaneously. By default this is set to 1, meaning that only one message can bd 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. **Note:** When this value is greater than 1, message order is **NOT** guaranteed.

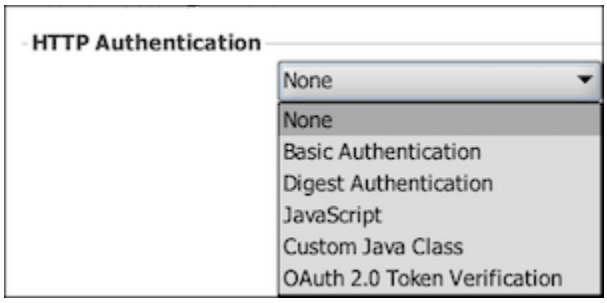
HTTP Authentication Settings

The HTTP Authentication Settings provide automatic user authentication with a variety of supported mechanisms. for HTTP-based source connectors.



Choose an Authentication Type

Select an option from the drop-down window.



Once an HTTP Authentication type is selected, fields associated with the type appear. For additional information on each HTTP Authentication type, select a links below.

- Basic HTTP Authentication *(on page 240)*
- Digest HTTP Authentication *(on page 241)*
- JavaScript HTTP Authentication *(on page 243)*
- Custom Java Class HTTP Authentication *(on page 244)*
- OAuth 2.0 Token Verification *(on page 245)*

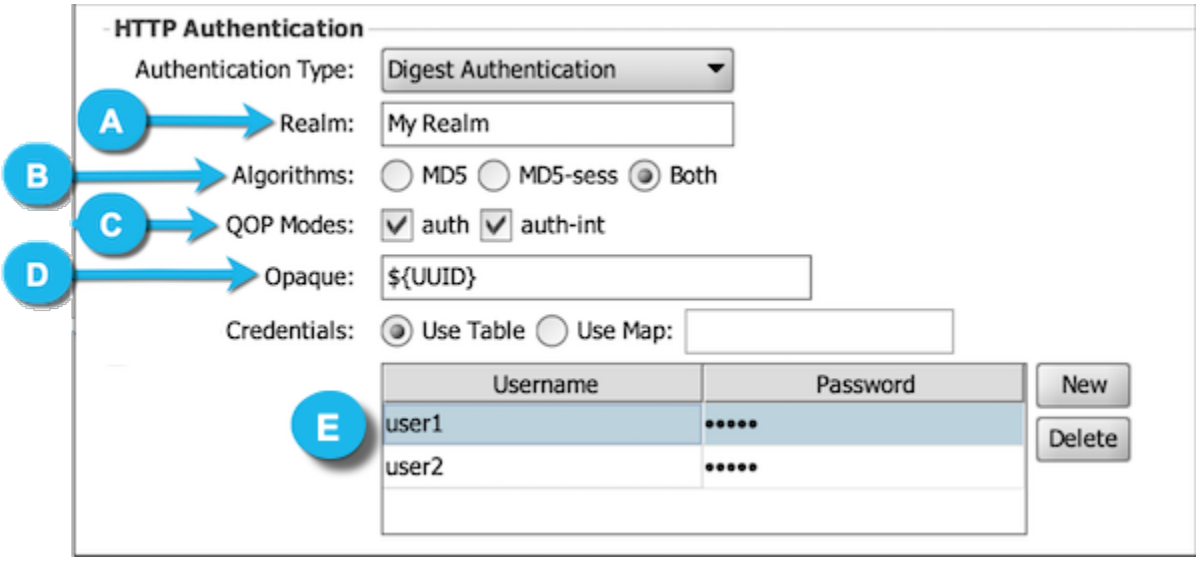
Basic HTTP Authentication

Provides Basic Authentication support for HTTP-based source connectors.

Item	Name	Description
A	Realm	The protection space for this server.
B	Use Table	• **Use Table:** When selected, use the Credentials table **(B)** to populate credentials. To Use the Table • Select **New** • Double-click the Username field and enter a username for the credential. • Double-click the Password field and enter a password for the credential. Note: At least one username/password pair is required for the credentials. • **Use Map:** When selected, the Java map specified by the following variable will is used to populate credentials. The map must have String keys and either String or List<String> values.

Digest HTTP Authentication

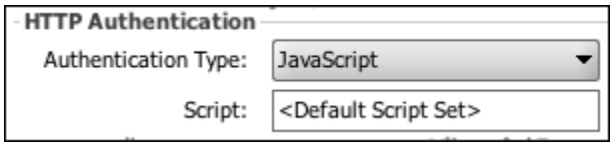
Provides Digest Authentication support for HTTP-based source connectors.



Item	Name	Description
A	Realm	The protection space for this server.
B	Algorithms	Specifies the digest algorithms supported by this server. • **MD5:** The security data A1 contains the username, realm, and password. • **MD5-sess:** The security data A1 will also contain the server and client nonces. • **Both**: Both MD5 and MD5-sess.
C	QOP Modes	The quality of protection modes to support. • **auth:** Regular authentication with client nonce and count in the digest. • **auth-int:** Same as auth (regular authentication), but also with message integrity protection enabled.
D	Opaque	A string of data that should be returned by the client unchanged. Velocity Variable Replacement *(on page 469)* 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	• **Use Table:** When selected, use the Credentials table **(E)** to populate credentials. To Use the Table • Select **New** • Double-click the Username field and enter a username for the credential. • Double-click the Password field and enter a password for the credential.Note:At least one username/password pair is required for the credentials. • **Use Map:** When selected, the Java map specified by the following variable will is used to populate credentials. The map must have String keys and either String or List<String> values.

JavaScript HTTP Authentication

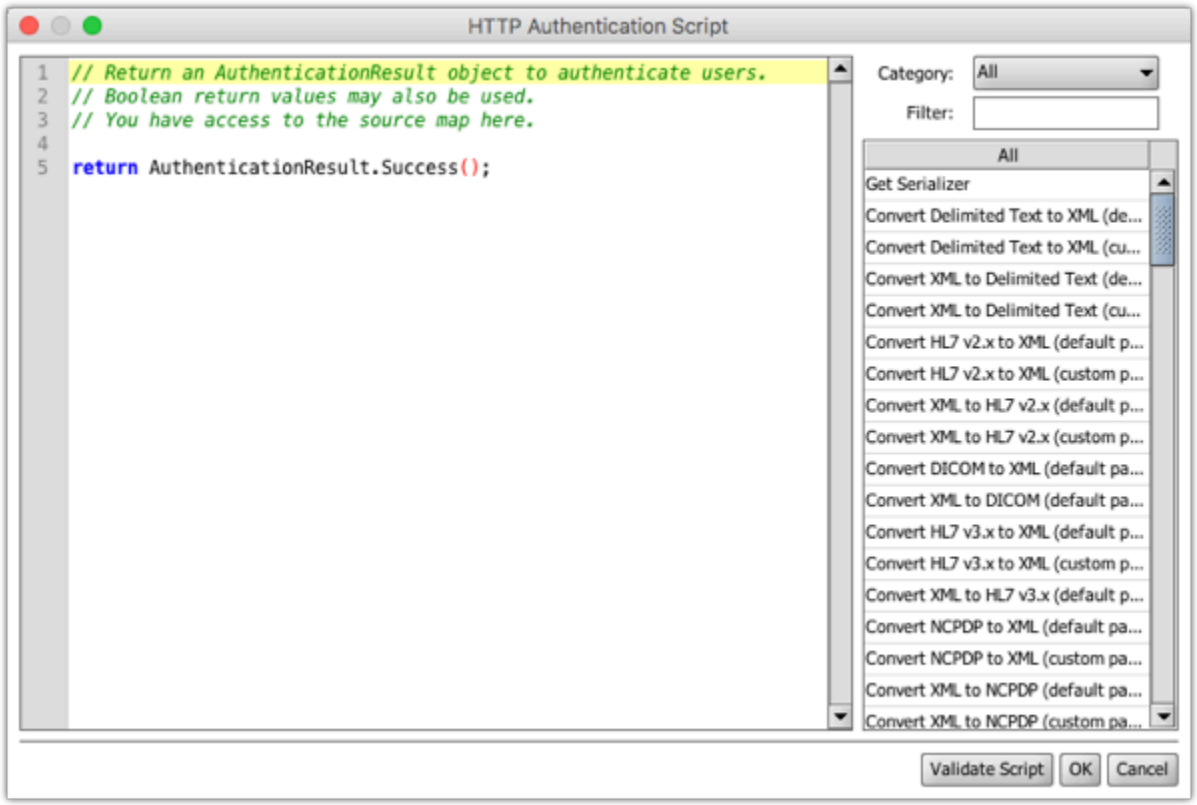
Allows you to authenticate users with a custom JavaScript script. With this script you have access to source map variables, and can choose whether to send a challenged or failure response back to the client.



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 have made any modification to the script, the **Script** field will show **<Custom Script Set>**.

Select 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 *(on page 459)*) . There are three types of results you can return:

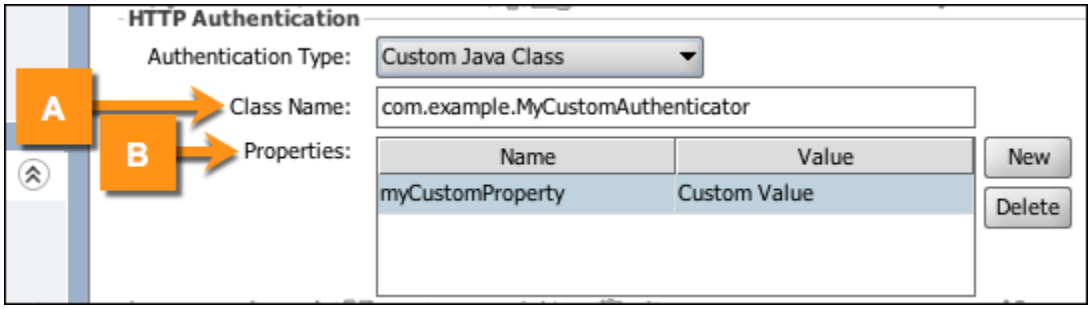
- **AuthenticationResult.Success():** The request will be accepted and processed through the channel.
- **AuthenticationResult.Challenged(authenticateHeader):** The request will not be processed through the channel. A 401 response will be sent back to the client, with a given WWW-Authenticate header 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 Mirth® Connect by NextGen Healthcare, see Mirth Connect and JavaScript *(on page 428)*.

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. • **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 be processed through the channel.

Note: 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 Mirth® Connect

- Source Connectors *(on page 340)*

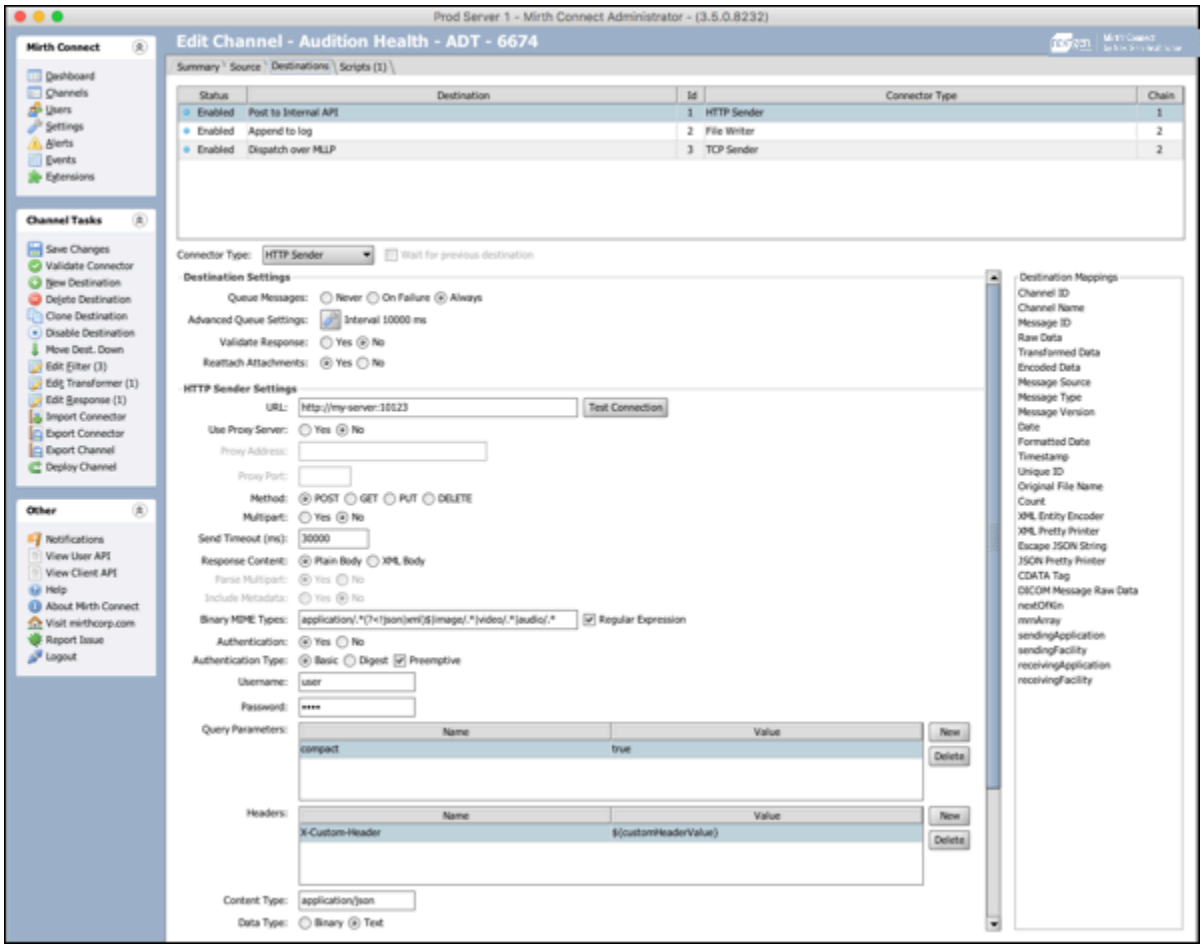
- Channel Reader *(on page 340)*
- DICOM Listener *(on page 341)*
- Database Reader *(on page 346)*
- File Reader *(on page 351)*
- HTTP Listener *(on page 362)*
- JMS Listener *(on page 366)*
- JavaScript Reader *(on page 371)*
- TCP Listener *(on page 372)*
- Web Service Listener *(on page 381)*

Additional source connectors are made available as commercial extensions *(on page 582)*:

- Email Reader *(on page 588)*
- Serial Connector *(on page 597)*
- FHIR Connector *(on page 590)*
- Interoperability Connector Suite *(on page 592)*

Destinations Tab

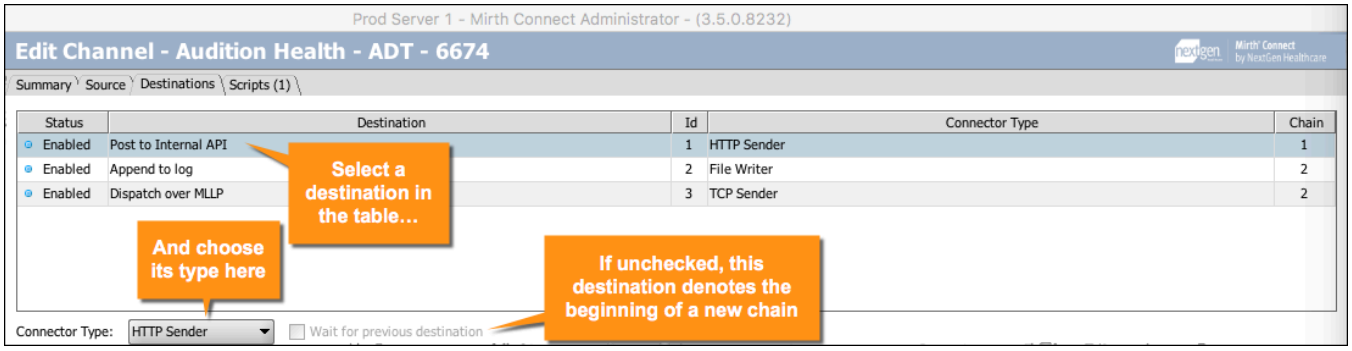
The Destinations tab is where destination connectors are configured. This includes the destination connector properties, the destination filter / transformer scripts, and the response transformer scripts. From this tab you can rename / reorder / enable / disable / clone destinations, and decide which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



Configuration of destination connectors is separated into the following sections:

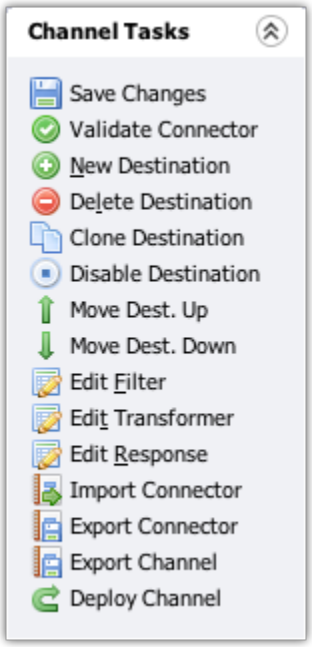
Destination Table

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



Column	Description
Status	Indicates whether the destination is enabled or disabled. Only enabled destinations process messages. 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.

Destination Tasks

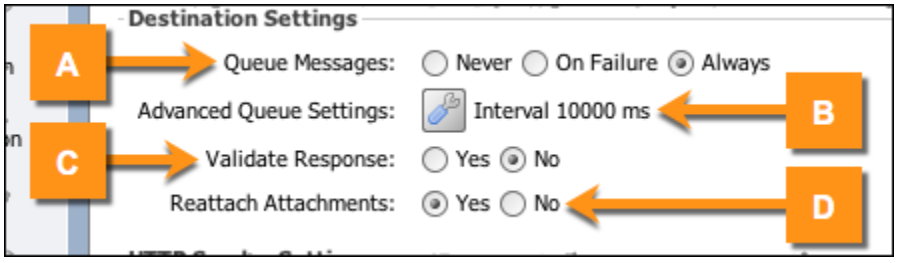


In addition to the general Edit Channel Tasks *(on page 256)*, several context-specific tasks are unique to the Destinations Tab *(on page 246)*:

Task Icon	Task Name	Description
New Destination	New Destination	Creates a new destination and adds it to the table above.
Delete Destination	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	Clone Destination	Copies the currently selected destination and adds it to the table above.
Enable Destination	Enable Destination	Marks this destination as ready to process messages at deploy time.
Disable Destination	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 Dest. Up	Move Destination Up	Moves the currently selected destination one row higher in the table above.
Move Dest. Down	Move Destination Down	Moves the currently selected destination one row lower in the table above.
Edit Response	Edit Response	Enters the Edit Transformer View *(on page 257)* for the currently selected destination's response transformer.

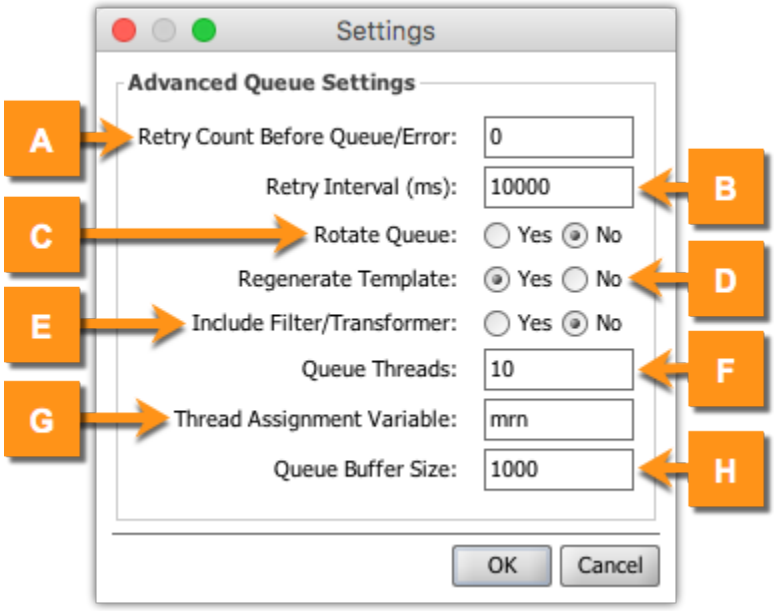
Destination Settings

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	• **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 *(on page 251)*	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 *(on page 323)*.
D	Reattach Attachments	• If enabled, replacement tokens using the ${ATTACH:...} syntax are automatically replaced with the associated attachment content before the message is sent. • If disabled, the tokens are expanded to the full ${ATTACH:channelId:messageId:attachmentId} syntax which can then be reattached in downstream channels. For additional information, see Attachment Handlers *(on page 219)*.

Advanced Queue Settings



Item	Name	Description
A	Retry Count Before Queue/ Error	The maximum number of times the connector attempts 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 is used for each attempt.
E	Include Filter/Transformer	If enabled, the filter and transformer is 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 **NOT** guaranteed if this value is greater than one, unless an assignment variable is used below.

Item	Name	Description
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 Mirth® Connect:

- Destination Connectors *(on page 384)*
  - Channel Writer *(on page 384)*
  - DICOM Sender *(on page 386)*
  - Database Writer *(on page 390)*
  - Document Writer *(on page 393)*
  - File Writer *(on page 395)*
  - HTTP Sender *(on page 400)*
  - JMS Sender *(on page 407)*
  - JavaScript Writer *(on page 410)*
  - SMTP Sender *(on page 411)*
  - TCP Sender *(on page 416)*
  - Web Service Sender *(on page 422)*

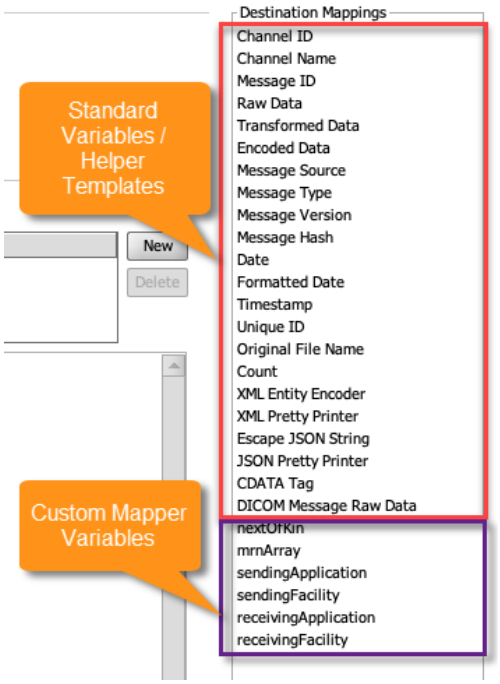
Additional destination connectors are made available as commercial extensions *(on page 582)*:

- Serial Connector *(on page 597)*
- NextGen Results CDR Connector *(on page 600)*

- FHIR Connector *(on page 590)*
- Interoperability Connector Suite *(on page 592)*

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 *(on page 281)* you have added in the current destination or in any previous destinations. For additional information on Velocity replacement, see Velocity Variable Replacement *(on page 469)*.



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

Name	Description
	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 *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* messages this will usually be the Sending Facility value in MSH.4.1.
Message Type	Depends on the inbound data type *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* 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 *(on page 323)* for the connector. For HL7 v2.x *(on page 329)* messages this will usually be the Version value in MSH.12.1.
Message Hash	The SHA-256 hash of the encoded data of the destination connector message. If no encoded data exists, the raw content is hashed instead.
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

Name	Description
	merged bytes to the outbound endpoint. Typically used by the DICOM Sender *(on page 386)* destination.

Scripts Tab

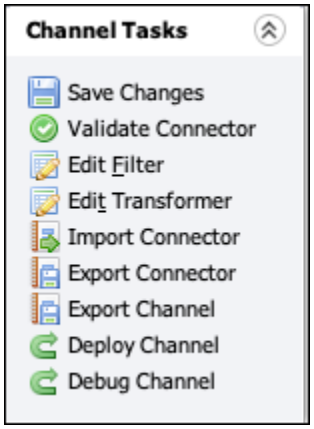
The Scripts tab is where channel-level scripts are configured. Select a script type from the drop-down list and edit the script in the text area.Select a script type from the drop-down and edit the script in the text area below. If a number appears next to "Scripts" in the tab, that number represents 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, such as loading a properties file from disk, or instantiating 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 is stored as the Processed Raw content and used to feed into the source filter/transformer.
- **Post Processor Script**: This script executes once for every message, after all destinations have

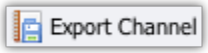
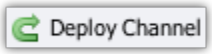
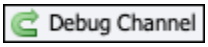
completed processing (not including queued messages which are processed asynchronously). You have access to "message" which is an ImmutableMessage object containing information about the state of all connector messages. This script may be used as a general tool to perform a custom cleanup script. It can also be used to return a custom response that may be sent back to the originating system.

Edit Channel Tasks



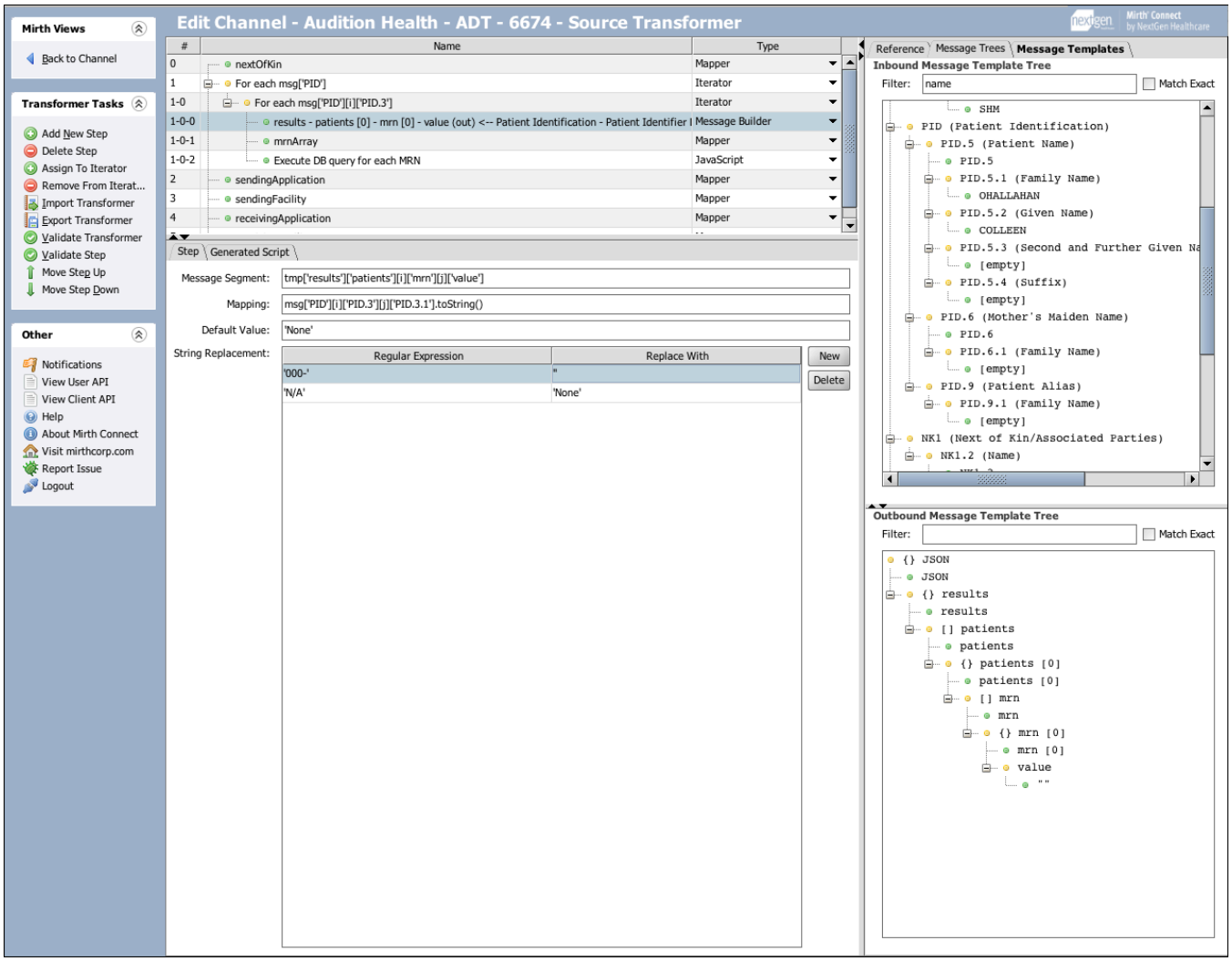
The following context-specific tasks are available throughout the Edit Channel View *(on page 203)*:

Task Icon	Task Name	Description
Save Changes	Save Changes	Saves a new revision of the current channel, if anything was actually changed.
Validate Connector	Validate Connector	Validates the currently viewed connector, ensuring that all connector properties are valid and able to be saved.
Edit Filter	Edit Filter	Enters the Edit Filter View *(on page 257)* for the currently viewed connector.
Edit Transformer	Edit Transformer	Enters the Edit Transformer View *(on page 257)* for the currently viewed connector.
Import 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 is added to the table.

Task Icon	Task Name	Description
	Export Connector	Exports the currently viewed connector to an XML file. This includes 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 *(on page 154)*.
	Deploy Channel	Deploys the current channel.
	Debug Channel *(on page 461)*	Displays the Debug Channel *(on page 461)* dialog which allows you to deploy the currently selected channel while debugging specific scripts within the channel.

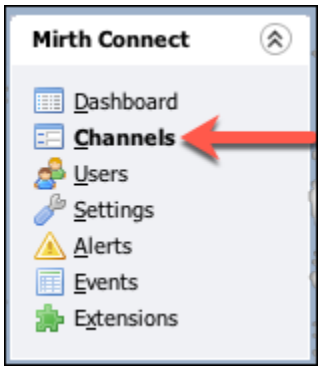
Edit Filter / Transformer Views

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.

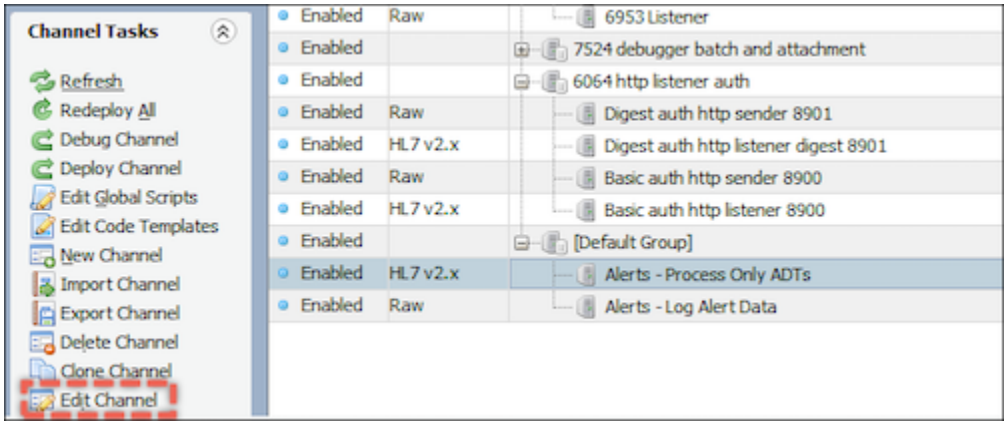


Navigation

Select the Channels link in the Mirth® Connect task pane at the top-left to enter the Channels View *(on page 148)*:

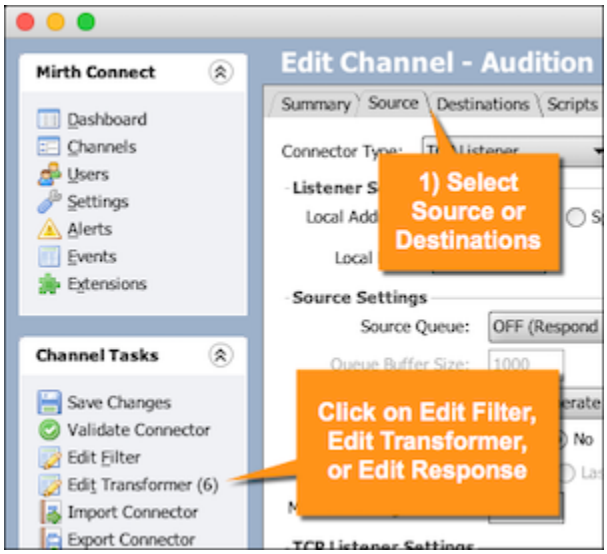


In the Channel Table *(on page 150)*, select the channel you wish to edit, and select the **Edit Channel** task to the left:



You can also double-click the channel row in the table.

Once in the Edit Channel View *(on page 203)*, select either the Source Tab *(on page 232)* or Destinations Tab *(on page 246)*. Select a destination if necessary. Then, select on the **Edit Filter**, **Edit Transformer**, or **Edit Response** tasks to the left. The Edit Response task (for editing the response transformer) is only visible for destination connectors.



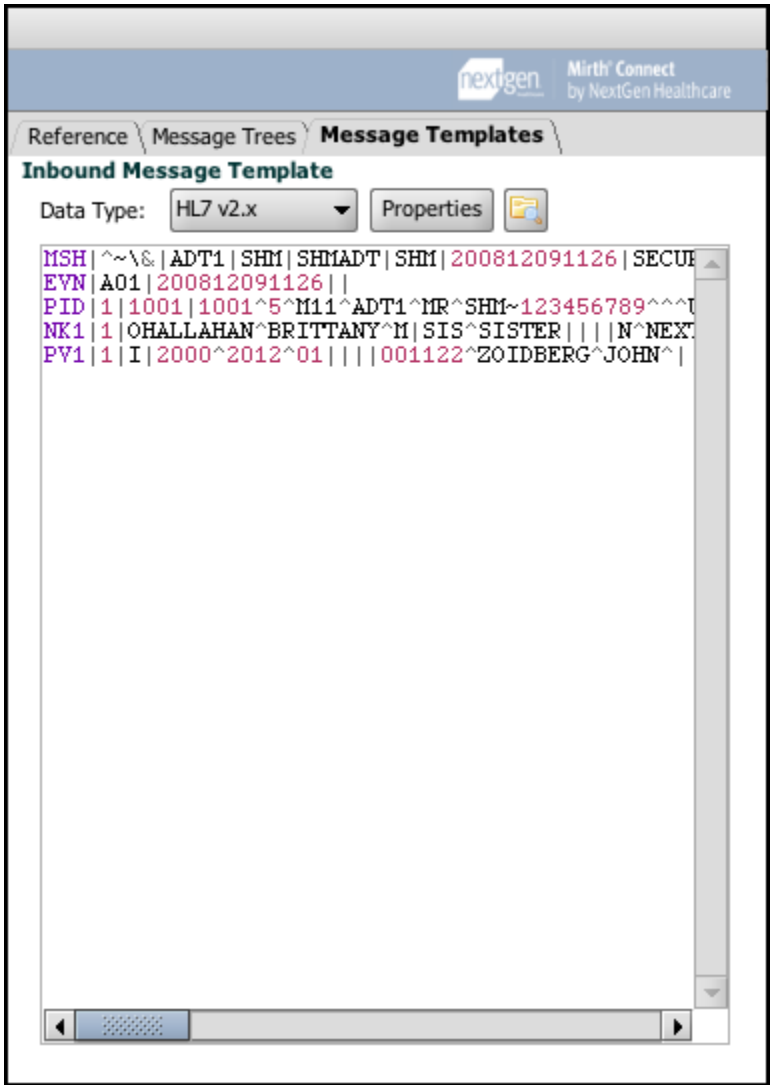
This section is separated into the following topics:

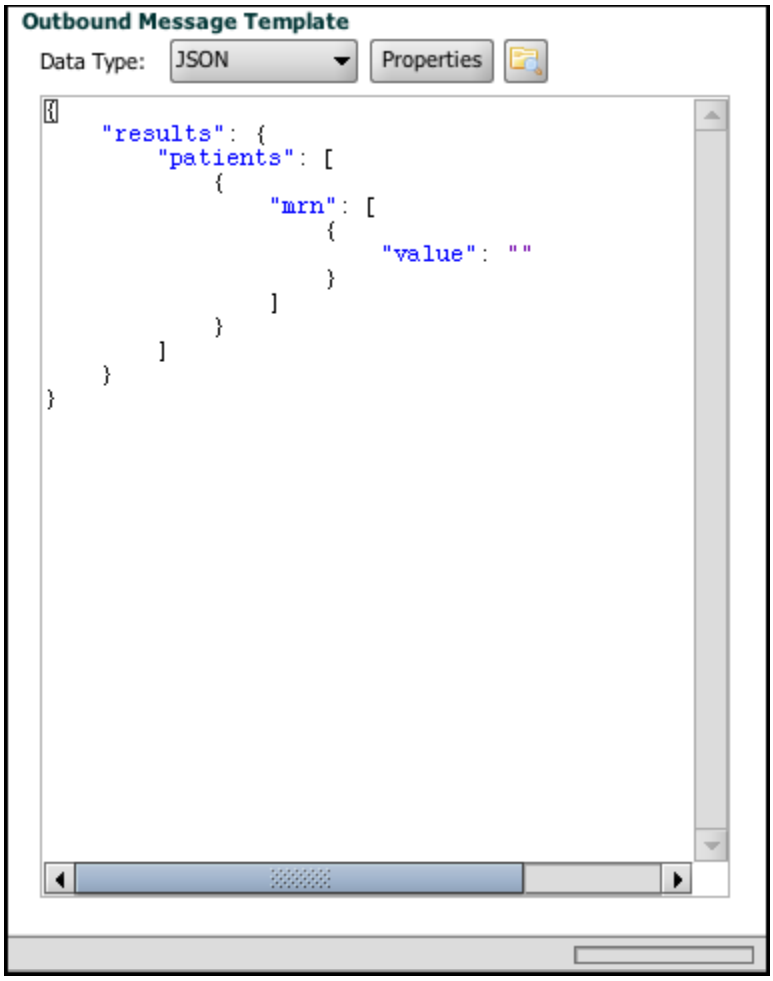
- Message Templates Tab *(on page 260)*
- Message Trees Tab *(on page 265)*
- Reference Tab *(on page 273)*
- Create New Rules/Steps *(on page 276)*
- Rule/Step Table *(on page 277)*
- Filter Rule Properties *(on page 278)*
- Transformer Step Properties *(on page 281)*
- Response Transformers *(on page 286)*
- Working With Iterators *(on page 288)*
- View Generated Script *(on page 294)*
- Filter Tasks *(on page 295)*
- Transformer Tasks *(on page 297)*

Message Templates Tab

The Message Templates tab is located on the right side of the Edit Filter / Transformer View and enables you to edit the inbound and 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 will see the Message Templates title appear bold in the tab list.

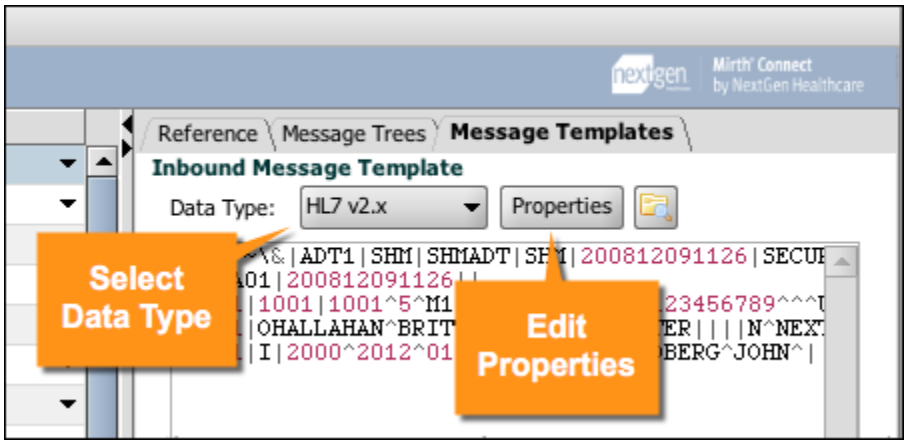




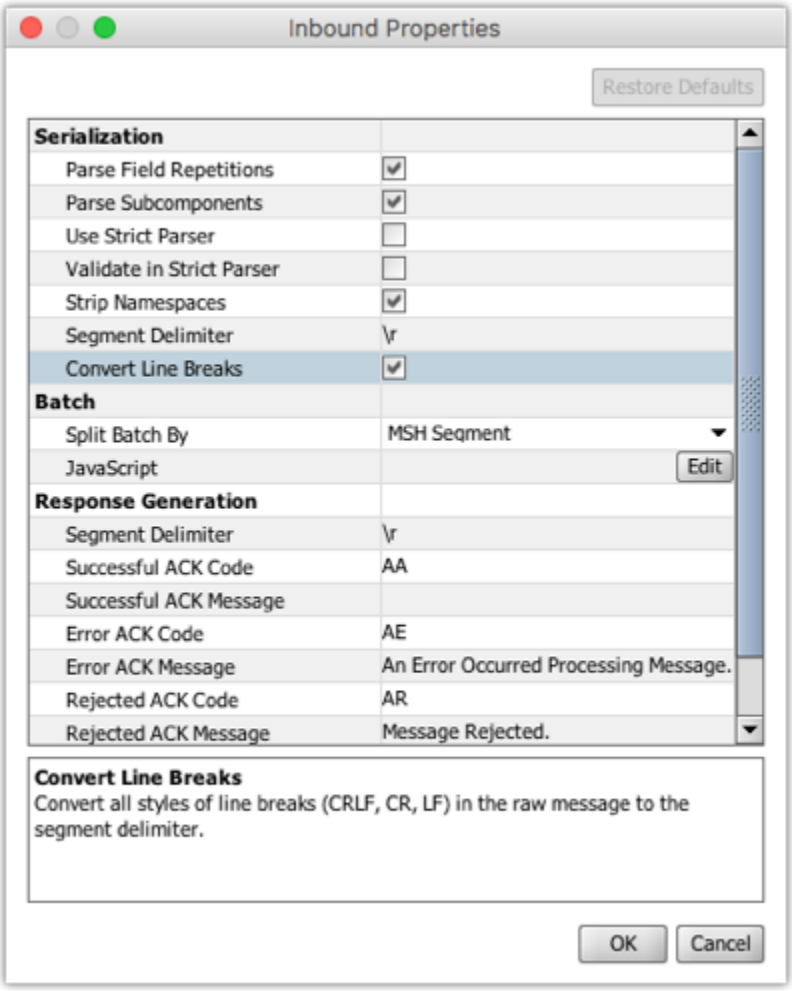
Note: The **Outbound** template is only visible for transformers and response transformers. When you edit a connector's filter, you only see the **Inbound** template.

Edit 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 selecting the **Properties** button, a dialog opens 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 Window *(on page 207)*. For additional information, see About Data Types *(on page 72)* and Data Types *(on page 323)*.

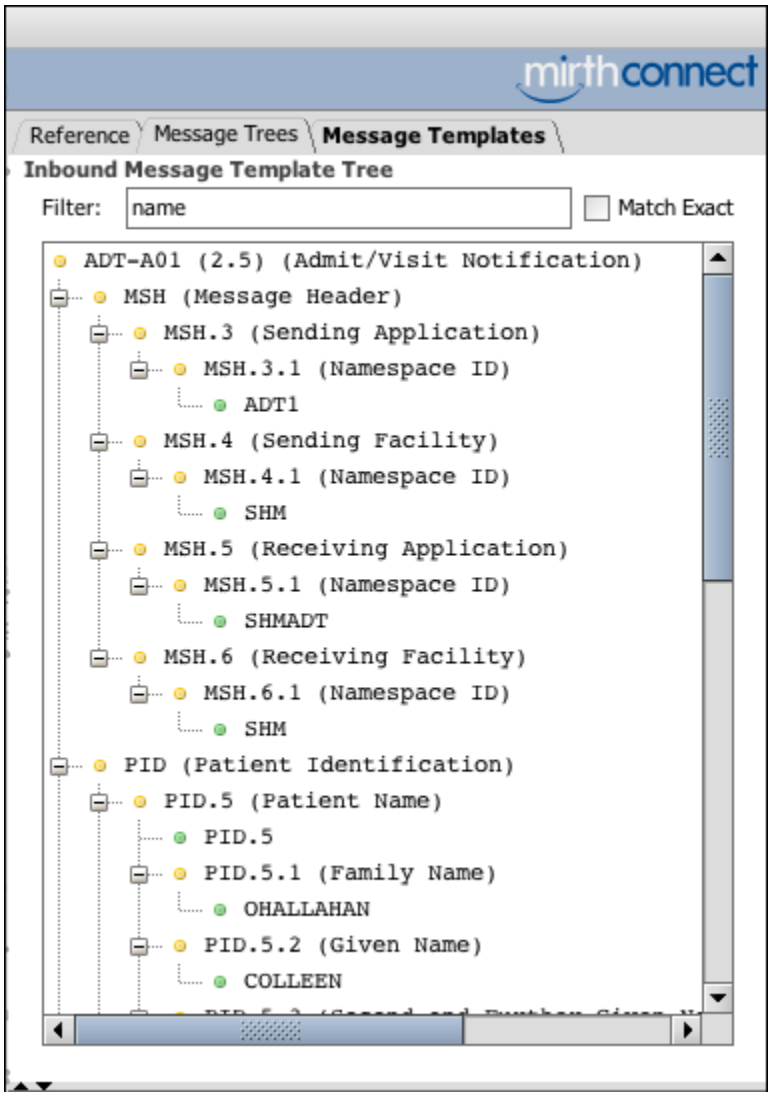


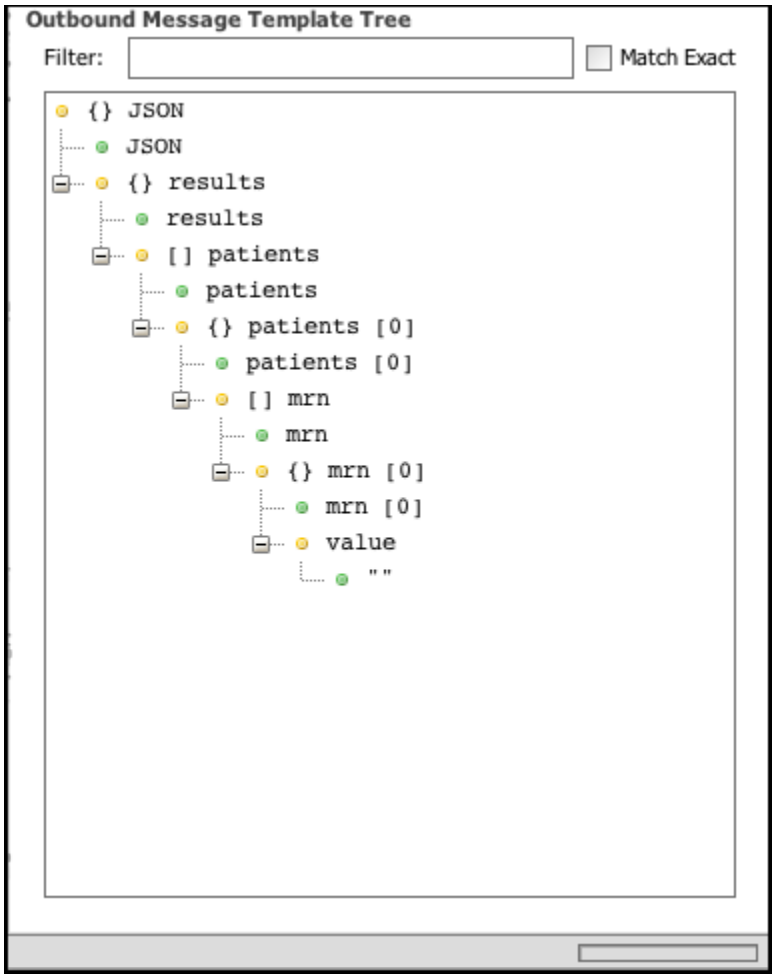
Specify a Template

You can use the folder icon button next to the **Properties** button to select a message from your local file system. You can also type / paste a message into the text area.

Message Trees Tab

The Message Trees tab is located on the right side of the Edit Filter/Transformer views and enable you to view a tree representation of your inbound/outbound message templates set in the Message Templates tab.

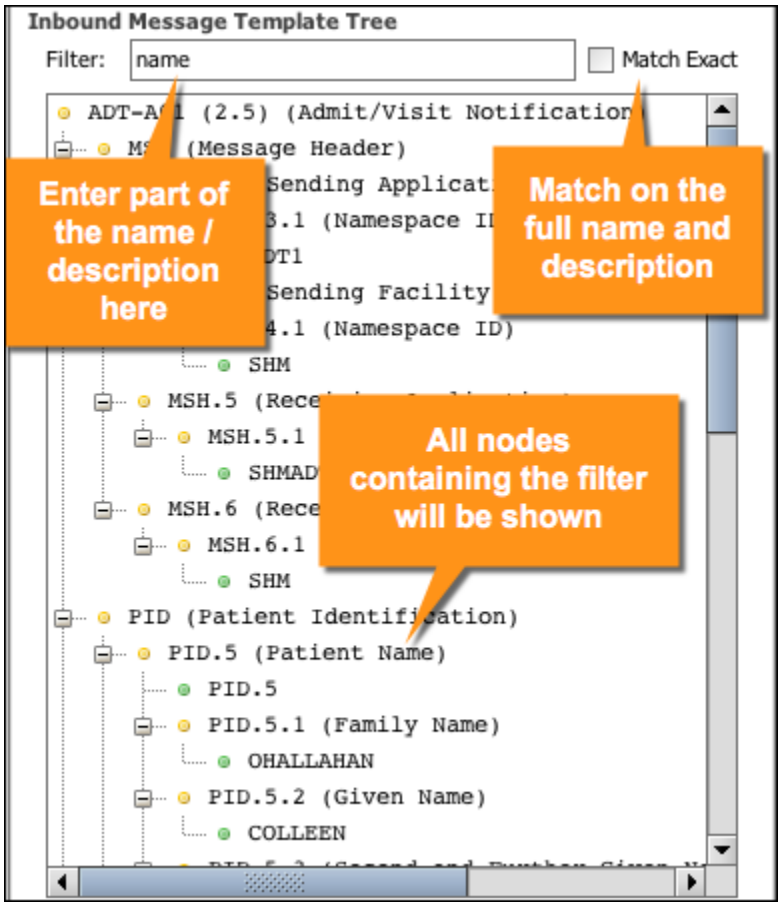

mirthconnect
Reference
Message Trees
Message Templates
Inbound Message Template Tree
Filter:
name
Match Exact
ADT-A01 (2.5) (Admit/Visit Notification)
MSH (Message Header)
MSH.3 (Sending Application)
MSH.3.1 (Namespace ID)
ADT1
MSH.4 (Sending Facility)
MSH.4.1 (Namespace ID)
SHM
MSH.5 (Receiving Application)
MSH.5.1 (Namespace ID)
SHMADT
MSH.6 (Receiving Facility)
MSH.6.1 (Namespace ID)
SHM
PID (Patient Identification)
PID.5 (Patient Name)
PID.5
PID.5.1 (Family Name)
OHALLAHAN
PID.5.2 (Given Name)
COLLEEN



Note: The **Outbound** template tree is only visible for transformers and response transformers. When you edit a connector's filter, you will 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 of the area.

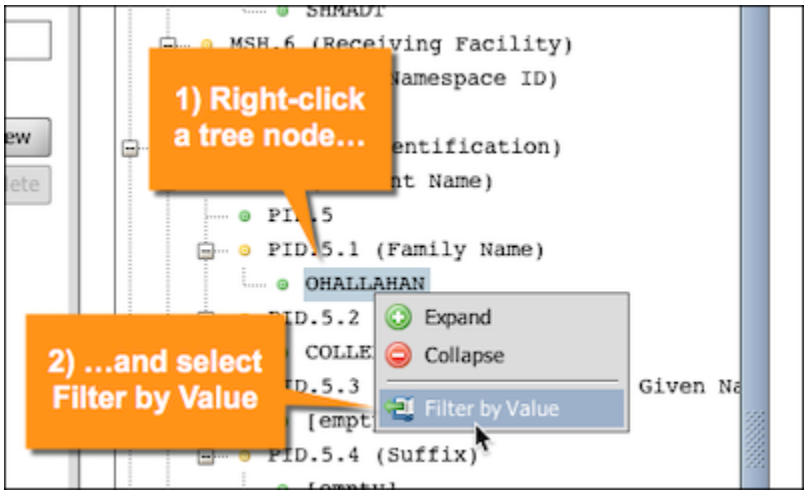


Create a Rule Builder Rule or Mapper Step

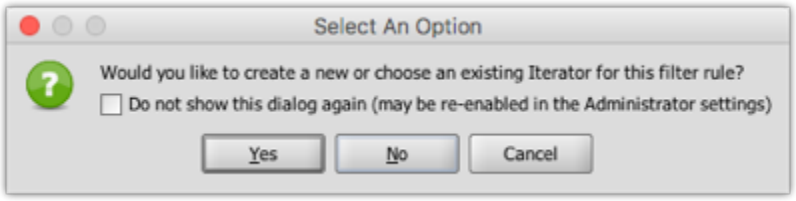
When editing a filter, you can create a Rule Builder Filter Rule *(on page 278)* directly from the message tree. When editing a transformer, you can create a Mapper Transformer Step *(on page 281)* in the same way.

Method 1

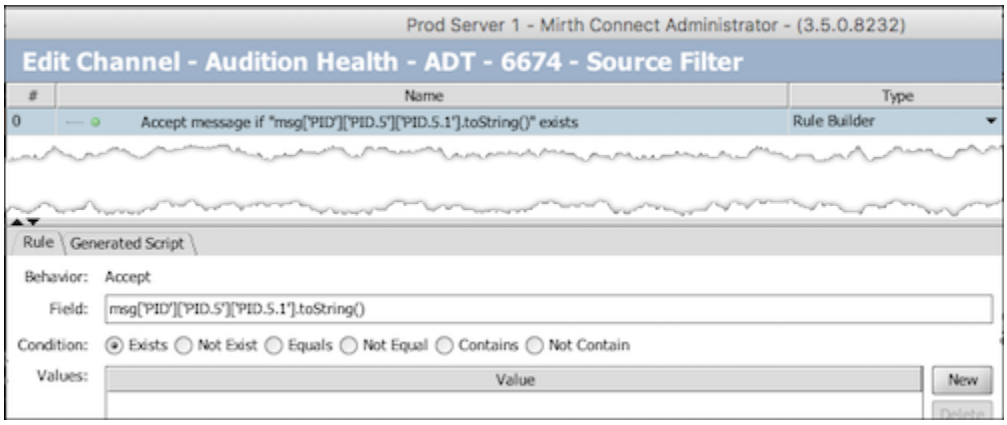
Right-click the node you wish to filter on or map, then select either **Filter by Value** or **Map to Variable**:



A dialog may be shown asking whether you want to add the rule/step as part of an Iterator. For additional information, see Working With Iterators *(on page 288)*

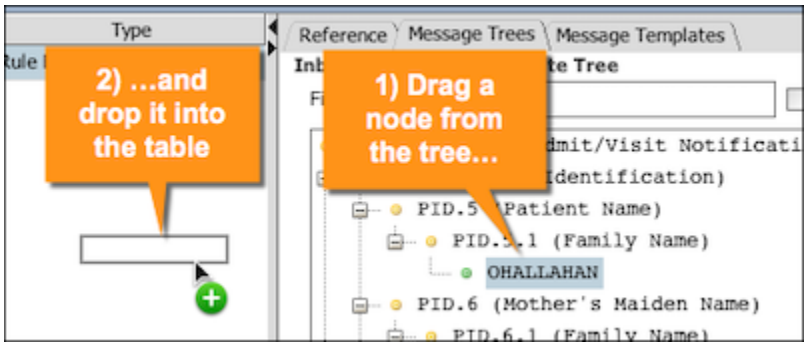


A Rule Builder or Mapper is automatically added to the table with the selected node populated appropriately.



Method 2

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



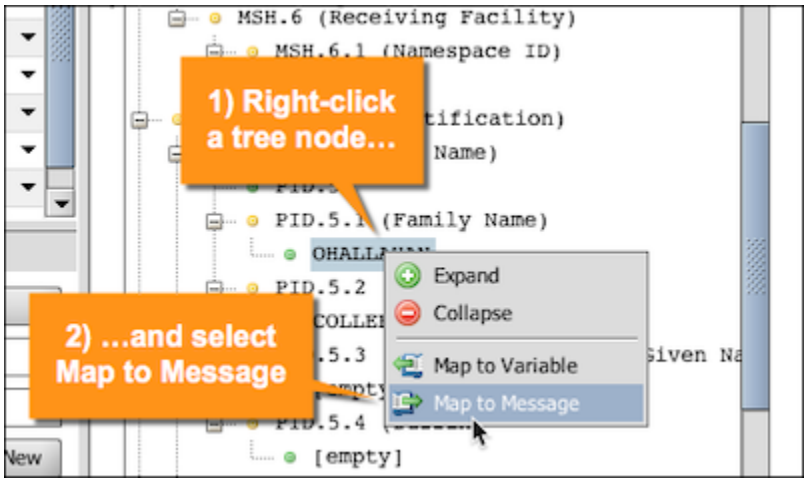
The same steps follow as in Method 1.

Create a Message Builder Step

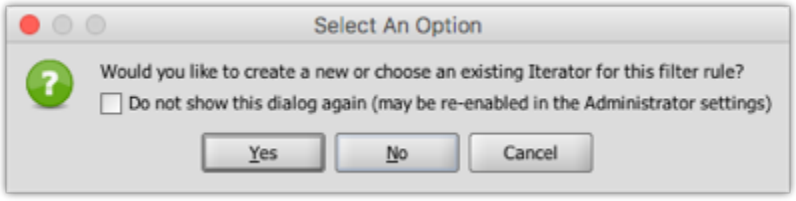
When editing a transformer, you can create a Message Builder Transformer Step *(on page 282)* directly from the message tree.

Method 1:

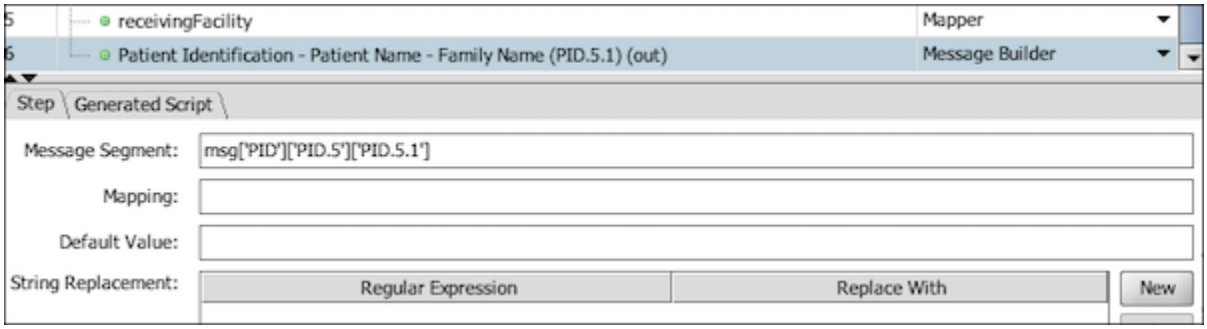
Right-click the node you want to modify, then select **Map to Message**.



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 *(on page 288)*.

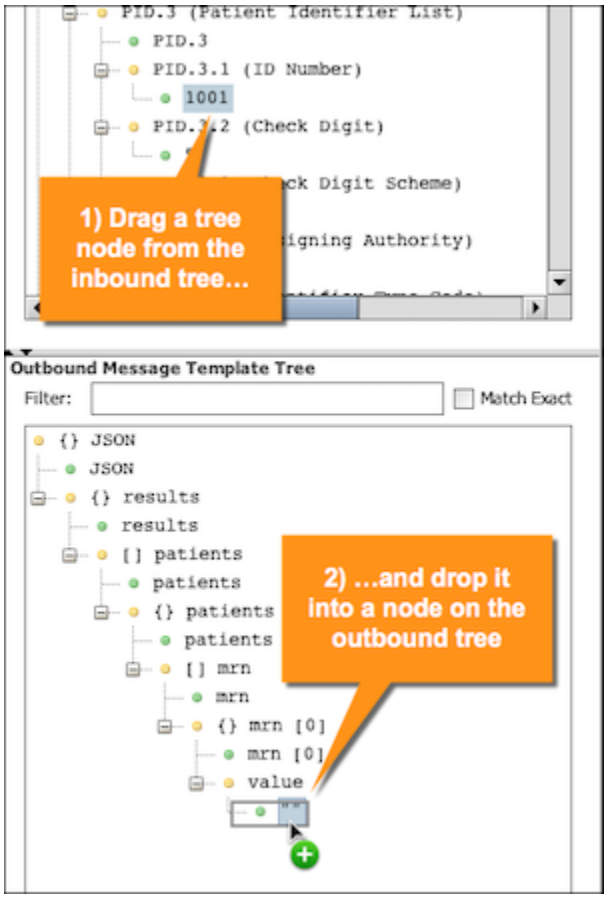


A Message Builder is automatically added to the table with the selected node populated appropriately.



Method 2

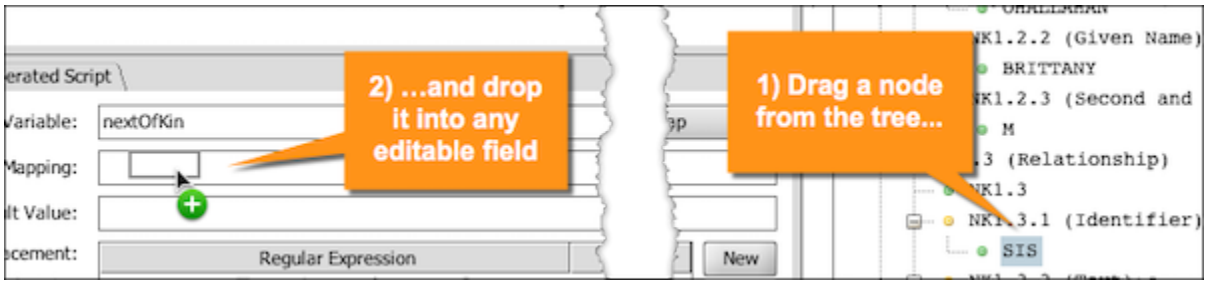
Drag the node you wish to map **from** the inbound message tree, and drop it into the node you wish to map **to** in the outbound message tree.



The same steps follow as in Method 1.

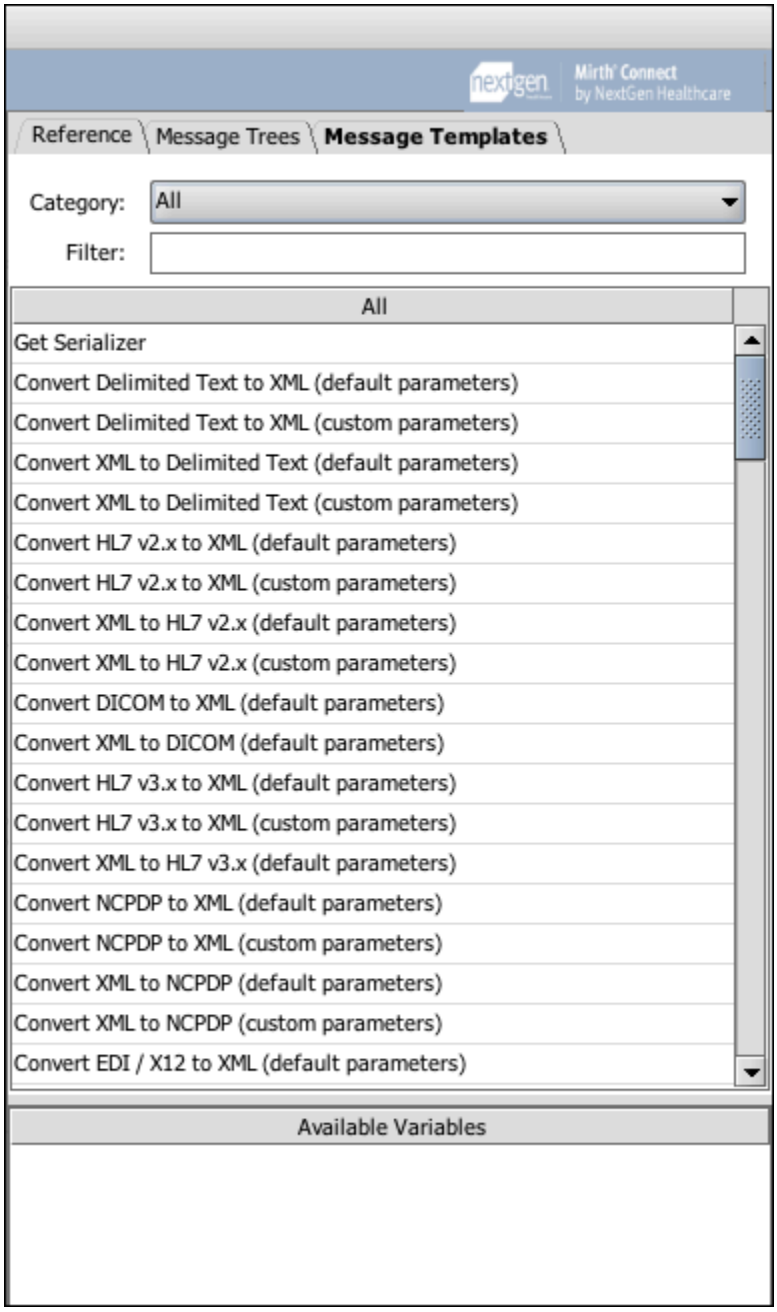
Drag-and-Drop Field Values

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.



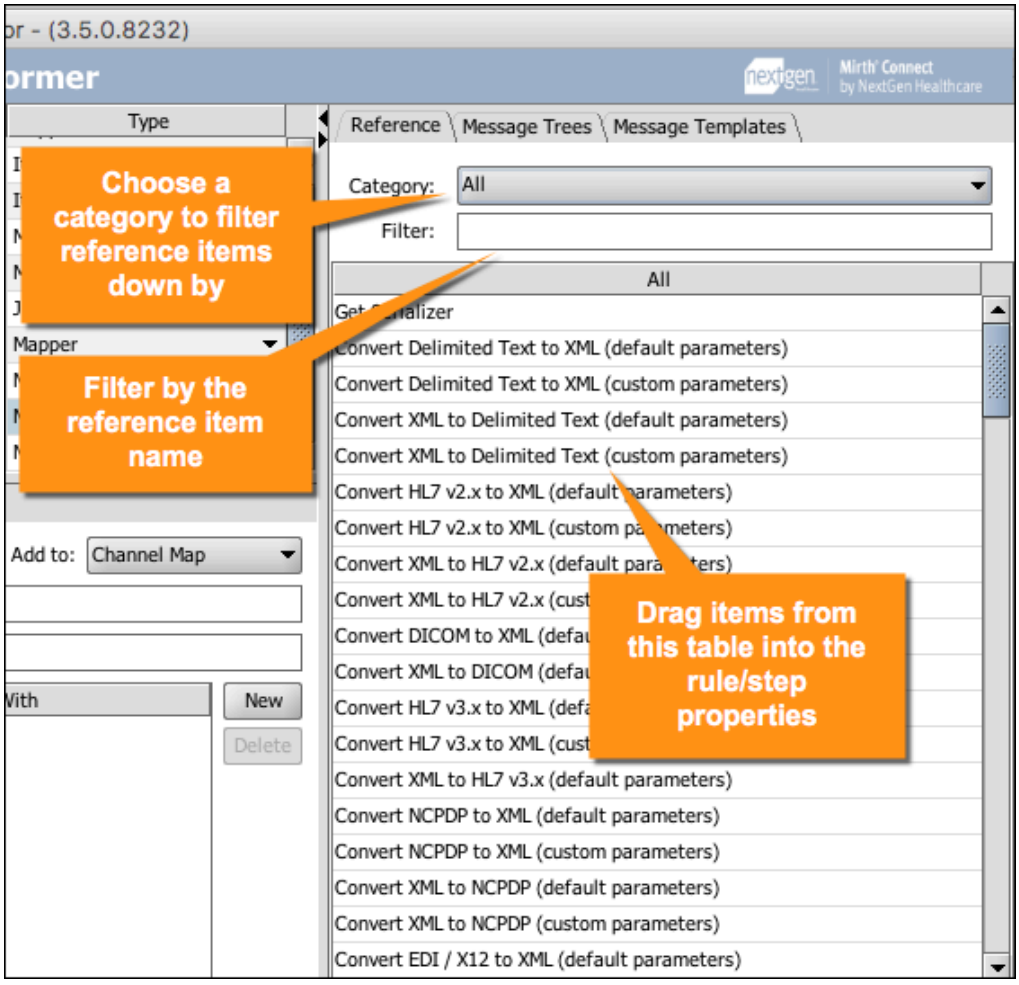
Reference Tab

The Reference tab is located on the right side of the Edit Filter / Transformer Views *(on page 257)* and provides helpful variables and templates you can use within your filter rules and transformer steps.



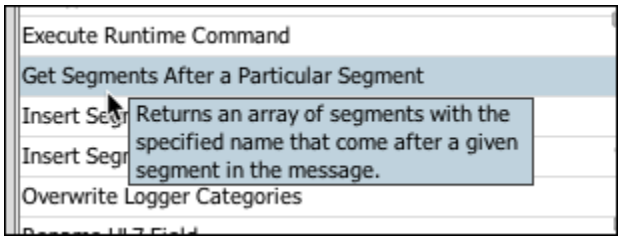
Reference List

The **Reference List** contains code templates (both built-in and custom) that you can drag into your filter rules and 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 .

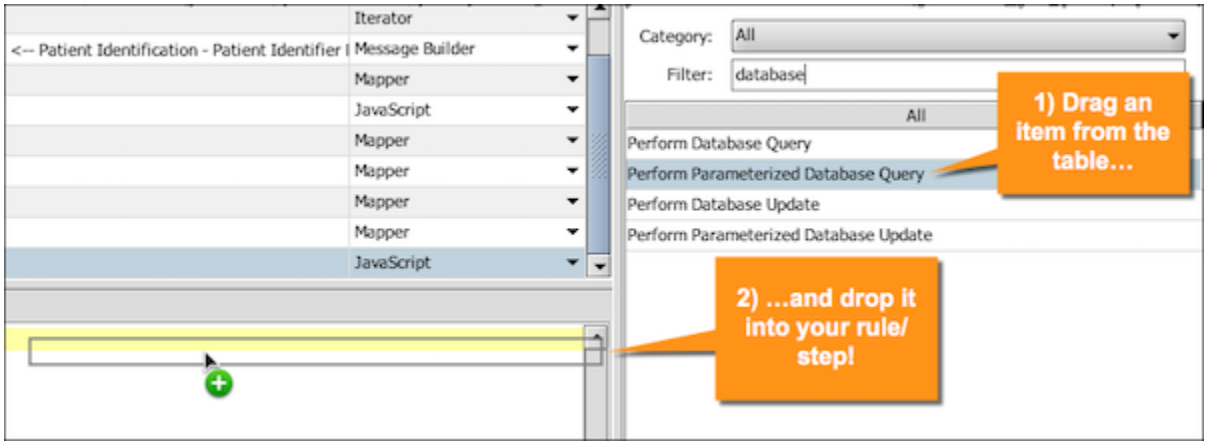


Use the **Category** and **Filter** components at the top of the area to quickly find the template you are looking for. For custom code templates, choose the **User Defined Functions** or **User Defined Code** options.

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



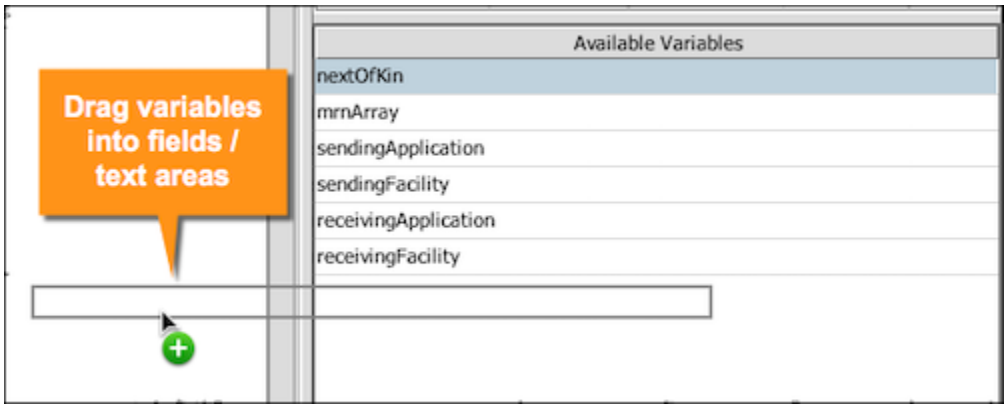
To use an item, simply drag it from the table into your rule/step properties window.



The corresponding code template/function call is automatically pasted into the field/text area.

Available Variables

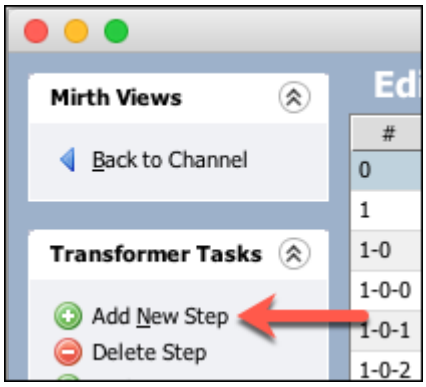
This is a convenience list shown in the Edit Transformer View *(on page 257)* when the Reference Tab *(on page 273)* is selected. It is automatically populated with the variables set in previous Mapper steps, allowing you to easily drag them into subsequent steps.



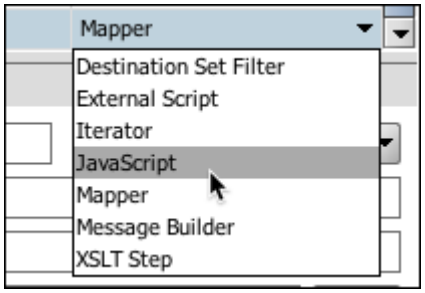
Create New Rules/Steps

As outlined in the Message Trees Tab *(on page 265)* section, new rules / steps can be generated easily from the inbound/outbound message trees. They can also be created manually.

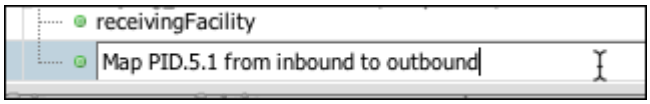
- Select the **Add New Rule/Step** task to the left:



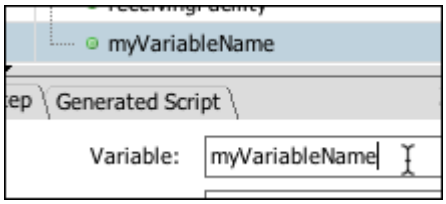
- A new rule/step is created and added to the table. To change the type of rule/step, double-click the cell in the **Type** column.



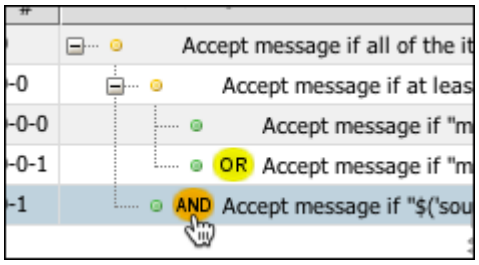
- To change the name, double-click the cell in the **Name** column.



- Some rule/step types have names auto-generated from the properties and so do not allow editing in the table.

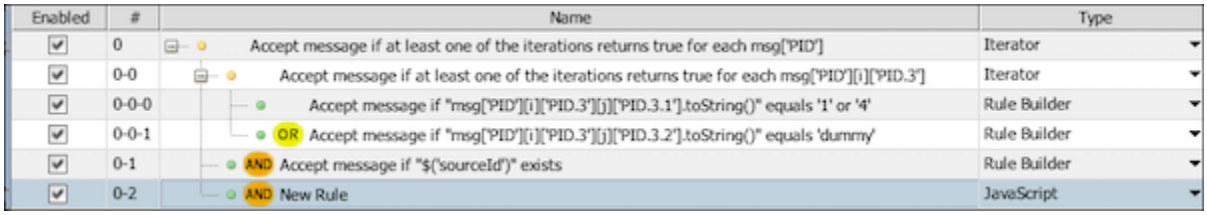


- If you are creating a filter rule, you can change the **operator** of the rule by selecting the **AND** or **OR** button next to the name.



Rule/Step Table

The table in the top half of the Edit Filter / Transformer Views *(on page 257)* shows all the rules/steps you currently have configured. If you have Iterators *(on page 288)*, the elements are organized into a tree-table with the children underneath each Iterator.



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

Column	Description
	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 select on the **AND** / **OR** icon next to the name to change the operator.
Type	The type of rule / step. Double-click this cell to select a different type.

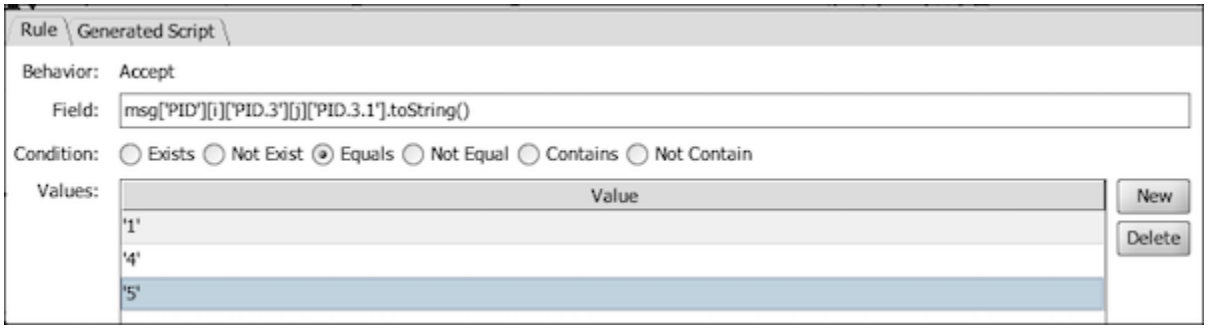
Filter Rule Properties

The following Filter Rules are supported by Mirth® Connect

- Rule Builder Filter Rule *(on page 278)*
- JavaScript Filter Rule *(on page 279)*
- External Script Filter Rule *(on page 279)*
- Iterator Filter Rule *(on page 280)*

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:

Item Name	Description
	• **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.

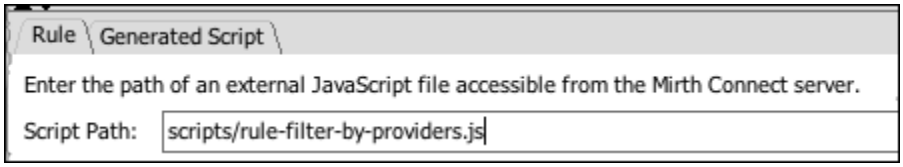
JavaScript Filter Rule

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 *(on page 435)*.

External Script Filter Rule

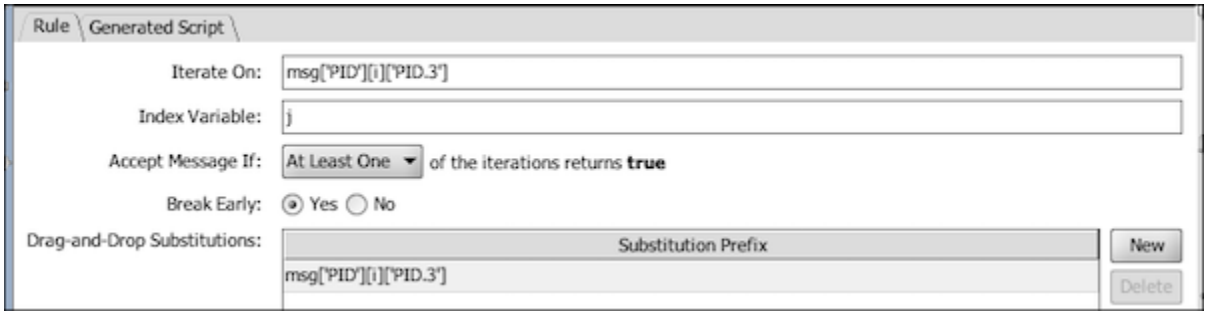
This rule functions the same way as the JavaScript Filter Rule *(on page 279)*, except that the script is

read from an external file when the channel is deployed. If the given path is not absolute, it will be relative to the Mirth® Connect 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 *(on page 288)*.



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. • **At Least One:** If the logical combination of the child rules returns true for **at least one** 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 **all** 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**.

Item Name	Description
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 *(on page 265)* into a child rule, it will automatically be replaced with **msg['PID'][i]['PID.3']['PID.3.1'].toString()**.

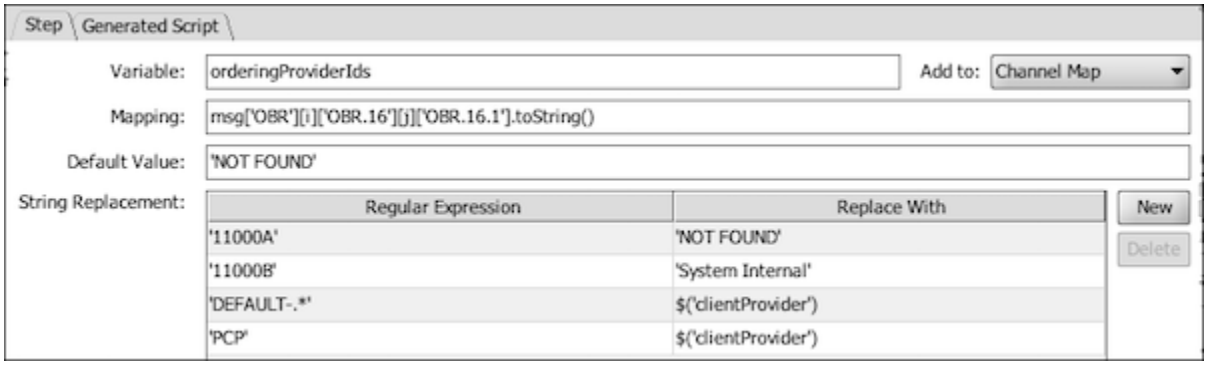
Transformer Step Properties

The following Transformer Steps are supported by Mirth® Connect:

- Mapper Transformer Step *(on page 281)*
- Message Builder Transformer Step *(on page 282)*
- JavaScript Transformer Step *(on page 283)*
- External Script Transformer Step *(on page 283)*
- XSLT Transformer Step *(on page 283)*
- Destination Set Filter Transformer Step *(on page 284)*
- Iterator Transformer Step *(on page 286)*

Mapper Transformer Step

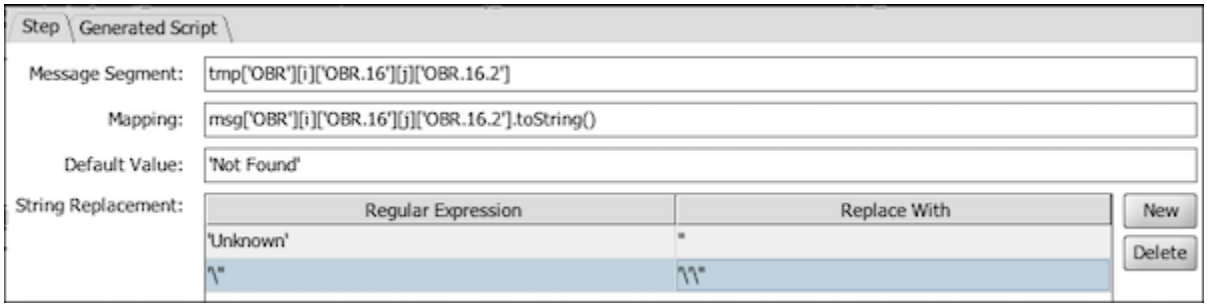
This step extracts data from a field in the message (or an expression) and places it into one of the available Variable Maps *(on page 449)*. Depending on the scope of the map, this variable will be available in subsequent steps, in the destination connector properties, or even in subsequent connectors.



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 *(on page 449)*.
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. • **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.

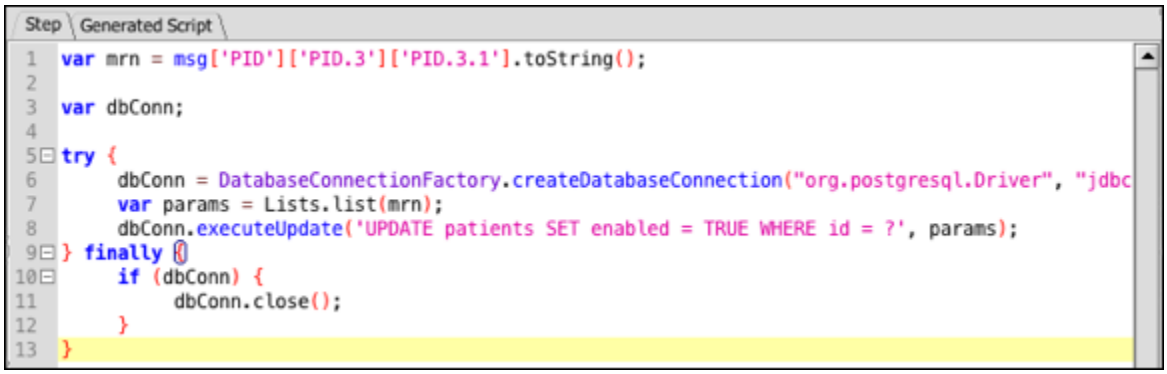


Item Name	Description
Message Segment	The field/location in the inbound or outbound message to place the value into.
Mapping	The value to place into the given message segment. This may be a field from the message, or any JavaScript expression.
Default Value	If the Mapping is not found or evaluates to an empty string, this value / expression is used instead.
String Replacement	This table allows you to perform replacements on the value before it gets inserted.

Item Name	Description
	• **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.

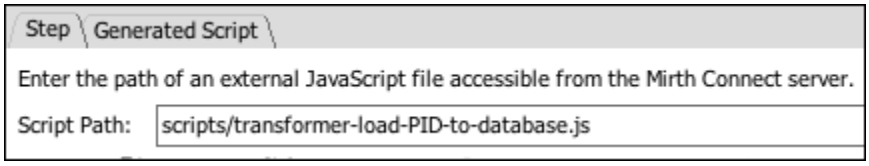
JavaScript Transformer Step

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 *(on page 435)*.



External Script Transformer Step

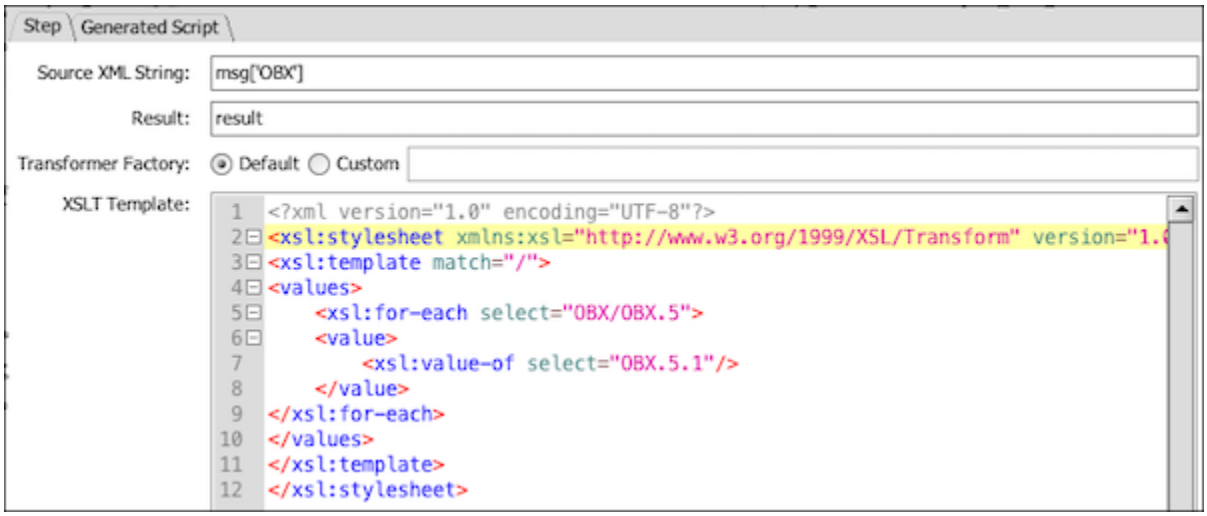
This step functions the same way as the JavaScript Transformer Step *(on page 283)*, except that the script is read from an external file when the channel is deployed. If the given path is not absolute, it will be relative to the Mirth® Connect installation directory.



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.

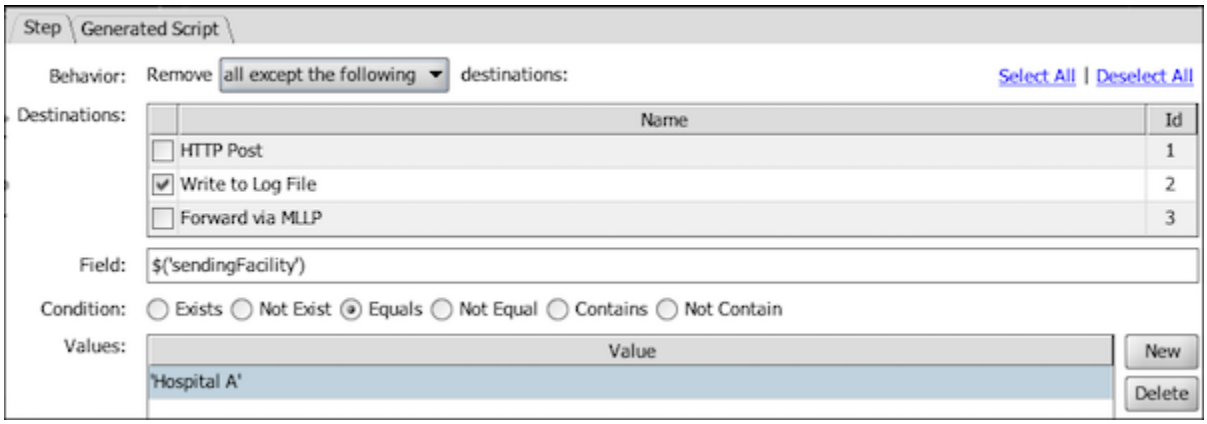


Item Name	Description
Source XML String	The XML string to transform.
Result	The key to use when storing the result into the channel map.
Transformer Factory	Select **default** to use the Java platform default TransformerFactory implementation class. Select **custom** to provide a custom TransformerFactory 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 *(on page 103)*, making it harder to find what you are 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 *(on page 459)*) . However this step allows easier access to the feature without having to write any JavaScript.

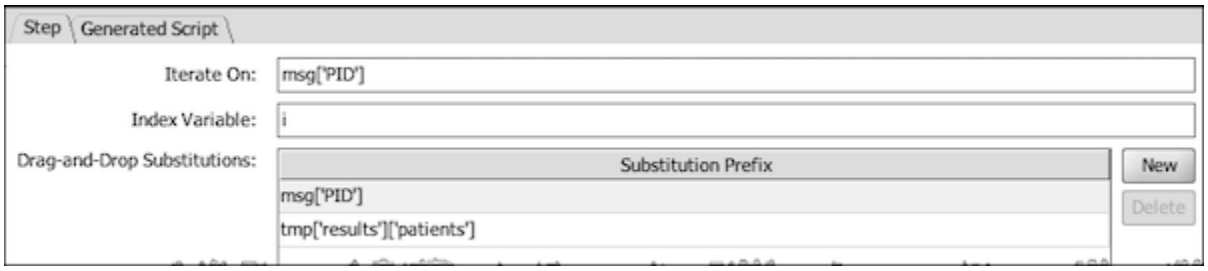


Item Name	Description
Behavior	Identifies which destinations will be removed (not processed) from the destination set. Available options are: • **the following:** The identified destination(s) are filtered if the conditions indicated are **true**. • **all except the following:** All destinations except those selected are filtered if the conditions indicated are **true**. This includes new destinations added after this step was created. • **all:** All destinations are filtered if the conditions indicated are **true**. This includes new destinations added after this step was created.
Destinations	Select the destinations to exclude or *not* exclude, depending on the behavior above. 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: • **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.

Item Name	Description
	• **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 *(on page 288)* .



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 *(on page 265)* 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 *(on page 246)*. 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 *(on page 76)*.

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.

Note: Response transformers will only execute if there is an actual response payload to transform. For example if you are using an HTTP Sender *(on page 400)* 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 is 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 *(on page 459)*.

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.';
}
```

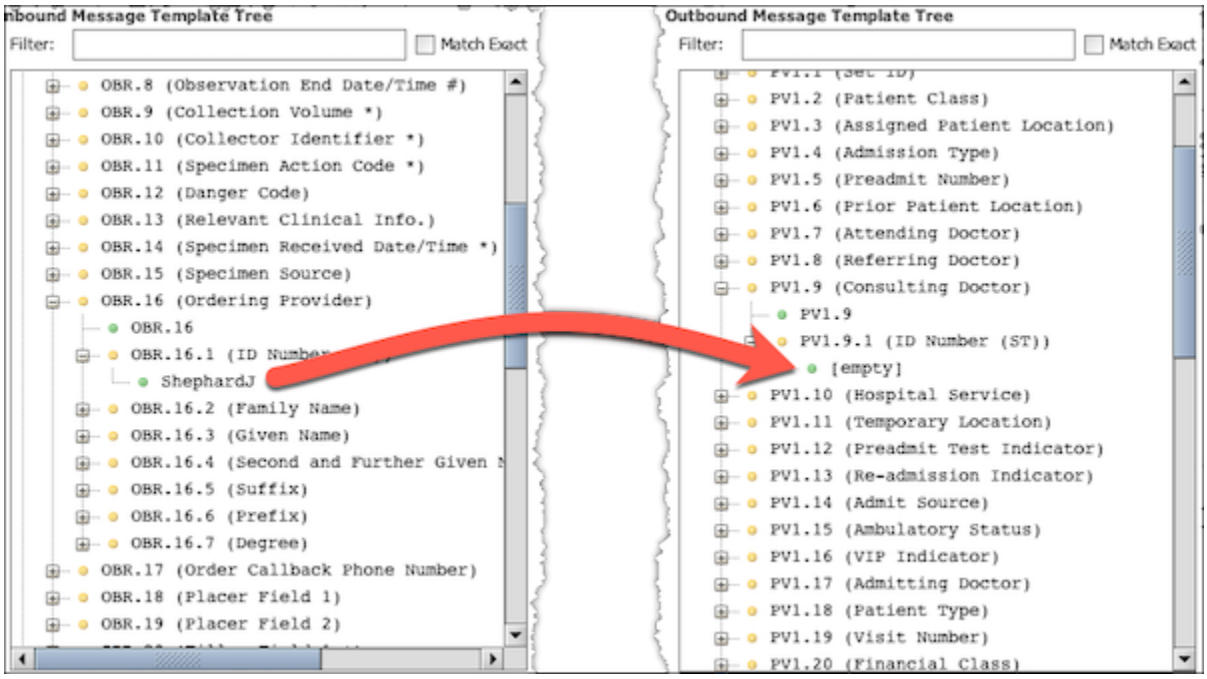
Route the response data to a downstream channel

```
if (responseStatus == SENT) {
    router.routeMessageByChannelId('channel ID here', response.getMes
sage());
}
```

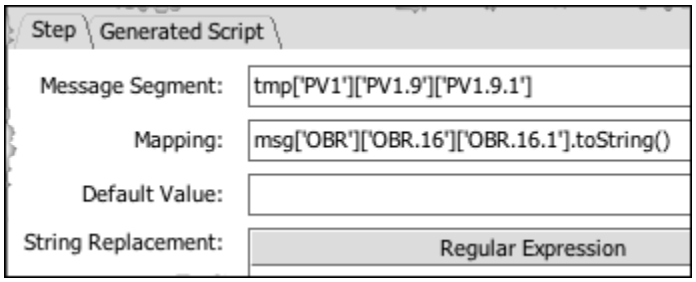
Working With Iterators

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 us 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.



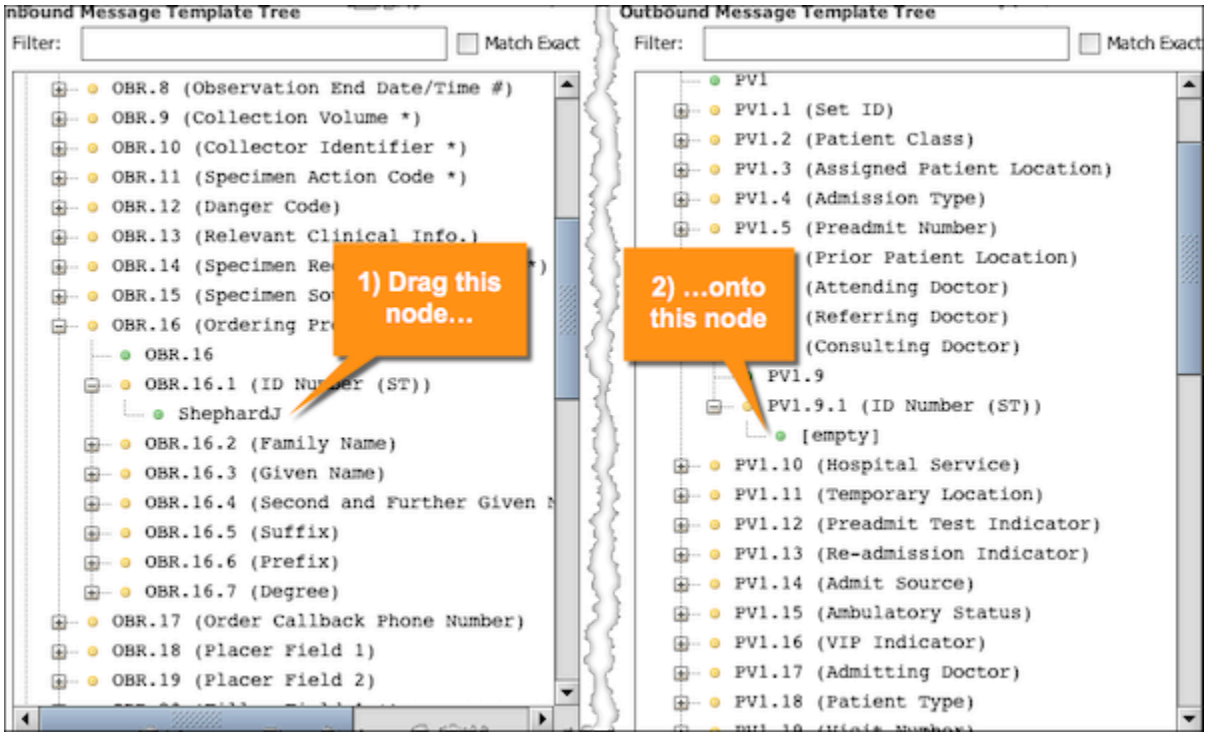
Typically you would do this with a Message Builder Transformer Step *(on page 282)*:

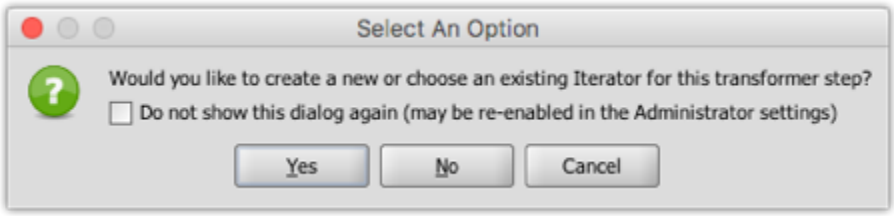


This works so far when there is only **one** OBR segment and **one** OBR.16 field. But what if you want to handle multiple segments or repeating fields? This is where Iterators come into play.

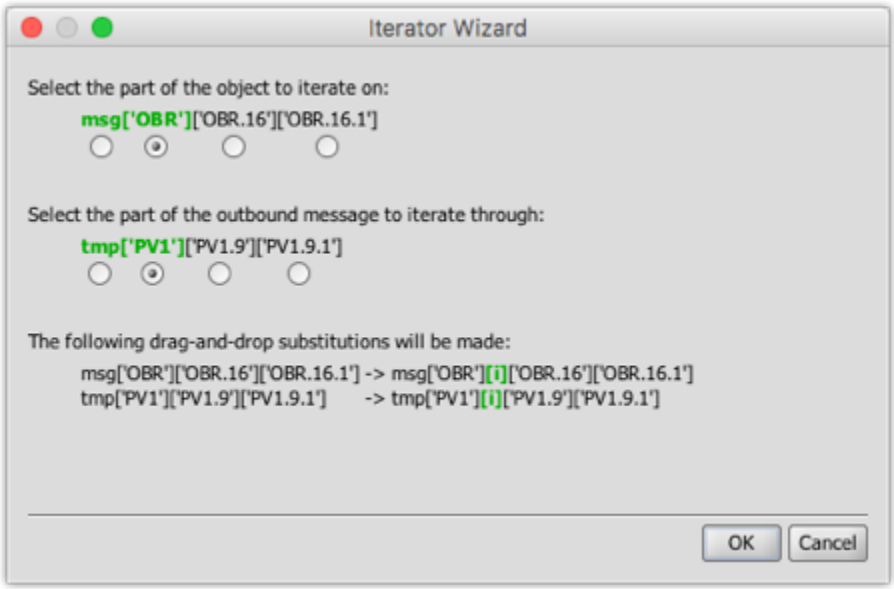
Creating Iterators From Drag-and-Drop

As explained in the Message Trees Tab *(on page 265)* 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.



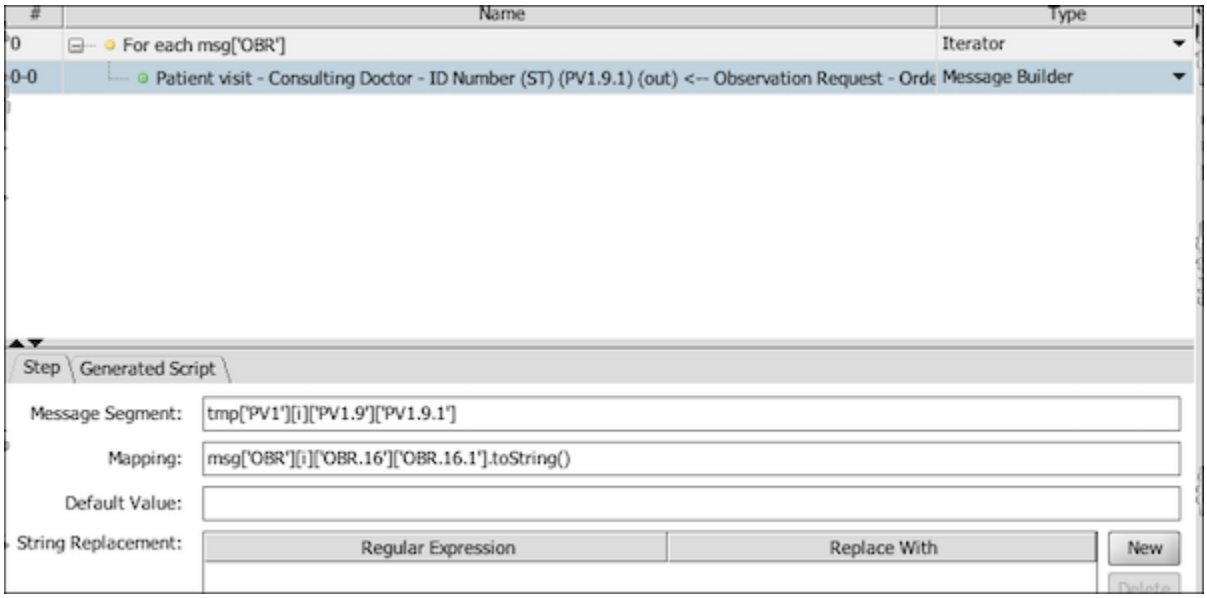


If **Yes** is chosen, the **Iterator Wizard** dialog is shown.



In this wizard dialog you can select your iteration target (what to iterate on). If you have 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 are essentially telling the wizard "where the **i** goes"

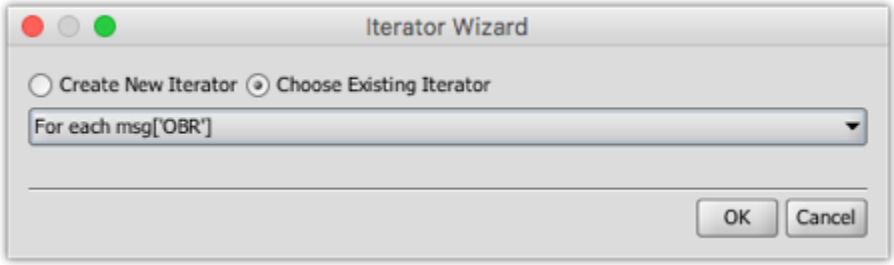
Once you select **OK**, the Iterator and subsequent rule/step will be created.



So far it is 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

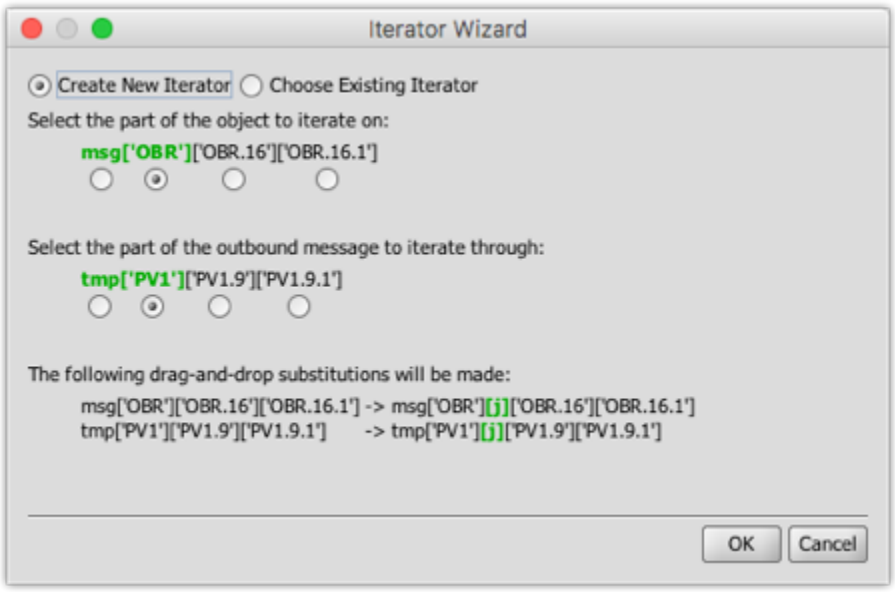
This task takes the currently selected rule/step, and either moves it to an existing Iterator, or creates an entirely new Iterator and puts the rule/step within it. When selecting the task, a dialog appears.



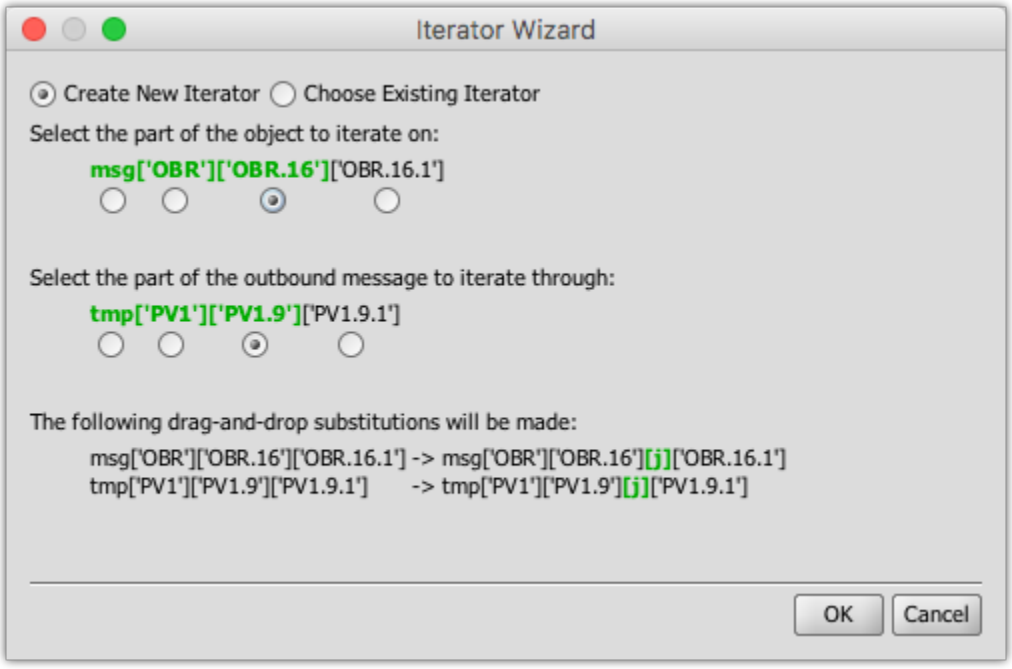
The **Choose Existing Iterator** option allows you to take the currently selected rule/step and move it

underneath a specific pre-existing Iterator. If your rule/step already belongs to an Iterator, by default this option is selected and the current parent is selected in the drop-down menu.

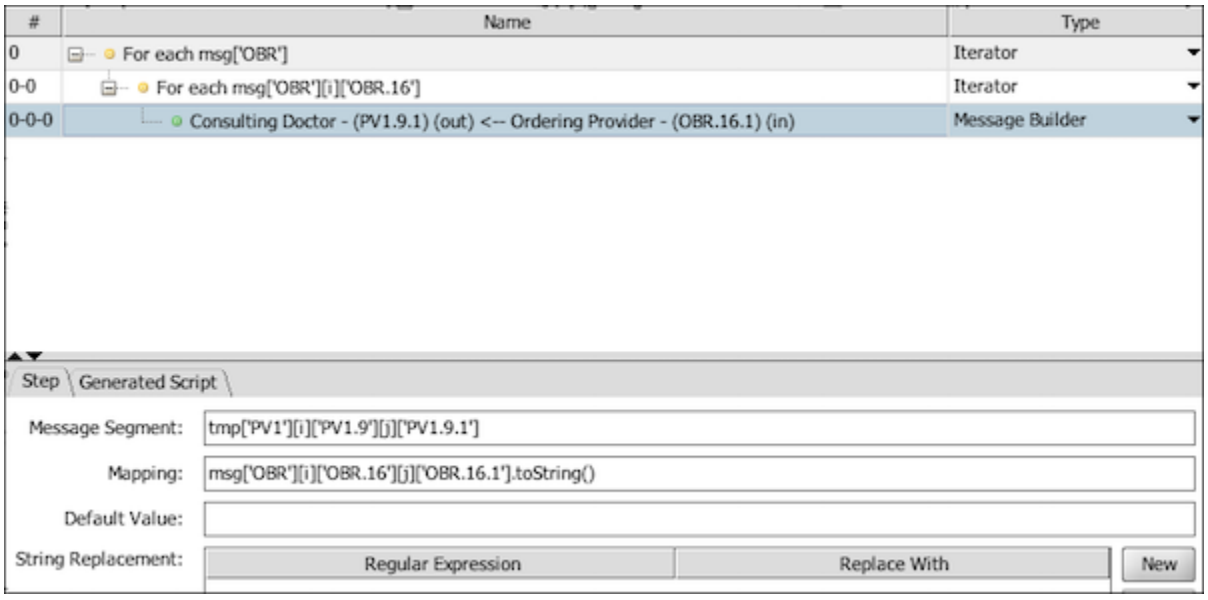
The **Create New Iterator** option is the same dialog shown before, where you have the option of choosing what to iterate on.



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 **i** index variable. We can then choose to iterate through each **OBR.16** field instead of on each OBR segment:.



After selecting **OK**, the step will now be placed inside a new nested Iterator. You can see from the following example that OBR.16.1 is being mapped into PV1.9.1, but now it is being done for each OBR segment, and in turn **for each** OBR.16 field repetition.

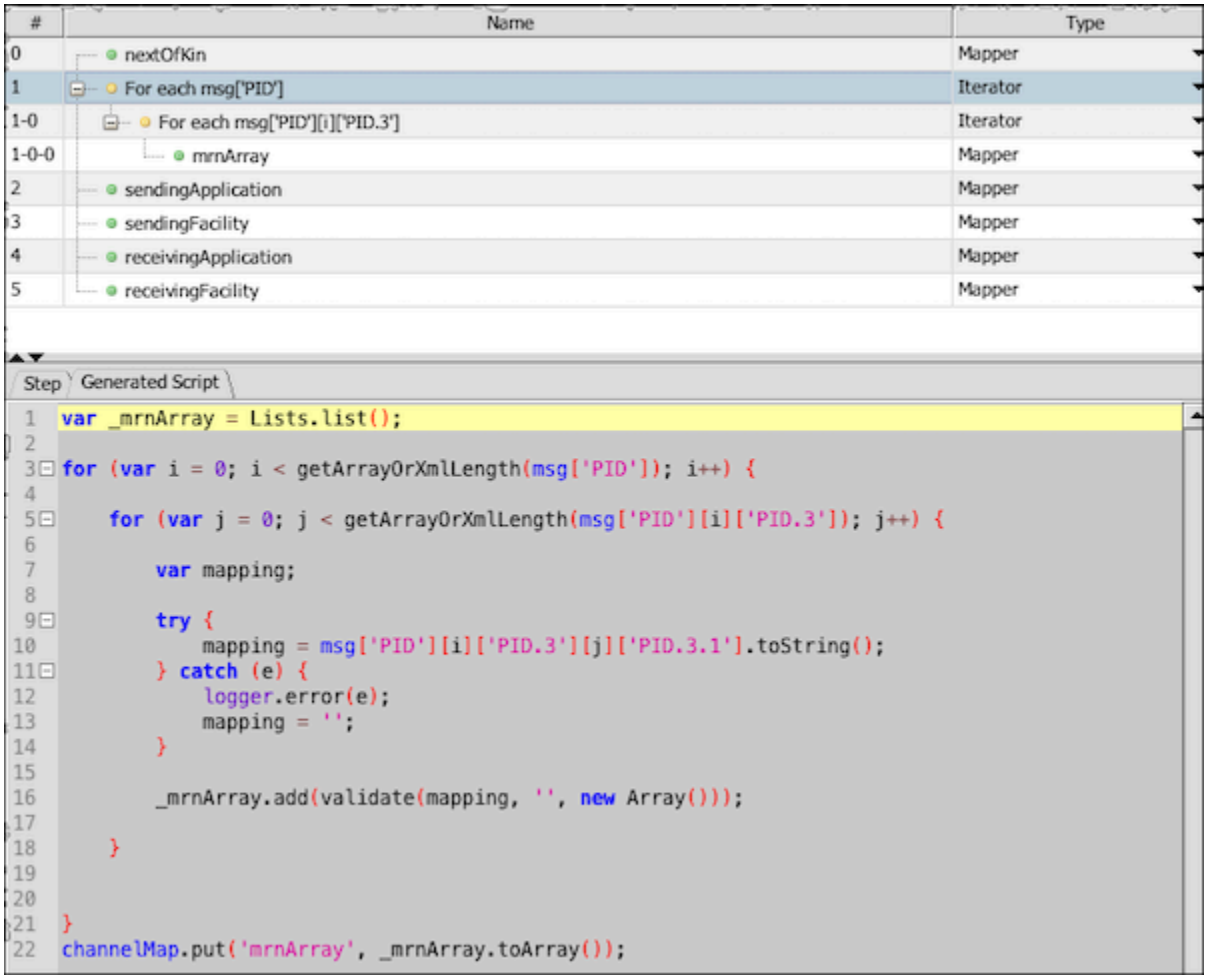


Remove From Iterators

If you decide later that a rule or step should not be part of an Iterator, it is easy to undo those changes. Select the rule or step, then select the **Remove From Iterator** task. The rule/step is moved one level higher in the tree. So if the rule/step is currently nested under multiple Iterators, select the task multiple times until it is at the depth you want.

View Generated Script

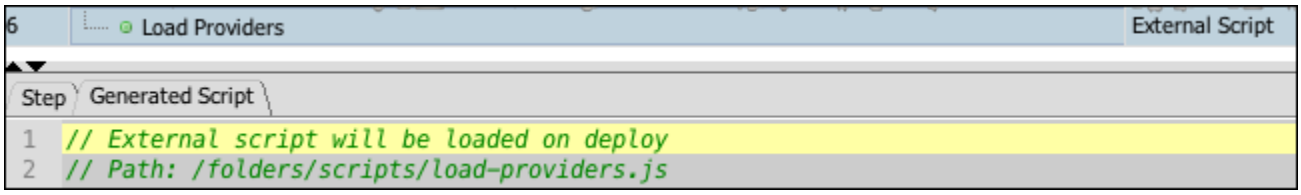
For all rule / step types, the properties panel shown in the bottom half of the screen is split into two tabs: **Rule/Step**, and **Generated Script**. Selecting the Generated Script tab shows the equivalent JavaScript that will execute when your channel is deployed and a message is sent through:



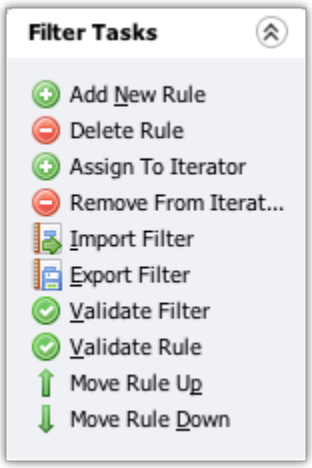
The script pane is not editable, but you can still select code and expand/collapse code folds. As shown in the screenshot above, selecting an Iterator rule/step shows you the script for the Iterator and all of its

children at once.

Note: External Script rules/steps are an exception and will **not** show the actual script that resides on the server:

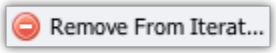
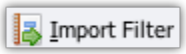
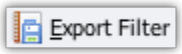
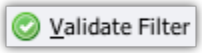
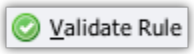
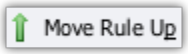
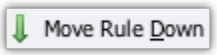


Filter Tasks



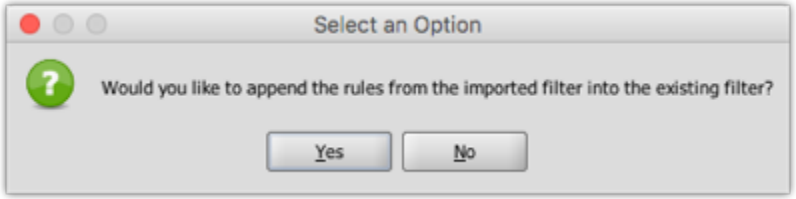
The following context-specific tasks are available throughout the Edit Filter View *(on page 257)*:

Task Icon	Task Name	Description
Add New Rule	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 is placed at the end of the children of the most immediate parent Iterator. Otherwise, the new rule is placed at the very end of the list at the bottom of the table.
Delete Rule	Delete Rule	Removes the currently selected rule from the table. If an Iterator rule is deleted, all of its children are also deleted.

Task Icon	Task Name	Description
	Assign To Iterator	Adds the selected rule to a new or existing Iterator. For additional information, see Working With Iterators *(on page 288)*.
	Remove From Iterator	Removes the selected rule from its current Iterator. For additional information, see Working With Iterators *(on page 288)*.
	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. For additional information, see Import Filter *(on page 296)*
	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 *(on page 288)*.
	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 *(on page 288)*.

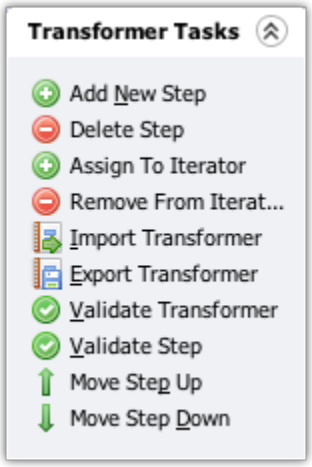
Import Filter

When this task is selected, you will be presented with a prompt.



If **Yes** is chosen, the rules from the filter is added at the end of the current list at the bottom of the table. If **No** is chosen, all rules currently in the table are deleted and replaced with the rules from the imported filter.

Transformer Tasks



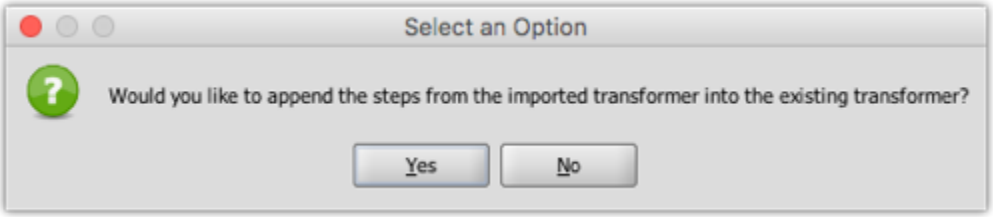
The following context-specific tasks are available throughout the Edit Transformer View *(on page 257)*:

Task Icon	Task Name	Description
Add New Step	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 is 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	Delete Step	Removes the currently selected step from the table. If an Iterator step is deleted, all of its children are also deleted.
Assign To Iterator	Assign To Iterator	Adds the selected step to a new or existing Iterator. For additional information, see Working With Iterators *(on page 288)*.
Remove From Iterat...	Remove From Iterator	Removes the selected step from its current Iterator. For additional information: Working With Iterators *(on page 288)*
Import Transformer	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. For additional information, see Import

Task Icon	Task Name	Description
		Transformer *(on page 298)*
Export Transformer	Export Transformer	Exports the current transformer (inbound/outbound data types and all steps) to an XML file.
Validate Transformer	Validate Transformer	Validates the entire transformer and all steps. This includes property validation and script syntax validation.
Validate Step	Validate Step	Validates the currently selected step. This includes property validation and script syntax validation.
Move Step Up	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 *(on page 288)*.
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 *(on page 288)*.

Import Transformer

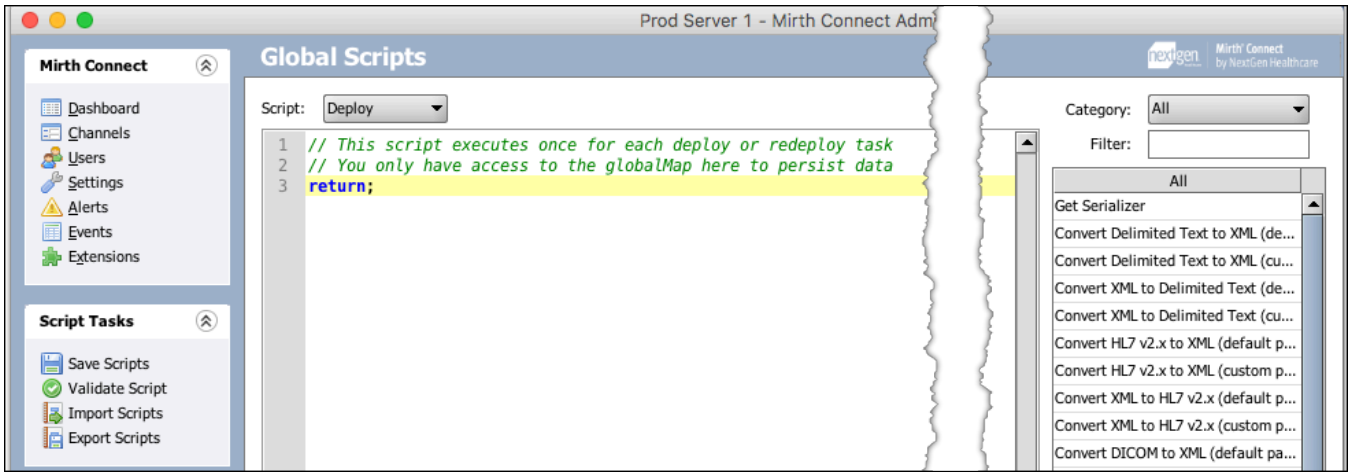
When this task is selected, you are presented with a prompt:



If **Yes** is chosen, the steps from the filter is 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 is overwritten.

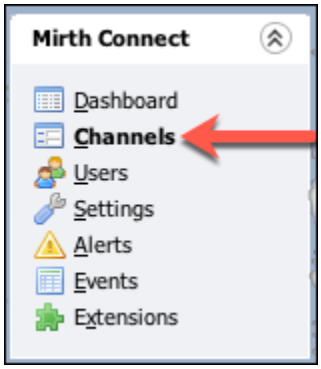
Edit Global Scripts View

This view is similar to the **Scripts** tab, in the Edit Channel view, except that the scripts here are global across all channels.

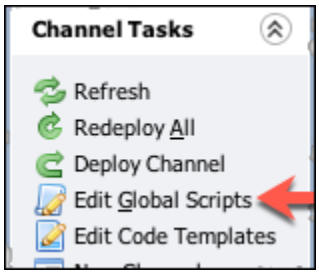


Navigation

Select the **Channels** link in the Mirth® Connect task pane on the top-left side of the window to enter the Channels View *(on page 148)*:



Select the **Edit Global Scripts** task in the Channel Tasks area.



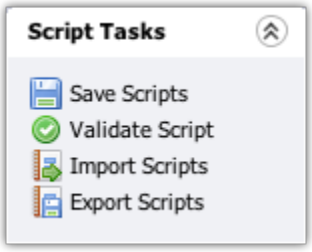
Edit Global Scripts

Select a script type from the drop-down menu and edit the script in the text area below. There is also a Reference List *(on page 274)* 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 deploys. You have access to the global / configuration maps here. Typically this script is used to perform a one-time operation, such as loading 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 *(on page 219)* 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 is executed and then the channel-level preprocessor.
- **Post Processor 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 post processor returned a response, it is available as the variable "response" in the global post processor. 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 is sent back to the originating system.

Tasks

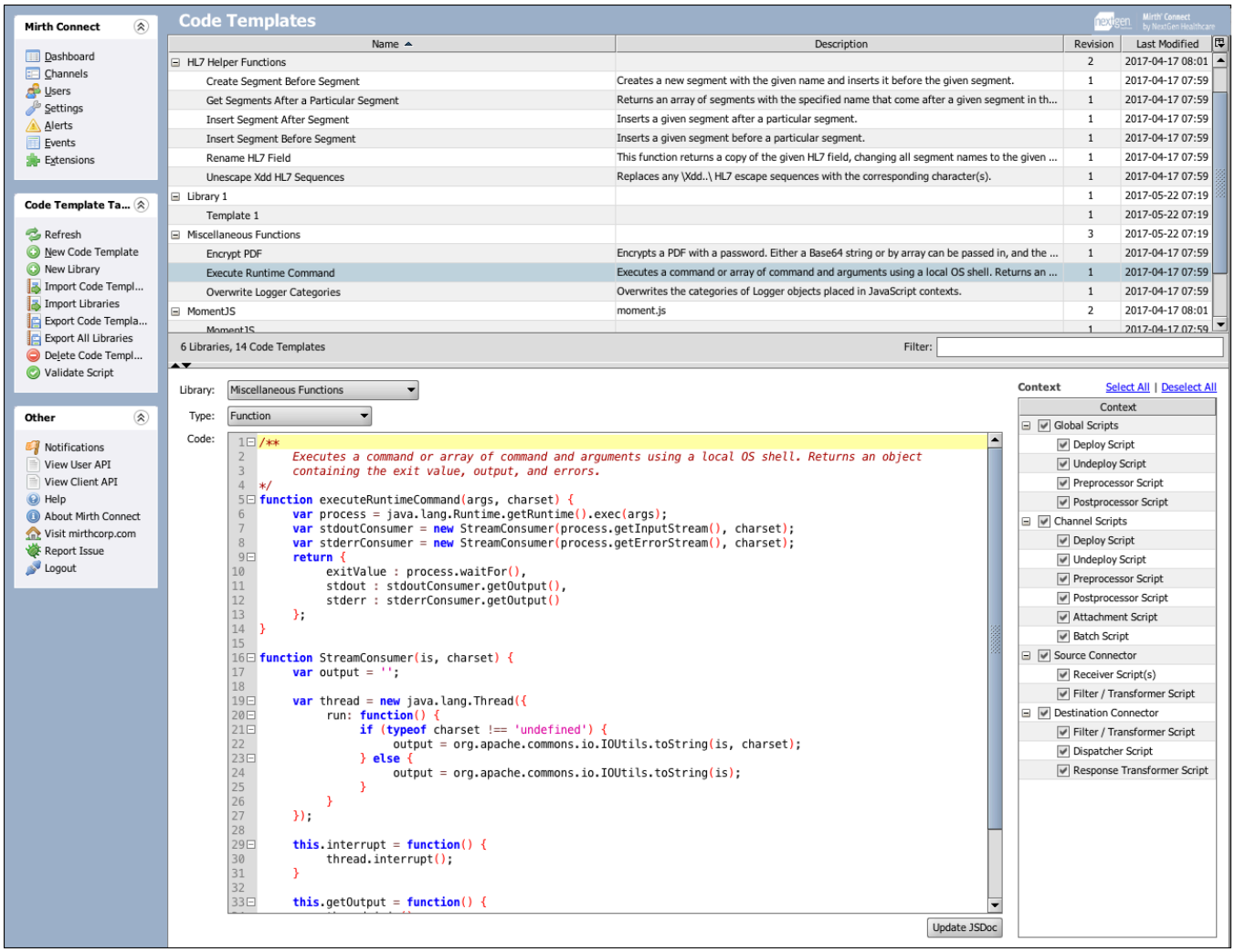


The following context-specific tasks are available:

Task Icon	Task Name	Description
Save Scripts	Save Scripts	Saves all global scripts.
Validate Script	Validate Script	Validates the currently viewed script, ensuring that there are no syntax errors.
Import Scripts	Import Scripts	Imports all global scripts from an XML file.
Export Scripts	Export Scripts	Exports all global scripts to an XML file.

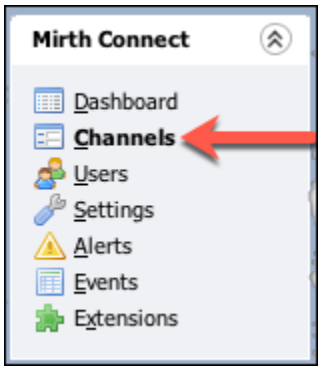
Edit Code Templates View

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 are available to the channel. This view is where code templates and libraries are configured for your Mirth® Connect server.

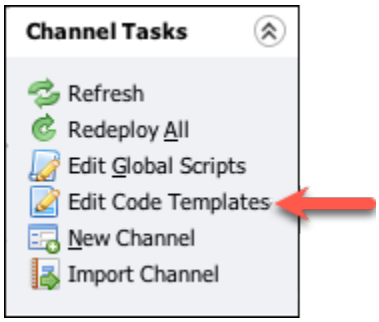


Navigation

Select the **Channels** link in the Mirth® Connect task pane at the top-left side of the window to enter the Channels View *(on page 148)*:



Select the **Edit Code Templates** link in the Channel Tasks task pane to the left.

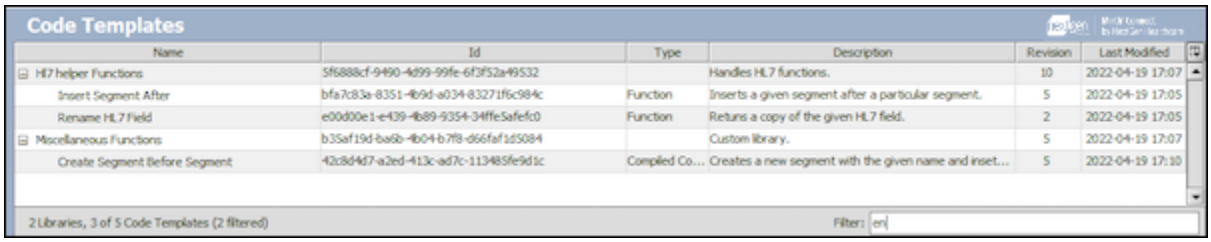


This section is separated into the following topics:

- Code Template Library Table *(on page 303)*
- Edit Library Panel *(on page 304)*
- Edit Code Template Panel *(on page 305)*
- Code Template Tasks *(on page 310)*

Code Template Library Table

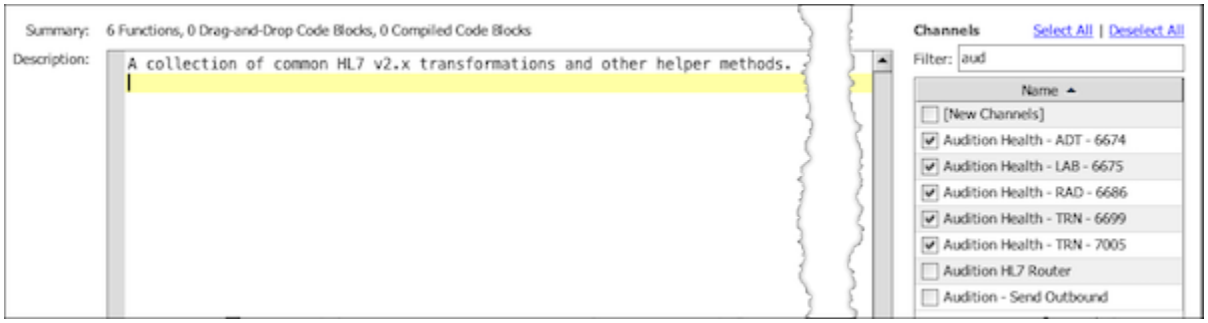
The table in the top section of the Edit Code Templates View *(on page 301)* shows all currently configured code templates and libraries. The name column is displayed as a tree, showing libraries at the hierarchical top level, and all child code templates at a lower level. The filter field at the bottom 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 Mirth® Connect, see Mirth Connect Administrator *(on page 79)*.



Column	Description
Name	The name of the code template or library. This column displays 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 *(on page 305)*.
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 **or** 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.

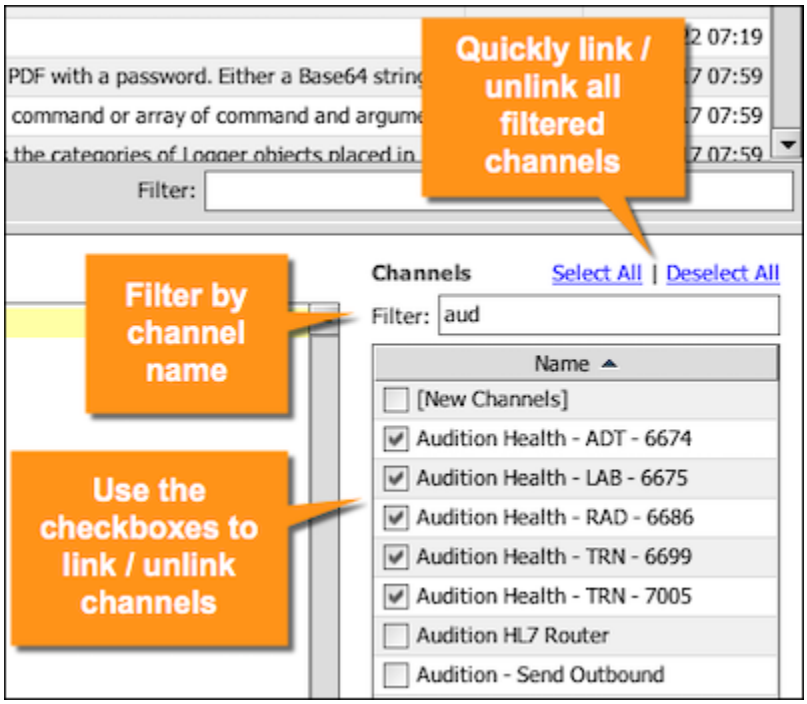
Edit Library Panel

When any library is selected in the Code Template Library Table *(on page 303)*, 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.



Link 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 can 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 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 *(on page 213)* in the Summary Tab *(on page 204)* of the Edit Channel View *(on page 203)*.

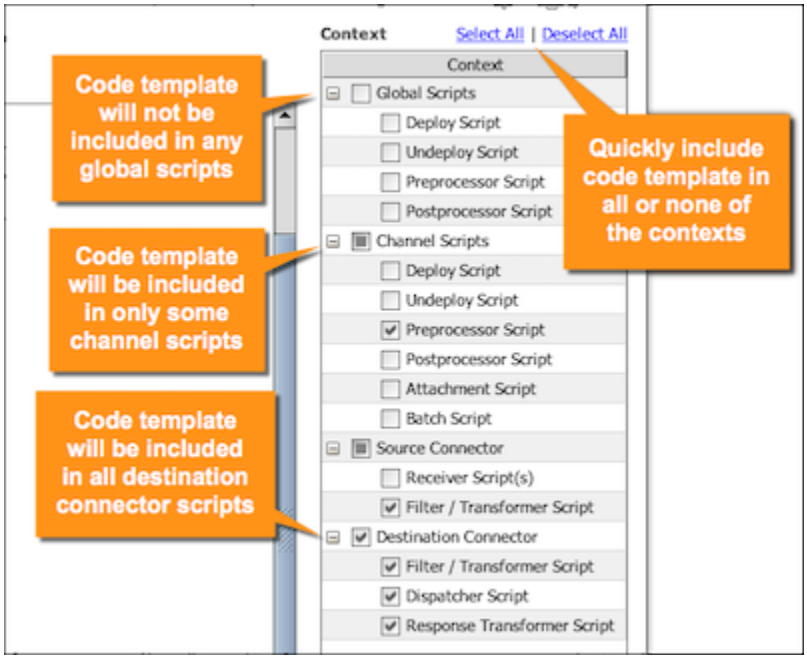
Edit Code Template Panel

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	The type of code template to create. • **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 **needed** can help optimize memory usage. For additional

Item	Name	Description
		information, see Code Template Contexts *(on page 307)*.
E	Update JSDoc	Generates / updates a JSDoc at the beginning of your code, with parameter / return annotations as needed. For additional information, see Use JSDoc in Code Templates *(on page 308)*.

Code Template Contexts



Similar to how a code template library *(on page 304)* 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, **all** global scripts, in a single select. The following groups are displayed:

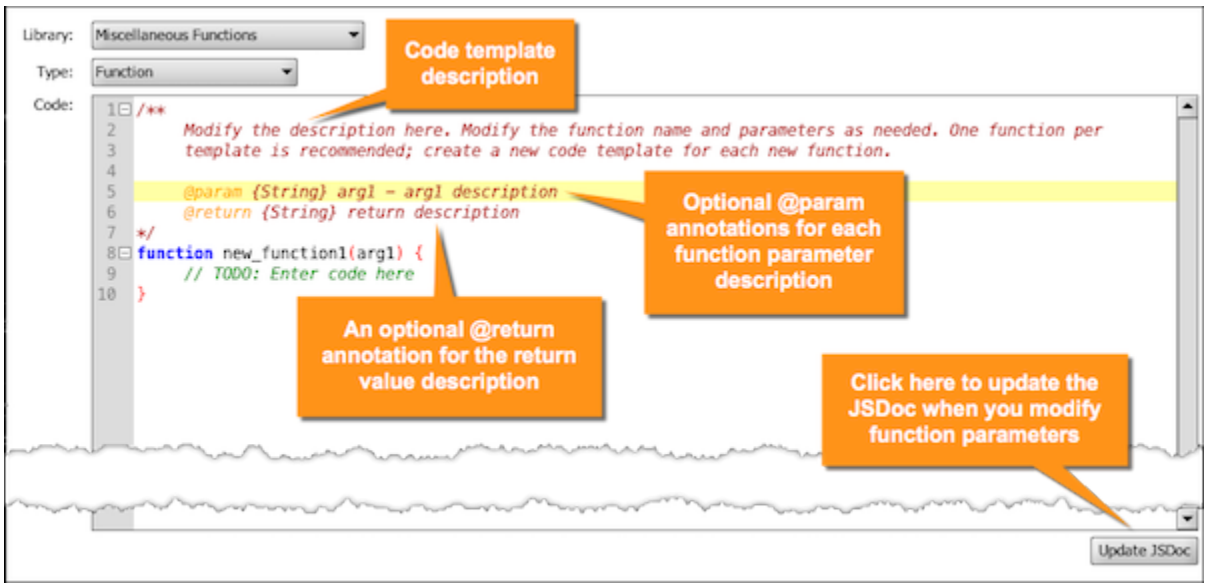
- **Global Scripts:** The global deploy/undeploy/preprocessor/postprocessor scripts, not specific to any particular channel. For additional information, see Edit Global Scripts View *(on page 299)*.
- **Channel Scripts:** The channel-level deploy/undeploy/preprocessor/postprocessor scripts (more info here *(on page 255)*) , as well as the JavaScript Attachment Handler *(on page 224)* and JavaScript Batch Adapter (for additional information, see Data Types *(on page 323)*) .

- **Source Connector:** The source filter/transformer script *(on page 257)*, and any script associated with the source connector. Examples of this include the JavaScript Reader *(on page 371)*, Database Reader *(on page 346)* (in JavaScript mode), and the JavaScript HTTP Authentication script *(on page 243)*.
- **Destination Connector:** The destination filter/transformer script *(on page 257)*, the response transformer *(on page 286)*, and any script associated with the destination connector. Examples of this include the JavaScript Writer *(on page 410)* and Database Writer *(on page 390)* (in JavaScript mode).

Use JSDoc in Code Templates

A JSDoc is a type of comment block used to annotate JavaScript scripts. For code templates in Mirth® Connect, 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 *(on page 445)*.

When you create a new code template, a sample JSDoc is created for you:



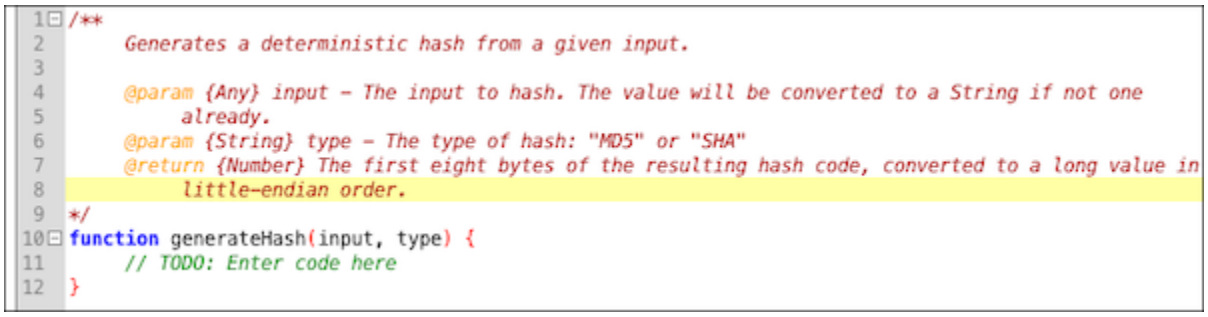
The first portion of the comment block is used for the description of the code template, and may contain multiple lines and blank lines. Following that, you can include any JSDoc annotations, your own custom annotations, or whatever you want, as long as it follows correct JavaScript syntax. Only the following annotations are recognized by Mirth Connect for the purpose of populating the auto-completion dialog in the JavaScript Editor *(on page 445)*:

- **@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`

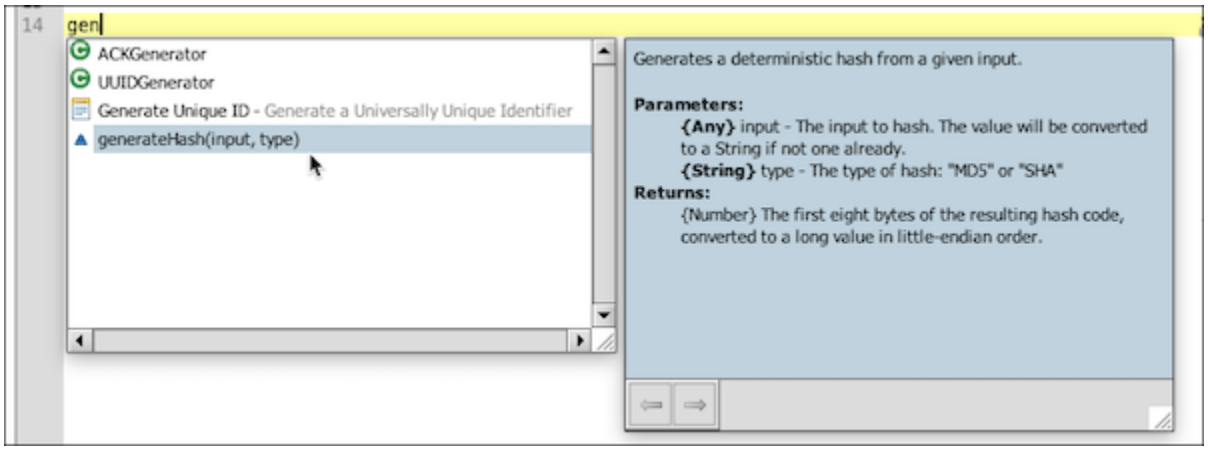
  Note that the type doesn't have to be an actual JavaScript type. It can be anything you want as long as it doesn't contain the "{}" characters, like "MyObject", "String/Number", etc.

- **@return:** Documents the return value for your function, if applicable. The format is:
    - `@return Description`

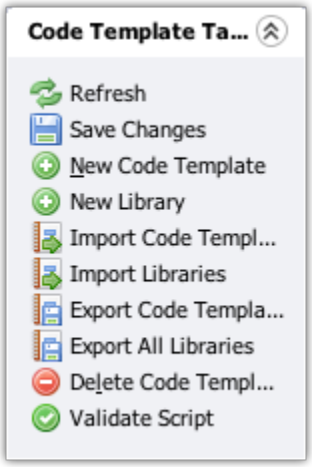
If you change the names or number of arguments in your function, you can use the **Update JSDoc** button at the bottom of the editor. This will automatically inject or update @param annotations as needed, after which you can change the Type and Description as needed.



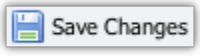
Once you have filled out your JSDoc appropriately, the function appears in the auto-completion dialog accessible from the JavaScript Editor *(on page 445)*:

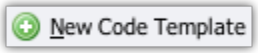
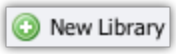
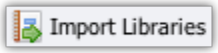
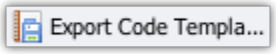
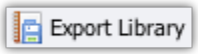
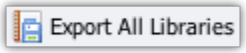
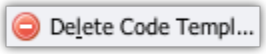
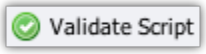


Code Template Tasks



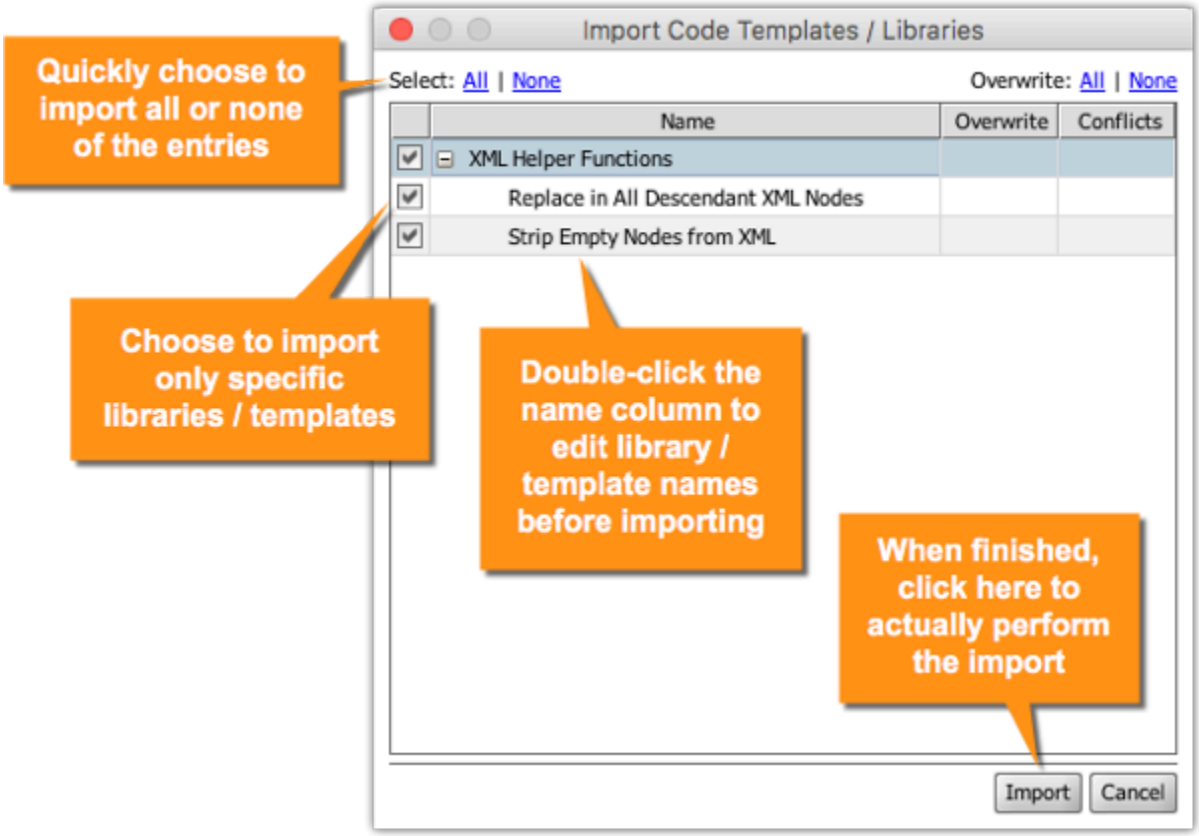
The following context-specific tasks are available throughout the Edit Code Templates View *(on page 301)*:

Task Icon	Task Name	Description
	Refresh	Refreshes the list of code templates / libraries. If there are unsaved changes, you are 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.

Task Icon	Task Name	Description
	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 becomes available.
	New Library	Creates a new code template library and adds it to the table. By default the library is not included in any channels. Use the Channels table in the Edit Library Panel *(on page 304)* to link the library to specific channels.
	Import Code Templates *(on page 307)*	Import a single code template or a list of code templates from an XML file.
	Import Libraries *(on page 311)*	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 Template	Deletes the currently selected code template, removing it from the table.
	Delete Library	Deletes the currently selected code template library, **and all code templates** belonging to the library.
	Validate Script	Validates the currently selected code template, ensuring that all properties are valid, and that the actual code has proper syntax.

Import Code Templates/Libraries

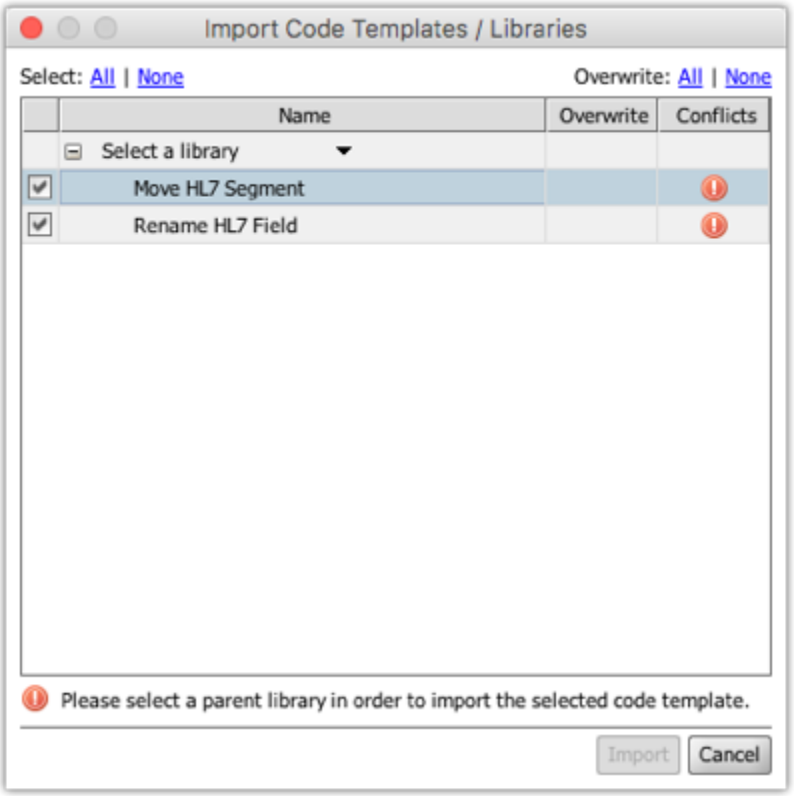
When importing code templates/libraries, either in the Edit Code Templates View *(on page 301)* or when importing a channel *(on page 154)* containing libraries, you are presented with a confirmation dialog.



All libraries and code templates contained in the import file/channel appear in the dialog. From here you can choose to import everything, or individually select specific entries to import. Once you have made your selections, select the **Import** button to actually perform the import operation.

- If a code template or library already exists in your Mirth® Connect server, the **Overwrite** column displays 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, the **Conflicts** column displays 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 are trying to import into, you will 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.

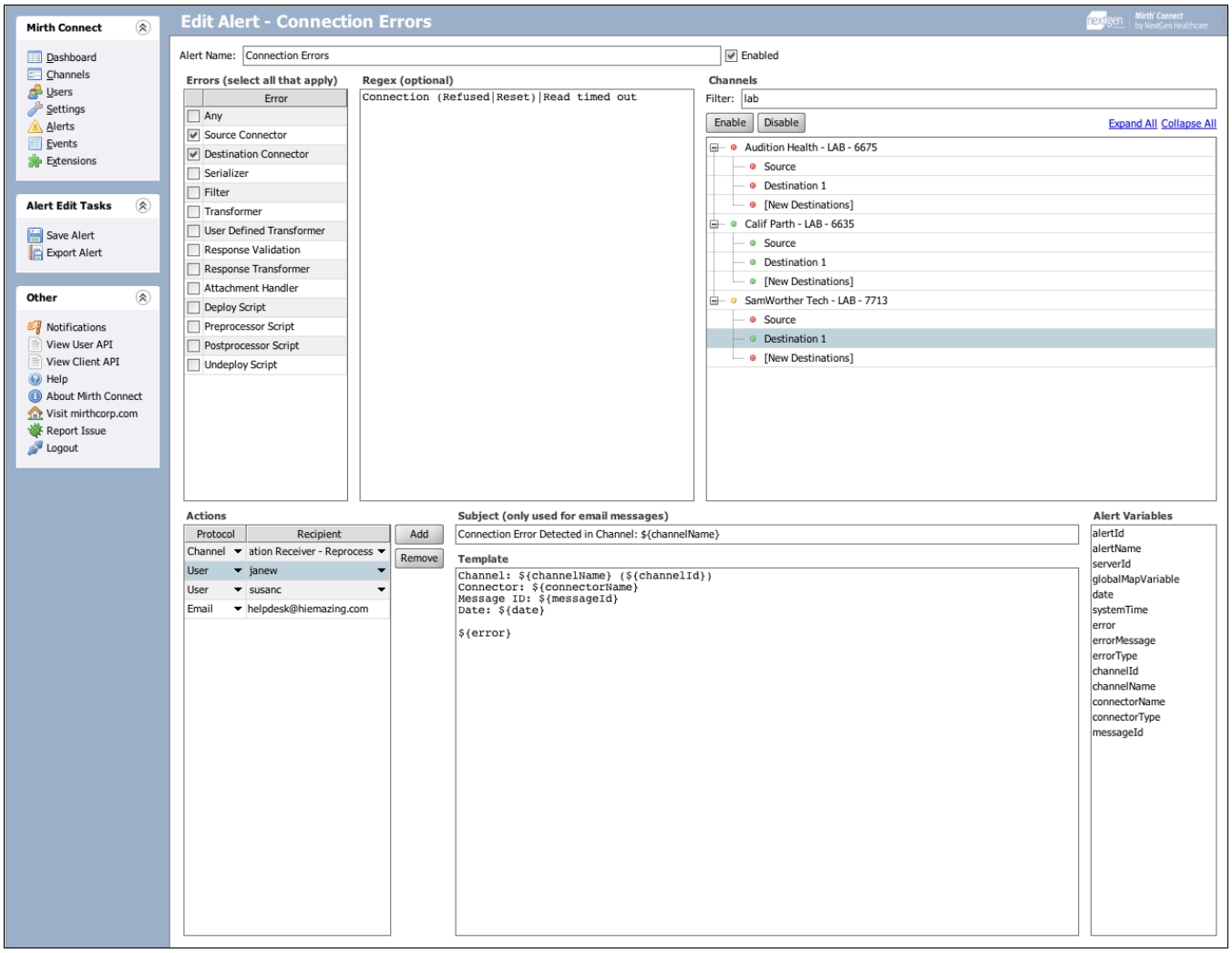
When importing only code templates by themselves, you must first select a library to import them into.



Edit Alert View

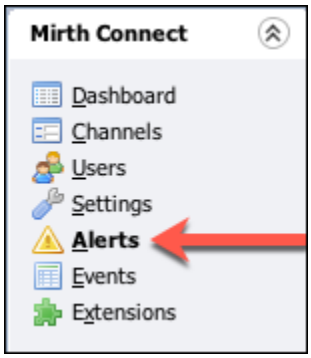
An alert is a process that listens for certain types of events and triggers based on configurable settings. From these triggers you can take various actions, like dispatching an e-mail to a user or specific address, or sending a message to a channel. Mirth® Connect by NextGen Healthcare comes with a built-in Error-based alerting system that listens for error events from selected channels. The Advanced Alerting *(on page 583)* 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.

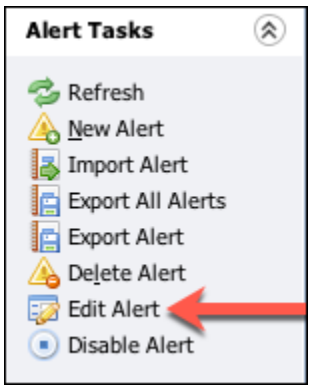


Navigation

Select the **Alerts** link in the Mirth® Connect task panel at the upper-left side of the window.



Select the alert you want to edit, and select the **Edit Alert** task in the Alert Tasks panel on the left.



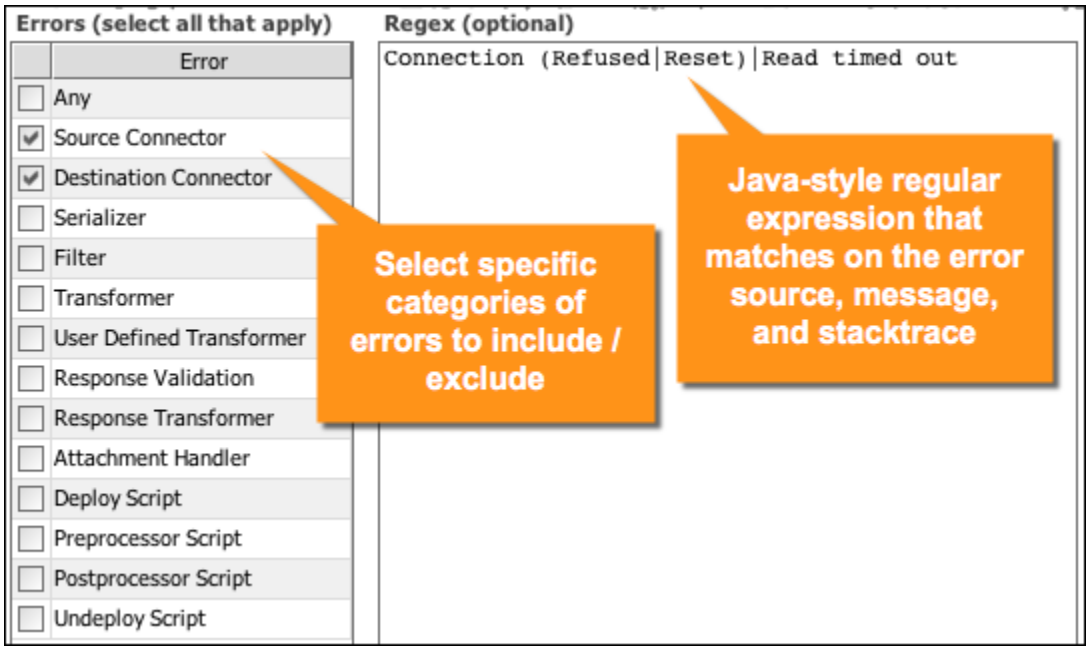
You can also double-click the alert from the Alerts Table *(on page 140)*.

This section is separated into the following topics:

- Alert Error Types and Regex *(on page 315)*
- Alert Enabled Channels *(on page 317)*
- Alert Actions *(on page 318)*
- Edit Alert Tasks *(on page 319)*

Alert Error Types and Regex

This section of the Edit Alert View *(on page 313)* 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.



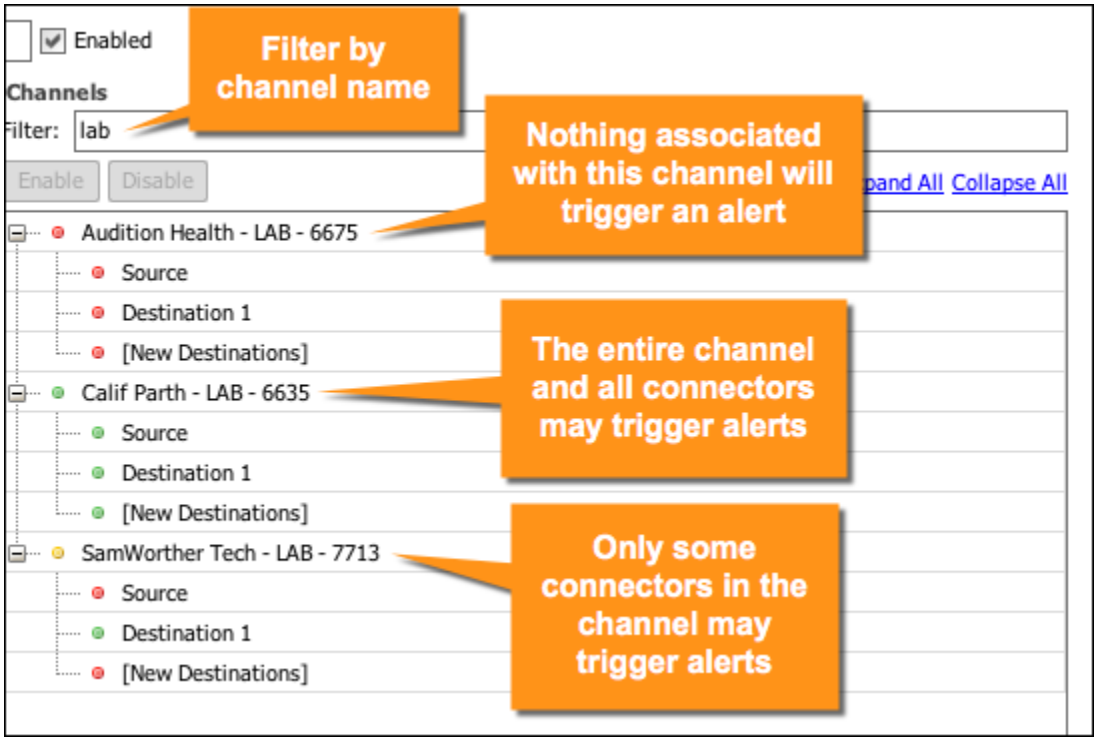
Alert Error Categories

- **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) *(on page 459)*.
- **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 *(on page 219)*.
- **Deploy Script**: An error that originated from a channel deploy script.
- **Preprocessor Script**: An error that originated from a global or channel preprocessor script.
- **Post Processor Script**: An error that originated from a global or channel post processor script.
- **Undeploy Script**: An error that originated from a channel undeploy script.

Alert Enabled Channels

This section of the Edit Alert View *(on page 313)* 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.

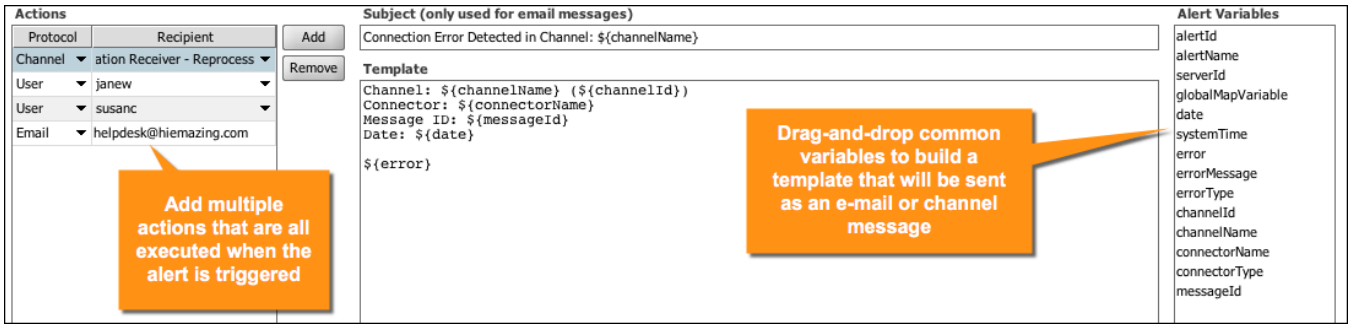


The **[New Channels]** option indicates that you want an alert to listen for error events on any **new** channels that are created for a channel after the alert was initially created. Similarly, the **[New Destinations]** option indicates that you want an alert to listen for error events on any **new** destinations

that are created for a channel after the alert was initially created.

Alert Actions

This section of the Edit Alert View *(on page 313)* 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.



Protocols

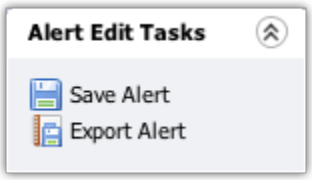
All protocols also support Velocity Variable Replacement *(on page 469)*, you can use any of the Alert Variables below, or any Global Map *(on page 453)* variable.

- **Channel**: The template is sent as a message to the channel selected in the Recipient column. The subject is not used for this protocol.
- **Email**: The subject and template is used to send an e-mail to the address specified in the Recipient column.
- **User**: Same as the Email protocol, the e-mail address used is the one configured for the selected user.
- **Role**: E-mails will be sent to *all* users within a particular Role-Based Access Control *(on page 599)* role. Only visible when User Authorization is installed.

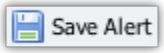
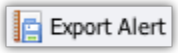
Alert Variables

- **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 *(on page 315)*.
- **channel Id**: 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.

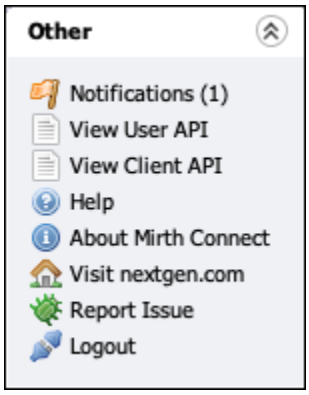
Edit Alert Tasks



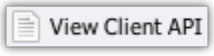
The following context-specific tasks are available throughout the Edit Alert View *(on page 313)*:

Task Icon	Task Name	Description
	Save Alert	Saves all changes made to the current alert. If the alert is enabled, it automatically listens for new events to trigger on. Note that since many alerts rely on dispatching e-mails, you are warned if your global SMTP settings are not configured on the Server Settings Tab *(on page 167)*.
	Export Alert	Exports the current alert to an XML file.

Other Tasks



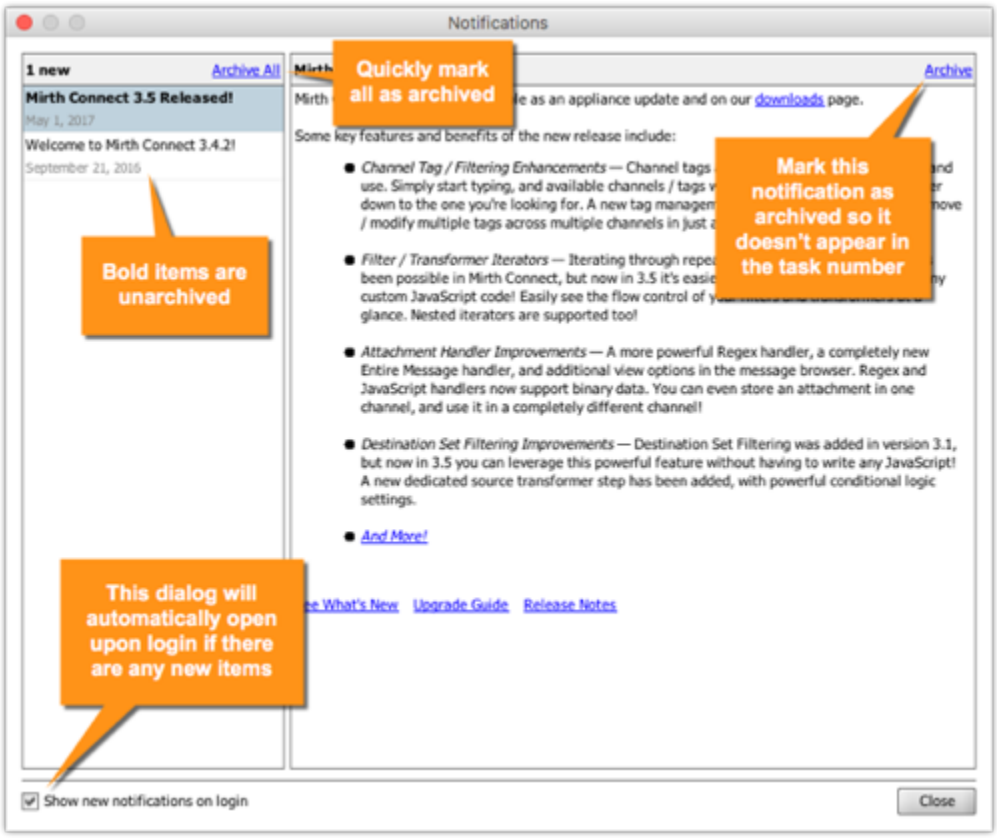
The following tasks are available at any time at the bottom-left of the task pane throughout the Mirth® Connect Administrator.

Task Icon	Task Name	Description
	Notifications *(on page 321)*	View all notifications received from Mirth® Connect headquarters. This enables you to be notified when there is a new version of Mirth® Connect 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 *(on page 321)*.
	View User API	Opens the User API (Javadoc) *(on page 459)* documentation in your default browser.
	View Client API	Open the Mirth Connect REST API *(on page 473)* documentation in your default browser.

Task Icon	Task Name	Description
	Help	Opens the Mirth® Connect Wiki page in your default browser.
	About Mirth® Connect	Opens the About window for Mirth Connect, 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 window are not an exhaustive list; look at the "docs/thirdparty" folder in the Installation Directory *(on page 478)* for a more complete list.
	Visit nextgen.com	Opens the NextGen Healthcare website in your default browser.
	Report Issue	Opens the Mirth public issue tracker in your default browser. Search the tracker for similar issues before creating a new issue. For defects/bugs, try to include full reproduction steps, error stacktraces, the Mirth® Connect / Java versions, and as much information as you can.
	Logout	Logs off the Mirth® Connect Administrator, and brings you back to the Login Dialog *(on page 55)*.

Notifications

This dialog shows all messages you have received from Mirth® Connect headquarters. This allows you to be notified when a new version of Mirth® Connect 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 *(on page 320)* 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 clear " **Show new notifications on login** ".

CHAPTER 7

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 *(on page 72)*.

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 Window *(on page 207)* 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 *(on page 238)*. 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 *(on page 238)*, 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 *(on page 249)*. 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).

Mirth® Connect supports the following data types:

- Delimited Text Data Type *(on page 324)*
- DICOM Data Type *(on page 327)*
- EDI / X12 Data Type *(on page 328)*
- HL7 v2.x Data Type *(on page 329)*
- HL7 v3.x Data Type *(on page 332)*
- JSON Data Type *(on page 333)*
- NCPDP Data Type *(on page 334)*
- Raw Data Type *(on page 335)*
- XML Data Type *(on page 336)*

An additional data type is made available as a commercial extension:

- ASTM E1394 Data Type *(on page 586)*

Also see:

- Batch Processing *(on page 337)*

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	Select the method for splitting the batch message. This option has no effect unless Process Batch is enabled in the Source Settings *(on page 238)*. The following options are available: • **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 *(on page 338)*.
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.

Name	Default Value	Description
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 *(on page 338)*.

DICOM Data Type

This data type works in conjunction with the DICOM Listener *(on page 341)* / DICOM Sender *(on page 386)* and the DICOM Attachment Handler *(on page 224)* 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">1.2.840.1000
8.5.1.4.1.1.4</tag00020002>
    <tag00020003 len="60" tag="00020003" vr="UI">1.3.46.670589.11.3
0.9.1062531302827752870602.13.1.1.1.0.0.1</tag00020003>
    <tag00020010 len="18" tag="00020010" vr="UI">1.2.840.10008.1.2</t
ag00020010>
    <tag00020012 len="18" tag="00020012" vr="UI">1.3.46.670589.1
7.1</tag00020012>
    <tag00020013 len="14" tag="00020013" vr="SH">ARCVTS04NOV99</tag00
020013>
    <tag00020016 len="14" tag="00020016" vr="AE">VTS_DCM_STORE</tag00
020016>
    <tag00080005 len="10" tag="00080005" vr="CS">ISO_IR 100</tag00080
005>
    <tag00080008 len="26" tag="00080008" vr="CS">ORIGINAL\PRIMARY\M_S
E\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 *(on page 238)*. The following options are available: • **JavaScript:** Use JavaScript to split messages. For additional information, see JavaScript Batch Script *(on page 338)*.
JavaScript		Enter JavaScript that splits the batch, and returns the next message. This script has access to 'reader', a Java

Name	Default Value	Description
		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 *(on page 338)*.

HL7 v2.x Data Type

This data type enables 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 enables 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 enables 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 do not 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 on 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 characters expected to occur after each

Name	Default Value	Description
		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 *(on page 238)*. The following options are available: • **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 *(on page 338)*.
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 *(on page 338)*.

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 *(on page 469)* with values from the current connector message.

Name	Default Value	Description
Successful ACK Message		The ACK message to respond with when the message processes successfully. This value supports Velocity Variable Replacement *(on page 469)* 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 *(on page 469)* 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 *(on page 469)* 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 *(on page 469)* 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 *(on page 469)* 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.

Name	Default Value	Description
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

Name	Default Value	Description
		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 *(on page 238)*. The following options are available: • **JavaScript:** Use JavaScript to split messages. For additional information, see JavaScript Batch Script *(on page 338)*.
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 *(on page 338)*.

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 *(on page 238)*. The following options are available: • **JavaScript:** Use JavaScript to split messages. For additional information, see JavaScript Batch Script *(on page 338)*.
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 *(on page 338)*.

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 *(on page 238)*. The following options are available: • **JavaScript:** Use JavaScript to split messages. For additional information, see JavaScript Batch Script *(on page 338)*.
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 *(on page 338)*.

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 *(on page 238)*. The following options are available:

Name	Default Value	Description
		• **JavaScript:** Use JavaScript to split messages. For additional information, see JavaScript Batch Script *(on page 338)*
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 *(on page 338)*

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	Disabled	Identifies the 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	Element Name	Select the method for splitting the batch message. This option has no effect unless

Name	Default Value	Description
		Process Batch is enabled in the Source Settings. The following options are available: • **Element Name**: Use the element name to split messages. Does not work with namespaces. • **Level:** Use the element level to split messages. Default is **1**. • **XPath Query**: Use a custom XPath Query to split messages. • **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 Mirth Connect 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 *(on page 238)*. To change how an incoming message is split into multiple messages, look at the **Batch Properties** section of the source inbound data type you are using. For example, the Delimited Text Data Type *(on page 324)* 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 *(on page 449)*

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. Since this value is returned from a map, it will be a java.lang.Boolean object. An example of how you could use this is: `if ($('batchComplete').booleanValue()) { // Code here will execute when batchComplete is true }`

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 *(on page 238)*, you can set this script from the Set Data Types Window *(on page 207)* 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 , 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();
```

CHAPTER 8

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 additional information on connectors in general, go here: About Channels and Connectors *(on page 60)*

Here is a list of source connectors supported by Mirth® Connect by NextGen Healthcare

- Channel Reader *(on page 340)*
- DICOM Listener *(on page 341)*
- Database Reader *(on page 346)*
- File Reader *(on page 351)*
- HTTP Listener *(on page 362)*
- JMS Listener *(on page 366)*
- JavaScript Reader *(on page 371)*
- TCP Listener *(on page 372)*
- Web Service Listener *(on page 381)*

Additional source connectors are available as commercial extensions *(on page 582)*:

- Email Reader *(on page 588)*
- Serial Connector *(on page 597)*
- FHIR Connector *(on page 590)*
- Interoperability Connector Suite *(on page 592)*

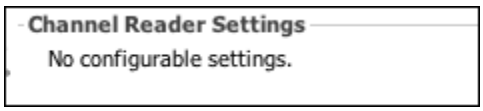
Channel Reader

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 *(on page 384)* or from a "router.routeMessage" call.

Supported property groups:

- Source Settings *(on page 238)*



Source Map Variables

If this connector receives a message from a Channel Writer *(on page 384)*, the following source map *(on page 449)* 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.

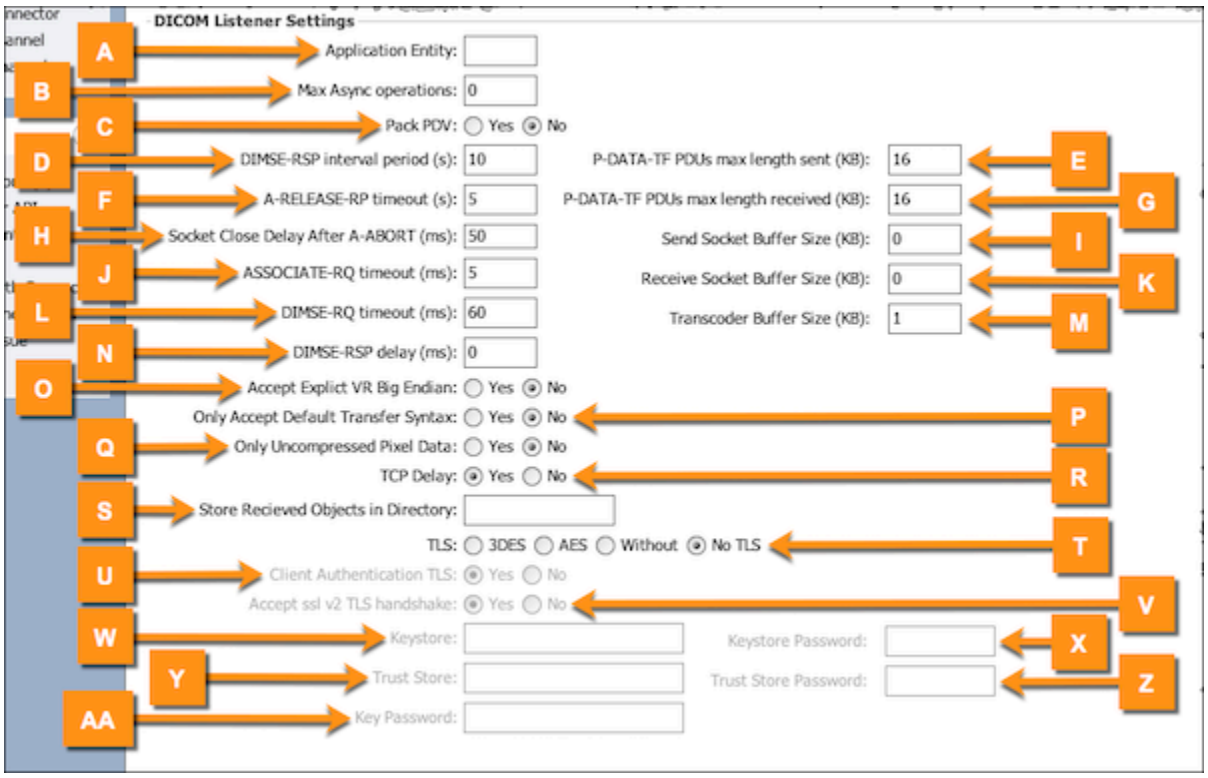
DICOM Listener

This source connector works in conjunction with the DICOM Attachment Handler *(on page 224)* and the DICOM Data Type *(on page 327)* to allow Mirth® Connect 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 *(on page 598)* extension.

Supported property groups:

- Listener Settings *(on page 233)*

- Source Settings *(on page 238)*



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.

Item	Name	Default Value	Description
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	Determines whether to receive data over an implicit SSL/TLS socket. The following options are available: • **3DES:** TLS will be used, with the **SSL_RSA_WITH_3DES_EDE_CBC_SHA** cipher suite.

Item	Name	Default Value	Description

			• **AES:** TLS will be used, with the following cipher suites:
			  • **TLS_RSA_WITH_AES_128_CBC_SHA**
			  • **SSL_RSA_WITH_3DES_EDE_CBC_SHA**
			• **Without:** TLS will be used without symmetric encryption, with the cipher suite **SSL_RSA_WITH_NULL_SHA**. 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.

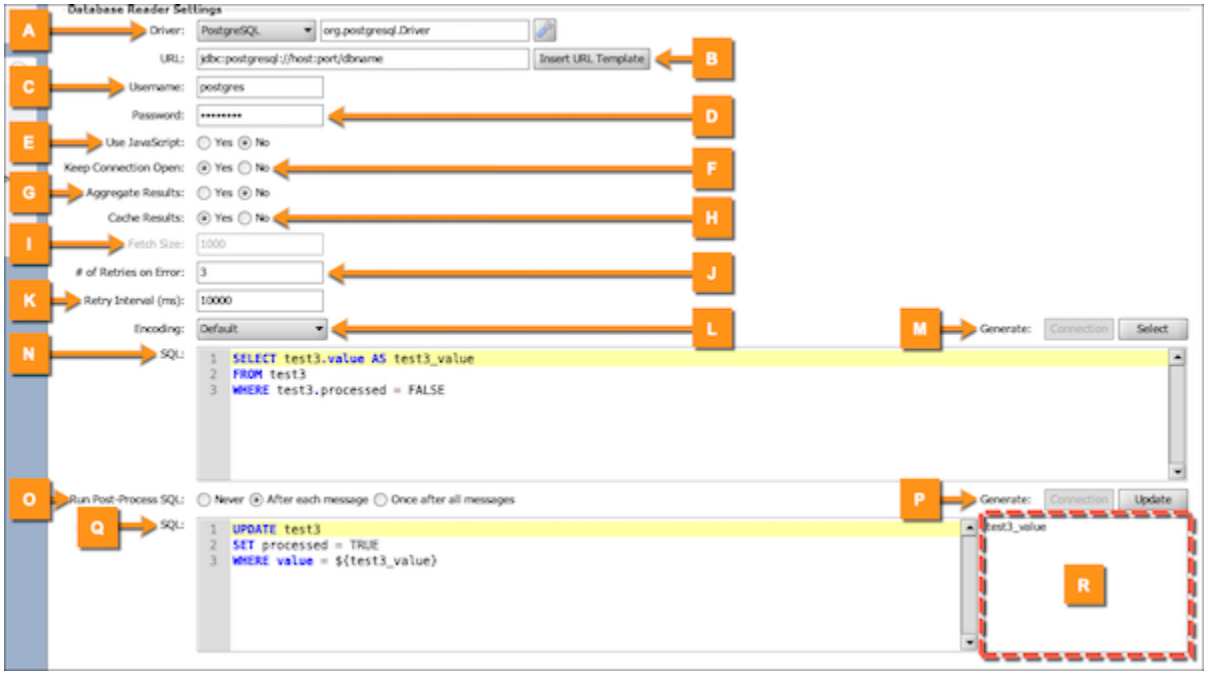
Key	Description
localPort	The port that that TCP socket is locally bound to.
remoteAddress	The IP address of the remote connecting client.
remotePort	The TCP port of the remote connecting client.
associateACProtocolVersion	The associate protocol version of the local SCP.
associateACImplClassUID	The associate 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	The type of user identity presented by the remote SCU, if available. Will be one of the following values: • USERNAME • USERNAME_PASSCODE • KERBEROS • SAML

Database Reader

This **Database Reader** 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 *(on page 234)*
- Source Settings *(on page 238)*



Item	Name	Default Value	Description
A	Driver		Specifies the type of database driver to use to connect to the database. The following values are supported by default:

Item	Name	Default Value	Description
			• Sun JDBC-ODBC Bridge • MySQL • Oracle • PostgreSQL • SQL Server / Sybase • SQLite Additional drivers can be added by editing dbdrivers.xml File *(on page 480)*.
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

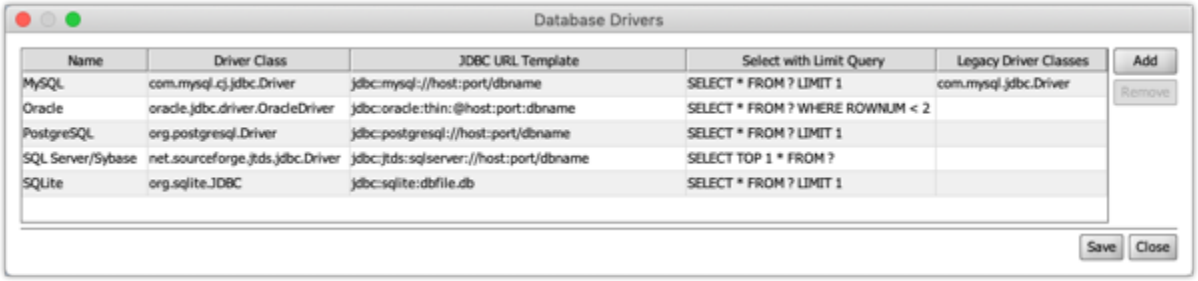
Item	Name	Default Value	Description
			once, so use this with caution.
H	Cache Results	Yes	Cache the entire result set in memory prior to processing messages.
I	Fetch Size	1000	The JDBC ResultSet fetch size to be used when fetching results from the current cursor position.
J	# of Retries on Error	3	The 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 Mirth® Connect is running on.
M	Generate		• **Connection**: This button is enabled when Use JavaScript is enabled. When selected, 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	Determines whether the post-process update script is active, and if so whether to execute it after each message or only once after all messages in the ResultSet have completed.If **Aggregate Results** is disabled:

Item	Name	Default Value	Description
			• **Never:** Do not run the post-process statement/script. • **After each message:** Run the post-process statement/script after each message finishes processing. • **Once after all messages:** Run the post-process statement/script only after all messages have finished processing. If **Aggregate Results** is enabled: • **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)		• **Connection**: This button is enabled when **Use JavaScript** is enabled and a post-process script is being used. When selected, 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.

Item	Name	Default Value	Description
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

Select 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 Class	Yes	The fully-qualified Java class name for the JDBC driver.
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

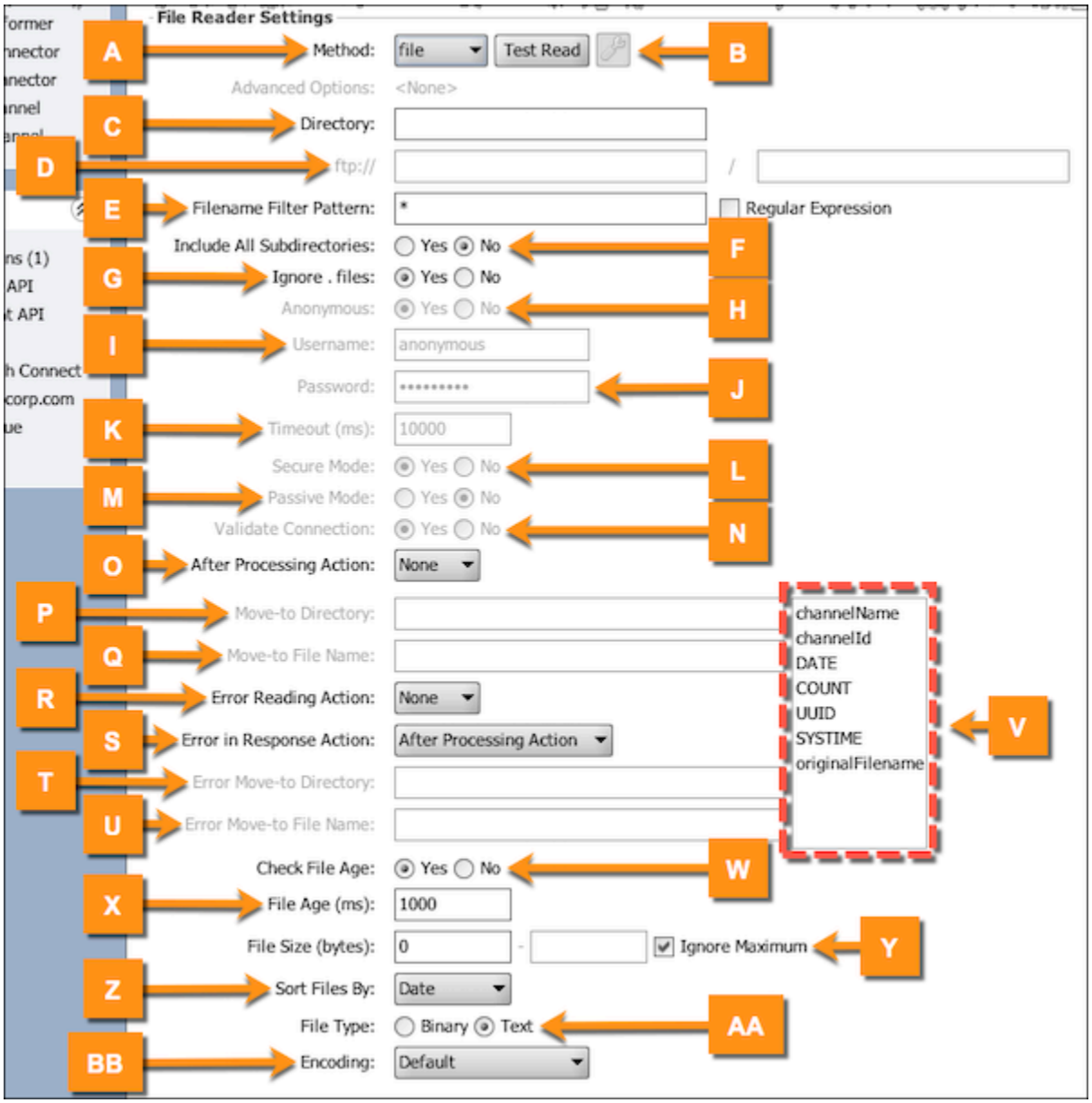
Column	Required	Description
		connector using one of these driver classes will have the corresponding entry selected in the Driver drop-down menu. The driver will be updated to the primary value if you select the entry from the drop-down menu again.

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 *(on page 598)* extension.

Supported property groups:

- Polling Settings *(on page 234)*
- Source Settings *(on page 238)*



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.

Item	Name	Default Value	Description
B	Advanced Options		If the file method supports advanced options, this button will be enabled. Advanced options are summarized in the Advanced Options label. For additional information, see: • Advanced FTP Options *(on page 357)* • Advanced SFTP Options *(on page 357)* • Advanced SMB Options *(on page 359)* • Advanced Amazon S3 Options *(on page 359)*
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.

Item	Name	Default Value	Description
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

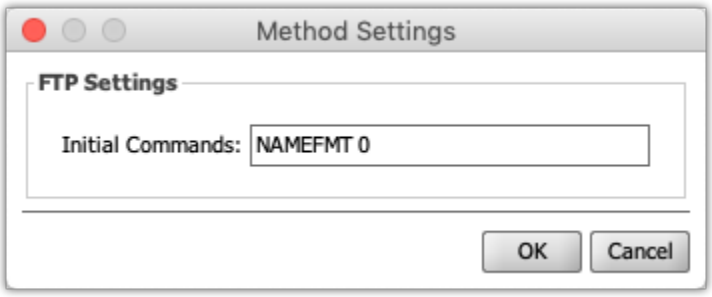
Item	Name	Default Value	Description
			this field is left empty, successfully processed files will not be renamed.
R	Error Reading Action	None	Select **Move** to move and/or rename files that have failed to be read in (for example, if an out-of-memory error occurs, or the network connection drops). Select **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 **ERROR** response is returned. This action is triggered when the Response selected in the Source Settings *(on page 238)* has a status of **ERROR**. 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 *(on page 238)*.
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 *(on page 238)* has a status of **ERROR**. 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 *(on page 238)* has a status of **ERROR**. 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		The variables listed here can be dragged-and-dropped into the Move-to fields to the left. • **channelName:** The name of the current channel.

Item	Name	Default Value	Description
			• **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.

Item	Name	Default Value	Description
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

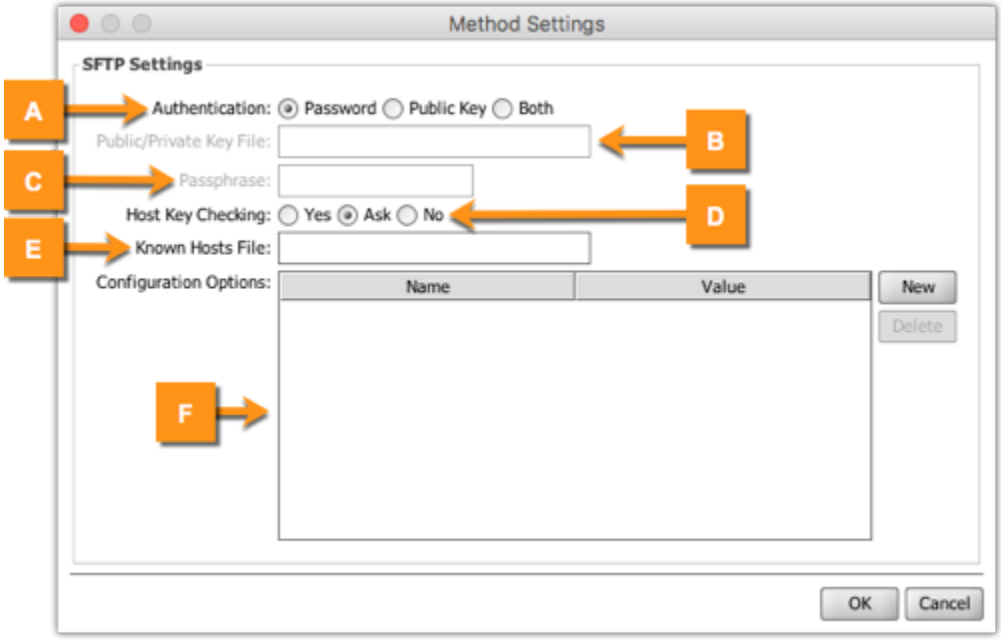
When the FTP file method is selected, these additional advanced options may be set:



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:

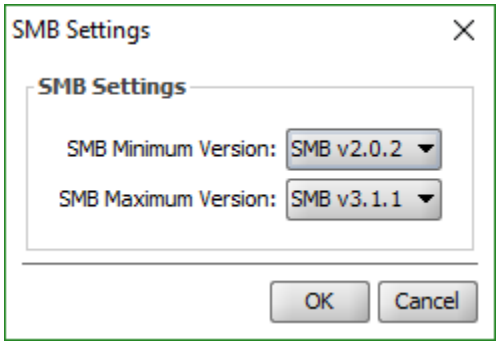


Item	Name	Default Value	Description
A	Authentication	Password	Determines how to authenticate to the SFTP server. Options include Password, Public Key, or Both.
B	Public/Private Key Files		The absolute file path of the public/private keypair used to gain access to the remote server.
C	Passphrase		The 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

Item	Name	Default Value	Description
			connecting to the remote server. For example, these can be used to enabled Kerberos authentication.

Advanced SMB Options

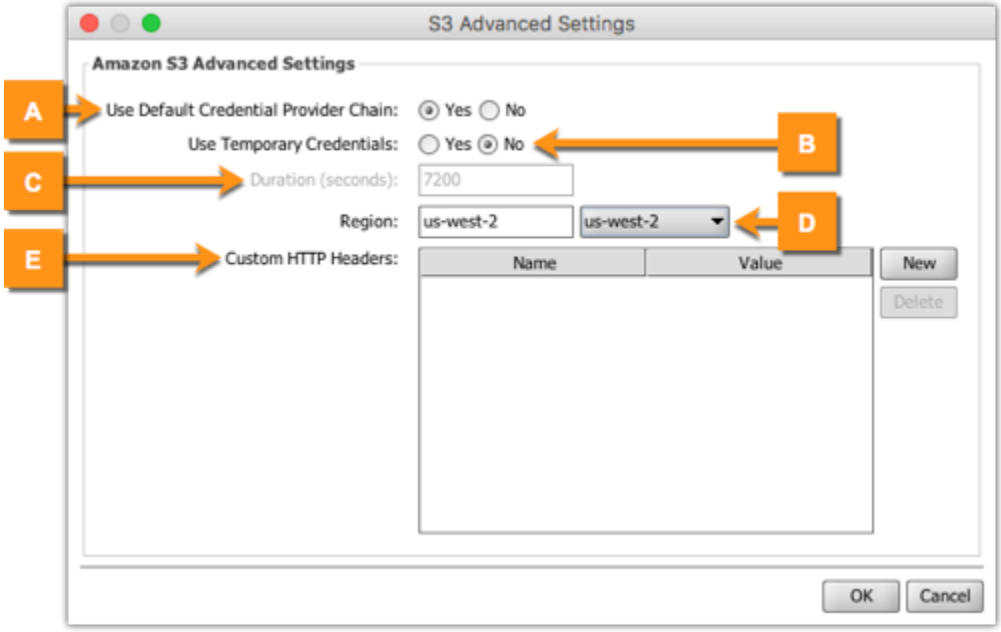
When the SMB file method is selected, these additional advanced options may be set:



Item	Default Value	Description
SMB Minimum Version	2.0.2	The minimum version of the SMB protocol to support. Note that if you are upgrading from an earlier version of Connect, this may still be set to v1. You should consider changing this to at least v2 for best security practices
SMB Maximum Version	3.1.1	The maximum version of the SMB protocol to support.

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	If enabled and no explicit credentials are provided, the default provider chain looks for credentials in this order: • **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 ~/.aws/credentials (location can very per platform) • **ECS container credentials:** Loaded from an Amazon ECS environment variable. • **Instance profile credentials:** Loaded from the EC2 metadata service. Note that if your File Reader has Anonymous enabled, this option will not be enabled.
B	Use Temporary	No	If enabled, the given credentials will be used

Item	Name	Default Value	Description
	Credentials		to request a set of **temporary** 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 *(on page 469)* here.
E	Custom HTTP Headers	pollId	These headers will be used on any S3 PUT operation. They are not used for GET operations.To add user-defined metadata tags to the S3 object, include a custom header that starts with "x-amz-meta-".For more information, check out the official Amazon S3 documentation.

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 **last** file in a poll. Note that if you have Batch Processing *(on page 337)* enabled, you will want to look at both

Key	Description
	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 are on the last file in the polling window, and also on the last message in that file.

Variables only applicable to the Amazon S3 mode

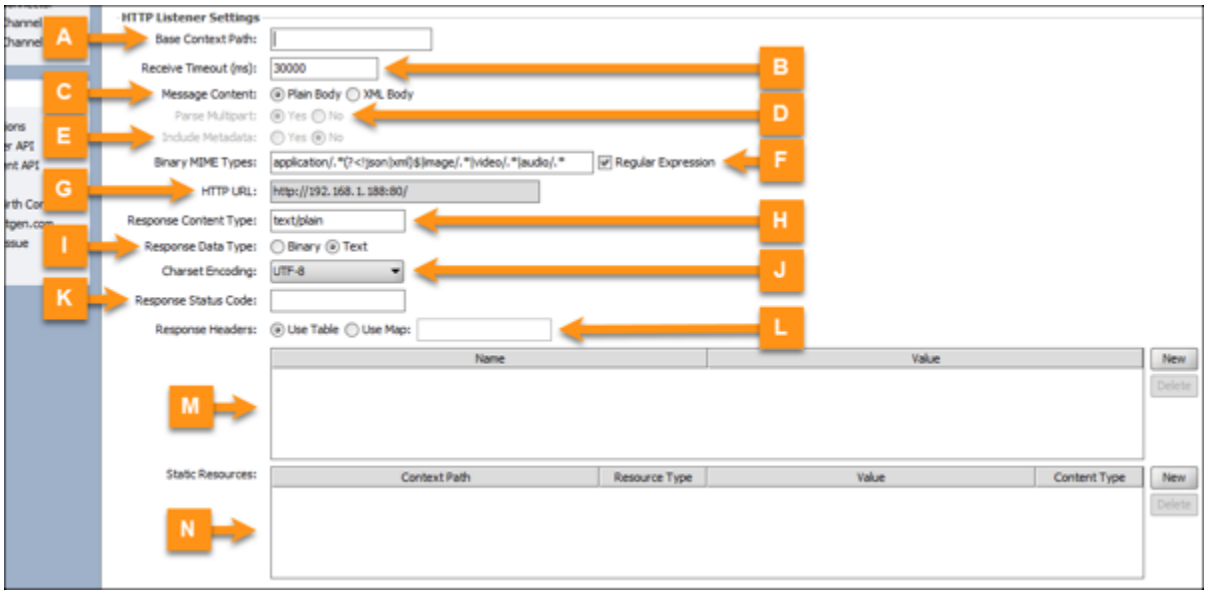
Key	Description
s3BucketName	The name of the S3 bucket that the file belongs to.
s3ETag	The hex encoded 128-bit MD5 hash of the file contents as computed by Amazon S3.
s3Key	The key under which the file is stored in S3.
s3Owner	The owner of the S3 object, if present.
s3StorageClass	The storage class 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 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 *(on page 598)* extension.

Supported property groups:

- Listener Settings *(on page 233)*
- Source Settings *(on page 238)*
- HTTP Authentication Settings *(on page 239)*



Item	Name	Default Value	Description
A	Base Context Path		The context path for the HTTP Listener URL. **Note:** 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	• **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	Applies only to Message Content XML Body. • Select **Yes** to automatically parse multipart requests into separate XML nodes.

Item	Name	Default Value	Description
			• Select **No** to always keep the request body as a single XML node.
E	Include Metadata	No	Applies only to Message Content XML Body.Select **Yes** to include request metadata (method, context path, headers, query parameters) in the XML content. **Note:** 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. • If **Regular Expression** is checked,, 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:** The host in the URL is 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.

Item	Name	Default Value	Description
I	Response Data Type	Text	• If **Binary** is selected, responses is decoded from Base64 into raw byte streams. • If **Text** is selected, responses is 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 Mirth® Connect by NextGen Healthcare 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 is returned for a successful message, and 500 is returned for an errored message. • If a Response is chosen in the Source Settings *(on page 238)*, the status of that response is used to determine a successful or errored response.
L	Response Headers Map Variable	Use Table	• **Use Table:** The table below will be used to populate response headers. • **Use Map:** The Java map specified by the following variable will be used to populate response headers. The map must have String keys and either String or List<String> values.
M	Response Headers		When using the **Use Table** option above, enter custom headers to send

Item	Name	Default Value	Description
			back to the originating client
N	Static Resources		Values in this table are automatically sent back to any request with the matching context path. There are three resource types: • **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.

JMS Listener

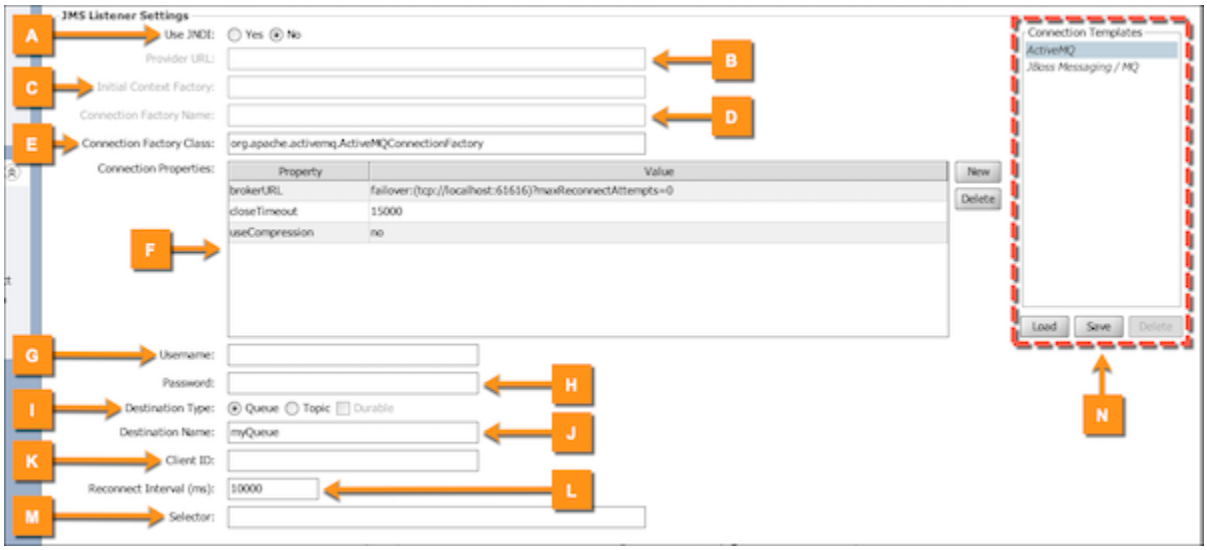
This source connector connects to an external JMS provider and reads messages from a queue or topic. It supports both JNDI and specifying a specific connection factory, as well as fine-tuned queries through a configurable selector. Once this connector is started, it will attempt to keep a persistent 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.

Note: Depending on the JMS connection provider you are using, you may need to include some external libraries (JARs) as a Library Resource *(on page 186)* and include them using the Set Dependencies *(on page 214)* dialog on the Channel Summary tab.

For example, to use connection templates included for ActiveMQ and JBoss, you should download the libraries from their official site in order to use them with the JMS connectors.

Supported property groups:

- Source Settings *(on page 238)*



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

Item	Name	Default Value	Description
			the actual value for the setting. The specific properties used here will vary depending on what connection factory class / provider you are 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

Item	Name	Default Value	Description
			other JMS Listener connectors. For additional information, see JMS Connection Templates *(on page 369)*.

JMS Connection Templates

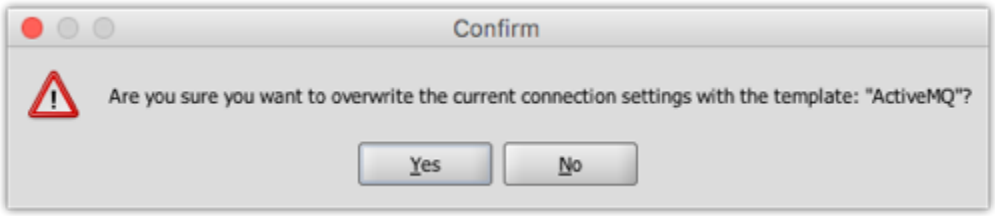
This section allows you to save the current state of your JMS Listener properties into a **template**, which may then be restored later if you make changes, or may also be applied to other JMS Listener connectors. Custom templates can be updated and deleted.

Note: Updating or deleting an existing template does not affect any connectors currently using that template.

Note: 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.

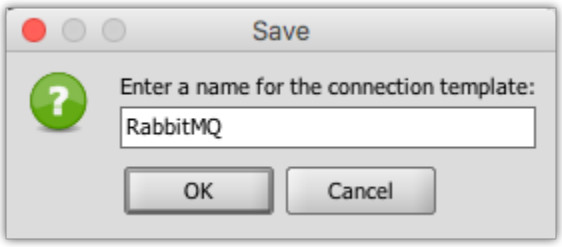
Loading Templates

Select a template in the Connection Templates list, then select the **Load** button. You will be prompted to overwrite your current JMS Listener settings:

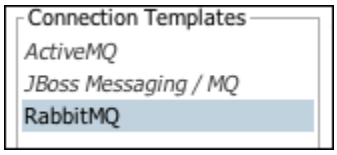


Creating New Templates

Configure your JMS Listener settings to the state you want to save, then select the **Save** button in the Connection Templates section. You will be prompted to give the template a name:

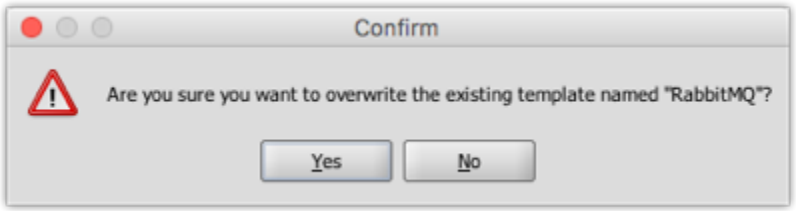


After selecting **OK**, the new template will appear in the Connection Templates list:



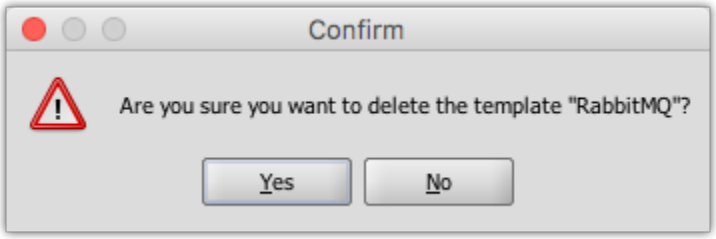
Updating Templates

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

Select a template in the Connection Templates list, then select the **Delete** button to delete a template. You will be prompted to confirm the action:

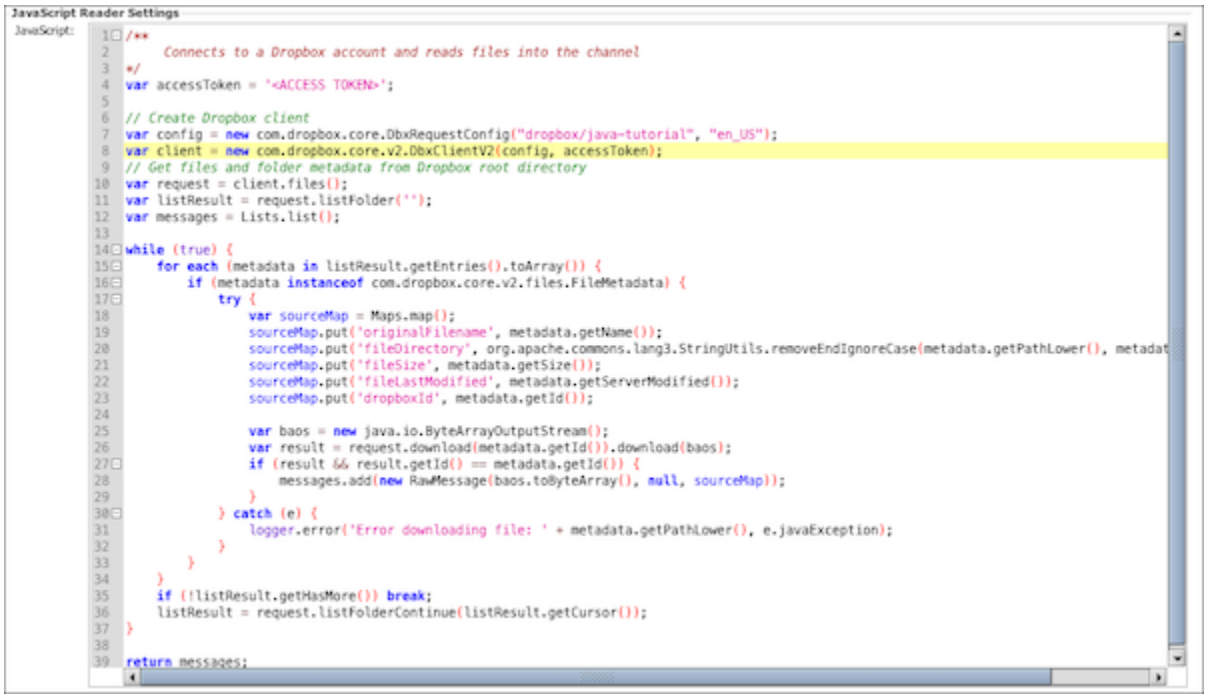


JavaScript Reader

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 *(on page 459)* to programmatically start/stop/deploy channels from within the script.

Supported property groups:

- Polling Settings *(on page 234)*
- Source Settings *(on page 238)*



JavaScript Reader Return Values

If you are 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.

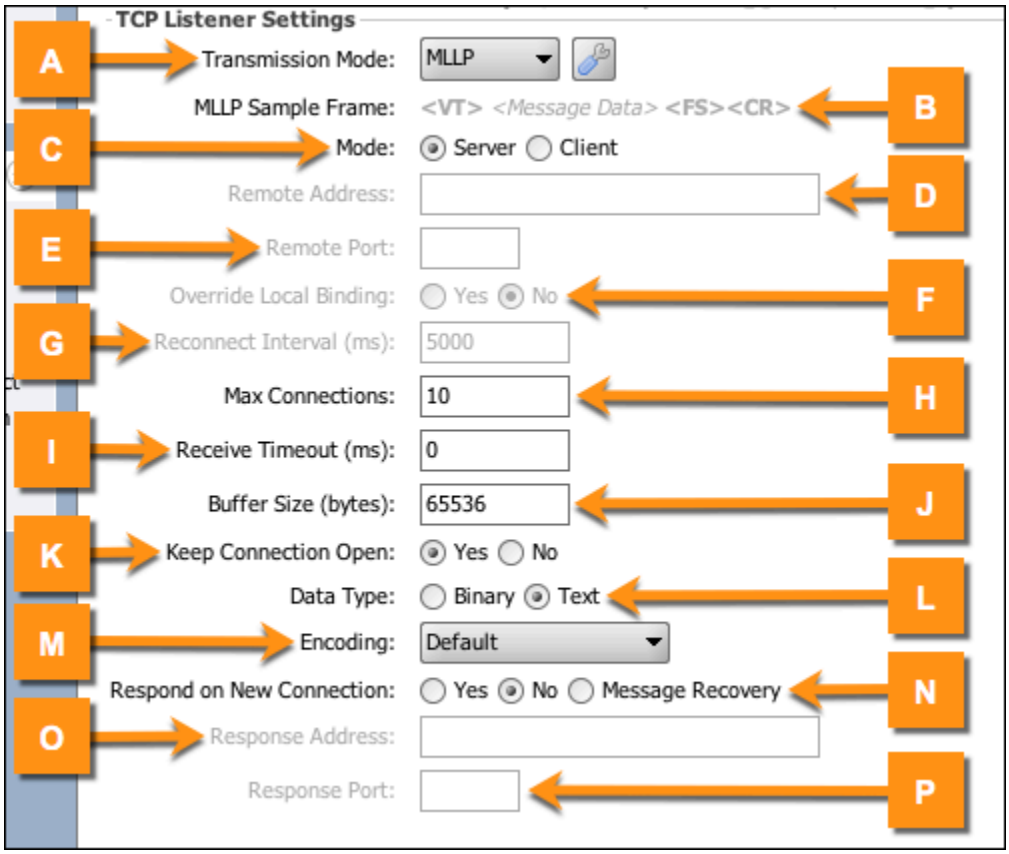
- **RawMessage:** This is a special object that contains not only the string message data, but also information about which destinations to dispatch to, and any source map variables you wish to inject. For additional information, see The User API (Javadoc) *(on page 459)*.
- **List:** If a Java List 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 *(on page 376)* 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 *(on page 233)*
- Source Settings *(on page 238)*



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 *(on page 376)*.
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. • Select **Client** to connect to a TCP Server. In Client mode, the Listener Settings *(on page 233)* is only used if **Override Local Binding** is enabled.
D	Remote Address		The domain name or IP address on which

Item	Name	Default Value	Description
			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	• 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. Only applicable for Client mode. **Note:** 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.
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 **Yes** to keep the socket open until the sending system

Item	Name	Default Value	Description
			closes it. When Yes is selected, the 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. • Select **No** to close the socket after a received message has finished processing.
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 Mirth® Connect by NextGen Healthcare 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

Item	Name	Default Value	Description
			recovery. Connections will be bound locally on the same interface chosen in the Listener Settings *(on page 233)* 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 *(on page 372)* / TCP Sender *(on page 416)* (and Serial Listener / Sender commercial extension):

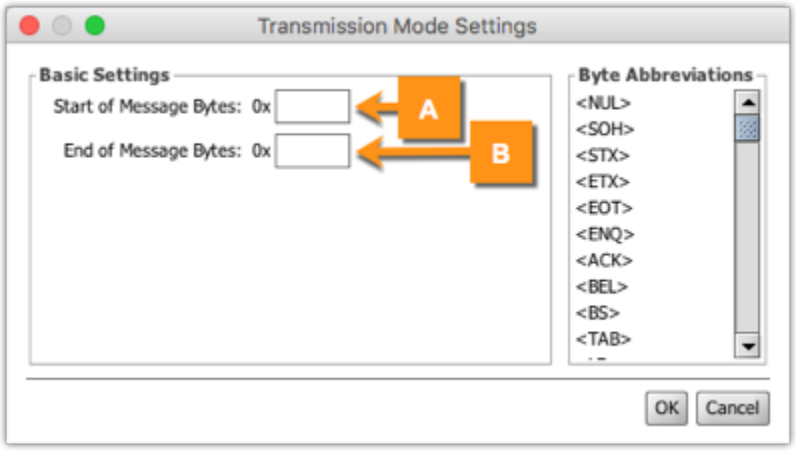
- Basic TCP Transmission Mode *(on page 377)*
- MLLP Transmission Mode *(on page 378)*

An additional transmission mode is made available via a commercial extension:

- ASTM E1381 Transmission Mode *(on page 585)*

Basic TCP Transmission Mode

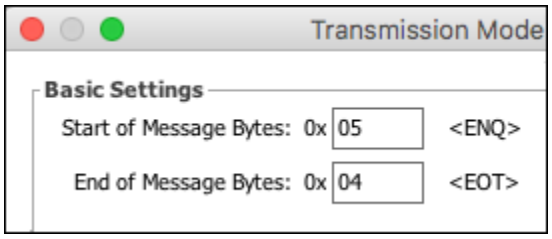
This transmission mode allows you to specify basic TCP frame data (beginning and ending byte sequences). This allows the source connector to know when a message has been fully received. Destination connectors also use these sequences when sending data outbound.



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.

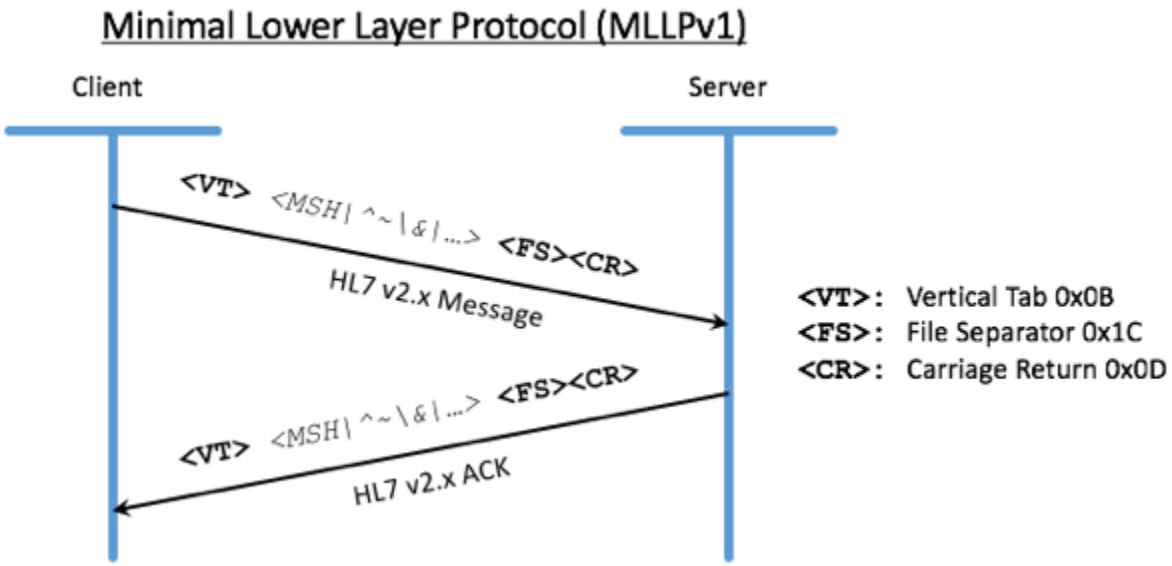
Byte Abbreviations

The **s** 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:

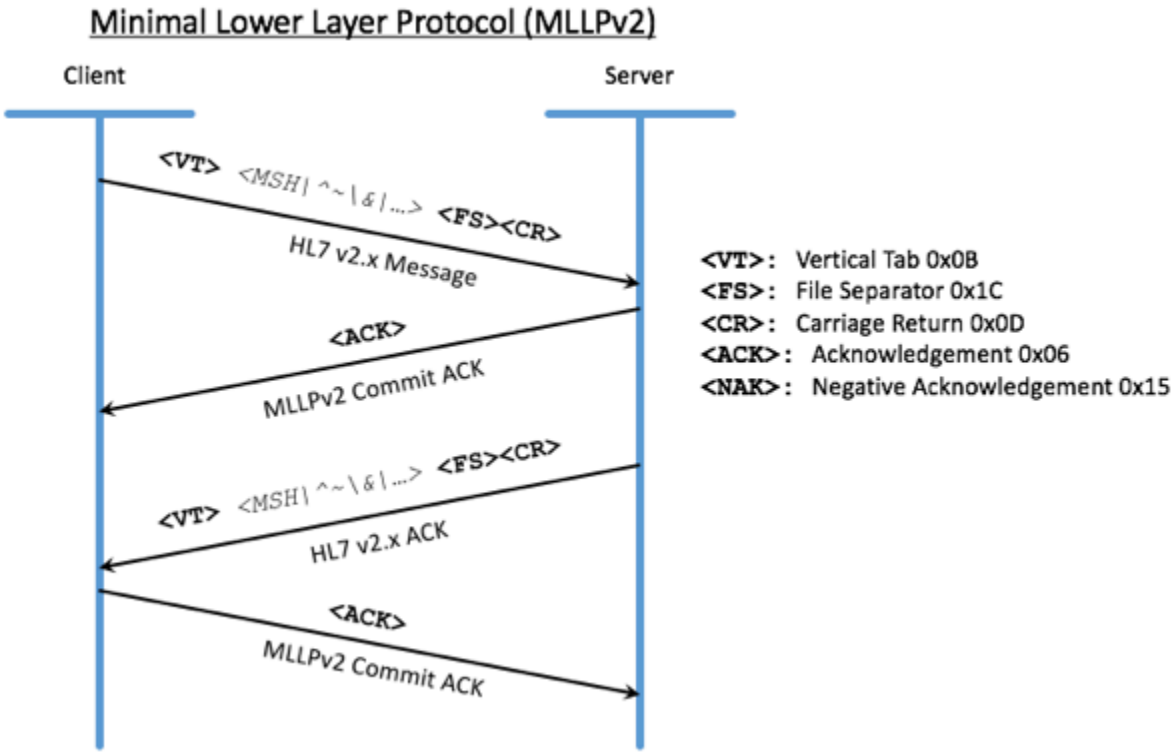


MLLP Transmission Mode

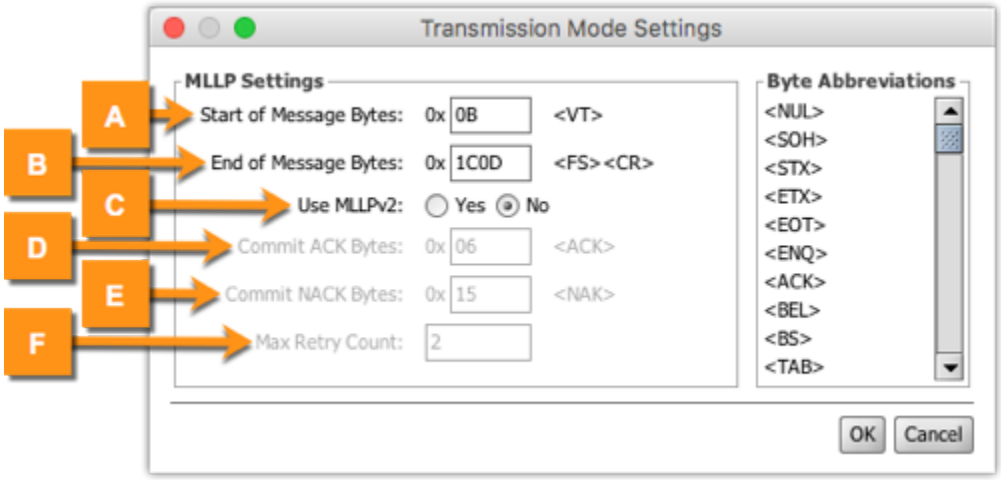
This transmission mode implements the Minimal Lower Layer Protocol (MLLP) specified by HL7, and is often used when transmitting HL7 v2.x *(on page 329)* messages. There are two versions of MLLP, v1 and v2. The first version is similar to the Basic TCP Transmission Mode *(on page 377)* in that it only specifies sequences for the start/end message bytes.



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.



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

Item	Name	Default Value	Description
			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

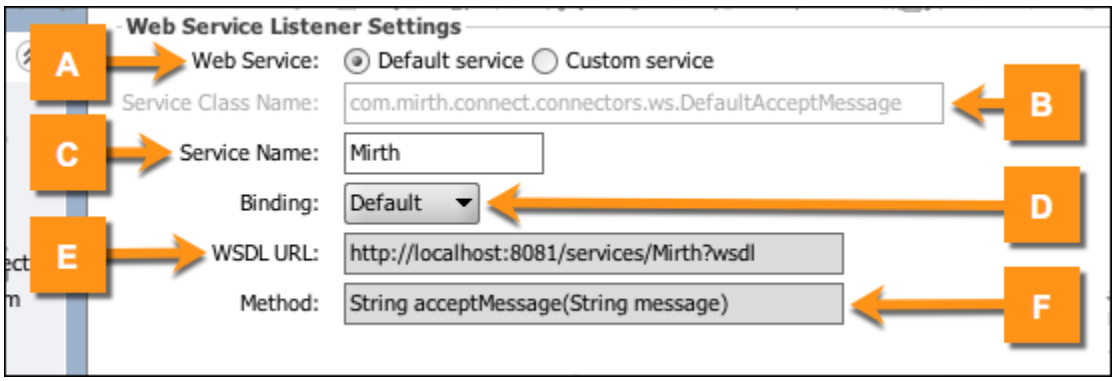
This section is the same as in the Basic TCP Transmission Mode *(on page 377)*.

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 *(on page 233)*
- Source Settings *(on page 238)*
- HTTP Authentication Settings *(on page 239)*



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.

Item	Name	Default Value	Description
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 dur 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

Item	Name	Default Value	Description
			"<Custom Web Service Methods>".

CHAPTER 9

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 *(on page 60)*.

Here is a list of destination connectors supported by Mirth® Connect

- Channel Writer *(on page 384)*
- DICOM Sender *(on page 386)*
- Database Writer *(on page 390)*
- Document Writer *(on page 393)*
- File Writer *(on page 395)*
- HTTP Sender *(on page 400)*
- JMS Sender *(on page 407)*
- JavaScript Writer *(on page 410)*
- SMTP Sender *(on page 411)*
- TCP Sender *(on page 416)*
- Web Service Sender *(on page 422)*

Additional destination connectors are made available as commercial extensions *(on page 582)*:

- Serial Connector *(on page 597)*
- NextGen Results CDR Connector *(on page 600)*

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 *(on page 340)* 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 *(on page 249)*

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 *(on page 469)*. 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		The map variables entered here will be included in the source map of the destination channel's message. **Note:** This table expects only variable names, not Velocity replacement tokens. For example, do not use ${varName}, instead use

Item	Name	Default Value	Description
			varName. The value will be extracted from all available Variable Maps *(on page 449)*.
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 *(on page 469)* is supported here.

Source Map Variables

When this connector sends a message to another channel, the following source map *(on page 449)* variables will be available on the downstream message:

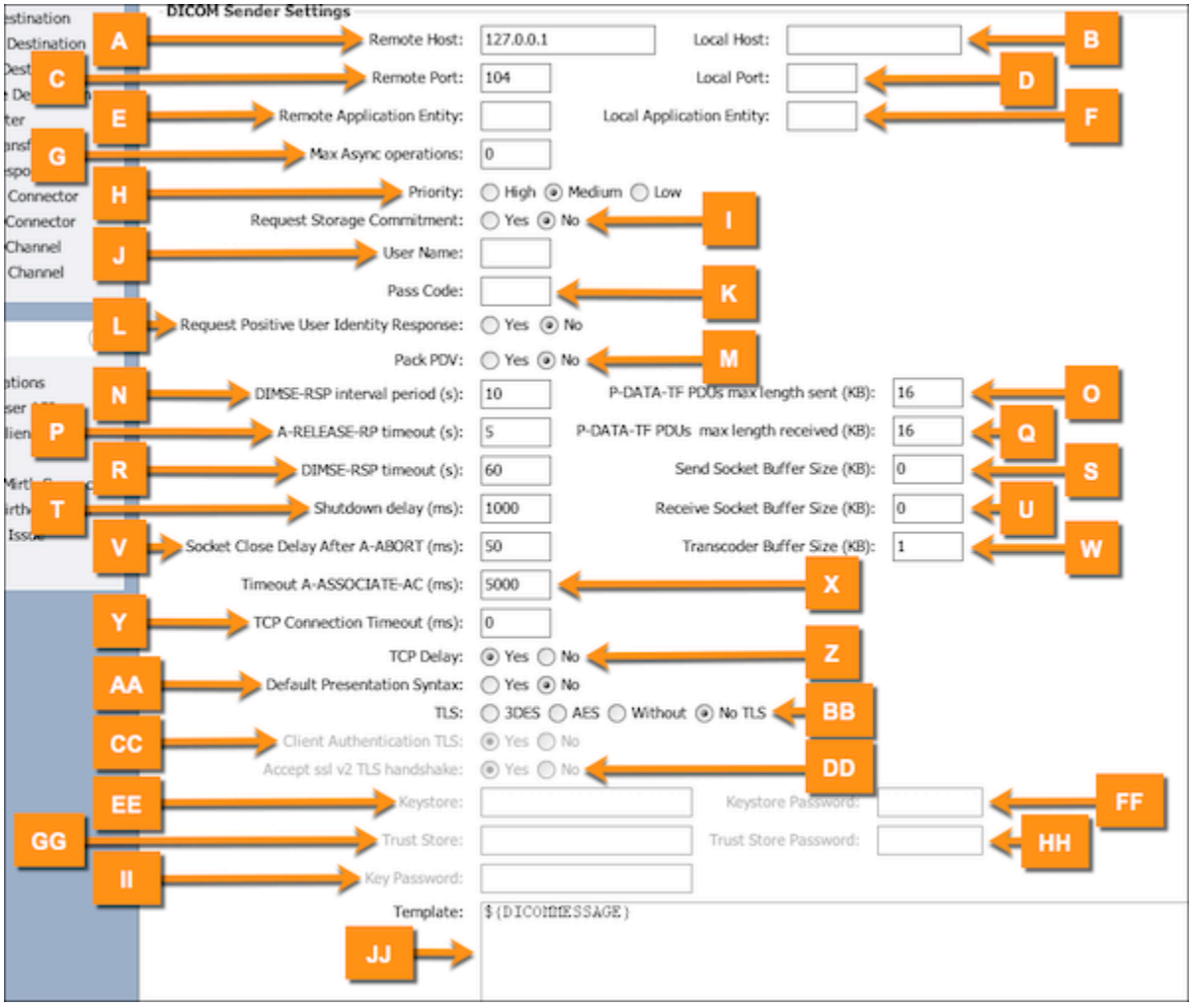
Key	Description
sourceChannelId	The unique ID of the channel that dispatched a message to the current channel.
sourceMessageId	The ID of the message from which the current message dispatch originated.
sourceChannelIds	If there are more than two channels in a Channel Writer -> Reader chain, this will 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 Properties *(on page 224)* and the DICOM Data Type *(on page 327)* to allow Mirth® Connect 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 *(on page 598)* extension.

Supported property groups:

- Destination Settings *(on page 249)*



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.

Item	Name	Default Value	Description
	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 committment 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	0	Send socket buffer size in KB

Item	Name	Default Value	Description
	Buffer Size (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:
			• **3DES:** TLS will be used, with the **SSL_RSA_WITH_3DES_EDE_CBC_SHA** cipher suite.
			• **AES:** TLS will be used, with the following cipher suites:
			  • **TLS_RSA_WITH_AES_128_CBC_SHA**
			  • **SSL_RSA_WITH_3DES_EDE_CBC_SHA**
			• **Without:** TLS will be used without symmetric encryption, with the cipher suite **SSL_RSA_WITH_NULL_SHA**. DICOM messages will be received unencrypted.

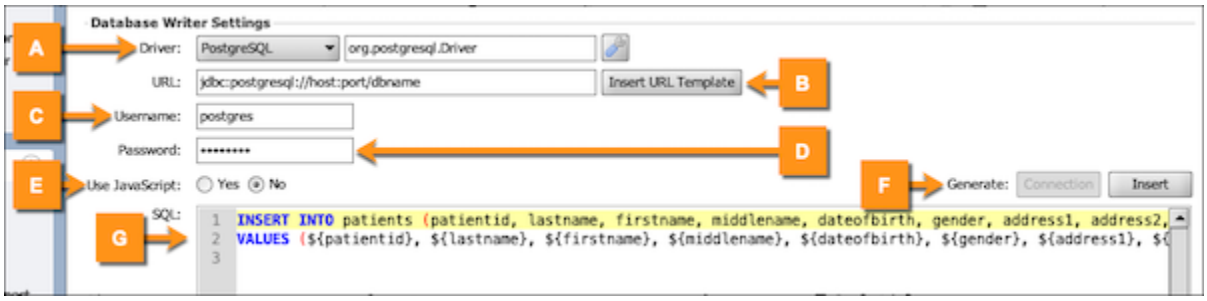
Item	Name	Default Value	Description
			• **No TLS:** No TLS will be used. DICOM messages will be received over a regular unencrypted socket.
CC	Client Authentication TLS	Yes	Enable client authentication for TLS. Only applicable if the TLS option is not set to **No TLS**.
DD	Accept ssl v2 TLS handshake	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 *(on page 469)* 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 *(on page 249)*



Item	Name	Default Value	Description
A	Driver		Specifies the type of database driver to use to connect to the database. The following values are supported by default: • Sun JDBC-ODBC Bridge • MySQL • Oracle • PostgreSQL • SQL Server / Sybase • SQLite Additional drivers can be added by editing the dbdrivers.xml File *(on page 480)*.
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

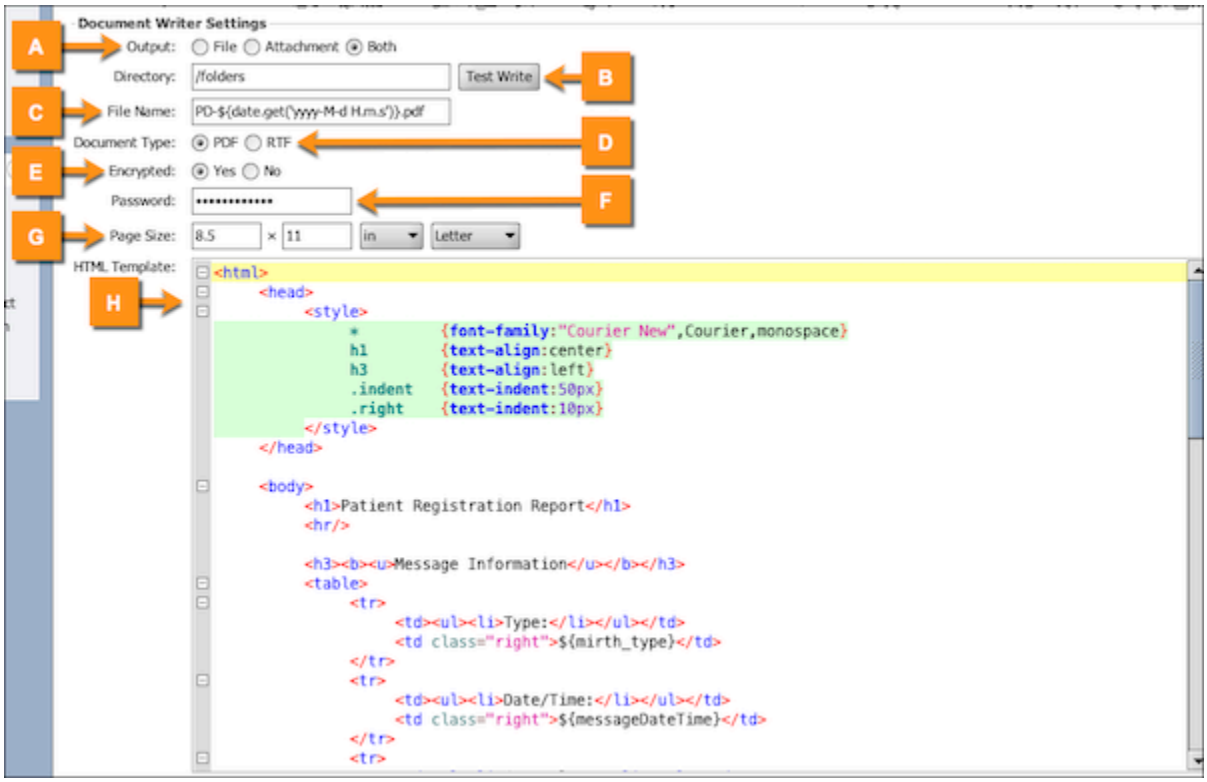
Item	Name	Default Value	Description
			JavaScript" is checked. However, it is used when the Generate Connection / Insert feature is used to generate code.
D	Password		The password to connect to the database with. This is not used when "Use JavaScript" is checked. However, it is used when the Generate 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		• **Connection**: This button is enabled when **Use JavaScript** is enabled. When selected, 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 *(on page 249)*



Item	Name	Default Value	Description
A	Output	File	Choose how to output the document. • **File:** Write the contents to a file.

Item	Name	Default Value	Description
			• **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

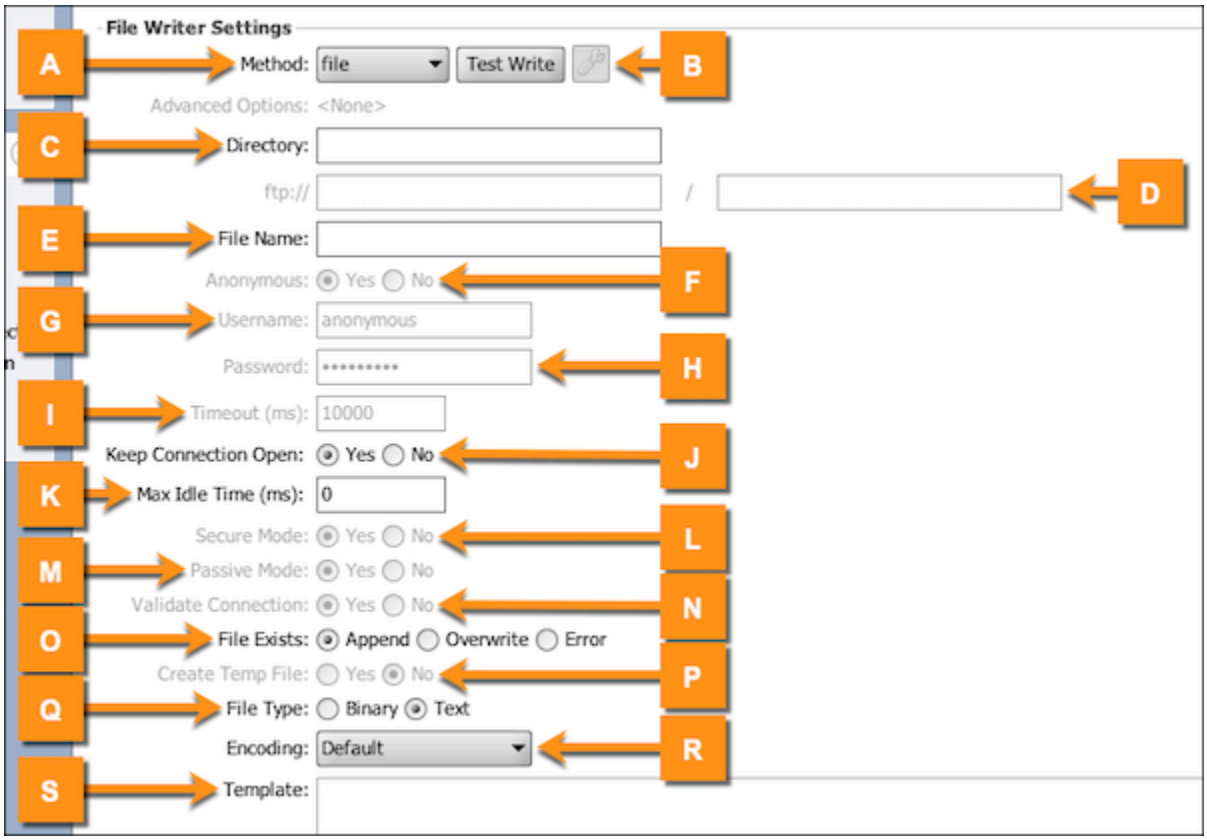
Item	Name	Default Value	Description
			formats.
H	HTML Template		This template is expected to be an HTML document, determining how to layout the PDF/RTF document. Custom embedded stylesheets are supported.

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 *(on page 598)* extension.

Supported property groups:

- Destination Settings *(on page 249)*



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

Item	Name	Default Value	Description
			information, see File Reader *(on page 351)*.
C	Directory		Only applicable to the File method. The directory (folder) to write the files to.
D	URL		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. 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	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	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.
H	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.
I	Timeout (ms)	10000	Applicable to the **FTP** / **SFTP** / **SMB** / **Amazon S3** methods. The socket

Item	Name	Default Value	Description
			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	Determines what to do when the file to be written already exists on the filesystem / remote server. • **Append:** Messages will be appended to the

Item	Name	Default Value	Description
			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.

Item	Name	Default Value	Description
			Velocity Variable Replacement *(on page 469)* is supported here.

Connector Map Variables

When the Amazon S3 mode is used, these variables will be available in the connector map.

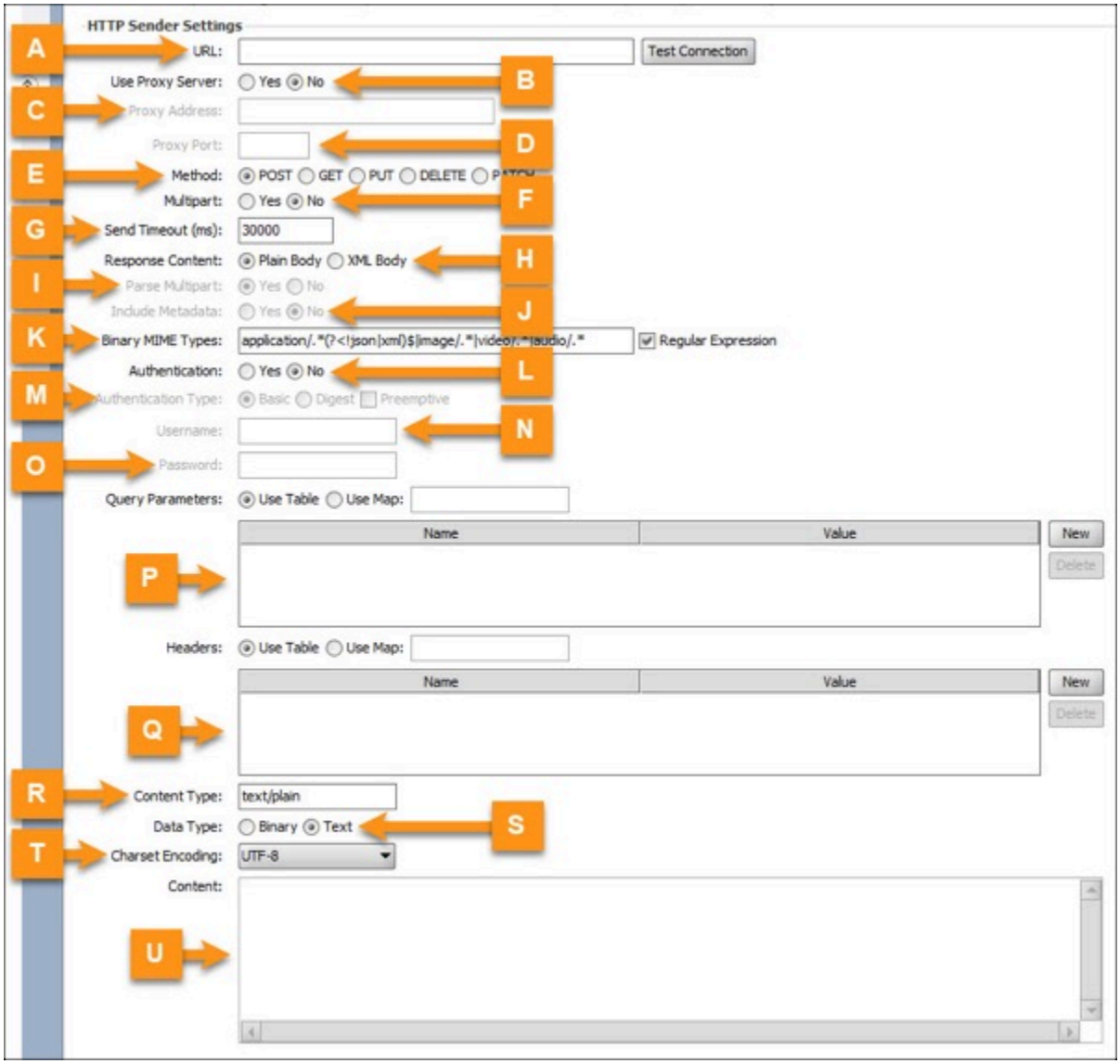
Key	Description
s3ETag	The hex encoded 128-bit MD5 hash of the file contents as computed by Amazon S3.
s3ExpirationTime	The expiration time for this object, only present if not null.
s3ExpirationTimeRuleId	The BucketLifecycleConfiguration rule ID for this object's expiration, only present if not null.
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 *(on page 598)* extension.

Supported property groups:

- Destination Settings *(on page 249)*



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 are forwarded to the proxy

Item	Name	Default Value	Description
			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 is first written to a local temp file and is then 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	• **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	Applies only to Response Content XML Body. • Select **Yes** to automatically parse multipart responses into separate XML nodes.

Item	Name	Default Value	Description
			• Select **No** to always keep the response body as a single XML node.
J	Include Metadata	No	Applies only to Response Content XML Body. • Select **Yes** to include response metadata (response code, headers) in the XML content. **Note:** 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 is encoded into a Base64 string. • If **Regular Expression** is unchecked, specify multiple entries with commas. • If **Regular Expression** is checked, enter a valid regular expression to match MIME types against.
L	Authentication	No	If enabled, a **Basic** or **Digest** Authorization

Item	Name	Default Value	Description
			header is automatically added to the request.
M	Authentication Type	Basic	Select either **Basic** or **Digest** authentication type. • If the **Preemptive** option is checked, the Authorization header is sent to the server with the initial request. • If the **Preemptive** option is unchecked, the Authorization header is only sent when the server requests it. **Note:** When **Digest** authentication is selected, 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	Use Table	• **Use Table:** The table below will be used to populate query parameters. • **Use Map:** The Java map specified by the following variable will be used to

Item	Name	Default Value	Description
			populate query parameters. The map must have String keys and either String or List<String> values. When using the **Use Table** option above, entries in this table will automatically be added to the request URI as query parameters. Multiple parameters with the same name are supported. **Tip:** 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	Use Table	• **Use Table:** The table below will be used to populate headers. • **Use Map:** The Java map specified by the following variable will be used to populate headers. The map must have String keys and either String or List<String> values. When using the **Use Table** option above, entries in this table will be added to the request as HTTP headers. Multiple headers with the

Item	Name	Default Value	Description
			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. This is decoded before it is sent out. • Select **Text** if the outbound message is textual. This is 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 Mirth® Connect 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-

Item	Name	Default Value	Description
			enclosing requests (POST / PUT / PATCH). Velocity Variable Replacement *(on page 469)* 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 *(on page 286)*.

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

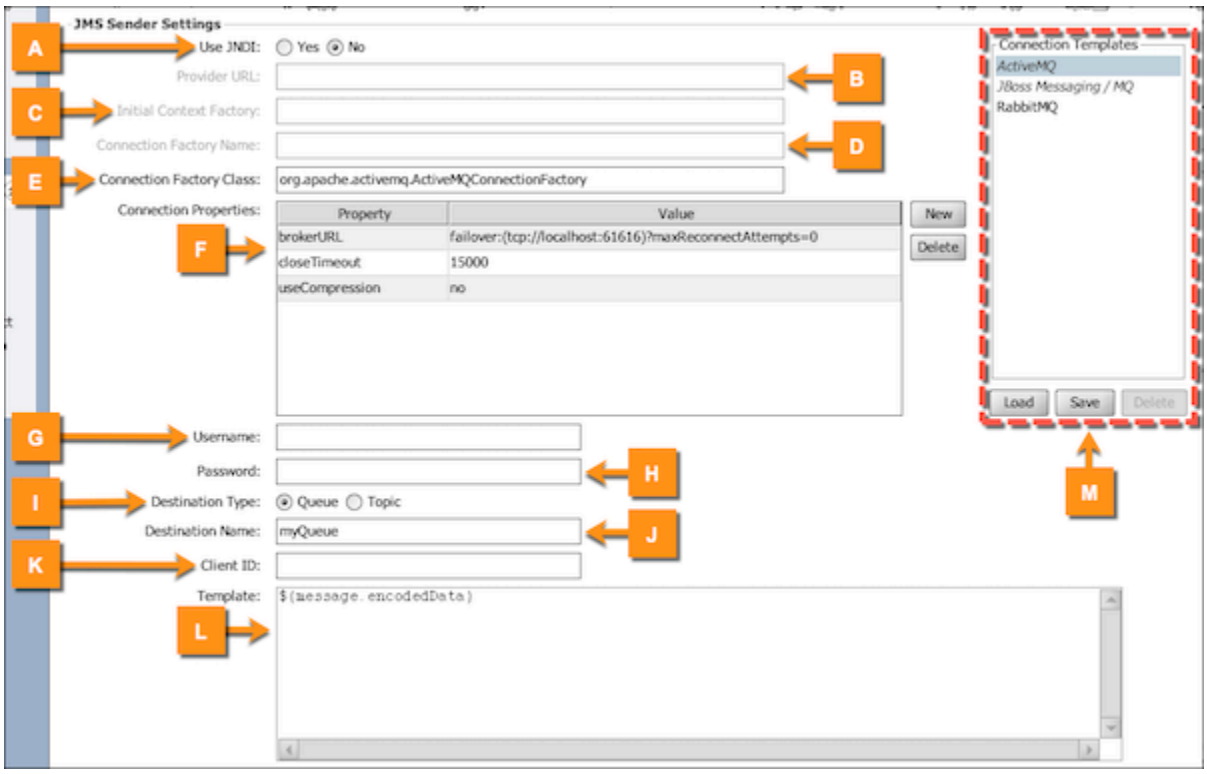
This destination connector connects to an external JMS provider and writes messages to a queue or topic. It supports both JNDI and specifying a specific connection factory. Once this connector dispatches a message, the connection to the JMS provider will be kept open and cached until the connector is 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.

Note: Depending on the JMS connection provider you are using, you may need to include some external libraries (JARs) as a Library Resource *(on page 186)* and include them using the Set Dependencies *(on page 214)* dialog on the Channel Summary tab.

For example, to use connection templates included for ActiveMQ and JBoss, you should download the libraries from their official site in order to use them with the JMS connectors.

Supported property groups:

- Destination Settings *(on page 249)*



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

Item	Name	Default Value	Description
			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 are 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 *(on page 469)* 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

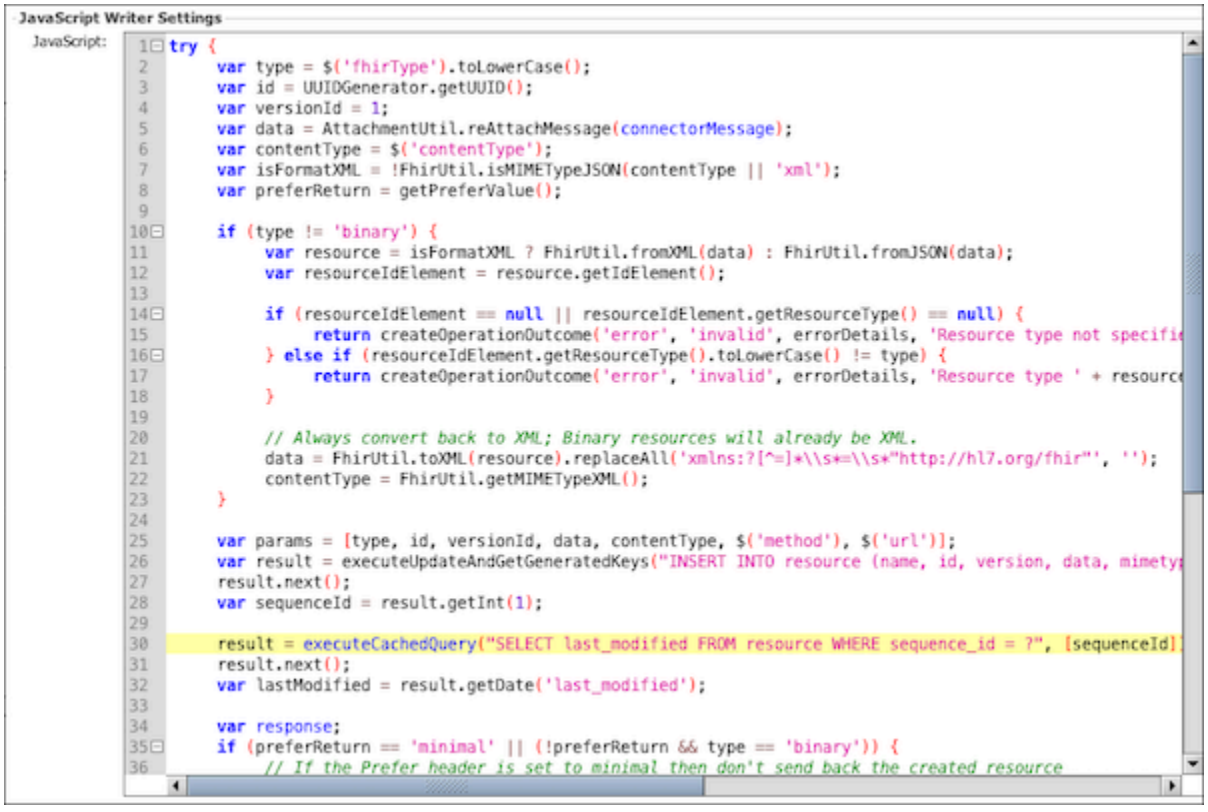
Item	Name	Default Value	Description
			information here: JMS Listener *(on page 366)*

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 *(on page 249)*



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:

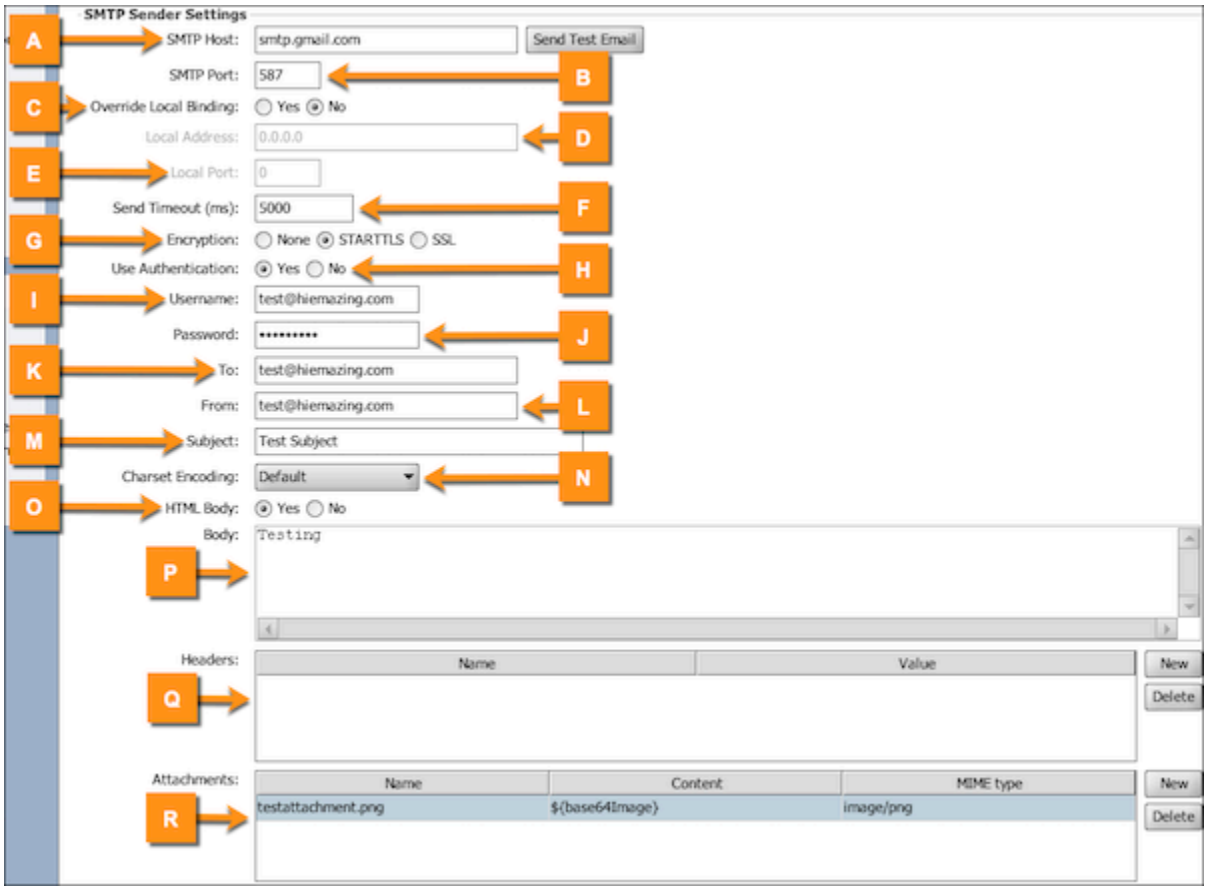
- **String:** Any string returned will be stored as the Response content for the destination, with a status of **SENT**.
- **Status:** An instance of the Status enum (for additional information, see The User API (Javadoc) *(on page 459)*) will cause no response content to be stored, but the connector 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 *(on page 249)*.
- **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) *(on page 459)*.
- **Empty String / null / undefined:** Returning any of these (including 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 *(on page 249)*



Item	Name	Default Value	Description
A	SMTP Host		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 Mirth® Connect being added to the server's "denylist".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.
B	SMTP Port	25	The port number of the SMTP server to send the e-

Item	Name	Default Value	Description
			mail messages to. Generally, the default port of 25 is used.
C	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 is 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 is 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. • **None:** No encryption will be

Item	Name	Default Value	Description
			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

Item	Name	Default Value	Description
			to use when converting the body, or Default to use the default character set encoding of the JVM Mirth® Connect by NextGen Healthcare 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 *(on page 469)* is supported here.
Q	Headers		• **Use Table:** The table below will be used to populate headers. • **Use Map:** The Java map specified by the following variable will be used to populate headers. The map must have String keys and either String or List<String> values. When using the **Use Table** option above, entries in this table will be included as SMTP headers in the e-mail dispatch.
R	Attachments		• **Use Table:** The table below will be used to populate attachments. • **Use List:** The

Item	Name	Default Value	Description
			Java list specified by the following variable will be used to populate attachments. The List must have AttachmentEntry values. The AttachmentEntry class is available in JavaScript scripts and is documented in The User API (Javadoc) *(on page 459)*. When using the **Use Table** option above, entries in this table will be added as attachments with the e-mail. The following columns are configurable: • **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").

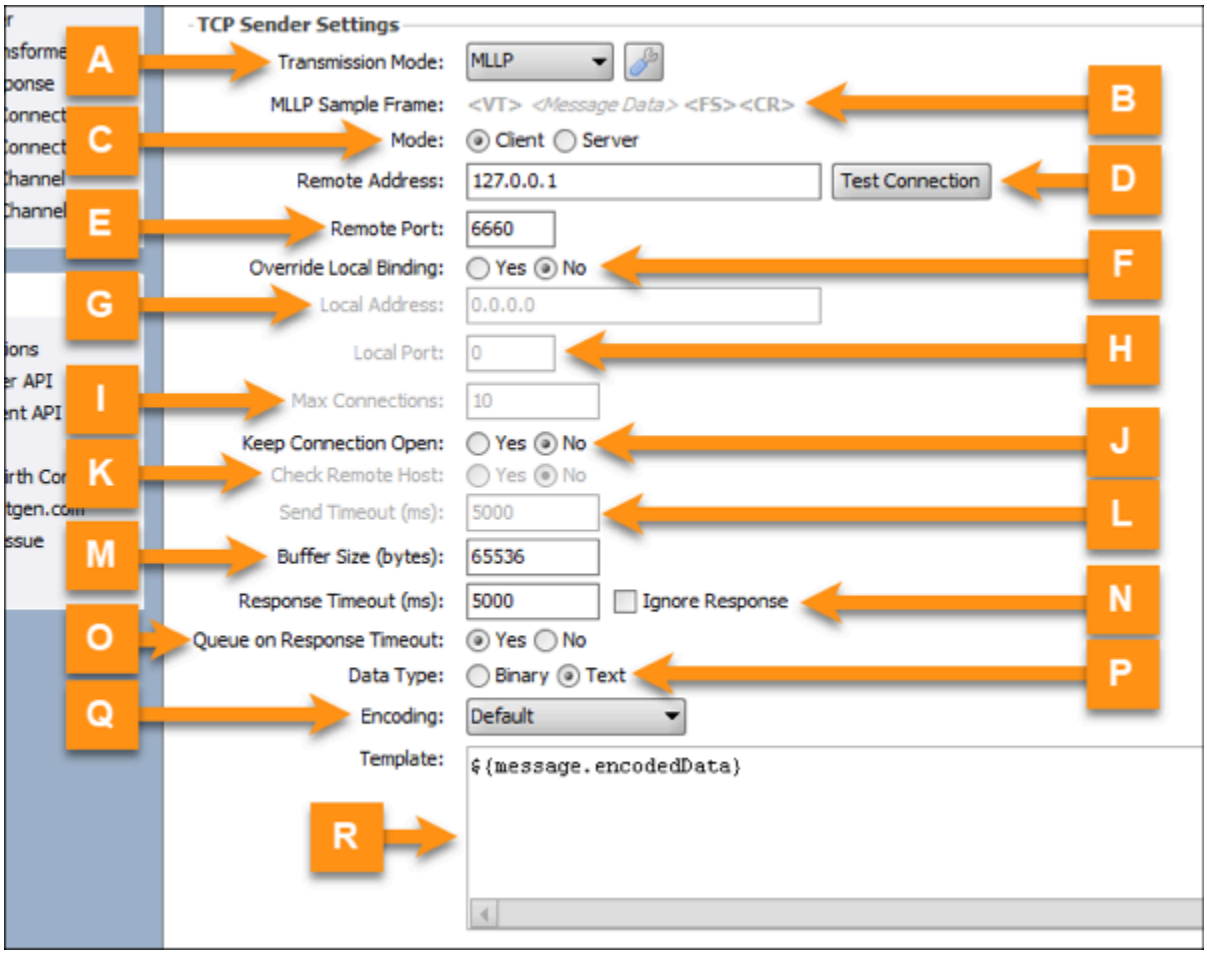
TCP Sender

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 *(on*

page 372) allow you to decide how to send outbound messages and receive responses.

Supported property groups:

- Destination Settings *(on page 249)*



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 *(on page 372)*
B	Sample Frame	**<VT>** <*Message Data*>	This is dependent on the

Item	Name	Default Value	Description
		<FS><CR>	Transmission Mode and displays an example of how an outgoing message frame is expected to look.
C	Mode	Client	• **Client:** The TCP Sender will initiate new connections outbound to a remote server, and then send messages outbound on that connection. • **Server:** The TCP Sender will open a server socket and listen for incoming connections. If queuing is enabled, messages will continue to queue until **at least one** connection has been established, at which point the messages will be sent to all currently connected client connections.
D	Remote Address	127.0.0.1	The domain name or IP address on which to connect. Press the **Test Connection** button to verify whether the server is able to open a TCP connection as the specified IP/port. Only applicable when **Client** mode is used above.
E	Remote Port	6660	The port on which to connect. Only applicable when **Client** mode is used above.

Item	Name	Default Value	Description
F	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 is up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail. Only applicable when **Client** mode is used above.
G	Local Address	0.0.0.0	The local address that the client socket will be bound to, if Override Local Binding is enabled. If **Server** mode is used above, this will determine what interfaces to listen on for incoming connections.
H	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 is up to the underlying OS to release the port before the next socket creation, otherwise the bind attempt will fail. If **Server** mode is used above, this will determine what port to listen on for incoming connections.
I	Max Connections	10	The maximum number of client connections to accept at once. After this number has been reached, subsequent socket requests will result in a rejection.

Item	Name	Default Value	Description
			Only applicable when the **Server** mode is used above
J	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. **Note:** 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. Only applicable when **Client** mode is used above.
K	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. Only applicable when **Client** mode is used above.
L	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

Item	Name	Default Value	Description
			connection will be kept open indefinitely. Only applicable when **Client** mode is used above.
M	Buffer Size (bytes)	65536	The size, in bytes, of the buffer to hold messages waiting to be sent. Generally, the default value is fine.
N	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.
O	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 **ERROR** when a timeout occurs. This setting has no effect unless queuing is enabled for the connector.
P	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).
Q	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 Mirth® Connect by NextGen Healthcareis running on.
R	Template	${message.encodedData}	The actual payload to send

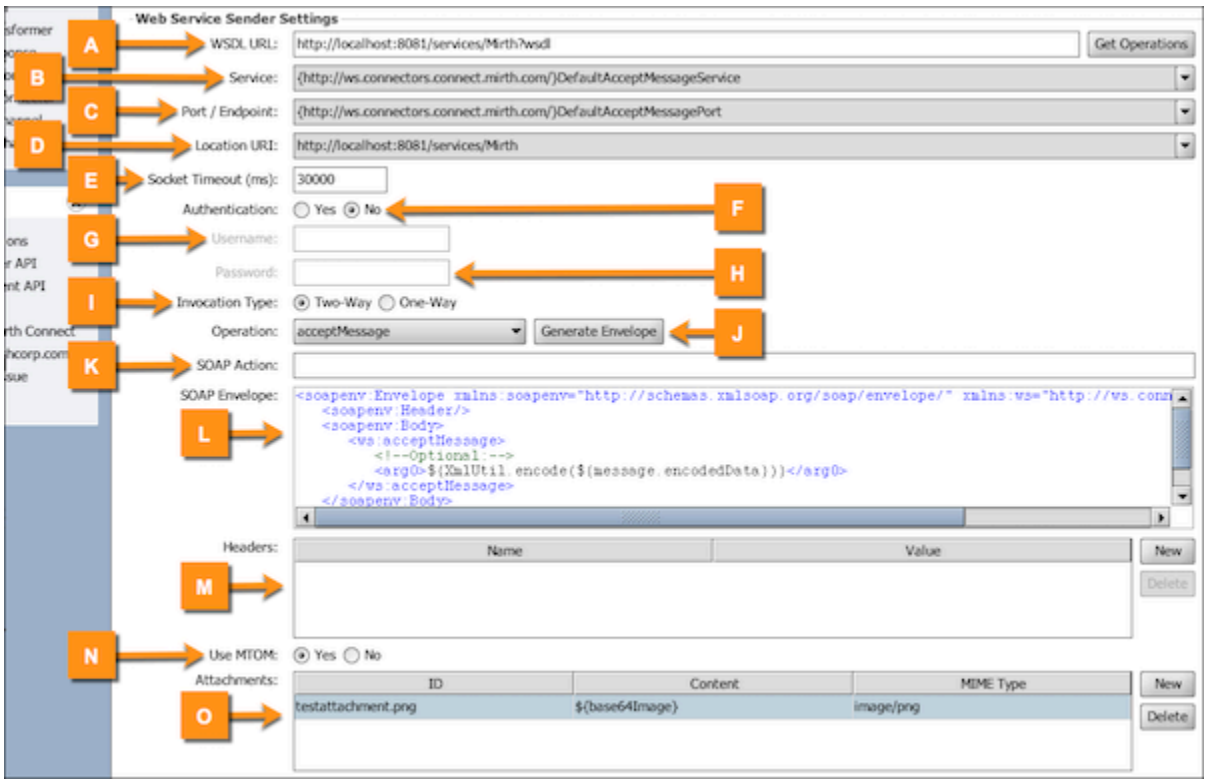
Item	Name	Default Value	Description
			to the remote server. By default the encoded data of this destination will be used. Velocity Variable Replacement *(on page 469)* is supported here.

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 *(on page 249)*



Item	Name	Default Value	Description
A	WSDL URL		The URL to the WSDL describing the web service and available operations. Select **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 selected 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 selected 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 selected 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

Item	Name	Default Value	Description
			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	Determines how to invoke the operation selected below. • **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

Item	Name	Default Value	Description
			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		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 *(on page 469)*. For more complex schemas, this button may produce an incomplete SOAP envelope. Please refer to your vendors to ensure you're sending the correct SOAP envelopes in your outgoing messages. **Note:** 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.
M	Headers		• **Use Table:** The table below will be used to populate headers. • **Use Map:** The Java map specified by the

Item	Name	Default Value	Description
			following variable will be used to populate headers. The map must have String keys and either String or List<String> values. When using the **User Table** option above, 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		• **Use Table:** The table below will be used to populate attachments. • **Use List:** The Java list specified by the following variable will be used to populate attachments. The List must have AttachmentEntry values. The AttachmentEntry class is available in JavaScript scripts and is documented in The User API (Javadoc) *(on page 459)*. When using the **Use Table** option above, entries in this table will be added as

Item	Name	Default Value	Description
			MTOM attachments along with the request. The following columns are configurable: • **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").

CHAPTER 10

Mirth Connect and JavaScript

JavaScript is a scripting language that can be used in a wide variety of places throughout Mirth® Connect to perform advanced routing and transformation.

About JavaScript

This section provides a basic explanation of how the language works:

- Variables *(on page 428)*
- Comments *(on page 429)*
- Arrays *(on page 429)*
- Operators *(on page 430)*
- Conditional Statements *(on page 432)*
- Loops and Iterations *(on page 433)*
- Exception Handling *(on page 435)*

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 Operators *(on page 430)*
- Assignment Operators *(on page 431)*
- Comparison Operators *(on page 431)*
- Logical Operators *(on page 432)*

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
+=	add and assign	x+=y	x=x+y (10+5)	x=15
-=	subtract and assign	x-=y	x=x-y (10-5)	x=5
*=	multiply and assign	x*=y	x=x*y (10*5)	x=50
/=	divide and assign	x/=y	x=x/y (10/5)	x=2
%=	modulus (division remainder) and assign	x%=y	x=x%y (remainder is 0 for the equation 10/2)	x=0

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
==	is equal to	x==8	false
		x==5	true
!=	is not equal to	x!=8	true
>	is greater than	x>8	false
<	is less than	x<8	true
>=	is greater than or equal to	x>=8	false
<=	is less than or equal to	x<=8	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 loops *(on page 433)*
- for each…in loops *(on page 433)*
- while loops *(on page 434)*
- do…while loops *(on page 434)*

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

`for (index=startValue; endCondition; incIndex) {`
`    // Code to execute`
`}`
	`for (var i=0; i<10; i++) {`
	`    logger.info("i = " + i);`
	`}`

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

`for each (var in object) {`
`  // Code to execute`
`}`
	`var sources = new Array ();`
	`sources [0] = "Customer 1";`
	`sources [1] = "Customer 2";`
	`sources [2] = "Customer 3";`
	`for each (src in sources) {`
	`  logger.info(src);`
	`}`

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

`while (condition) {`
`  // Code to execute`
`}`
	`var index = 0;`
	`var found = false;`
	`while (!found) {`
	`  if (myArray[index++] == "ABC") {`
	`  found = true;`
	`  }`
	`}`

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

`do {`
`  // Code to execute`
`} while (conditional);`
	`var index = -1;`
	`do {`
	`  var val = getValue(++index);`
	`} while (val != "ABC");`

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

`try {`
`  // Code to execute`
`} catch (exception) {`
`  // Exception handling`
`  // code to execute`
`} finally {`
`  // Code to always execute`
`}`
	`try {`
	`  var x = undefinedVarName;`
	`} catch (e) {`
	`  logger.error("Error: "+e);`
	`}`

Using JavaScript in Mirth Connect

This section shows you how to use JavaScript to perform various messaging operations within Mirth® Connect.

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>|^~\&amp;</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 only 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 *(on page 445)*.

Adding Segments to a Message

Segments are a group of fields that can contain varying types of data. Each segment exists independently and can be used in multiple messages.

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:

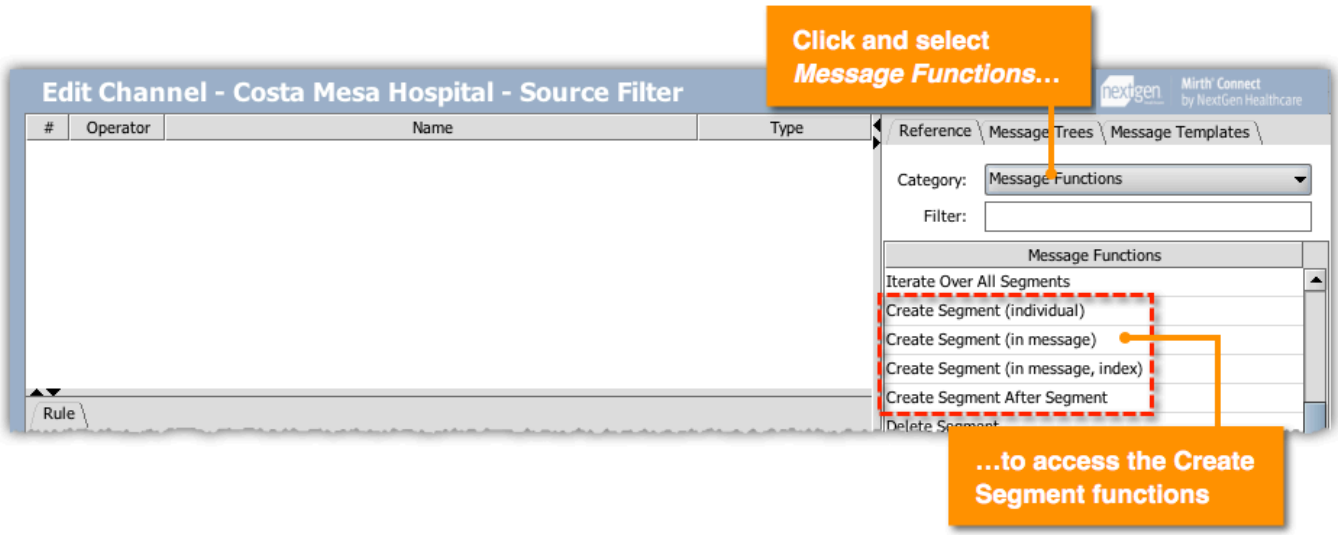
```
var seg = new XML("<ZZZ><ZZZ.1><ZZZ.1.1>My Value</ZZZ.1.1></ZZZ.1></Z
ZZ>");
msg['QRF'] += seg;
```

Insert the += operator into a message after a particular segment or at the end of the message. Mirth® Connect 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.	`var newSeg = createSegmen t('ZYX'); newSeg ['ZYX']['ZYX.1']['ZY X.1.1'] = "My Value"; msg += newSeg;`
Create Segment (in message)	Creates an XML object for the segment at the end of a specified message.	`createSegment ('ZYX', msg); msg ['ZYX']['ZYX.1']['ZY X.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']['OB X.3.2'] = "Glucose";`

Create Segment After Segment	Creates an XML object for the segment and adds it after the target segment.	`createSegmentAfter("ZZZ", ms g ["QRF"]); msg ["ZZZ"]["ZZZ.1"]["ZZ Z.1.1") = "My Value";`
		
		`which function call is equiv alent to:`
		
		`msg ["QRF"] += createSegmen t("ZZZ");`

To access these functions, navigate to an **Edit Channel** page > **Channel Tasks** panel > **Edit Filter** or **Edit Transformer**. On the **Reference** tab, select the **Category** bar > **Message Functions**, among which you will find the **Create Segment** functions. For additional information, see Reference List *(on page 274)*.



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 *(on page 288)* 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 Mirth® Connect, 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 *(on page 274)*.

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 *(on page 186)*) containing your .jar file. Then include it on the channel in the Library Resources *(on page 214)* dependencies tab.

Regular Expressions

These are used to search, match, or manipulate text strings based on patterns. In Mirth Connect 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,” “abbbc,” etc.
+	Plus	Indicates 1 instance or multiple instances of the previous character	ab+c – matches “abc,” “abbc,” “abbbc,” 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()**

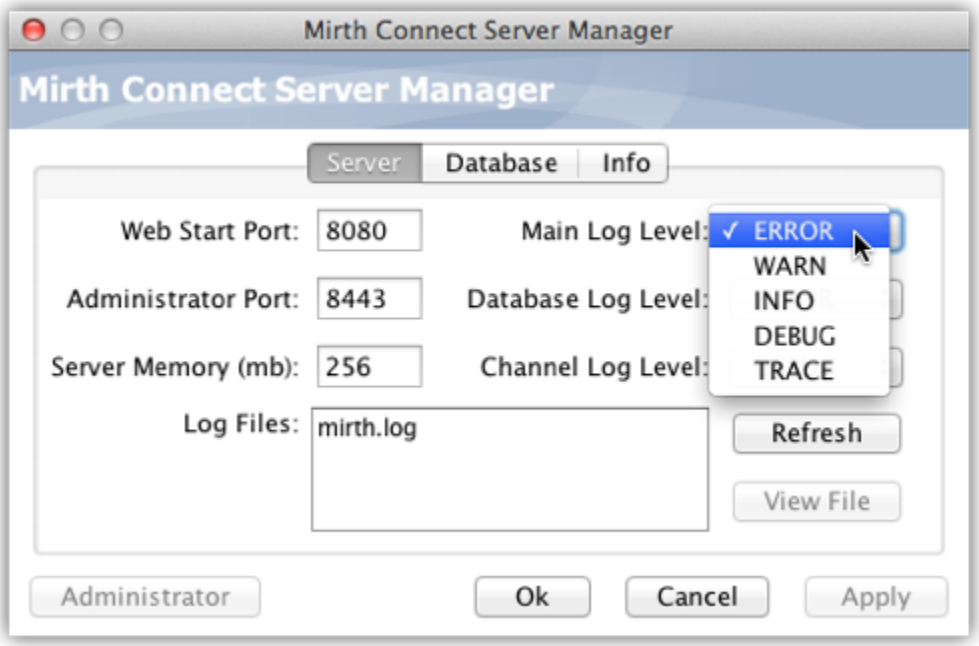
Example:

```
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 Mirth® Connect on a Mac (**Applications** > **Mirth Connect** > **mcmanager**) or PC:



Note: Mirth Appliance users can change the logging level from the Mirth® Connect **Settings** page via **Applications** > **Mirth Connect** > **Manage from the Appliance UI**.

Generating a Hash with JavaScript

Use **HashUtil.generate()** to create a hex hash of a message passing through a channel in JavaScript contexts: **var hash = HashUtil.generate(message)**;

We provide 3 overloaded versions:

- **HashUtil.generate(Object)**
  - Takes in any object and returns a hex hash of the object as a String. This defaults to using the platform's default character encoding and the SHA-256 hashing algorithm.
  - Example
    - `HashUtil.generate("Hello, World!")`

- **HashUtil.generate(String, String, String)**
  - Takes in data as a string, a character encoding as a string, and a hashing algorithm as a string and returns a hex hash of the data as a string.
  - Example
    - `HashUtil.generate("Hello, World!", "UTF-8", "SHA-256")`

- **HashUtil.generate(byte[], String)**
  - Takes in data as a byte[] and a hashing algorithm as a string and returns a hex hash of the data as a string.
  - Example
    - `HashUtil.generate("Hello, World!".getBytes(), "SHA-256")`

Supported Character Encodings (for more information, please see documentation on Charset):

- US-ASCII
- ISO-8859-1
- UTF-8
- UTF-16BE
- UTF-16LE

- UTF-16

Supported Algorithms (for more information, please see documentation on MessageDigest and MessageDigest Algorithms):

- MD2
- MD5
- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512
- SHA-512/224
- SHA-512/256

Using the JavaScript Editor

Mirth® Connect 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 *(on page 445)*
- Remapping Editor Shortcut Keys *(on page 449)*
- Using the Auto-Completion Popup in the JavaScript Editor *(on page 448)*

Using the Context Menu in the JavaScript Editor

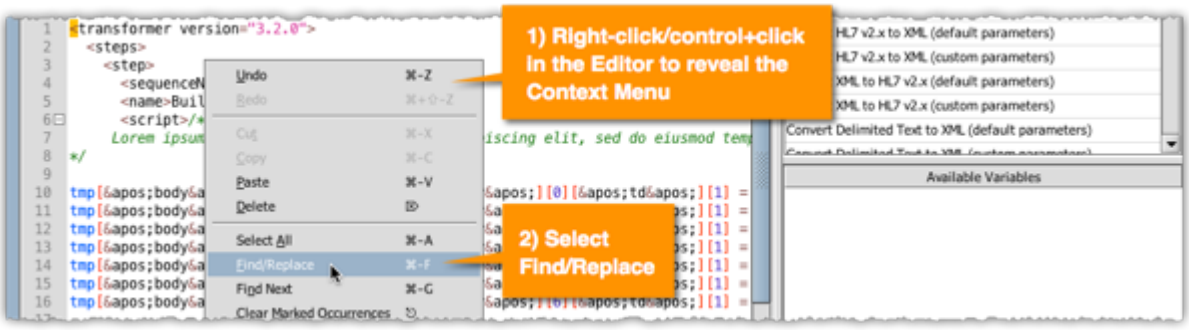
The Context menu contains items that let you Undo/Redo actions; Select/Copy/Cut/Paste/Delete code; Find/Replace code; collapse/expand sections of code; and characterizes tabs/whitespace/line endings.

- Finding/Replacing Code in the JavaScript Editor *(on page 446)*
- Folding in the JavaScript Editor *(on page 447)*

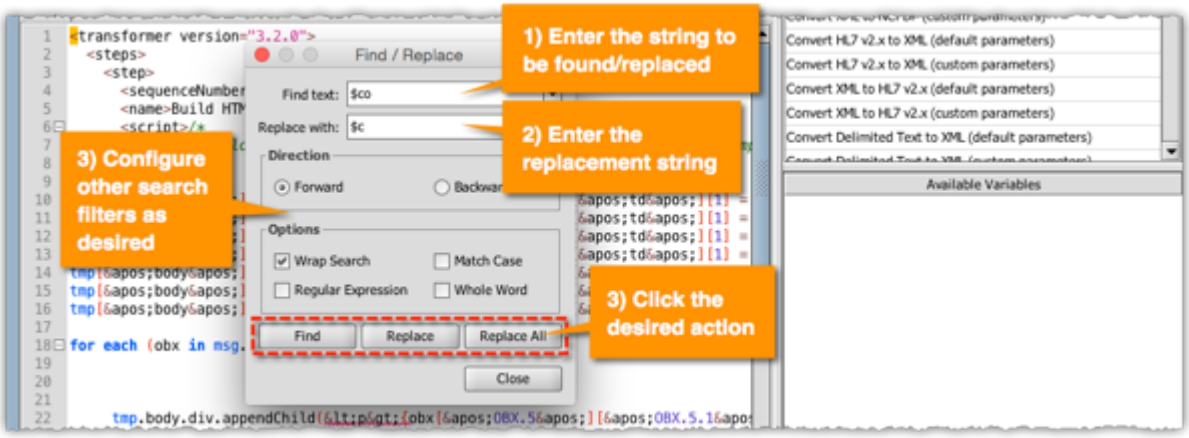
Finding/Replacing Code in the JavaScript Editor

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**) .



Note: If you are merely searching for all incidents of "$co" in the code, select 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, select the **Replace** button (to replace only the first incident of the string) or the **Replace All** button (to replace all incidents of the string).

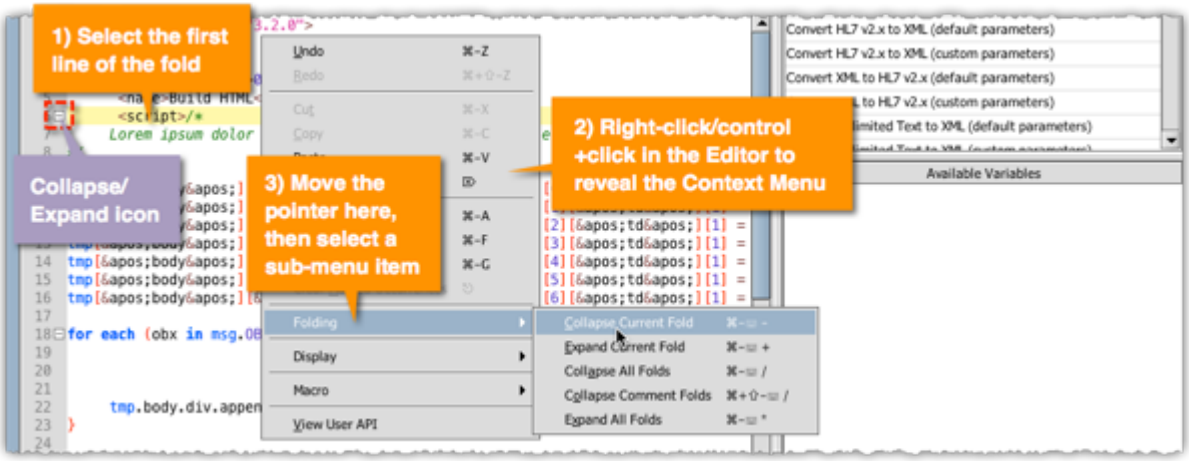
Note: Mirth® Connect performs the selected action and highlights the strings in the Editor.

6. Select the **Close** button.

Folding in the JavaScript Editor

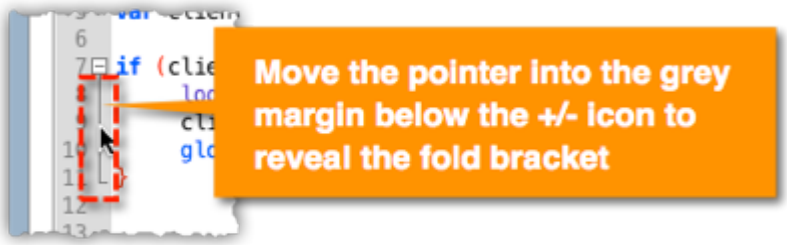
Folding, which refers to collapsing/expanding portions of code in the JavaScript Editor, can be accomplished manually by selecting the + / - icon in the left margin of the JS Editor or from the Editor's Context menu (which offers more ways to collapse/expand folds faster than you can do manually). Use 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 select the first line of the desired fold.



Note: 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.

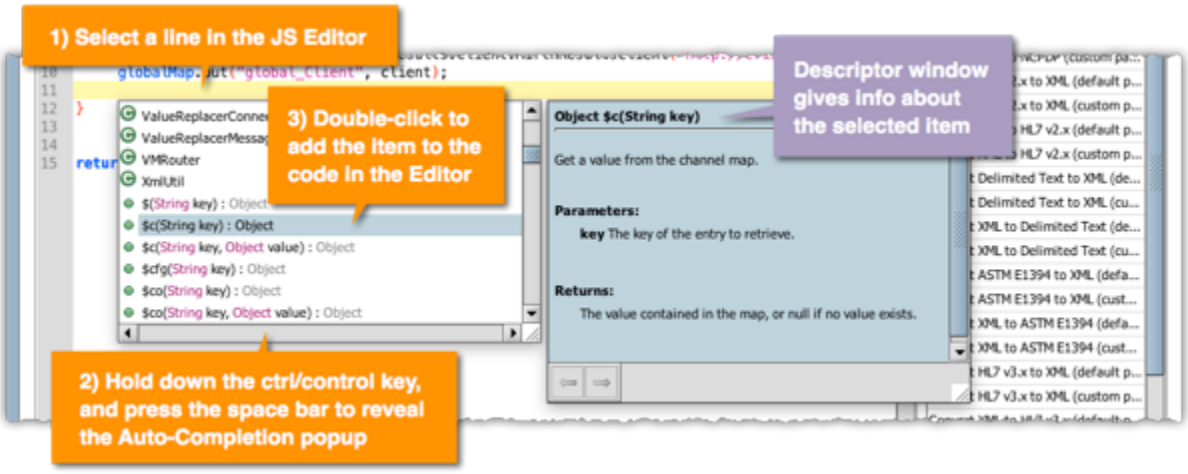


2. Right-click or **control+click** in the Editor.
3. On the context menu, move the pointer over **Folding**, and select an option on the sub-menu (in this case, **Collapse Current Fold**) .

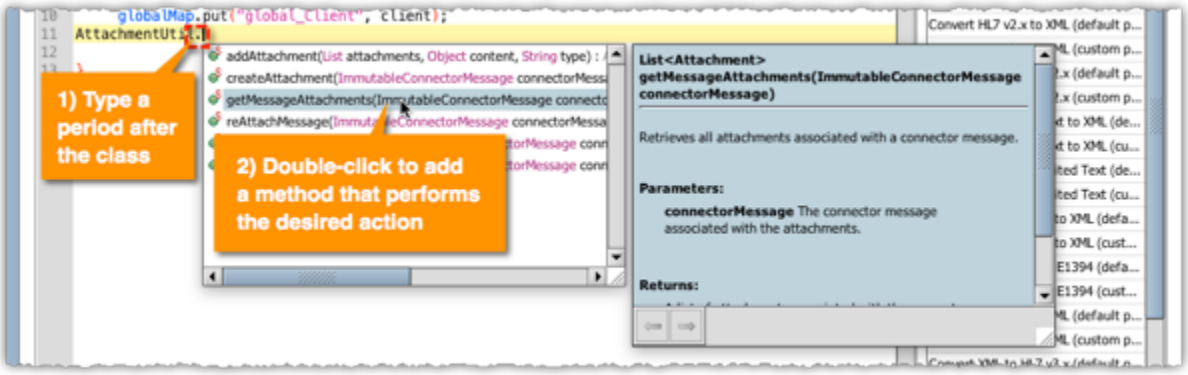
Note: The **Collapse Current Fold** action is performed.

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 *(on page 459)* or even the Message Template *(on page 260)* 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, select 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.



Note: 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 *(on page 174)* in the Settings View *(on page 166)*. You only have to do this once, even if you log into multiple, separate Mirth® Connect instances.

Variable Maps

Throughout the Message Processing Lifecycle *(on page 68)*, 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 *(on page 253)* list will display all available connector/channel map variables for example. They are also used in other ways, such as populating Custom Metadata Columns *(on page 230)*.

The following variable maps exist throughout Mirth® Connect:

Name	JavaScript Variable	Get/Put Shortcut Variable
Response Map *(on page 452)*	responseMap	$r
Connector Map *(on page 450)*	connectorMap	$co
Channel Map *(on page 451)*	channelMap	$c
Source Map *(on page 451)*	sourceMap	$s
Global Channel Map *(on page 452)*	globalChannelMap	$gc
Global Map *(on page 453)*	globalMap	$g
Configuration Map *(on page 453)*	configurationMap	$cfg

The table above also shows the precedence of these maps when referencing them in Velocity *(on page 469)* or when using the generic lookup function. For additional information, see The Variable Map Lookup Sequence *(on page 454)*.

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 *(on page 400)* destination that makes a request to a remote service, and then in the Response Transformer *(on page 286)* you store a particular response HTTP header in the channel map. As long as the next destination connector is in the same chain *(on page 68)*, 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 *(on page 340)* or an upstream process can inject source map variables. For example, the File Reader *(on page 351)* 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 *(on page 255)* 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 *(on page 238)*) .

- 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 *(on page 255)*.

This map is useful for storing stateful, non-serializable objects like a database Connection. It is in-memory only, meaning that if Mirth® Connect 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');

Note: By default the "Clear global channel map on deploy" option is enabled on the Summary Tab *(on page 204)*. 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 *(on page 313)*. You can also store global map values in the global scripts *(on page 299)*.

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 Mirth® Connect 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');

Note: By default the "Clear global map on redeploy" option is enabled on the Server Settings Tab *(on page 167)*. 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 *(on page 313)*. Unlike the global map however, this map is editable only from the Configuration Map Settings Tab *(on page 182)*, 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 Mirth Connect. For example, you could store a variable like "clientAdtPort" and then use Velocity *(on page 469)* to reference that variable ("${clientAdtPort}") in a TCP Listener *(on page 372)*. That way you can export the channel on one Mirth® Connect 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');

The Variable Map Lookup Sequence

In many cases when referencing map variables in Mirth® Connect, you do not 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 *(on page 469)*:

- ${variableName}

When you do this, Mirth® Connect 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 *(on page 299)*) , or it may be **all** maps (in the case of a filter / transformer script *(on page 257)*) . 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 *(on page 449)* section:

- var dataSource = globalChannelMap.get('dataSource');

Attachment JavaScript Functions

Attachment Handlers *(on page 219)* 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, Mirth® Connect 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 *(on page 249)*) .

Throughout the message lifecycle you can retrieve, modify, and add new attachments with built-in helper functions.

- Built-In Attachment Functions *(on page 455)*
- The AttachmentUtil Class *(on page 457)*
- The Attachment Object *(on page 458)*
- Examples *(on page 459)*

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 are 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 The Attachment Object *(on page 458)* 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 are 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 *(on page 455)* are also available from the AttachmentUtil utility class available from the User API *(on page 459)*. 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) *(on page 459)*

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 *(on page 459)*. 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

Note: When you retrieve an attachment using the Built-In Attachment Functions *(on page 455)* or The AttachmentUtil Class *(on page 457)*, 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).getContentStrin
g());

ccd.id.@root = 'testing';

updateAttachment(attachmentId, ccd.toString(), 'text/xml', true);
```

Read in a PDF from the filesystem and add it as an attachment:

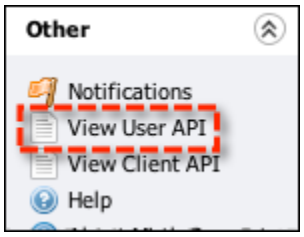
```
var fileBytes = FileUtil.readBytes('/path/to/file.pdf');

// Pass in true for Base64Encode, since the content isn't already Bas
e64 encoded
addAttachment(fileBytes, 'application/pdf', true);
```

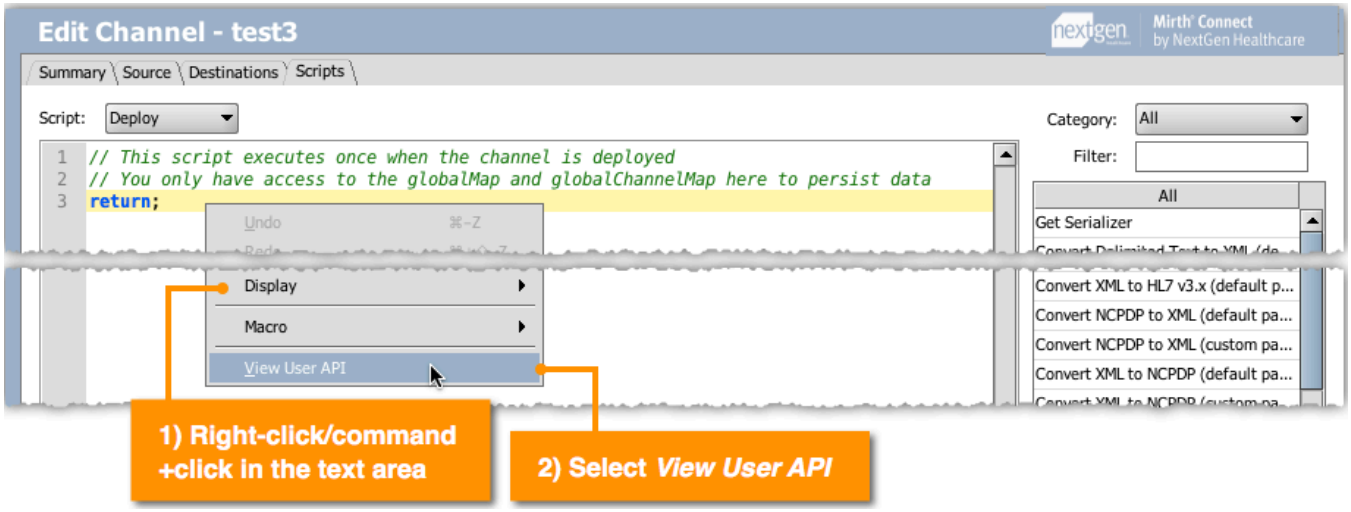
The User API (Javadoc)

The user API (application programming interface) is a collection of Java classes and methods that helps you interact with channels and the message that is being processed. It also provides helper methods for common tasks such as date formatting.

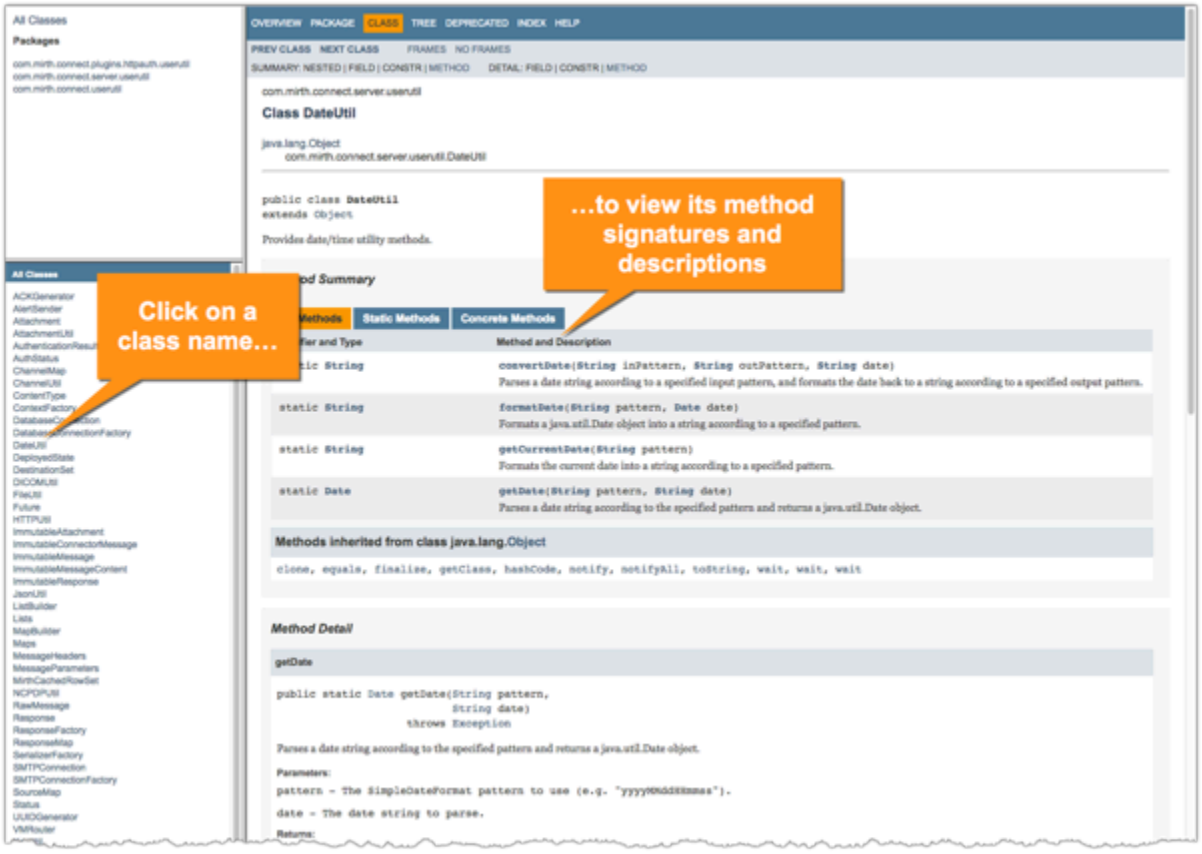
Wherever you are editing JavaScript code in Mirth® Connect, you can view the API by selecting the **View User API** function in the **Other** panel, which is available on all Mirth® Connect Administrator views.



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.



Note: You can also view the API via this link: http://javadocs.mirthcorp.com/connect/3.6.0/user-api/

Mirth Connect Debugger

The **Mirth Connect Debugger** tool allows you to troubleshoot JavaScript errors throughout the Mirth® Connect application. The Debugger can be used for the following scripts and connectors:

- Deploy, Undeploy, Preprocessor, and Post Processor scripts
- Attachment/Batch scripts
- Source Connectors:
    - Filters/Transformers
    - Database Reader and JavaScript Reader types

- Destination Connectors:
    - Filters/Transformers.
    - Database Writer and JavaScript Writer types
    - Response transformers

Before You Begin

If your Java application does not interact directly with a user, use the following steps to change the Headless mode to false. Setting this mode to false means that your Java application:

- Displays windows or dialog boxes.
- Accepts keyboard or mouse input.
- Uses any heavyweight AWT (Abstract Windowing Toolkit) components.

To Edit the mcserver.vmoptions file

1. Access the folder in which Mirth® Connect is installed.

    Note: The mcserver.vmoptions file can typically be found in the following location:

    - **Windows**: C:\Program Files\Mirth® Connect\mcserver.vmoptions
    - **MacOS**: /Applications/Mirth® Connect/mcserver.vmoptions

2. Open the **mcserver.vmoptions** file.
3. Set **-Djava.awt.headless=false**.

```
-server
-Xmx256m
-Djava.awt.headless=false
-Dapple.awt.UIElement=true
```

4. Save the updated file.

Use the Debugger

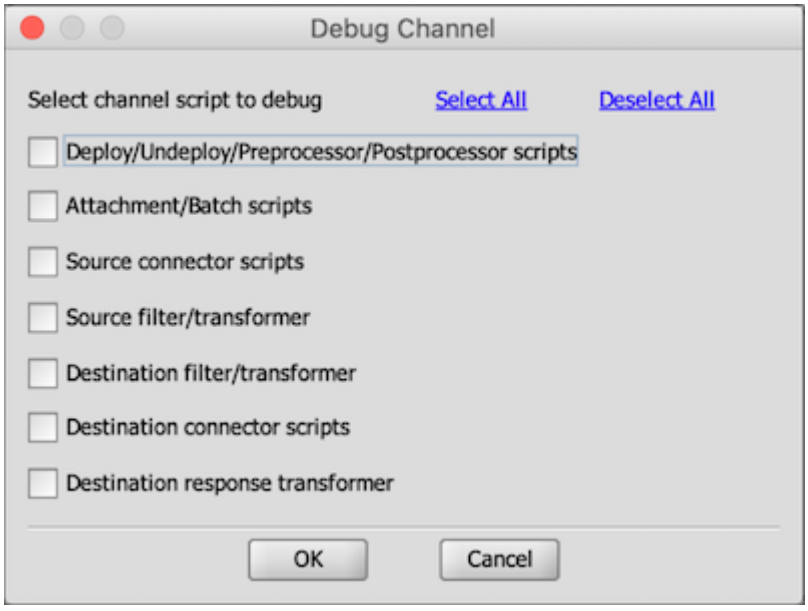
Note:

- mcservice.exe is currently not supported for Debug mode.
- Debug mode is not recommended for Clustering environment (Basic or Advanced Clustering).

1. Within Mirth® Connect, select **Channels**.
2. In the **Channels** list, select the channel you would like to debug.

   Note: You can currently only debug one channel at a time.

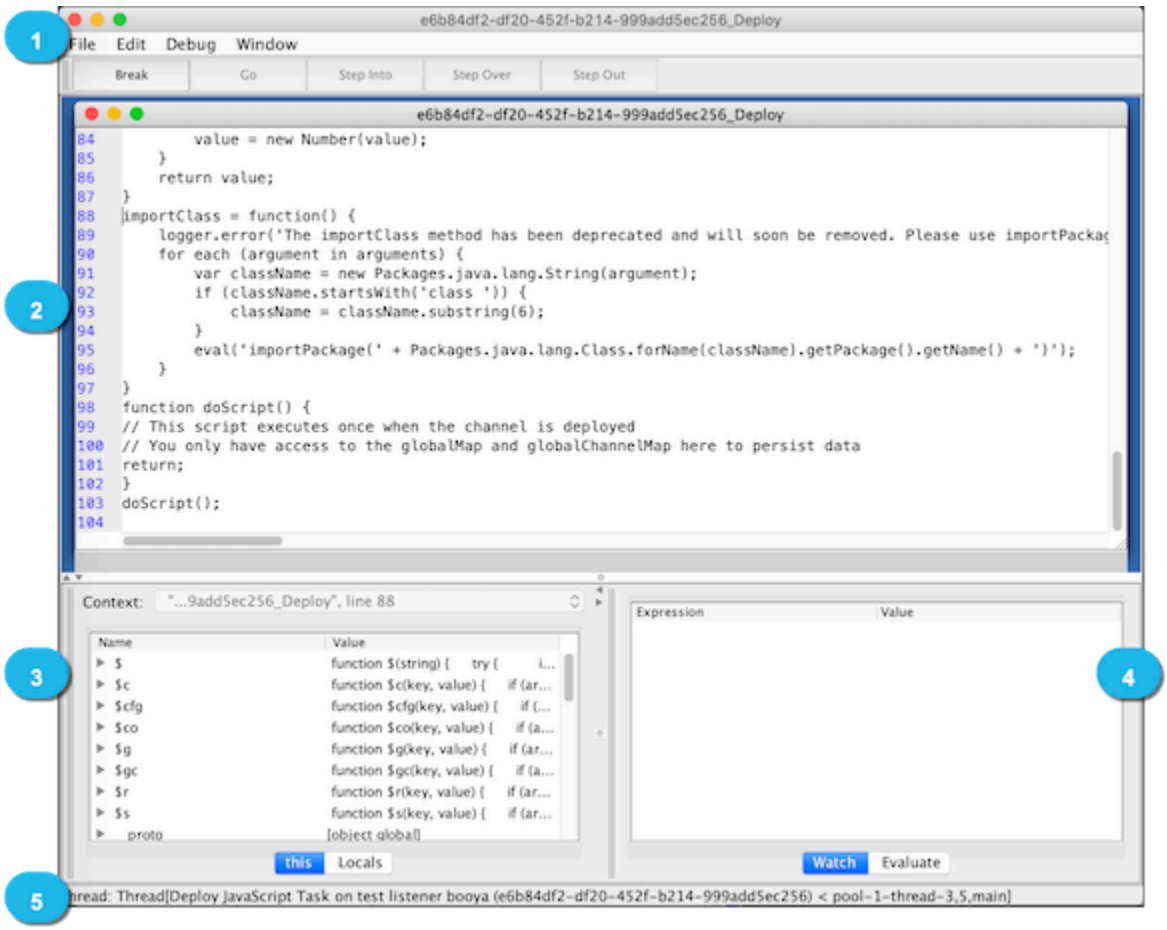
3. In **Channel Tasks**, select **Debug Channel**.
4. On the **Debug Channel** window, select the channel script(s) you would like to use to debug the selected channel.



5. Select **OK**. A separate debugger window opens for each selected script, and for each destination connector.
6. Process the selected channel (For Example: send a message). The Debugger window(s) will pop-up based on the options you selected.
7. To exit the debug mode, close the debugger window(s) or undeploy the channel.

Note: The channel will not stop or undeploy properly if the debugger is waiting for user operation, such as selecting “Go”.

Debugger Window



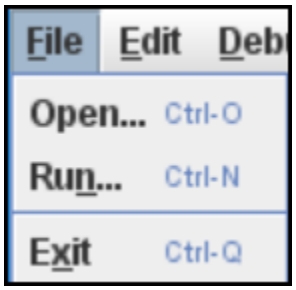
The following context-specific tasks are available:

Number	Name	Description
1	**Title And Menu Bar**	The **Title** identifies the Chanel ID and the Event Name or Connector Identifier for the selected channel. File Edit Debug Window The **Menu Bar** allows you to perform actions on the

Number	Name	Description
		Debugger window.
2	**Coding Area**	The **Coding** area displays Mirth® Connect code and custom JavaScript associated with the channel debugger
3	**Local Variables Area**	The **Local Variables** area shows the values and values and the current state of local variables associated with the debugger results.
4	**Expressions Area**	The **Expressions** area allows you to evaluate an expression to determine what a value is. You can also evaluate a snippet of code in this area.
5	**Thread Information Bar**	The **Thread Information** bar identifies the following information: • Connector type (Ex: JavaScript Writer) • Channel Name (Ex: “Debugger Channel”) • Channel ID • Location of JavaScript (Ex: destination “Physician (2)” where “Physician” is the name of a destination connector and “2” is the destination number)

Debugger Menus

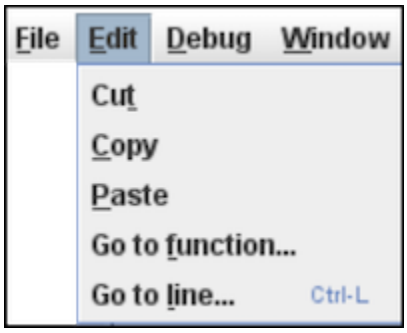
File Menu



Name	Description
Open	Opens a dialog which allows you to select JavaScript from the system which

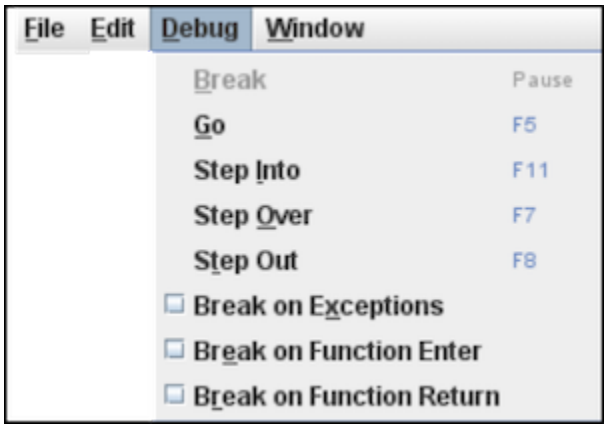
Name	Description
	is pulled into a new window in the debugger workspace.
Run	Allows you to select JavaScript from the system which is pulled into the debugger window for execution. The new test interruption point will be the next executable line.
Exit	Finishes the current test and closes the debugger window.

Edit Menu



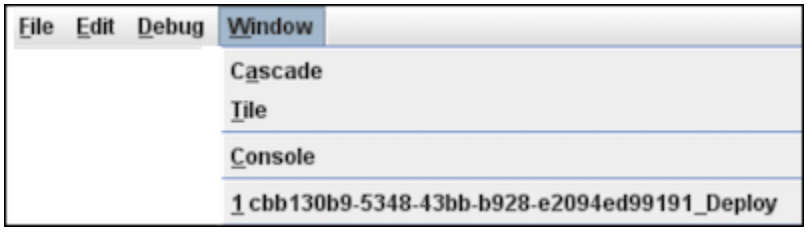
Name	Description
Cut	Copies the highlighted text to the system clipboard and removes the text from the debugger window.
Copy	Copies the highlighted text to the system clipboard.
Paste	Places the system clipboard text into the current location of the cursor in the debugger window.
Go to function...	Moves the cursor to a specific named function within the script.
Go to line...	Moves the cursor to a specific line within the script.

Debug Menu



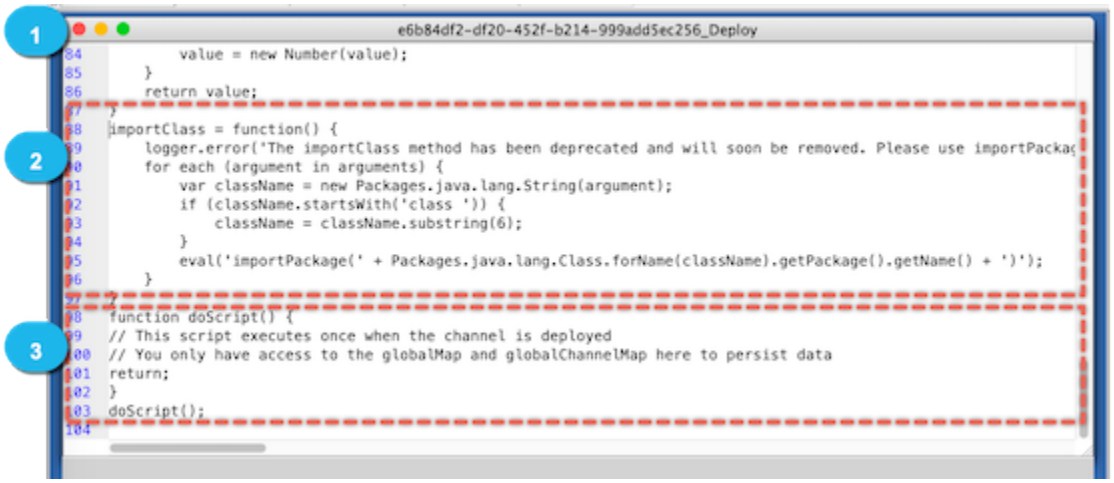
Name	Description
Break	Pauses to interrupt the running code in the test.
Go	Continues running the test until the end or to the next breakpoint.
Step Into	Increments the execution by one line. If the next line is inside a function, then the interruption will step into the function allowing the view of the context within.
Step Over	If the current function has been called by another function, then Step Over returns to the previous function. Otherwise it continues running the test to the end or to the next breakpoint (like the Go function).
Step Out	Progresses the execution to the next line outside of the current function.
Break on Exceptions	Continues running the test script until an exception occurs.
Break on Function Enter	Continues and then breaks once you reach a new function.
Break on Function Return	Continues and then breaks once you reach the end of the current function.

Window Menu



Name	Description
Cascade	Stacks all windows with the debugger work space.
Tile	Places all windows side-by-side in the debugger work space.
Console	Brings focus to the Console Window. The Console Window allows users to perform small scripts, test the state of variables, and display logs.

Coding Area



Number	Name	Description
1	**Title**	The **Title** identifies the Chanel ID and the Event Name or Connector Identifier for the selected channel.
2	**Internal Code**	This is the **thread interruption point** which appears where the internal code of the debug option starts (i.e., the "**importClass – function()** ". line). As you test your script, you can see the state of various objects and variables in the lines prior to the interruption line.
3	**Custom JavaScript**	Any custom JavaScript displays normally within the "**doScript()** " boundary.

CHAPTER 11

Velocity Variable Replacement

The Apache Velocity template engine is used throughout Mirth® Connect 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 *(on page 449)*, and then use a Velocity template to inject those variables into the TCP Sender *(on page 416)* settings.

- Basic Syntax *(on page 469)*
- Conditional Statements *(on page 470)*
- For Loops *(on page 470)*

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 *(on page 454)*. 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 ${list.size()} total valu
e#if($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>
```

CHAPTER 12

Mirth Connect Command Line Interface

The Command Line Interface is an alternate, lightweight way to interact with your Mirth® Connect 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 Mirth® Connect 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 do not need to enter it every time.

Once connected, you will 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:

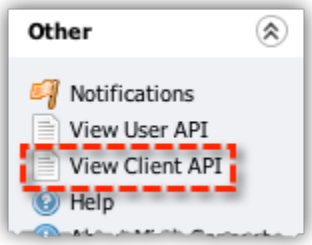
- In a terminal / shell, run the **mccommand** executable script, with the following options:
  - `mccommand -a https://localhost:8443 -u user -p pass -s script.txt`
- Change the address, username, and password as necessary

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.

CHAPTER 13

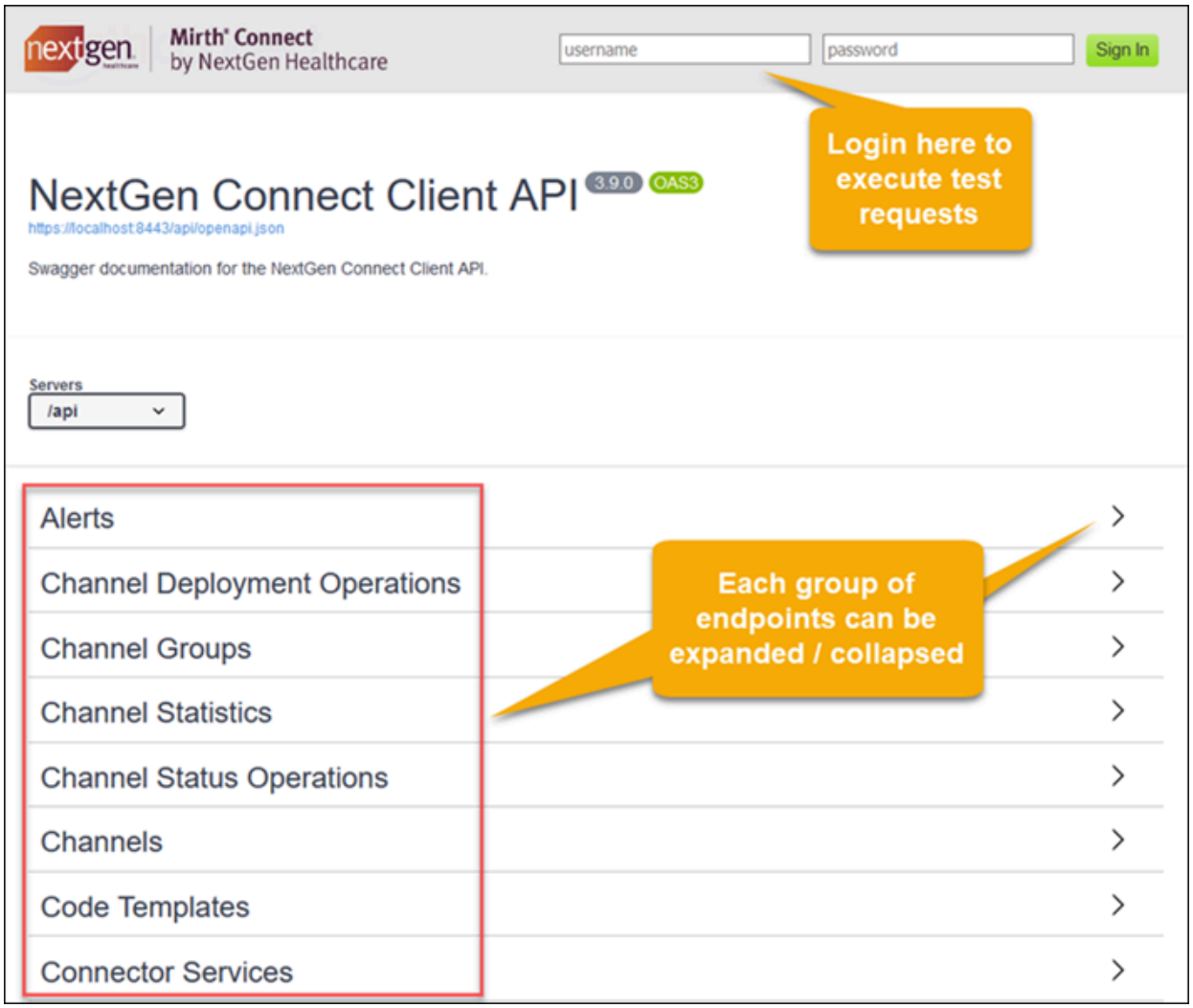
Mirth Connect REST API

The Administrator *(on page 79)* and Command Line Interface *(on page 471)* both use a well-defined API to communicate with the Mirth® Connect 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 select View Client API in Administrator via Other Tasks *(on page 320)*:

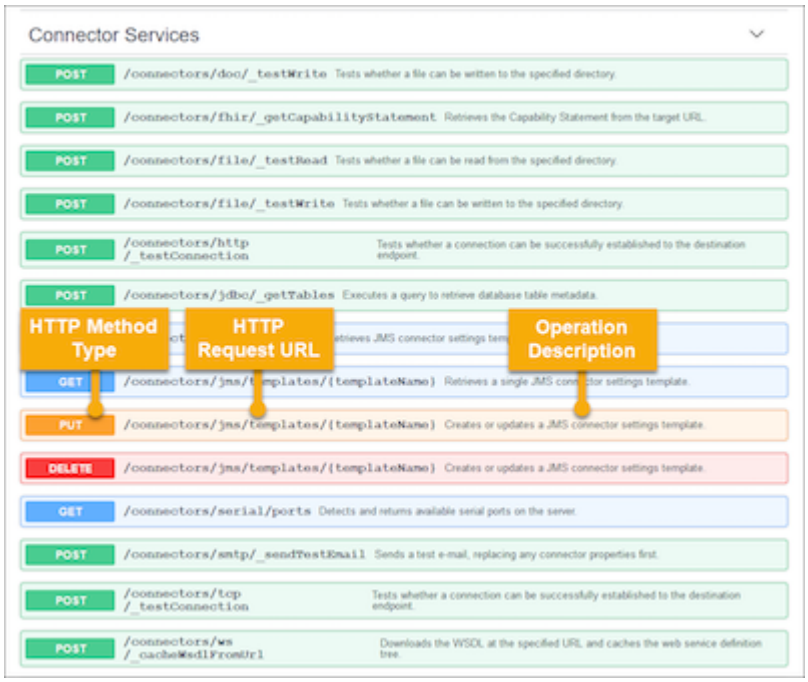


- Select the **View Client API** action to open the API documentation in your default browser.

Note: You may see a warning in your browser if your Mirth® Connect instance is still using the auto-generated self-signed certificate. You can select **Advanced** and allow the browser to continue to the page. To resolve this warning, look at Application Data Directory *(on page 478)* for instructions on installing your own certificate.



- Select the arrow button on one of the endpoint group rows to view its operations. This list shows the operation's HTTP method type, request URL, and a description for each endpoint.



- Select on one of the endpoint rows to view its details. If desired, you can select a specific operation to view only that op's details. Depending on the type of endpoint, the following may be available:
    - **Parameters**
    - **Request Body**
    - **Responses**

Authentication

Mirth® Connect 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):

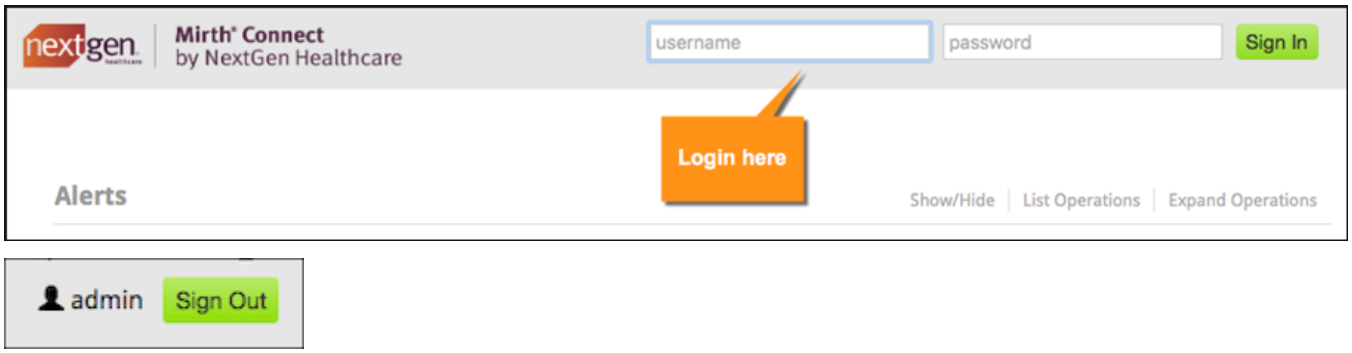
- `Authorization: Basic YWRtaW46YWRtaW4=`

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=uysqrtx9lle36ernybizstps;Path=/api;Secure`
- Invoke the actual API endpoints of your choice, passing in the same cookie as a header:
  - `Cookie: JSESSIONID=uysqrtx9lle36ernybizstps;Path=/api;Secure`
- Once you are 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:



CHAPTER 14

Installation Directory

This is the location you installed or extracted Mirth® Connect into. Typically only administrators need to have access to this, and even then usually only once when first setting up. For example this where you can edit initial configuration to change the database *(on page 49)*Mirth® Connect points to.

Application Data Directory

This directory stores configuration files and temporary data created by Mirth® Connect 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 *(on page 485)*. The main files/folders of note here are as follows:

- configuration.properties *(on page 478)*
- extension.properties *(on page 479)*
- keystore.jks *(on page 479)*
- server.id *(on page 480)*
- temp *(on page 480)*

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 *(on page 182)*. If you instead edit the properties file manually, Mirth® Connect 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 and disabling can be done using 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 File *(on page 485)*.

Warning: 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 Mirth® Connect 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 Mirth® Connect, 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.
- Have your new keypair ready to import in a PCKS #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 -src
      storetype PKCS12 -srcstorepass mystorepass -srckeypass myk
      eypass -srcalias myalias -destkeystore keystore.jks -dests
      toretype JCEKS -deststorepass 81uWxplDtB -destkeypass 81uW
      xplDtB -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 Mirth® Connect server.

server.id

This is the unique ID for your server instance. It is auto-generated when Mirth® Connect starts up for the first time.

temp

This stores all temporary files created by the Mirth® Connect 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 File *(on page 485)*.

Configuration Directory

This directory contains the main configuration files Mirth® Connect needs in order to start up correctly. Modifying the configuration requires a restart of Mirth® Connect. The following files are found here:

- dbdrivers.xml File *(on page 480)*
- log4j2.properties File *(on page 482)*
- log4j2-cli.properties File *(on page 484)*
- mirth.properties File *(on page 485)*
- mirth-cli-config.properties File *(on page 500)*

dbdrivers.xml File

Note: This file is used by the and 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, can
not be empty
    selectLimit = defines the select statement used for retrieving co
lumn information, empty means use the generic query (which could be s
low)
-->
<drivers>
    <driver class="sun.jdbc.odbc.JdbcOdbcDriver" name="Sun JDBC-ODBC
Bridge" template="jdbc:odbc:DSN" selectLimit="" />
    <driver class="com.mysql.jdbc.Driver" name="MySQL" template="jdb
c:mysql://host:port/dbname" selectLimit="SELECT * FROM ? LIMIT 1" />
    <driver class="oracle.jdbc.driver.OracleDriver" name="Oracle" tem
plate="jdbc:oracle:thin:@host:port:dbname" selectLimit="SELECT * FROM
? WHERE ROWNUM &amp;lt; 2" />
    <driver class="org.postgresql.Driver" name="PostgreSQL" templat
e="jdbc:postgresql://host:port/dbname" selectLimit="SELECT * FROM ? L
IMIT 1" />
    <driver class="net.sourceforge.jtds.jdbc.Driver" name="SQL Serve
r/Sybase" template="jdbc:jtds:sqlserver://host:port/dbname" selectLim
it="SELECT TOP 1 * FROM ?" />
    <driver class="org.sqlite.JDBC" name="SQLite" template="jdbc:sqli
te: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 select 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).

log4j2.properties File

This file is used to specify where to write log files to and how verbose the logs should be. It is split up into the following sections:

```
rootLogger = ERROR,stdout,fout
```

This is the "root" logger property which is 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
appender.console.type = Console
appender.console.name = stdout
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = %-5p %d{yyyy-MM-dd HH:mm:ss.SSS}
[%t] %c: %m%n
```

This appender ensures that all logging can redirect to STDOUT, if you are running Mirth Connect from an attached shell.

```
# file appender
dir.logs = logs
```

```
appender.rolling.type = RollingFile
appender.rolling.name = fout
appender.rolling.fileName = logs/mirth.log
appender.rolling.filePattern = logs/mirth.log.%i
appender.rolling.policies.type = Policies
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size = 500KB
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.max = 20
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = %-5p %d{yyyy-MM-dd HH:mm:ss.SSS}
[%t] %c: %m%n
```

This appender writes logs 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
logger.mirth.name = com.mirth.connect.server.Mirth
logger.mirth.level = INFO

# Mirth® Connect server logging
logger.donkeyEngineController.name = com.mirth.connect.server.control
lers.DonkeyEngineController
logger.donkeyEngineController.level = INFO
logger.recoveryTask.name = com.mirth.connect.donkey.server.channel.Re
coveryTask
logger.recoveryTask.level = INFO
logger.fileReceiver.name = com.mirth.connect.connectors.file.FileRece
iver
logger.fileReceiver.level = WARN
```

These entries help display certain helpful information when logging on and when messages are recovered.

```
# Mirth® Connect channel logging
logger.transformer.name = transformer
logger.transformer.level = DEBUG
logger.preprocessor.name = preprocessor
logger.preprocessor.level = DEBUG
logger.postprocessor.name = postprocessor
logger.postprocessor.level = DEBUG
```

```
logger.deploy.name = deploy
logger.deploy.level = DEBUG
logger.undeploy.name = undeploy
logger.undeploy.level = DEBUG
logger.filter.name = filter
logger.filter.level = DEBUG
logger.db-connector.name = db-connector
logger.db-connector.level = DEBUG
logger.js-connector.name = js-connector
logger.js-connector.level = DEBUG
logger.attachment.name = attachment
logger.attachment.level = DEBUG
logger.batch.name = batch
logger.batch.level = DEBUG
logger.response.name = response
logger.response.level = DEBUG
logger.shutdown.name = shutdown
logger.shutdown.level = 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
logger.sql.name = java.sql
logger.sql.level = ERROR
```

Turn the verbosity up here to see more information about which database statements are being executed across the server.

log4j2-cli.properties File

This is the same as the log4j2.properties file, except that it is specific to the Command Line Interface. You can edit the root logger level, and the STDOUT appender settings:

```
rootLogger=ERROR, stdout

# stdout appender
appender.console.type=Console
appender.console.name=stdout
appender.console.layout.type=PatternLayout
appender.console.layout.pattern=%-5p %d [%t] %c: %m%n
```

mirth.properties File

This is the main configuration file that tells Mirth® Connect where to store application data, what web server ports to listen on, and which database to connect to. You can also set other security and encryption options.

The following properties are supported:

Property	Default Value	Description
Directories		
dir.appdata	appdata	The location of the Application Data Directory.
dir.tempdata	${dir.appdata}/temp	The location of the temporary files directory, by default set inside of the Application Data Directory.
Ports		
http.port	8080	The HTTP port to make the web server available from. This is used to access the launch page and download signed client resources from. If this property is omitted or commented out, the web server will only start up on the HTTPS port.
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 REST API *(on page 473)* traffic (which includes the Administrator *(on page 79)* and CLI *(on page 471)*) .
Password Requirements		
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.minnumeric	0	Minimum numeric characters, 0 for no minimum.

Property	Default Value	Description
password.minspecial	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.
Keystore		
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 Application Data Directory.
keystore.storepass	81uWxplDtB	The password for the keystore file itself. It is a good idea to change this from the default value. 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. It is a good idea to change this

Property	Default Value	Description
		from the default value. 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.
Server		
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 **client-side** heap size to set in the Java Web Start JNLP. Users may override this on the launch page. Note that this is **not** the same as the server-side max heap size.
administrator. maxheapsizeoptions	256m,512m,1g,2g	The client-side max heap size options

Property	Default Value	Description
		to give the user from the launch page and from the Server Manager.
configurationmap.location	file	Determines whether the configuration map will be stored as a file, or in the database. Valid values: file, database
configurationmap.path	${dir.appdata}/configuration.properties	The location of the configuration map properties file. Usually this is in the Application Data Directory.
extension.properties.provider		The fully-qualified class (extending ExtensionStatusProvider) that controls extension enabled/disabled flags. If absent or set to "file", the default behavior is used, reading from ${dir.appdata}/extension.properties.
donkey.statsupdateinterval	1000	The interval on which to update channel statistics across all channels.
license.key		If you have any commercial extensions installed, enter your license key here. Contact the help desk through our Success Community to get a license key.
rhino.optimizationlevel	-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.
rhino.languageversion	ES6	The ECMAScript/JavaScript version that the Rhino engine should use. Valid values: ES6, DEFAULT, 1.0-1.8.
server.api.require-requested-with	true	If set to true, the Mirth® Connect REST API will require all incoming requests to contain an "X-Requested-With" header. This protects against Cross-Site Request Forgery (CSRF) security vulnerabilities.
server.api.sessionstore	false	If set to true, the web server sessions are stored in the database.

Property	Default Value	Description
server.api.sessionstoretable	sessiondata	The table name to use for web server session data.
server.api.sessioncache	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.
server.api.sessionmaxinactiveinterval	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.
Security		
http.stricttransportsecurity	true	HTTP Strict Transport Security (HSTS) is a web security policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking. It enables web servers to declare that web browsers (or other complying user agents) should automatically interact with it using only HTTPS connections, which provide Transport Layer Security (TLS/SSL), unlike the insecure HTTP used alone
https.client.protocols	TLSv1.3,TLSv1.2	The protocols to support by default for all TLS/SSL/HTTPS client traffic. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager enables you to adjust protocol settings on a per-connector basis, rather than having to change the value for the entire server.
https.server.protocols	TLSv1.3,TLSv1.2,SSLv2Hello	The protocols to support by default for all TLS/SSL/HTTPS server traffic.

Property	Default Value	Description
		CAUTION: Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager enables you to adjust protocol settings on a per-connector basis, rather than having to change the value for the entire server.
https.ciphersuites	See Default Supported Cipher Suites *(on page 498)*	The cipher suites to support by default for all TLS/SSL/HTTPS server traffic. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager *(on page 598)* enables you to adjust protocol 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. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks.
server.api.allowhttp	false	If enabled, API access will be allowed through the regular HTTP port. Generally you should **not** 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

Property	Default Value	Description
		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.contentsecurity policy	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 are 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 are embedding API access in a webpage.
xstream.denytypes		A comma-separated list of fully-qualified classes to deny XStream from serializing or deserializing. The classes can include wildcards. This list takes precedence over the list of allowed types.
xstream.allowtypes		A comma-separated list of fully-qualified classes to allow XStream to serialize or deserialize. The classes can include wildcards. Classes are denied unless they are explicitly allowed, through this property or `xstream.allowtypehierarchies`, or are essential for Mirth Connect to function.
xstream.allowtypehierarchies		A comma-separated list of fully-qualified type hierarchies to allow

Property	Default Value	Description
		XStream to serialize or deserialize. This means that these classes and their subclasses are allowed. Classes are denied unless they are explicitly allowed, through this property or `xstream.allowtypes`, or are essential for Mirth® Connect to function.
Database		
database	derby	The database type to use for the Mirth® Connect backend database. Options: • derby • mysql • postgres • oracle • sqlserver By default Mirth® Connect ships with an embedded Apache Derby database for quick testing / development purposes. For production instances, you should change the database type *(on page 49)* to one of the other supported options.
database.url	dbc: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.

Property	Default Value	Description
database.connection.maxretry	2	On startup, if a database connection cannot be made for any reason, Mirth® Connect will wait and attempt again this number of times. By default, will retry 2 times (so 3 total attempts).
database.connection.retrywaitinmilliseconds	10000	The amount of time (in milliseconds) to wait between database connection attempts. By default, will wait 10 seconds between attempts.
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 *(on page 497)*	true	If enabled, the database connection pool will be split into read-only and read/write pools. More information here *(on page 497)*.
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

Property	Default Value	Description
		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.
Encryption		
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 Mirth® Connect server is next started. To update the password, overwrite database.password and on next server startup it will be automatically encrypted and updated again.

Property	Default Value	Description
encryption.algorithm	AES/CBC/PKCS5Padding	The algorithm to use for symmetric encryption. This applies to messages, exports, and anything that is used along with the keystore to encrypt / decrypt. You must include the explicit mode and padding settings with the algorithm. The mode must also require an initialization vector. **Note:** The default value is updated in version 4.3. • If you did not explicitly set this property, then no action is needed. Mirth Connect will automatically start using the new default. • If you did explicitly set this property, then NextGen Healthcare highly recommends you to update it to include the mode/ padding options as well. • Support for "AES" (without any mode/padding specified) will be removed in a future version.
encryption.charset	UTF-8	The charset to use when encoding textual data into bytes before encryption.
encryption.keylength	128	The key length to use for symmetric encryption.
encryption.fallback.algorithm		This is automatically set when you upgrade to 4.3. It is used to decrypt old data that may have been encrypted in a prior version of Mirth® Connect.
encryption.fallback.charset		This is automatically set when you upgrade to 4.3, if your default JVM charset is not UTF-8. After decryption, it is used to decode old data that may have been encrypted in a prior version

Property	Default Value	Description
		of Mirth Connect.
digest.algorithm	PBKDF2WithHmacSHA256	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. If you are using a Cures-certified version of Mirth® Connect, you must use one of the following digest algorithms for your environment to meet certification requirements: • SHA256 • SHA384 • SHA512
digest.saltsizeinbytes	8	The amount of bytes to use for random-generated salt values to store along with hashes and digests.
digest.iterations	600000	The amount of times to run an input message through the digest algorithm. The appropriate value for this property depends on the algorithm used.
digest.usepbe	1	0 for disabled and 1 for enabled. If enabled, the digest algorithm will be assumed to be a Password-Based Encryption function and will be instantiated using a SecretKeyFactory and PBEKeySpec, rather than a MessageDigest object.
digest.keysizeinbits	256	Only used if `digest.usepbe` is enabled. The size (in bits) of the key to use for the Password-Based Encryption function.
digest.fallback.algorithm	SHA256	The old digest algorithm, used for verifying old password hashes.
digest.fallback.saltsizeinbytes	8	The old digest salt size, used for verifying old password hashes.

Property	Default Value	Description
digest.fallback.iterations	1000	The old digest iterations, used for verifying old password hashes.
digest.fallback.usepbe	0	The old digest PBE flag, used for verifying old password hashes.
digest.fallback.keysizeinbits	256	The old digest PBE key size, used for verifying old password hashes.
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 enable 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", enables 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_EMPTY_RENEGOTIATION_INFO_SCSV

Note: 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.

New Default Digest Algorithm in Mirth Connect 4.4

In Mirth® Connect version 4.4, the default digest/hash algorithm was changed from SHA256 to PBKDF2WithHmacSHA256. In addition, the default iteration count was changed from 1000 to 600000. This was done to conform with modern password security recommendations.

If you have already explicitly set `digest.algorithm`, then no changes are made, Mirth® Connect uses the digest algorithm that you have set.

If you haven't already explicitly set `digest.algorithm`, then the new default algorithm is used going forward. New "fallback" properties are set to the old defaults, so old digest values can still be verified. When a user updates their password, the digest value is updated to use the new algorithm at that time.

Update the Digest Algorithm

To ensure you can still log on with your existing credentials (which used the old digest settings), you want to make sure to set the "fallback" digest settings in the mirth.properties file.

1. Set the "fallback" digest settings in the mirth.properties file.
   For example:

   ```
   digest.fallback.algorithm = SHA256
   digest.fallback.saltsizeinbytes = 8
   digest.fallback.iterations = 1000
   digest.fallback.usepbe = 0
   digest.fallback.keysizeinbits = 256
   ```

2. Update the `digest.*` settings to the new algorithm you want to use.

After restarting Mirth® Connect, it will use the new digest settings for all new digest values. It will also use the "fallback" settings to verify old digest values.
To use the new digest value for your stored password hash in the database, update your user password in

the Administrator.

mirth-cli-config.properties File

This properties file allows you to launch the Mirth Connect Command Line Interface *(on page 471)* without having to include any command line arguments like the username / password. The following properties are supported:

- **address:** The HTTPS address of the Mirth® Connect 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 do not 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 *(on page 478)* include the following:

- **cli-lib:** Libraries used by the Command Line Interface *(on page 471)*. These should not be changed.
- **client-lib:** Libraries used by the Mirth Connect Administrator *(on page 79)*. These should not be changed.
- **custom-lib:** Custom libraries can be dropped in this folder, to be used by the default library resource *(on page 186)*.
- **docs:** Contains license information for Mirth® Connect and all third-party libraries. Also contains the Javadoc documentation used by User API *(on page 459)*.
- **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 log4j2.properties file *(on page 482)*.
- **manager-lib:** Libraries used by the Server Manager *(on page 44)*. These should not be changed.

- **mccommand(.exe):** The entrypoint script for launching the Command Line Interface *(on page 471)*.
- **mcmanager(.exe):** The entrypoint script for launching the Server Manager *(on page 44)*.
- **mcserver(.exe):** The entrypoint script for launching the Mirth Connect 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 Mirth Connect 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 *(on page 473)*.
- **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 Mirth® Connect server. These should not be changed.
- **server-launcher-lib:** JARs in this directory will be loaded into the main server thread context classloader at startup. This is required if you are using any custom log4j2 appender libraries.
- **webapps:** All WAR files in this directory will automatically be published by the Mirth® Connect web server at startup.

CHAPTER 15

Frequently Asked Questions

This section addresses the most common questions users have about Mirth® Connect by NextGen Healthcare.

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.

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.

What is the Mirth® Connect license, and how much does it cost?

Mirth® Connect is released under the Open Source Initiative (OSI) approved MPL 2.0 (see Mozilla Public License 2.0), and it costs you absolutely nothing! You can download Mirth® Connect free via the Mirth® Connect download page.

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.

Is there a difference between the free, open-source Mirth® Connect download and the supported version of Mirth® Connect?

No; however, support subscriptions *(on page 582)* 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.

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.

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.

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 Release Notes.

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 NextGen offers.

Is Mirth® Connect hard to install and configure?

No. In fact, we have 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.

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.

As a member of the Mirth Solutions community, how can I get more help?

NextGen Healthcare provides numerous commercial support options *(on page 582)* 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.

How do I become a Mirth® Connect expert?

NextGen Healthcare provides extensive Mirth® Connect training and a certification program. More information here: Training *(on page 602)*

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 5.0+
- 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.

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.

Does Mirth® Connect use the Mule ESB?

No. Prior to the release of Mirth® Connect 3.0, Mule was eliminated from the application.

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).

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.

What message standards does Mirth® Connect support?

- HL7 v2.x *(on page 329)*
- HL7 v3.x *(on page 332)*
- Delimited Text *(on page 324)* (CSV, tab-delimited, fixed-width, etc.)
- DICOM *(on page 327)*
- EDI / X12 *(on page 328)*
- XML *(on page 336)*

- JSON *(on page 333)*
- NCPDP *(on page 334)*
- Raw *(on page 335)* (supports **any** data format!)
- ASTM E1394 *(on page 586)* (Commercial extension)

What transfer protocols does Mirth® Connect support?

- MLLP *(on page 378)* (v1 and v2)
- TCP/IP *(on page 372)*
- HTTP *(on page 362)*
- Flat Files *(on page 351)*
- Databases *(on page 346)*
- SFTP *(on page 351)*
- FTP *(on page 351)*
- SMB *(on page 351)*
- WebDAV *(on page 351)*
- IMAP / POP3 *(on page 588)* (Email)
- SMTP *(on page 411)* (Email)
- DICOM *(on page 341)*
- JMS *(on page 366)*
- SOAP Web Services *(on page 381)*
- PDF/RTF Documents *(on page 393)*
- Custom Java and JavaScript *(on page 371)*
- ASTM E1381 *(on page 585)*
- Serial / RS-232 *(on page 597)*

How do I transform a data segment?

Use the following 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.

How can I increase the text size when using the Mirth® Connect Administrator on a high DPI monitor and the text is small and difficult to read?

In Windows, do the following:

1. Ensure you're using Java 9 or greater *(on page 55)*.
2. Navigate to your Java bin folder.
3. Right-click on **java.exe** and edit properties.
4. Go to the Compatibility tab and select **Change high DPI settings**.
5. Check the box next to **Override high DPI scaling behavior** and select **System** in the drop-down menu.
6. Open the Mirth® Connect Administrator Launcher.
7. For **Java Home**, select **Custom** and enter the path to your Java installation folder.
8. Launch the Mirth® Connect Administrator.

In Linux, do the following:

1. Install **openjdk** version 9 or greater *(on page 55)*.
2. Install **openjfx**.
    - You can run `sudo apt install openjfx` on DPKG-based distributions.
3. In a Terminal, set **GDK_SCALE** to 2.

- Use the command `export GDK_SCALE=2`

4. From the same Terminal session, launch the Mirth® Connect Administrator Launcher
    a. Navigate to the Mirth® Connect Administrator Launcher installation folder.
    b. Use the command `./launcher`.
5. For **Java Home**, select **Custom** and enter the path to your Java installation folder.
6. Launch the Mirth® Connect Administrator.

CHAPTER 16

Channel Development Best Practices and Tips

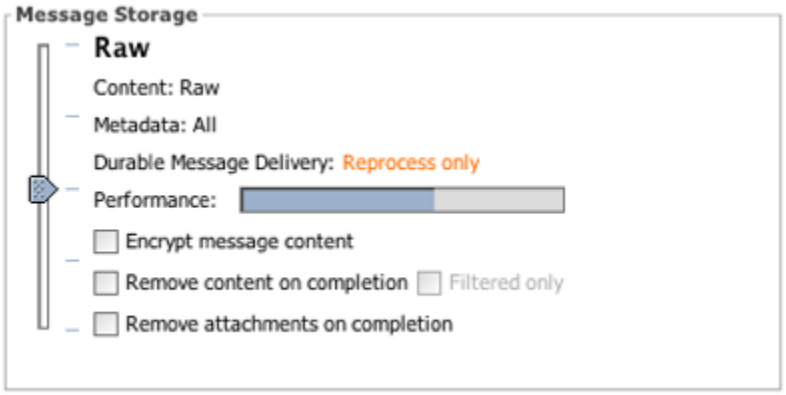
- Channel Performance *(on page 509)*
- Channel Configuration *(on page 513)*
- Other Tips *(on page 517)*

Channel Performance

- Adjust the Message Storage slider so that the channel only retains the message data that you will actually need. *(on page 510)*
- If large messages are expected, use an Attachment Handler to improve throughput and reduce memory and disk usage. *(on page 510)*
- Enable source queuing if an auto-generated acknowledgement is sufficient for the upstream system. *(on page 511)*
- Enable destination queuing if you do not need to respond to the originating system from your destination. *(on page 511)*
- 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 *(on page 511)*
- Increase the Queue Threads value if the downstream system can accept multiple concurrent connections. *(on page 512)*
- Clear the "Wait for previous destination" checkbox, unless you need destinations to process messages one after another. *(on page 512)*
- Use the "Destination Set Filter" feature in your source transformer when you need to route messages to only one (or a subset) of destinations in your channel. *(on page 512)*

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 Mirth® Connect'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 to 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 do not 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.

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.

Clear the "Wait for previous destination" checkbox, unless you need destinations to process messages one after another.

Clearing 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.

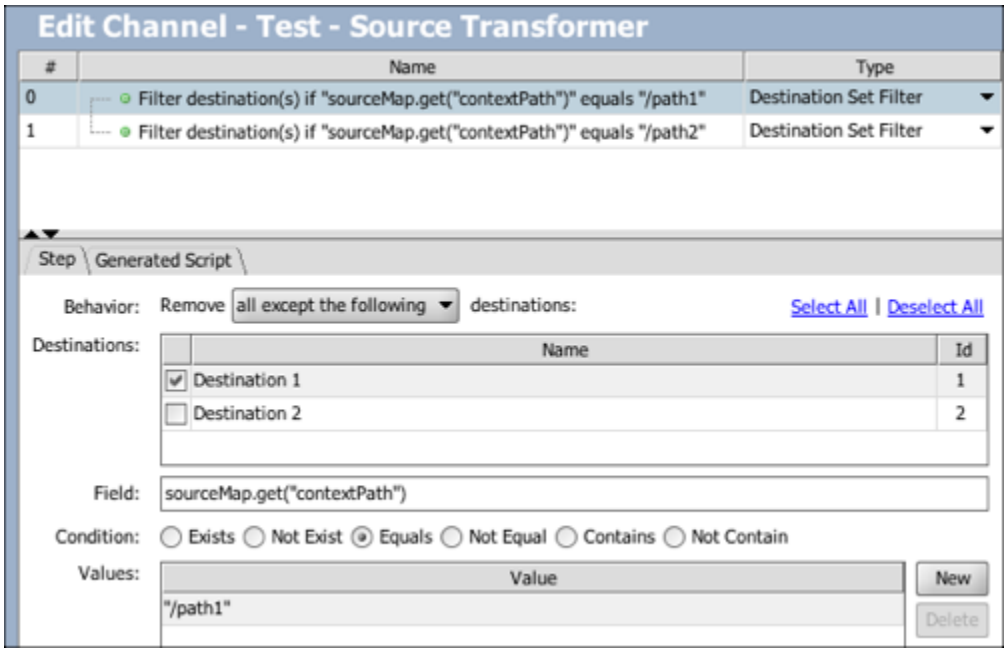
Use the "Destination Set Filter" feature in your source transformer when you need to route messages to only one (or a subset) of destinations in your channel.

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:



Channel Configuration

- Use tags to categorize your channels. *(on page 514)*
- Place any reusable JavaScript code into Code Template functions and organize them into Code Template Libraries. *(on page 514)*
- Use Custom Meta Data Columns to increase the performance of searching the message history on a particular message field. *(on page 514)*
- Use Deploy/Start Dependencies if one channel depends on another to operate correctly. *(on page 515)*
- 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. *(on page 516)*

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 nextgen.com or the Mirth® Connect What's New page for more information.

Place any reusable JavaScript code into Code Template functions and organize them into Code Template Libraries.

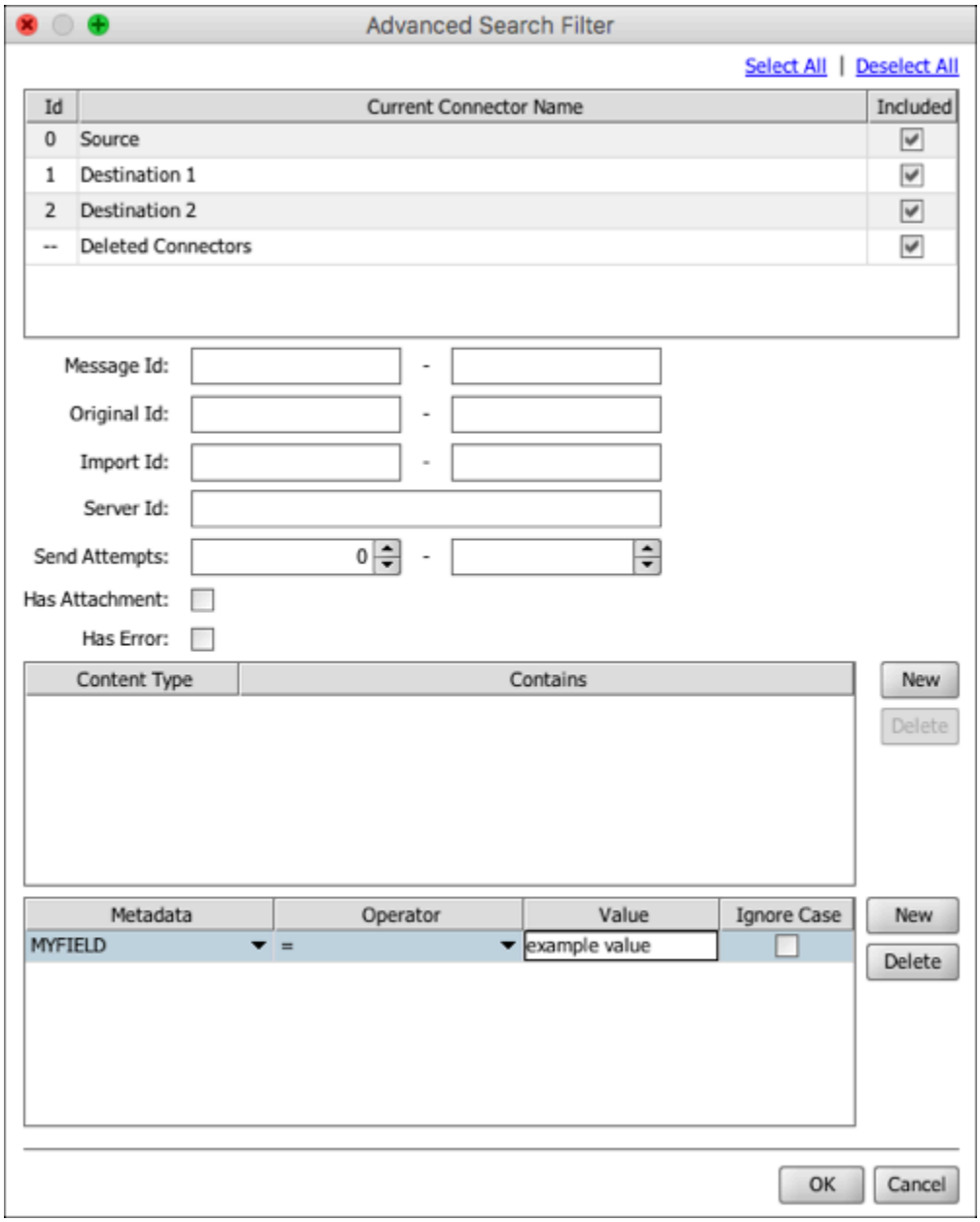
This prevents code duplication and keeps your JavaScript code from being scattered across your channels. Put reusable code into Code Templates and keep only channel-specific code in your channels. Code template libraries can also be enabled for only specific channels, to prevent namespace clashes and 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.

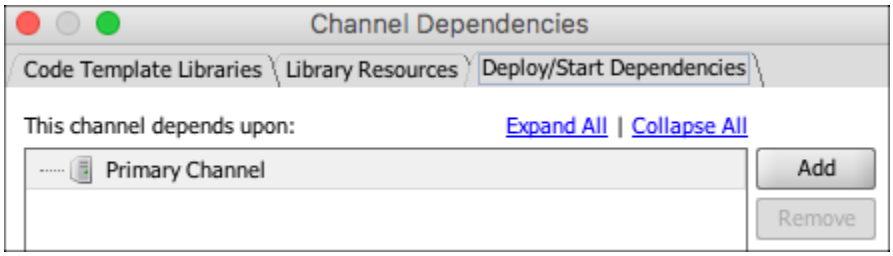
If there is a particular element of data from your messages that you may need to search by later, configure the channel to map the data into a custom meta data field. Mirth Connect will then index this data to improve search performance when searching for particular values. To do this, first create a 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, select "Advanced..." to open the Advanced Search Filter. Then add search criteria for the custom field in the table

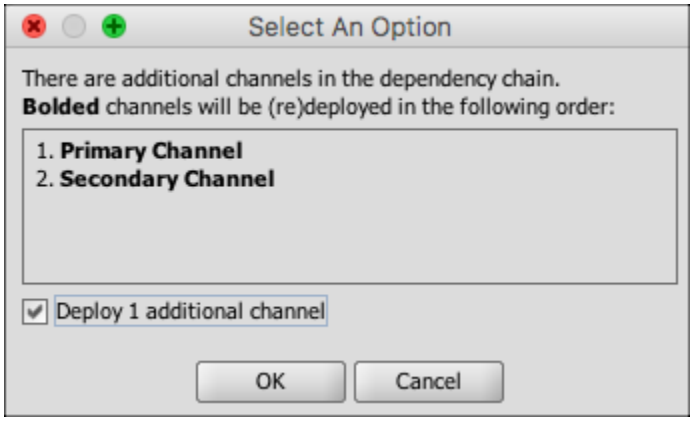


Use Deploy/Start Dependencies if one channel depends on another to operate correctly.

This can be configured in the Channel Summary view > Set Dependencies > Deploy/Start Dependencies.



By defining channel dependencies, you can protect yourself against deploying a channel without deploying another channel that it depends on to operate correctly. Mirth® Connect 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.

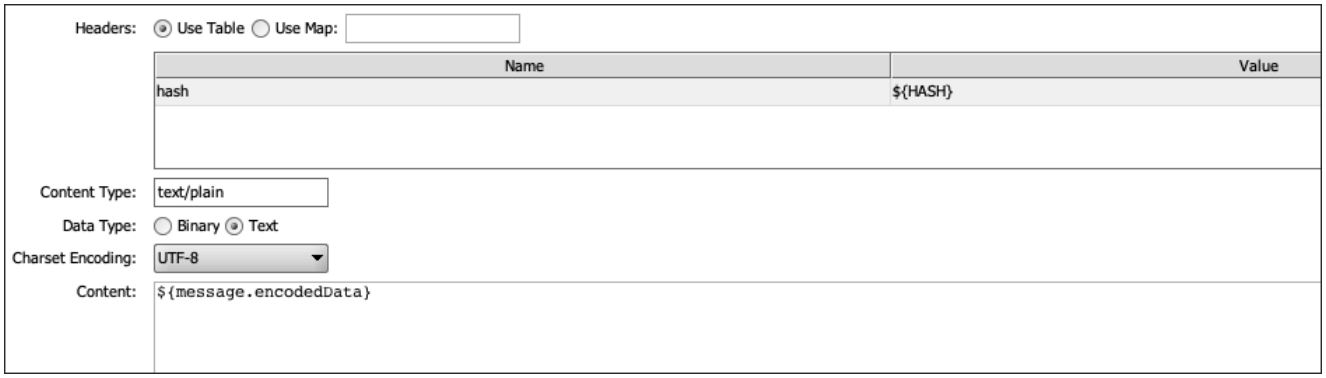
Message Integrity

Use hashing to ensure message integrity *(on page 517)*

Use hashing to ensure message integrity

A SHA-256 hash is automatically generated for each outgoing message from a destination connector. See the **Message Hash**variable on Standard Variables/Templates *(on page 253)*. Using that variable, the hash value can be sent to the destination alongside the message content. The recipient can use the hash to validate that the message hasn't been altered.

An example use case is to include the hash value in a header using an HTTP Sender.



To validate the integrity of incoming messages, you can use JavaScript code in a Source Filter. For example, using an HTTP Receiver and assuming the hash value is being sent in a header named **hash**, you can use the following code in a JavaScript Filter step:

Validate Message Hash

```
// Get the received hash from the request header

var receivedHash = sourceMap.get('headers').getHeader('hash');

// Generate a hash from the received message content

var generatedHash = HashUtil.generate(connectorMessage.getRawData(), 'UTF-8',
'SHA-256');

// Accept the message only if the hashes match

return receivedHash != null && generatedHash != null && generatedHash == receivedHash;
```

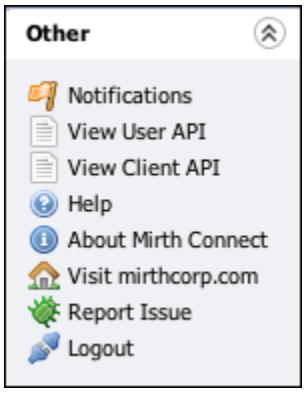
Other Tips

Do not reference Mirth Connect internal Java classes in your JavaScript code. *(on page 518)*

Do not reference Mirth Connect 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 Mirth® Connect 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 Mirth® Connect. Future versions of Mirth® Connect may deprecate certain functions, in which case a warning message will appear in your server log.

CHAPTER 17

Security Best Practices

- Secure Installation and Deployment *(on page 519)*
- Secure Configuration *(on page 525)*

Secure Installation and Deployment

- Installation Directory *(on page 519)*
- Networking *(on page 520)*
- Web Server Certificate *(on page 521)*

Installation Directory

In general, the **principle of least privilege** should be followed, and only Administrators should have access to the installation directory that Mirth® Connect is installed into. If installed on a dedicated machine, that machine should only be accessible to said administrators. If installed on a shared machine, it should be done in such a way that the Connect installation directory is only accessible by administrators.

Notes on specific folders inside the installation directory:

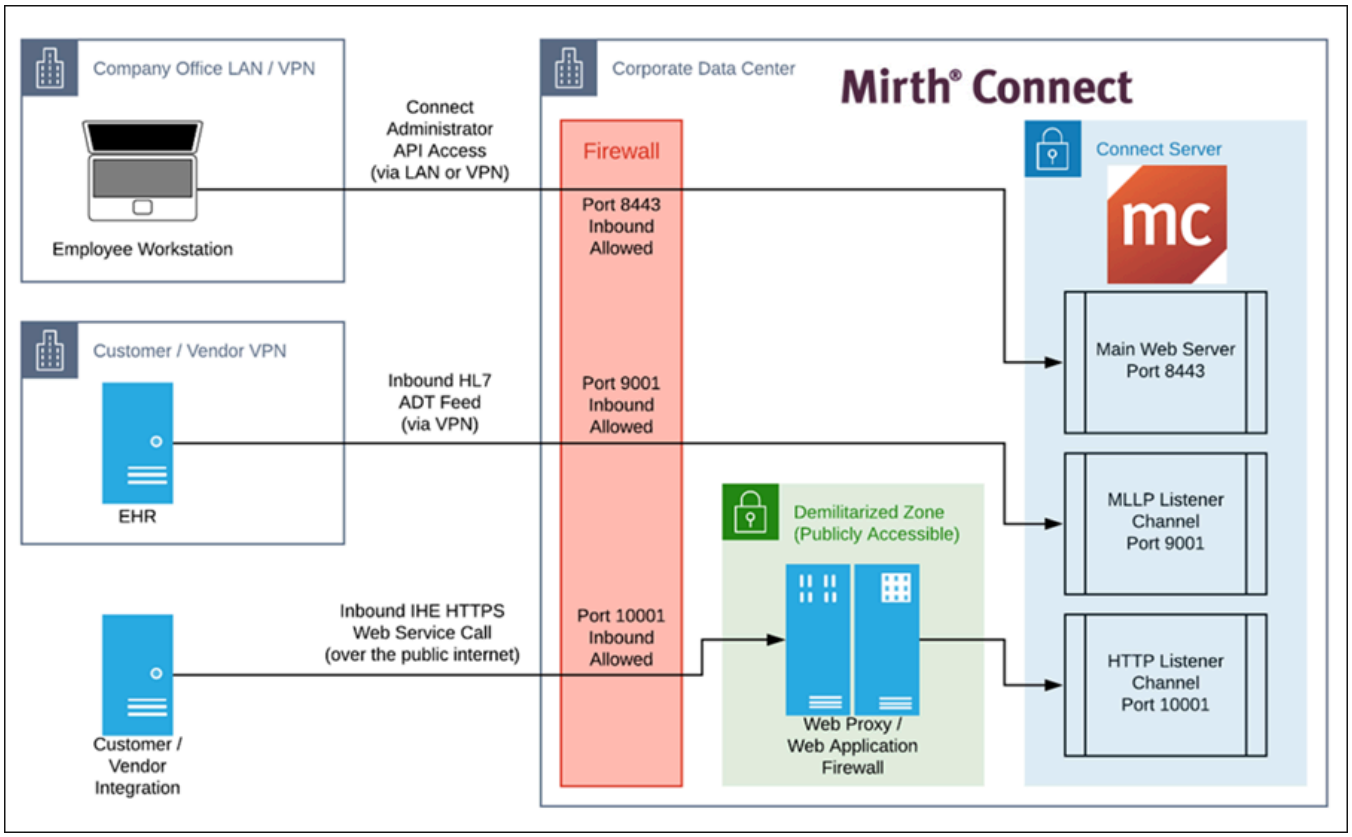
- **appdata:** The Application Data Directory *(on page 478)* stores configuration and temporary files used by Mirth® Connect during runtime. Take note of:
    - **configuration.properties:** By default this file is used to store the current state of the Configuration Map *(on page 182)*. However in mirth propertie *(on page 485)*s you can choose to store the configuration map in the database, or you can select a different file location. For example if you have a separate network storage location specifically for secure files, you could choose to store the configuration map file there instead.
    - **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. Any messages/channels/etc that are encrypted by Mirth® Connect use the encryption key stored in this file, so if this file is lost, those encrypted messages will be **irrecoverable**. The location of this file can be configured in the mirth.properties File *(on page 485)*. This file is encrypted, and the store-password / key-password are set in as well.

- **conf:** The Configuration Directory *(on page 480)* contains the main configuration files Mirth® Connect needs to start up correctly. Take note of:
    - **mirth.properties:** This is the main configuration file for Mirth Connect. It contains sensitive information such as your database username/password and the storepass/keypass for the keystore. You can change the location of various files and folders, such as the appdata folder or the keystore.jks file

Networking

By it is very nature, Mirth® Connect is an application that typically has a lot of inbound and outbound connections. It is up to you (or your IT staff) to decide how lenient or strict you want your firewall to be. Typically all inbound access is denied by default, and specific IP/port rules are added for specific channels. Outbound connectivity is less of a concern, and typically all outbound connections are allowed.



By default, Mirth® Connect listens on two ports: 8080 and 8443. These ports are used by the web server that serves up the main launch page, as well as API access (which the Administrator GUI and CLI use). The specific ports used can be configured in mirth.properties:

- http.port
- https.port

You can disable plain HTTP altogether by simply commenting out the "http.port" entry. This ensures that only secure HTTP (HTTPS) is used for all main web server traffic.

Web Server Certificate

The certificate used for the main web server is stored in the **keystore.jks** file. If not present, Mirth® Connect will automatically create a new certificate upon startup. Since this auto-generated certificate is self-signed, you will see security warnings in your browser if you navigate to the HTTPS landing page.

You can replace this auto-generated certificate with your own company cert. This could still be self-

signed or signed by an internal service if API/HTTPS access will only occur on your private company network. If you expose the API/HTTPS traffic over a publicly accessible DNS, you should replace this certificate with one that is signed by a CA (certificate authority).

For instructions on how to replace the web server certificate, look here: Application Data Directory *(on page 478)*

Connect to Database using SSL/TLS

By default, Mirth® Connect is configured to connect to an embedded Apache Derby database for quick deployment, development, and testing. When using Mirth® Connect 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+

There is more information on changing the database type here:

- Changing the Database Type *(on page 49)*

And more information about configurable options in mirth.properties here:

- mirth.properties File *(on page 485)*

Typically these database connections are not encrypted by default. The instructions to enable SSL/TLS traffic for database connections differ depending on the database and JDBC driver you are using. Here are some instructions for each of the supported backend databases. For more information, consult the manuals for the specific database and JDBC driver you are using.

Importing the Database Server Certificate

If the certificate your database is using is self-signed or not signed by a common trusted Certificate Authority (CA), then you will likely need to import that certificate into your default Java truststore in order to connect to your database using SSL/TLS.

Your default Java truststore is typically the "cacerts" JKS file that usually resides inside your Java installation folder under "jre/lib/security", and the default password is "changeit".

You can also override the default truststore by supplying the "-Djavax.net.ssl.trustStore" JVM option into the mcserver.vmoptions or mcservice.vmoptions files.

Using keytool:

```
keytool -importcert -keystore /path/to/javahome/jre/lib/security/cacerts -storepass
changeit -alias myalias -file /path/to/cert/server.crt -noprompt
```

Make sure to change the paths and alias above as needed. Depending on your permissions you may need to perform this as sudo/Administrator to edit the Java cacerts truststore.

You can also use a GUI tool like Portecle to import the cert.

PostgreSQL

The simplest way to enable SSL/TLS traffic is to add "ssl=true" to your JDBC URL. For example:

```
jdbc:postgresql://localhost:5432/mirthdb?ssl=true
```

There is also an "sslmode" property you can use for more granular options:

```
jdbc:postgresql://localhost:5432/mirthdb?sslmode=verify-full
```

And many other options as well. Consult these online documentation pages for more information:

- https://www.postgresql.org/docs/current/ssl-tcp.html
- https://jdbc.postgresql.org/documentation/head/ssl.html
- https://jdbc.postgresql.org/documentation/head/connect.html#ssl

MySQL

By default, the MySQL Connector/J JDBC driver will automatically connect via SSL/TLS if the server supports it. To force encrypted traffic at all times, use the sslMode parameter:

```
jdbc:mysql://localhost:3306/mirthdb?sslMode=VERIFY_IDENTITY
```

The above mode makes sure that SSL/TLS is always used, the server cert is trusted, and hostname verification is also performed.

For a more lenient and less secure connection (but still using SSL/TLS), you can also use the VERIFY_CA option which skips the hostname verification, or the REQUIRED option which doesn't require the server cert to be trusted and will work for self-signed certs.

Consult these online documentation pages for more information:

- https://dev.mysql.com/doc/refman/8.0/en/using-encrypted-connections.html
- https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-configuration-properties.html

Oracle

By default SSL/TLS is not used when connecting to Oracle via the JDBC thin client. To enable it, use the protocol TCPS in the expanded URL syntax:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SE
```

Note that the Oracle server must first be configured to support that TCPS protocol as well. Consult these online documentation pages for more information:

- https://docs.oracle.com/database/121/DBSEG/asossl.htm#DBSEG074
- https://docs.oracle.com/en/database/oracle/oracle-database/20/jjdbc/client-side-security.html#GUID-2BD2F189-A58C-4A85-8524-CFD9BB9AC575

SQL Server

Using the built-in jTDS JDBC driver, you can enable SSL/TLS by using the "ssl" option:

```
jdbc:jtds:sqlserver://localhost:1433/mirthdb?ssl=authenticate
```

The above option will require SSL/TLS traffic and require that the server certificate is trusted. If you want to accept self-signed certs, you can reduce the security level and use "ssl=require" instead.

Using the Microsoft JDBC driver, you can enable SSL/TLS by using the "encrypt" option:

```
jdbc:sqlserver://localhost:1433;databaseName=mirthdb;encrypt=true;trustServerCertificate=true
```

Consult these online documentation pages for more information:

- http://jtds.sourceforge.net/faq.html
- https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine
- https://docs.microsoft.com/en-us/sql/connect/jdbc/using-ssl-encryption

Secure Configuration

- Encryption Settings *(on page 525)*
- Encrypt Database Password *(on page 527)*
- Plain HTTP Main Web Server *(on page 527)*
- Default TLS/SSL Settings *(on page 527)*
- Default Supported Cipher Suites *(on page 528)*
- Password Requirements *(on page 532)*
- SSL Manager Extension *(on page 533)*

Encryption Settings

You can change the default encryption settings for Mirth® Connect as you see fit. The following can be set in mirth.properties:

Property	Default Value	Description
encryption.algorithm	AES/CBC/PKCS5Padding	The algorithm to use for symmetric encryption. This applies to messages, exports, and anything that is used along with the keystore to encrypt / decrypt. You must include the explicit

Property	Default Value	Description
		mode and padding settings with the algorithm. The mode must also require an initialization vector. **Note:** The default value is updated in version 4.3. • If you did not explicitly set this property, then no action is needed. Mirth® Connect will automatically start using the new default. • If you did explicitly set this property, then NextGen Healthcare highly recommends you to update it to include the mode/padding options as well. • Support for "AES" (without any mode/padding specified) will be removed in a future version.
encryption.charset	UTF-8	The charset to use when encoding textual data into bytes before encryption.
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

Property	Default Value	Description
		password hashing.

Encrypt Database Password

To automatically encrypt the database password using the encryption key stored in the keystore, set this in mirth.properties:

- encryption.properties = 1

Plain HTTP Main Web Server

By default the main web server listens on both plain HTTP and secure HTTPS ports. If you only want to listen on the secure port and not allow plain HTTP at all, comment out this setting in mirth.properties:

- http.port

Default TLS/SSL Settings

The following properties can be set to modify the default TLS settings:

Property	Default Value	Description
https.client.protocols	TLSv1.3,TLSv1.2	The protocols to support by default for all TLS/SSL/HTTPS client traffic. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager *(on page 598)* enables you to adjust protocol settings on a per-connector basis, rather than having to change the value for the entire server. **Note:** TLSv1.3 is only available when running Mirth Connect on Java 11 or later.

Property	Default Value	Description
https.server.protocols	TLSv1.3,TLSv1.2,SSLv2Hello	The protocols to support by default for all TLS/SSL/HTTPS server traffic. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager *(on page 598)* enables you to adjust protocol settings on a per-connector basis, rather than having to change the value for the entire server. **Note:** TLSv1.3 is only available when running Mirth Connect on Java 11 or later.
https.ciphersuites	See Default Supported Cipher Suites *(on page 528)*	The cipher suites to support by default for all TLS/SSL/HTTPS server traffic. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks. **Note:** The SSL Manager *(on page 598)* enables you to adjust protocol 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. **CAUTION:** Changing this property could leave your server vulnerable to certain SSL-based attacks.

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_EMPTY_RENEGOTIATION_INFO_SCSV

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.

Cipher Suites Removed From Earlier Versions

The following cipher suites were removed from mirth.properties starting in version 3.5 to address vulnerabilities like SWEET32. Upon upgrade the cipher suites should automatically be updated to remove any that use TripleDES. Check your mirth.properties file to make sure that they are not

included:

- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

The following cipher suites were removed from mirth.properties starting in version 4.0. Check your mirth.properties file to make sure they are not included:

- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

The following cipher suites were removed from mirth.properties starting in version 4.3. Check your mirth.properties file to make sure they are not included:

- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384

- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_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_DSS_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA

If you need to use one of the removed cipher suites on a specific connection, the SSL Manager Extension *(on page 533)* extension enables you to modify protocols / cipher suites per connector. As a result, your overall server settings remain secure.

New Protocol/Cipher Suite Support in Java 11

Java 11 added support for **TLS v1.3**. This should be added to your supported protocols list in mirth.properties when you upgrade to version 3.7 or later. If you do not want to support TLS v1.3, you can remove that from the list.

Java 11 also added support for new and more secure cipher suites. Upon upgrade to version 3.7 or later they should be automatically added. Check your mirth.properties file for these cipher suites:

- 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

Password Requirements

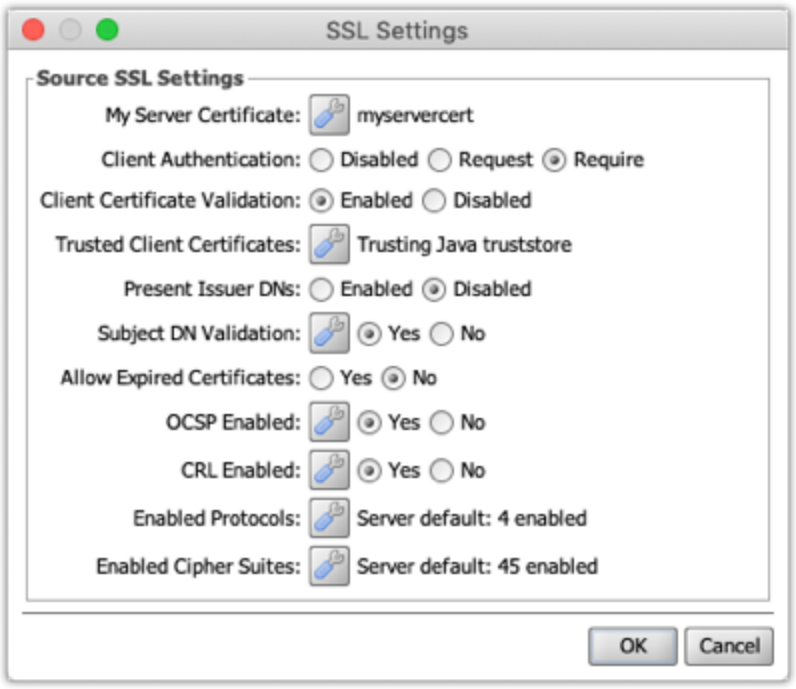
The following properties can be set in mirth.properties to enforce requirements on user passwords:

Property	Default Value	Description
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.minnumeric	0	Minimum numeric characters, 0 for no minimum.
password.minspecial	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.

SSL Manager Extension

This section only applies if you have the SSL Manager *(on page 598)* extension installed. This guide will summarize the available settings, but there is a separate user guide specifically written for the SSL Manager that goes into depth about all the available features. Contact our help desk to obtain a copy of the SSL Manager guide.

Source Connector Settings

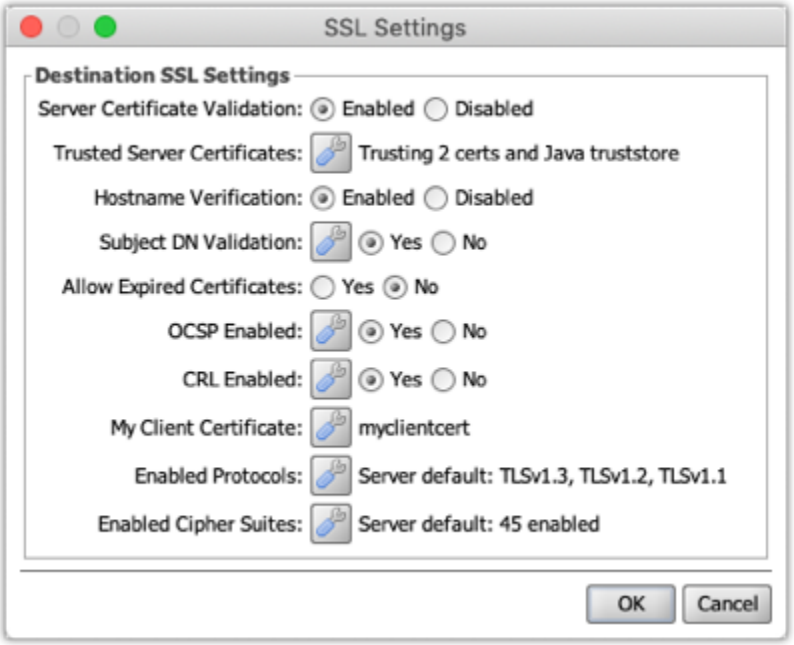


- **Client Authentication**: Client authentication (sometimes called mutual or bi-lateral authentication) can provide additional security as both the client **and** server must present certificates that the other can choose to trust or not. The **Trusted Client Certificates**section determines which client certs (or intermediate/root certs) to trust.
    - **Present Issuer DNs:** This is only applicable when Client Authentication is used. If enabled, during the SSL handshake the server will respond with a list of accepted client issuer distinguished names (DNs). Disabling this can provide extra security as potential attackers are given less information about the correct security parameters.
    - **Subject DN Validation:** This is only applicable when Client Authentication is used. If enabled, only client certificates with subject distinguished names (DNs) matching the given list will be allowed. If a client certificate not matching any of the trusted DNs is

presented, the SSL connection / handshake will fail.

- **Allow Expired Certificates:** The default value for this is **No**, meaning that when an expired certificate is encountered, the SSL handshake will fail. It is recommended not to enable this unless you need to for legacy purposes or for development/testing.
- **OCSP Enabled:** Select **Yes** to enable Online Certificate Server Protocol (OCSP) checking for all local and remote certificates. The issuer of the response certificate must be trusted as well in order to verify signatures.
- **CRL Enabled:** Select Yes to enable Certificate Revocation List (CRL) checking for all local and remote certificates. The issuer of the CRL must be included in your trusted certificates as well in order to verify signatures.
- **Protocols / Cipher Suites:** The server defaults (set in mirth.properties) are used by default here. However you can choose to override this and enable/disable certain protocols or cipher suites here. For example, you may need to enable a less secure cipher suite in order to communicate with an external legacy system. Any settings you override here will *only* affect this connector, not any other connector or the overall server settings. So you can still allow less secure settings for a particular connection without affecting anything else on your Mirth Connect server.

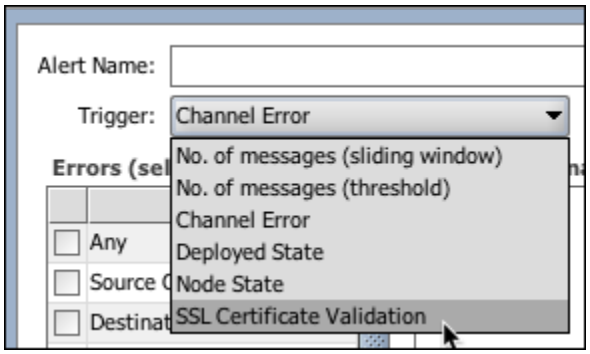
Destination Connector Settings



- **Trusted Server Certificates:** This section determines which server certs (or intermediate/root certs) to trust.
- **Hostname Validation:** If enabled, validation will fail if the Subject CN (or Subject Alternative Name) presented in the server certificate does not match the actual endpoint the connector is dispatching to. If disabled, valid certificates will be accepted even if the host name does not match.
- **Subject DN Validation:** If enabled, only server certificates with subject distinguished names (DNs) matching the given list will be enabled. If a server certificate not matching any of the trusted DNs is presented, the SSL connection / handshake will fail.
- **Allow Expired Certificates:** The default value for this is **No**, meaning that when an expired certificate is encountered, the SSL handshake will fail. It is recommended not to enable this unless you need to for legacy purposes or for development/testing.
- **OCSP Enabled:** Select **Yes** to enable Online Certificate Server Protocol (OCSP) checking for all local and remote certificates. The issuer of the response certificate must be trusted as well in order to verify signatures.
- **CRL Enabled:** Select **Yes** to enable Certificate Revocation List (CRL) checking for all local and remote certificates. The issuer of the CRL must be included in your trusted certificates as well in order to verify signatures.
- **My Client Certificate:** Client authentication (sometimes called mutual or bi-lateral authentication) can provide additional security as both the client **and** server must present certificates that the other can choose to trust or not. To enable this, simply provide a client cert for this setting and it will be presented to the server. It is up to the server to validate or ignore the client cert.
- **Protocols / Cipher Suites:** The server defaults (set in mirth.properties) are used by default here. However you can choose to override this and enable/disable certain protocols or cipher suites here. For example, you may need to enable a less secure cipher suite in order to communicate with an external legacy system. Any settings you override here will *only* affect this connector, not any other connector or the overall server settings. So you can still allow less secure settings for a particular connection without affecting anything else on your Connect server.

Advanced Alerting - SSL Manager Trigger

If the Advanced Alerting *(on page 583)* extension is installed along with the SSL Manager, there will be a new trigger type available:



With this new type you can setup alerts that trigger when a certificate has expired or been revoked, or when it has been rejected by a connector because of any Subject DN Validation settings you have configured.

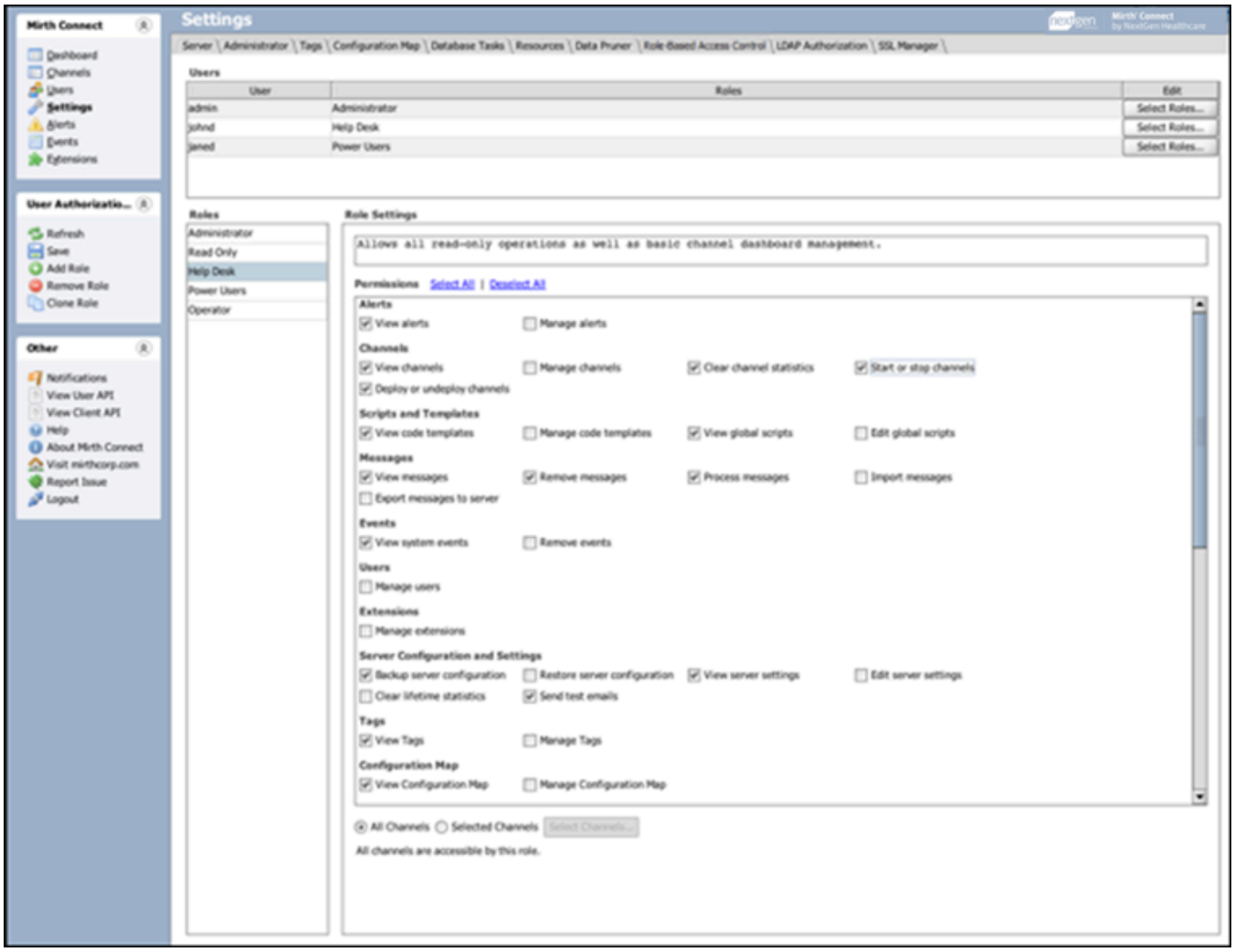
Operational Procedures

- Users/Permissions *(on page 536)*
- Auditing *(on page 537)*

Users/Permissions

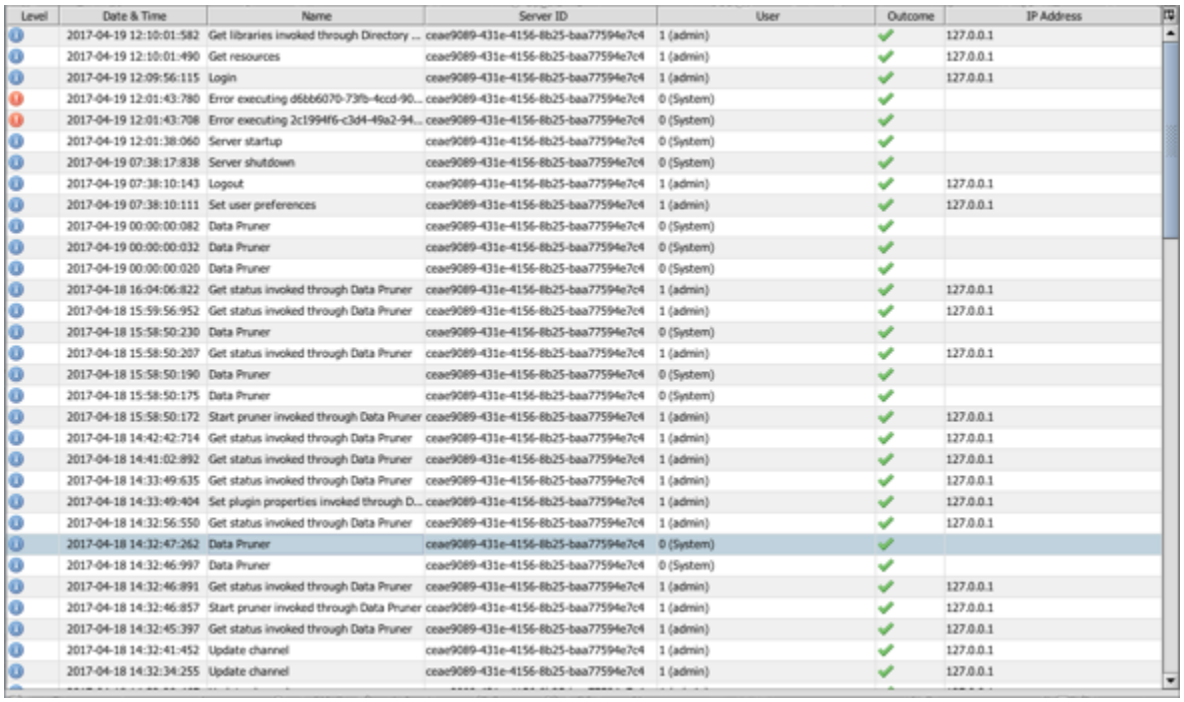
It is a good practice to periodically review your list of users and remove any accounts that are old or are no longer in use.

If you have the Role-Based Access Control *(on page 599)* extension installed, you can also review privileges for each role and the roles assigned to each user.



Auditing

Actions performed by users in the Administrator GUI (or using the API) are recorded as Events. For each event you can view the timestamp, user account (if applicable), and IP address (if applicable) from which the request originated. You can also search the events table for specific event types, users, or IP addresses.



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

Environmental Upkeep

Not only is it important to properly configure Mirth® Connect, but it is also critical to keep all aspects of your environment secure as well.

- Mirth Connect Software *(on page 538)*
- Operating System *(on page 539)*
- Java *(on page 539)*

Mirth Connect Software

Mirth® Connect makes use of many libraries for various purposes and occasionally it is discovered that one or more of these libraries contains a security vulnerability. These vulnerabilities are dealt with either by updating the library, patching them in-house, or eliminating the library altogether. Once work has been completed, a new Mirth® Connect version will be released containing updates to mitigate the vulnerability so it is important to keep your version of Mirth® Connect up to date. When logging in to the Mirth Connect Administrator you will receive a message when a new version is available. If you are using a Mirth Appliance, follow the usual steps in the Control Panel to install an update, otherwise use

one of the installers to install an update.

Operating System

Every operating system receives security updates occasionally. it is best to keep up with these updates as they can affect any software, including Connect, installed on the system. If you are using a Mirth Appliance, follow the usual steps in the Control Panel to install Control Panel updates, otherwise use your operating system's updater to keep up with security updates.

Java

Today there are many Java distributions and they usually get frequent security updates. Since Connect runs in a Java Virtual Machine it is highly recommended to keep up to date on your Java security updates to ensure Connect and all data passing through it stays secure. If you are using a Mirth Appliance, follow the usual steps in the Control Panel to install an update, otherwise follow the update instructions for your particular flavor of Java.

CHAPTER 18

Troubleshooting

This section is meant to capture some of the most common errors that may occur when using Mirth® Connect. It is by no means an exhaustive list. For additional help, you may be interested in our Commercial Support/Extensions *(on page 582)* or Training *(on page 602)* offerings.

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 *(on page 478)*. The "mirth.log" file is the most recent log.

Configuration

Often times issues occur because of mistakes in initial configuration. The main configuration for Mirth® Connect by NextGen Healthcare is done in the mirth.properties file *(on page 485)* in the "conf" folder inside your Installation Directory *(on page 478)*.

Mirth Connect Engine fails to start up

- A common cause of this is port conflicts. Check the Logs *(on page 540)*, you may see something like this:
    - ```
      ERROR 2017-06-23 09:17:22,757 [Main Server Thread] com.mir
      th.connect.server.Mirth: http.port port is already in use:
      8080
      ERROR 2017-06-23 09:17:22,758 [Main Server Thread] com.mir
      th.connect.server.Mirth: https.port port is already in us
      e: 8443
      ```
    - Change the "http.port" and "https.port" in your Configuration *(on page 540)* 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.mir
      th.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 *(on page 540)* settings, or the machine running Mirth® Connect by NextGen Healthcaredoes 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.mir
      th.connect.server.Mirth: Failed to migrate database schema
      com.mirth.connect.model.util.MigrationException: Failed to
      execute script:
      ```

      If may mean that you have changed the "database.url" setting correctly, but forgot to also change the "database" setting. For example if you are switching from Derby to PostgreSQL, make sure to change the "database" setting to "postgres".

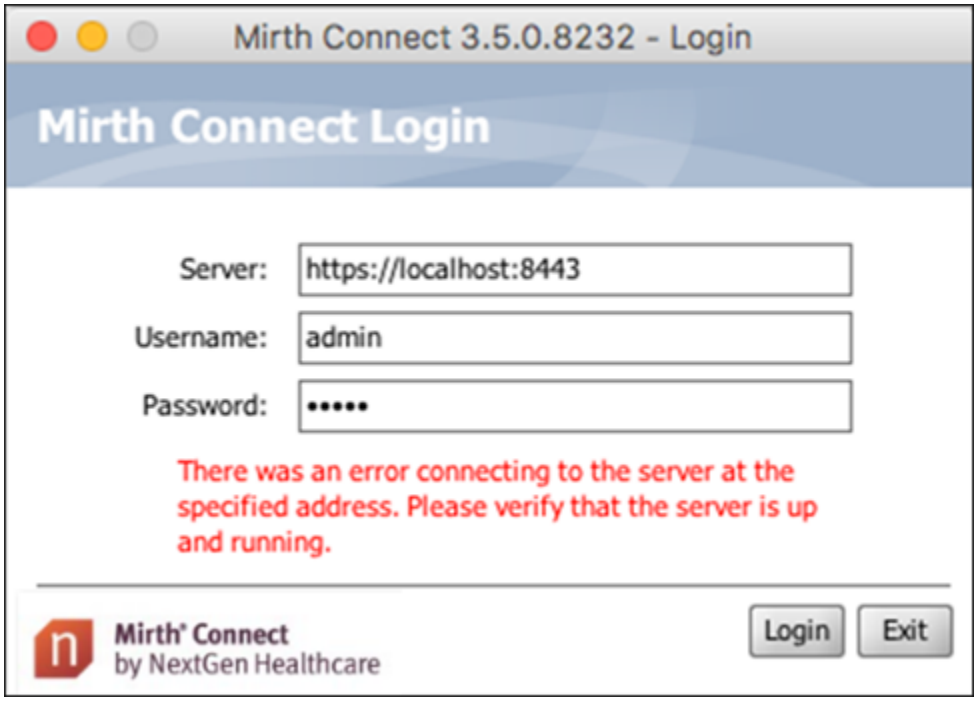
- Another possibility is that you have all the database settings correct, but the schema you are trying to connect to doesn't yet exist:

    - ```
      ERROR 2017-06-23 09:35:10,668 [Main Server Thread] com.mir
      th.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&quot; by default). Then Mirth® Connect will automatically create all the necessary tables once it starts up.
      ```

Unable to Launch Mirth Connect Administrator

The login window indicates if there is any issue:



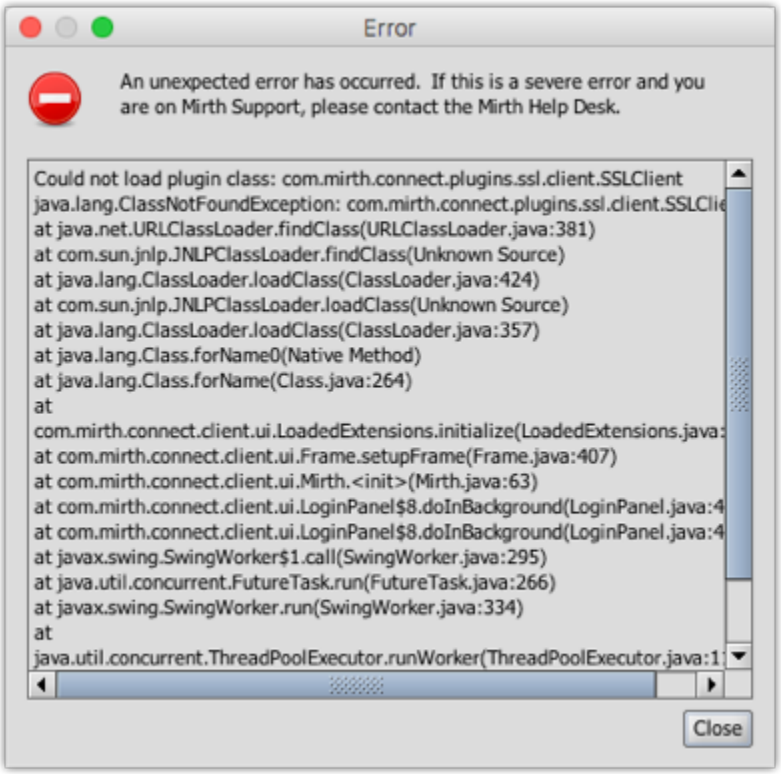
- If you do not have connectivity from your local machine to the Mirth® Connect server over the HTTPS port (default 8443), you will see the following:

  Warning: 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 are using is correct, and verify whether you have network connectivity to the server.
  - Verify that the Mirth® Connect server is actually running. If you installed as a service, you can view the current status from the Server Manager.

Clearing your Java Cache

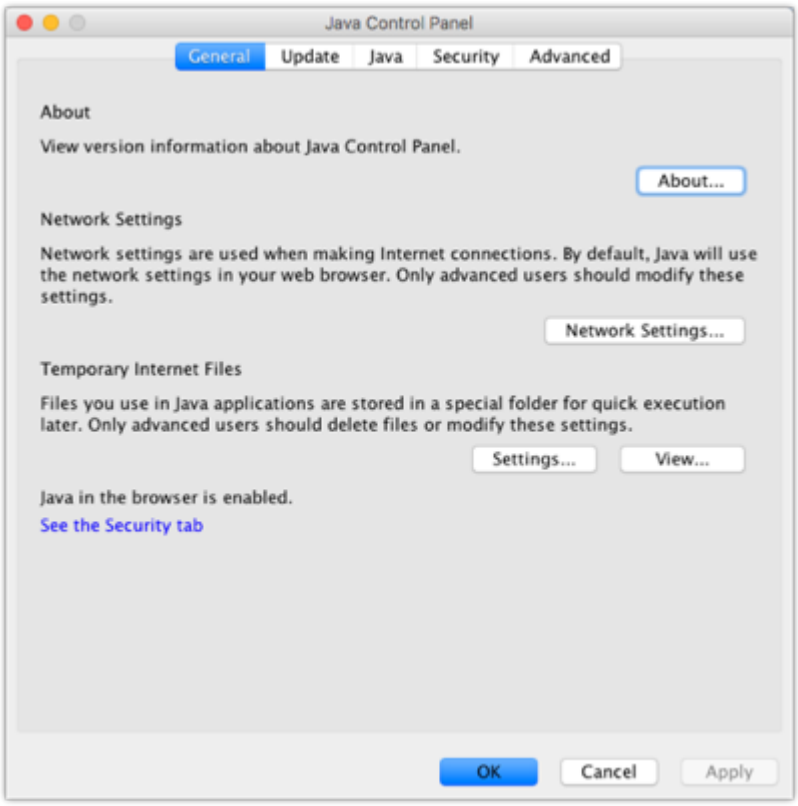
If you have recently upgraded Mirth® Connect or installed a new extension *(on page 196)*, you may find that your current Administrator shortcut no longer works:



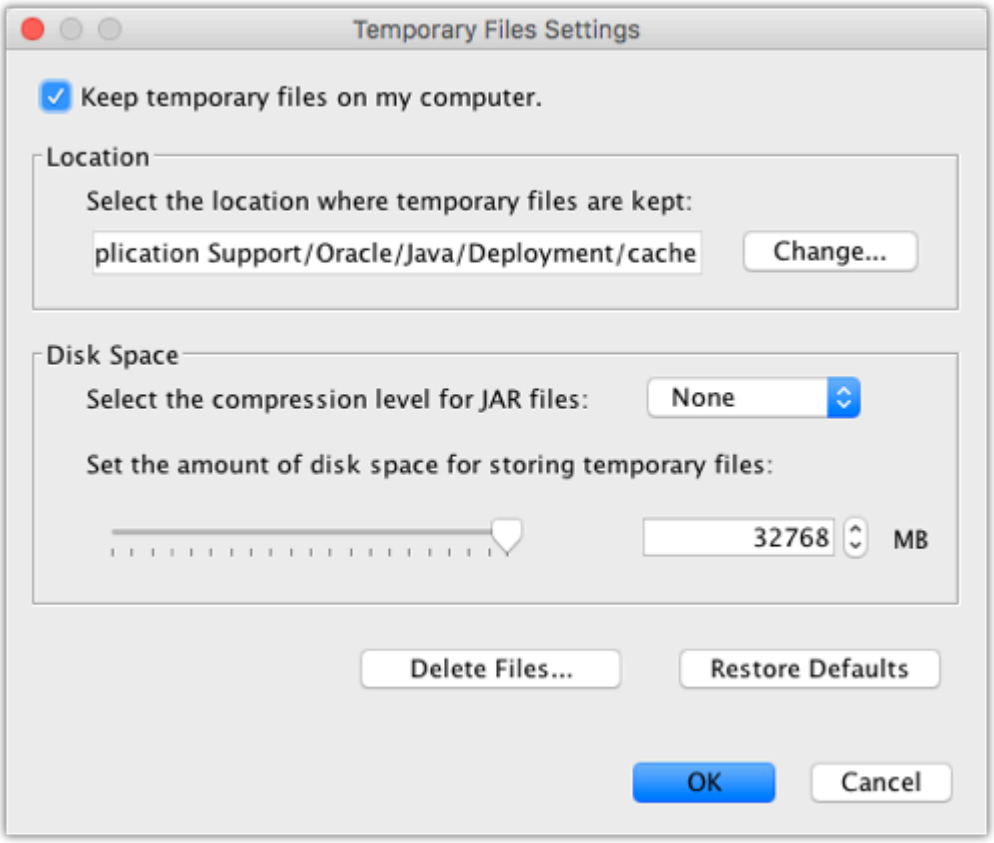
This can be resolved by clearing your Java cache. You can do that directly from the command line, or by using the Java Control Panel.

Using the Java Control Panel

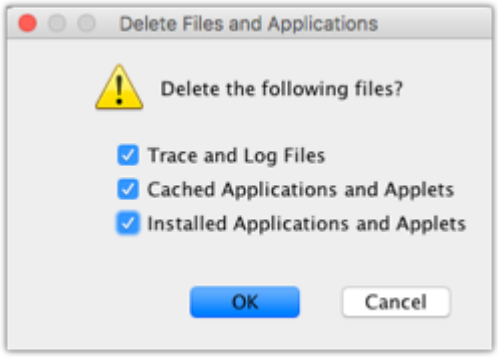
1. In your System Preferences (on OSX) or the Control Panel (on Windows), open the Java section. A new dialog will appear:



2. Under **Temporary Internet Files**, select on **Settings...**



3. Select on **Delete Files...**



4. Check all three boxes and then select **OK**.

Using the Command Line

1. Open up a terminal / command prompt, and type this:

- `javaws -uninstall`

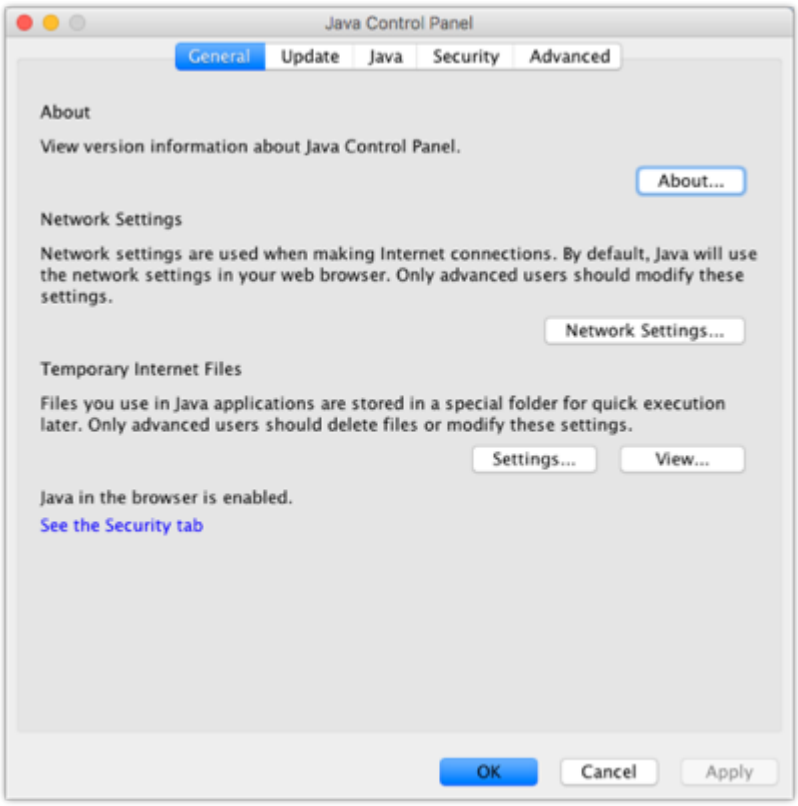
- Wait for the operation to finish. Note that if you have multiple versions of Java installed on your machine, you may have to use the **javaws** executable specific to the version you launch the Mirth® Connect Administrator with.

Opening the Java Client Console

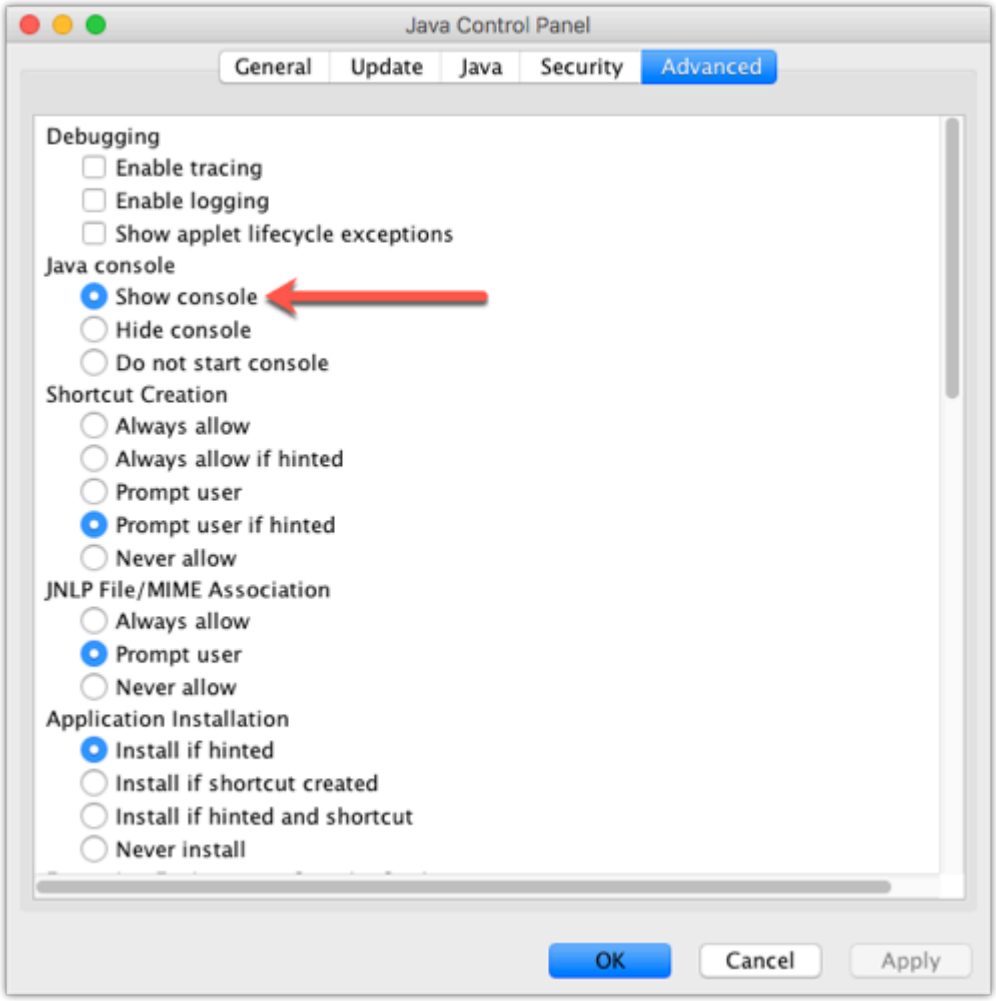
If an error occurs on the Mirth® Connect server, you will typically see one or more entries appear in the Logs *(on page 540)*. If an error occurs in the Mirth Connect Administrator *(on page 79)*, it is possible that it is isolated to the client-side only, and so nothing will appear in the server-side logs. In cases like this, it is 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:

1. In your System Preferences (on OSX) or the Control Panel (on Windows), open the Java section. A new dialog will appear:

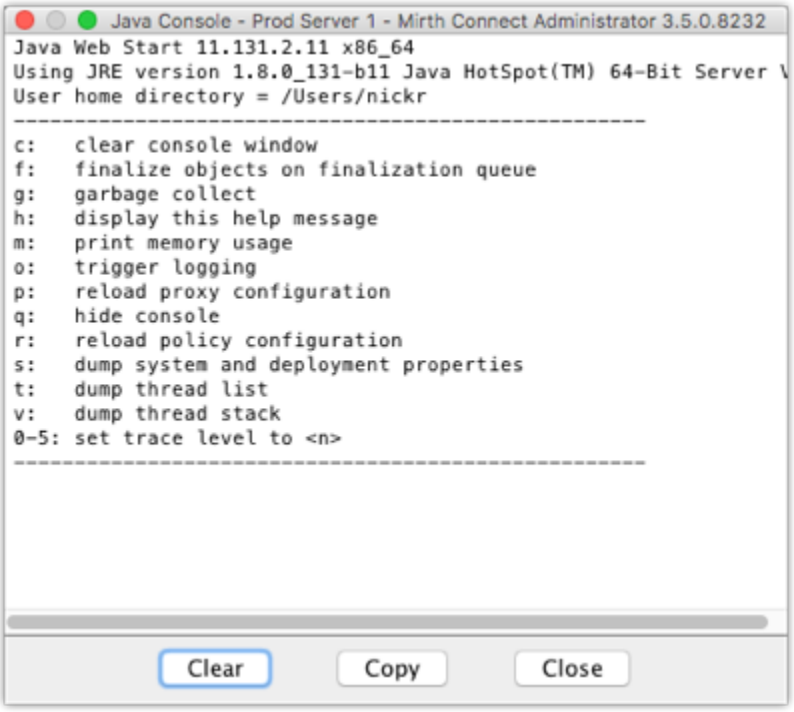


2. Select on the **Advanced** tab.
3. Select **Show console**.



4. Select **Apply**, then select **OK**.

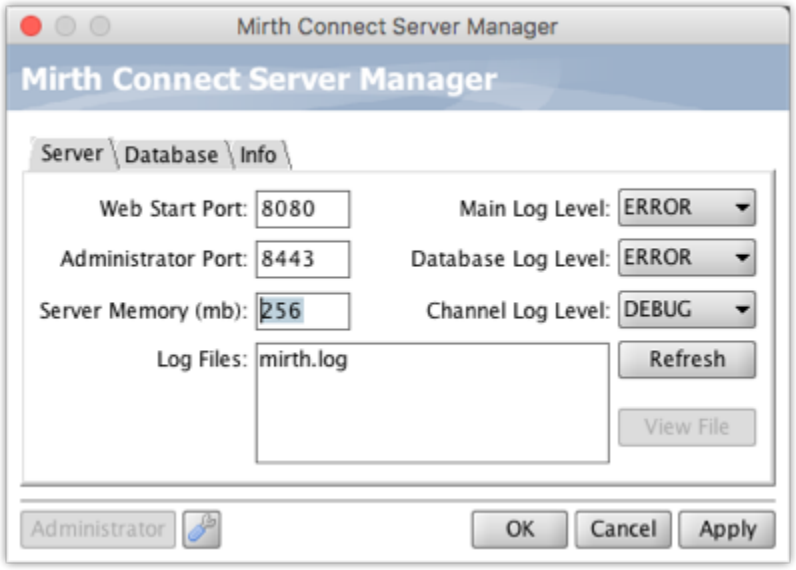
Now when you launch the Administrator, you will see a second window also pop-up:



Any exceptions that are logged to stdout / stderr on the client side will appear in that console.

Out of Memory Errors

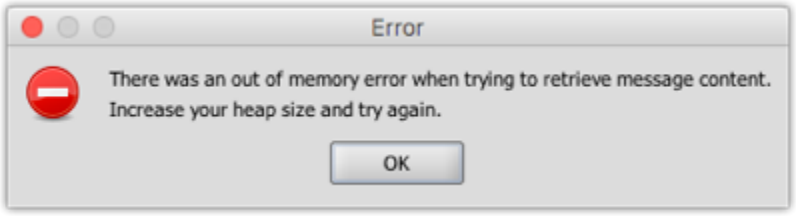
By default, the max Java heap size for the Mirth® Connect server is set to 256 MB. For large production instances you typically want to increase this value. You can do this from the Server Manager:



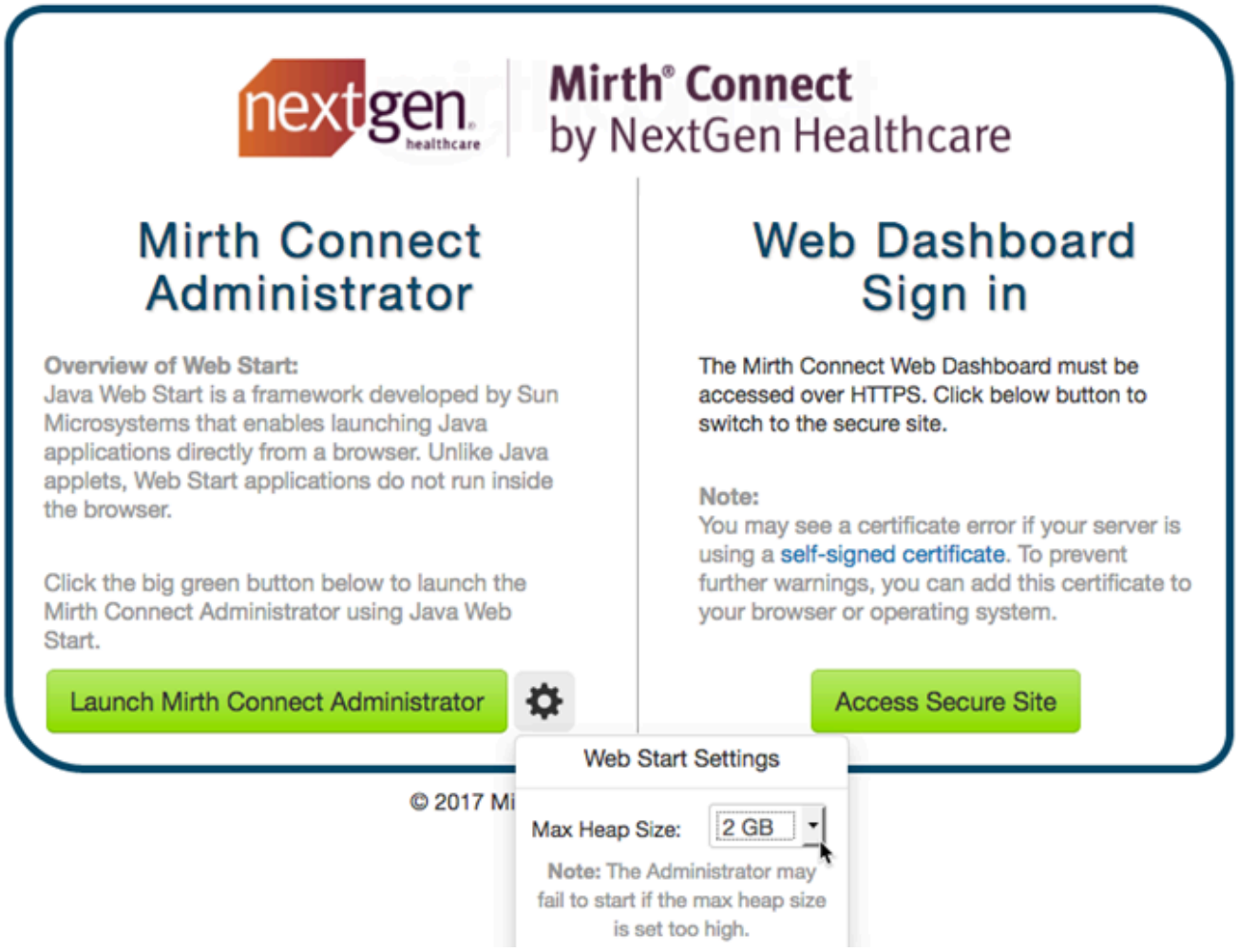
Or, you can edit the appropriate ***.vmoptions** file in your installation directory.

Note: When changing the max Java heap size, you must restart your Mirth® Connect 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:



Select the cog icon next to the **Launch** button, and change the max heap size. Then select 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.

Manually reset a password in the database

If you forgot or lost your password and cannot login, you can reset the password in the database

manually. You will need to use a third-party SQL client, either a command line tool or a graphical client like SQuirreL SQL Client.

- First locate the ID of the user you want to reset the password for.

```
SELECT * FROM PERSON;
```

- The users are in the PERSON table, and the "admin" user usually has an ID of 1:

	id [PK] integer	username character var
1	1	admin
2	5	recovery
3	6	test
*		

- Issue an UPDATE statement to the PERSON_PASSWORD table, using the ID of the user located in the previous steps. Change the question mark in the statement below to the correct ID.

```
UPDATE PERSON_PASSWORD SET PASSWORD = 'YzKZIAnbQ5m+3llggrZvNtf5f
g69yX7pAplfYg0Dngn/fESH93OktQ==' WHERE PERSON_ID = ?;
```

The above example uses a salted hash for the password "admin", assuming that the digest algorithm *(on page 485)* is still set to SHA-256. After issuing the above UPDATE statement, you should be able to login with a password of "admin". After logging in change the password as you see fit.

CHAPTER 19

Upgrade Guide

This section provides instructions for upgrading your version of Mirth® Connect and highlights important changes to the application in previous versions that you should be aware of before upgrading.

Before You Upgrade

Before upgrading, NextGen Healthcare highly recommends that you backup both your Mirth® Connect Server Configuration and your Mirth® Connect database using the following steps.

Note: `mirth_home` directory.

1. **Backup server configuration:** Before proceeding, it is important that you backup your existing server configuration. This can be done through the Mirth Connect Administrator (**Settings>Server Tab>Server Tasks – Backup Config**). The server configuration includes channels, code templates, alerts, and global scripts. User accounts and stored messages will not be backed up.
2. **Backup database:**

- If you use the embedded Derby database, create a backup of the "*mirth_home*`/appdata/mirthdb"` folder. Note that **appdata** is the database folder as defined in the "**`mirth_home/conf/mirth.properties"`** file. For additional information, see mirth.properties File *(on page 485)*.
- If you use an external database (for example, PostgreSQL, MySQL), create a backup of your database using your preferred database administration tool (for example pgAdmin, MySQL Administrator).

Changes to some custom configuration files may need to be manually copied over after the upgrade. If you have made changes to any of the following files, copy them to a location outside of your existing **mirth_home** directory before running the new Mirth Connect installer. After Mirth® Connect is upgraded, update the newly installed files with your custom changes.

1. **Custom database drivers:** If you have added custom database drivers they will be preserved in custom-lib during the upgrade. However, you will need to back up your additions to "`mirth_home/conf/dbdrivers.xml".`
2. **Java JVM parameters:** If you have added any Java JVM parameters or changed any of the default heap size settings, create a backup of any "`mirth_home/*.vmoptions"` files
3. **CLI configuration:** If you have made any changes to the CLI configuration file, create a backup of "`mirth_home/conf/mirth-cli-config.properties".`:

Note: Custom extensions may not work after the upgrade and you may see errors in the log. You will need to install an updated extension that is compatible with your version of Mirth® Connect.

Upgrade Mirth Connect

Note: For Linux users upgrading from 4.3.0, new URL links have been added to the Mirth® Connect windows of the setup wizard to enable navigating to the desired version from the main document/wiki pages. To get the updated links, you must uninstall and then re-install Mirth® Connect.

1. Make sure that you have exited and stopped the **Mirth Connect Administrator**, **Server Manager**, and **Server/Service** before proceeding.
2. Run and complete the new Mirth® Connect installer, choosing the update option using the same installation directory.

After the Upgrade

1. **Java JVM parameters:** Restore the changes in your backed up "`*.vmoptions`" files to the corresponding `*.vmoptions` files in "***mirth_home***".
2. **CLI configuration:** Restore the changes in your backed up "`mirth_home/conf/mirth-cli-config.properties`" file.
3. **Start/Restart the server**.
4. **Custom extensions:** Install new versions of any custom extensions.

Version-Specific Upgrade Instructions

If you are upgrading to one of the versions listed in the following sections, view the version-specific upgrade instructions. If you are upgrading through multiple versions, be sure to review all version upgrade instructions that you are upgrading through.

4.5.0 Upgrade Notes

Updated DUO Authentication to use the Universal Prompt

We have updated the DUO Authentication to use the new Universal Prompt because the Traditional Prompt will no longer be accessible after March 30, 2024. If you are using DUO Authentication with the Multi-Factor Authentication Plugin, do the following:

1. Upgrade to 4.5.0 from a previous version.
2. Start Mirth® Connect (login will look the same for DUO authentication).
3. Go to your DUO Account.
4. Select Applications and Mirth® Connect Application.
5. Select `Show New Universal Prompt`.
6. Restart Mirth® Connect (login will change to the new prompt).

If you choose not to upgrade before March 30, 2024, you need to disable your DUO setting.

Removed Libraries

We have removed and replaced the following libraries from all Mirth® Connect Components:

1. Apache commons-lang-2.6.jar -> Apache commons-lang3-3.13.0.jar

2. Apache commons-digester-2.0.jar -> Apache commons-digester3-3.2.jar

3. jdom-1.1.1.jar -> jdom2-2.0.6.1.jar

If you are referencing these libraries in your code, please refer to the upgraded libraries instead.

We have removed the following libraries from all Mirth® Connect Components:

1. Jasypt-1.7.1.jar

2. policy-xmlbeans-1.5.jar

3. soapui-4.0.1.jar

4. soapui-xmlbeans-4.0.1.jar

5. stax2-api-3.1.4.jar

6. woodstox-core-5.0.3

7. xbean-fixed-2.4.0.jar

If you are referencing any of these libraries in your code, please include them in your custom libraries instead.

Change of Functionality of "Generate Envelope" in the Web Service Sender

As noted above, we removed the SoapUI library as a dependency of Mirth® Connect. That library

previously provided the functionality for generating SOAP envelope templates in the Web Service Sender, as well as in the Interoperability Suite Senders. In 4.5.0, the "Generate Envelope" function may produce different SOAP envelopes than it did in previous versions, and it may produce incomplete SOAP envelopes for more complex schemas. Please refer to your vendors to ensure you're sending the correct SOAP envelopes in your outgoing messages.

Health Data Hub Plugin

If you previously experienced issues upgrading to 4.2.0 or later when using the Health Data Hub Plugin, then upgrading to 4.5.0 will fix those issues. You can upgrade directly to 4.5.0 from any version of Mirth® Connect whether it is before or after 4.2.0.

4.4.2 Upgrade Notes

Added a Readme File to Show Valid XML Type Message Export Information

An EXPORTREADME.txt file is now included with exported messages. This text file includes a link to a website with file format specifications.

4.4.1 Upgrade Notes

XStream Now Uses an Allowlist Instead of a Denylist

XStream is changed to use an allowlist instead of a denylist. This means that only a strict list of types are allowed to be serialized or deserialized in Mirth® Connect. This will likely have no effect on your Mirth® Connect installation. However, if you have custom channel implementations that happen to serialize or deserialize types that are not allowed by default, you may encounter `ForbiddenClassException` errors or something similar. To fix this problem, you'll need to allow those specific types by adding one or both of the following properties to your `mirth.properties` file:

- `xstream.allowtypes` - A comma-separated list of fully-qualified classes to allow. These classes can include wildcards. Example values are `com.my.package.MyClass`, `com.my.package.MyClass$MyInnerClass`, and `com.my.package.*`.

- `xstream.allowtypehierarchies` - A comma-separated list of fully-qualified type hierarchies to allow. This means that these classes and their subclasses will be allowed. An example value is `com.my.package.MyParentClass`.

Note: You can still use the property, `xstream.denytypes`, that allows you to explicitly deny types from being serialized or deserialized using XStream. It takes a comma-separated list of fully-qualified classes and can include wildcards. This denylist takes precedence over the allowlist.

4.4.0 Upgrade Notes

Added New Functionality to the Mirth® Connect Setup Wizard (Installation Process)

The following new features are added to the Mirth® Connect Setup Wizard:

1. You can download and install your Mirth® Connect commercial extensions at the time Mirth® Connect is installed or upgraded. This simplifies the process so that users no longer have to download commercial extensions prior to the installation.

2. When upgrading Mirth® Connect, the Setup Wizard displays the existing values for the destination directory, license key, network ports, and password requirements from your mirth.properties file rather than the standard default values or the values that were entered previously.

Note: For Linux users upgrading from 4.3.0, new URL links have been added to the Mirth® Connect windows of the setup wizard to enable navigating to the desired version from the main document/wiki pages. To get the updated links, you must uninstall and then re-install Mirth® Connect.

Default Digest Algorithm Changed

The default digest/hash algorithm is changed from SHA256 to PBKDF2WithHmacSHA256. In addition, the default iteration count is changed from 1000 to 600000. Support for Argon2 is also added. This was done to conform with modern password security recommendations.

If you had already explicitly set digest.algorithm, then no changes are made, and Mirth® Connect will

still use the digest algorithm that you have set.

If you had not already explicitly set digest.algorithm, then the new default algorithm will be used going forward. New "fallback" properties will be set to the old defaults so old digest values can still be verified. When a user updates their password, the digest value will be updated to use the new algorithm at that time.

Updating the Digest Algorithm

To make sure you can still log onwith your existing credentials (which used the old digest settings), you want to make sure to set the "fallback" digest settings in mirth.properties. For example:

- digest.fallback.algorithm = SHA256
- digest.fallback.saltsizeinbytes = 8
- digest.fallback.iterations = 1000
- digest.fallback.usepbe = 0
- digest.fallback.keysizeinbits = 256

Then, update the digest.* settings to the new algorithm/etc you want to use.

After restarting Mirth® Connect, it will use the new digest settings for all new digest values. It will also use the "fallback" settings to verify old digest values.

To use the new digest value for your stored password hash in the database, update your user password in the Administrator.

4.3.0 Upgrade Notes

New Functionality for the Mirth® Connect Setup Wizard

New features are added to the Mirth® Connect Setup Wizard that include, but are not limited to, the following:

1. Added more information on the screens with links to the What's New, Upgrade Guide, and documentation. This helps users understand what information to enter on the screens.

2. Added the ability to install your extensions at the time the application is installed or upgraded. This simplifies the process so that users no longer have to install extensions manually and restart the service.

3. Added the ability to enter your license key for extensions. This will default to the `license.key` value from your mirth.properties file during an upgrade.

4. Added the updated logo and a list of each dialog screens so that users know exactly where they are in the process.

Resource Classloaders Load Classes Child-First By Default

Resources now load classes "child-first" instead of "parent-first" by default. In cases where a resource uses a class that is already included with Mirth® Connect (for example a PostgreSQL driver), previously the class from the parent classloader was loaded instead of the custom class that the user wanted to use. By loading "child-first", users can achieve the desired behavior.

Note: If you still need to load "parent-first", you can enable the "Load Parent-First" option on individual resources.

Updated Deprecated Docker Base Images

The OpenJDK and AdoptOpenJDK base images are deprecated, so Mirth® Connect is now built with the eclipse-temurin image instead. If you were customizing your Mirth® Connect Docker containers, you may need to update your configuration to be compatible with eclipse-temurin.

Updated Encryption Settings

The default encryption algorithm is updated from AES to AES/CBC/PKCS5Padding. In addition, before being encrypted, strings will be encoded using UTF-8 instead of the JVM default charset.

- If you did not have `encryption.algorithm` already overridden in mirth.properties, then no action is required. Any newly encrypted data will use the new default algorithm, and old encrypted data will still be able to be decrypted (using the old default algorithm).

- If you had `encryption.algorithm` overridden and set to **AES**, then you should consider updating that, for example to **AES/CBC/PKCS5Padding** or **AES/GCM/NoPadding**. Mirth® Connect will warn you on startup with this message as well. Support for **AES** (without any mode/padding specified) will be removed in a future version.

Disabled TLS Cipher Suites

The default TLS cipher suites are updated. Weaker, potentially exploitable cipher suites are disabled as a best practice.

The following cipher suites are disabled:

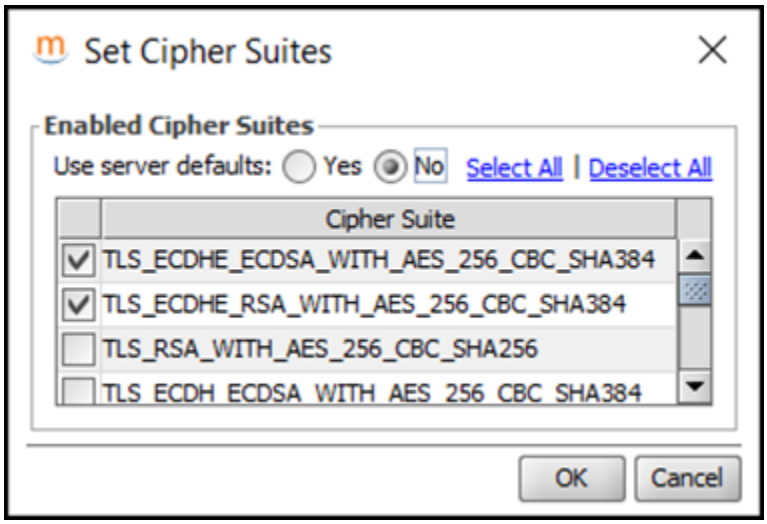
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_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_DSS_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA

If you updated your `https.ciphersuites` in mirth.properties previously, you will see a `https.ciphersuites.old` property which contains your previous values. If you didn't update `https.ciphersuites`, then `https.ciphersuites` will be updated removing the cipher suites above.

If any of your connectors, either connecting to external servers, or allowing clients to connect are using these older cipher suites then it is possible they may fail a Transport Layer Security (TLS) handshake after upgrading to 4.3.0.

If you encounter a TLS handshake error after upgrading, it is best to let the external entity know they need to switch to a more secure cipher suite.

If this is not possible, if you are using the SSL Manager commercial extension, you can select a weaker cipher suite for the connector which needs to allow the weaker legacy cipher suites.



If you are not using the SSL Manager commercial extension, you can manually add back any of the cipher suites removed above to your `https.ciphersuites` property (in mirth.properties) to restore previous, less secure, behavior for all of your connectors.

Removed the View User Guide Option

The **View User Guide** option is removed from the **Other** menu. The **Help** option will continue to navigate users to docs.nextgen.com to access the online documentation.

Administrator Launcher

Mirth® Connect 4.3.0 is signed with an updated certificate. You will need to update the Administrator Launcher to the latest version, 1.3.1, to avoid seeing security warnings.

4.2.0 Upgrade Notes

Removed the Remove All Events Button

We removed the **Remove All Events** button in the Events screen to comply with Cures certification security requirements. If developers need to remove all events, they can use the **Prune Events** in the Data Pruner tab of the Settings screen. This is the preferred method to remove or prune events.

Changed the `patient_id` Query Parameter Naming Scheme for the Remove Message API Request

For the `DELETE /channels/{channelId}/messages/{messageId}` (Remove Message) API request, the `patient_id` query parameter is changed to `patientId`.

Renamed the User Authorization Extension to Role-Based Access Control

The User Authorization extension is changed to the more standard name, Role-Based Access Control. All references to "User Authorization" in the user and extension guides, application user interface elements, and the GitHub Wiki, are changed to "Role-Based Access Control". Going forward, the User Authorization extension will be referred to as the Role-Based Access Control extension.

Moving Java Home Preference Checks

The JAVA_HOME environment variable is now higher in the list when launching the server.

If you only have one version of Java installed on your system, then this doesn't matter. But if you have multiple versions installed, you may want to check to see that the Mirth® Connect server is using the version of Java that you expect. You'll be able to see what version is being used in the server logs at startup.

Note: You can also set the `INSTALL4J_JAVA_HOME_OVERRIDE` environment variable to have the server always use a specific Java installation regardless of what other environment variables may be set. On macOS/Linux, you can also edit the mcserver/mcservice file and uncomment `INSTALL4J_JAVA_HOME_OVERRIDE` at the top instead of setting it as an environment variable.

4.1.0 Upgrade Notes

Warning: Open issue with SMB File Reader/Writer

In 4.1.0, you may encounter errors with the SMB mode of the File Reader/Writer connector, relating to a third-party library. If you currently use the SMB mode for reading or writing files, it's best to wait until a patch version is available.

If you've already upgraded to 4.1.0 and you're encountering this issue, there is a workaround documented in the issue: #5327.

Update log4j Library

Because of the upgrade from log4j to log4j2, the old log4j.properties file has been replaced with the new log4j2.properties file.

Note: If you have custom properties in your current log4j.properties file, you will need to these will need to be applied to the new log4j2.properties file

.

4.0.0 Upgrade Notes

Database Reader XML Casing

In 3.12.0 we updated Apache Commons DBUtils and made some associated changes in our code. As part of that change, the XML created by the Database Reader was using the same case that appeared in the actual SQL query, whereas before 3.12.0 everything was normalized to lowercase. In 4.0.0, we made some more updates in the Connect codebase to, again, normalize the casing to all lowercase.

- If you are upgrading to 4.0.0 from any version before 3.12.0 there is no change for you and database queries will use the casing they always have.
- If you are upgrading to 4.0.0 from 3.12.0 and had made changes adding `AS` clauses in your queries to work around the casing change in 3.12.0 there is no change for you.
- If you experienced database query errors after upgrading to 3.12.0 due to casing in the query statements, after upgrading to 4.0.0 you should no longer encounter those errors.

TLS Protocols and Cipher Suites

Default TLS protocols and cipher suites have been updated. Weaker, potentially exploitable protocols and cipher suites have been disabled as a best practice.

TLS Protocols

TLSv1.1 has been disabled and the default protocols are now:

```
https.client.protocols = TLSv1.3,TLSv1.2
https.server.protocols = TLSv1.3,TLSv1.2,SSLv2Hello
```

Cipher Suites

The following cipher suites have been disabled:

- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

If you had updated your `https.ciphersuites` in mirth.properties previously, you will see a `https.ciphersuites.old` property which contains your previous values. If you had not updated `https.ciphersuites` then `https.ciphersuites` will have been updated removing the cipher suites above.

Impact

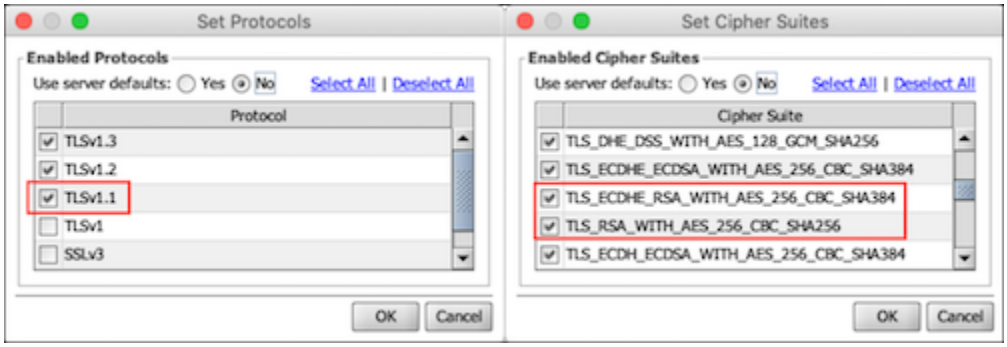
If any of your connectors, either connecting to external servers, or allowing clients to connect were using these older protocols or cipher suites then it is possible they may fail a TLS handshake after upgrading to 4.0.0.

Resolution

If you encounter a TLS handshake error after upgrading it is best to let the external entity know they need to switch to a more secure protocol and/or cipher suite.

If this is not possible:

SSL Manager Options



If you are using the SSL Manager commercial extension, you can select TLSv1.1 and/or a weaker cipher suite for the connector(s) which need to allow these weaker legacy protocols and cipher suites.

Server Wide Options

If you are not using the SSL Manager commercial extension, you can manually add back TLSv1.1 to your `https.client.protocols` or `https.server.protocols` or any of the cipher suites removed above to your `https.ciphersuites` property (in mirth.properties) to restore previous, less secure, behavior for all of your connectors.

HTTP User Agent

When HTTP Connectors send a message, the `user-agent` header will no longer include information about the Apache or Java library versions (example `user-agent=[Apache-HttpClient/4.5.13 (Java/1.8.0_181)]`) and will instead send "Mirth® Connect" as the user agent. This

default user agent can be overridden with a different user agent by populating your own `user-agent` header in the HTTP Sender's headers table.

HTTP Server Header

In previous versions of Connect, HTTP connectors and the Connect web server, which hosts the Dashboard and API documentation, would automatically add a server header which included the version of the Jetty library (example `server=[Jetty(9.4.21.v20190926)]`). Jetty would also show a "Powered by Jetty" message (example `Powered by Jetty://9.4.21.v20190926`) on error pages. To prevent external entities or malicious tools from easily learning about the libraries and versions used within Connect, after upgrading to 4.0.0, neither the Jetty header nor the "Powered by Jetty" message will be included in HTTP responses.

FHIR Extension

The FHIR extension is now part of NextGen's commercial extensions and no longer freely available. If you would like to use the FHIR extension, please contact mirthconnectsales@nextgen.com for more information.

3.12.0 Upgrade Notes

Preventing XML External Entity Vulnerabilities

In 3.12.0, we address a potential security risk regarding XML External Entity Processing. See this OWASP article for more details about the vulnerability.

In order to mitigate risk, we now disallow DOCTYPE declarations in all areas the Mirth® Connect that parse XML.

How this can affect you

If any of your channels or code templates are constructing XML messages that contain DOCTYPE declarations, there will now be an error when that XML is parsed by Connect.

Similarly, if any of your channels receives XML messages that contain DOCTYPE declarations, those messages will error upon parsing.

PDF Generation Updates

In 3.12.0 Connect no longer uses iText for generating PDFs and is now using openhtmltopdf. Also, the

PDFBox library has been updated from version 1.8.4 to 2.0.24.

Using PDFBox Classes

Since the PDFBox library has been updated from 1.x to 2.x, if you are using any classes from PDFBox directly, you may need to change how you have implemented your PDF solutions around this library as it is possible this major update could contain breaking changes.

PDF Generation and Images

Previously, while local paths to images may have worked without being proper URIs, it was never intended functionality, and with the change of PDF generation libraries in 3.12.0, you must now use a proper file URI when adding images to your PDF.

For example, this will **not** work:

```
<img src="/data/images/image1.png"></img>
```

But this **does** work:

```
<img src="file:///data/images/image1.png"></img>
```

3.11.0 Upgrade Notes

Database Connection Retries on Server Startup

We have added the following new properties to `mirth.properties` to support database connection retries:

- `database.connection.maxretry` - On startup, the maximum number of retries to establish database connections in case of failure. Defaults to `2`.
- `database.connection.retrywaitinmilliseconds` - On startup, the time, in milliseconds, to wait before retrying to establish a database connection in case of failure. Defaults to `10000`.

Interoperability Connector Suite

Logging Raw SOAP Payloads and WS-Security Details

Enable this feature by adding the following line to your `log4j.properties` file:

```
log4j.logger.com.mirth.connect.connectors.interop.server.InteropReceiver =
        WARN
```

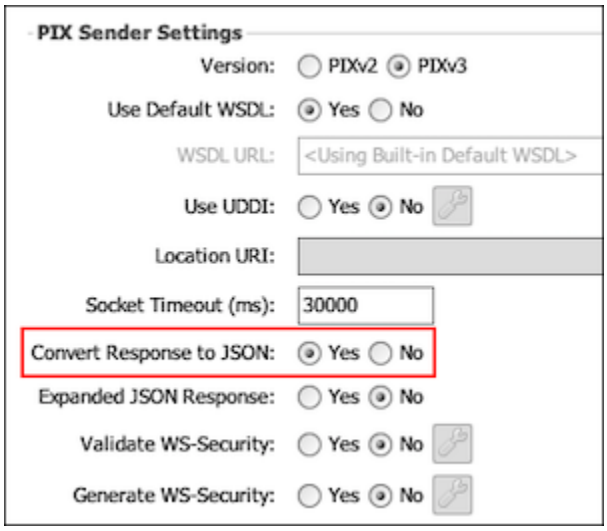
Toggling Automatic Conversion to JSON

In the settings for all Interoperability Listener Connectors, we have added two new toggle-able options:

- `Convert to JSON` - If enabled, automatically convert SOAP payloads to JSON. This is enabled by default and on migration.
- `Convert JSON Response to XML` - If enabled, automatically convert responses from JSON to XML. This is enabled by default and on migration.

In the settings for all Interoperability Sender Connectors, we have added the following new toggle-able option:

- `Convert Response to JSON` - If enabled, automatically convert responses from XML to JSON. This is enabled by default and on migration.



3.10.0 Upgrade Notes

Advanced Clustering Sync Intervals

In 3.10.0 we have split many of the clustering tasks off from the Cluster Sync Interval into their own intervals. Here's a list of the new intervals and what they control:

- Dashboard Status Sync Interval: The frequency in milliseconds to query the cluster to update channel dashboard statuses.
- Cluster Log Sync Interval: The frequency in milliseconds to query the cluster to update alert logs, connection logs, and server logs.
- Cluster State Sync Interval: The frequency in milliseconds to query the cluster to update alert states, connection statuses, and the global map.

Cluster Sync Interval continues to handle the frequency in milliseconds to query the cluster for new tasks to execute. These tasks handle membership changes, message recovery, role changes, and server name changes.

3.9.0 Upgrade Notes

SMB Versions in a File Reader/Writer

We added support for SMB versions 2 and 3 when using a File Reader/Writer. With "smb" as the selected method, select the wrench icon to view the SMB settings. There you can select the minimum and maximum supported SMB versions.

When upgrading Connect, existing channels will use "SMB v1" as the minimum version and "SMB v3.1.1" as the maximum version, in order to preserve functionality. However, SMB v1 is outdated and poses security risks, so we recommend updating your channels to use later versions of SMB. This may require also updating your SMB servers accordingly.

DICOM Sender Storage Commitment

In 3.9.0 we changed the functionality of the DICOM Sender option "Request Storage Commitment". Before 3.9.0, if a DICOM Sender with this option set to "Yes" failed to get a response to its storage commitment request, the sender would log an error to the server log but the message would still be considered successful. Now, in 3.9.0, if a DICOM Sender with this option set to "Yes" fails to get a response to its storage commitment request, the message will get the Error state. This means there is a potential for DICOM Senders to start encountering error messages after the update, but we feel that this is the proper behavior. You must either ensure the system you are sending to supports storage commitment or you can set "Request Storage Commitment" to "No".

3.8.0 Upgrade Notes

In version 3.8 we upgraded the MySQL JDBC driver to version 8.0.16, so that the latest MySQL 8.x can be used. However the new driver doesn't support versions of MySQL older than 5.6. Since these older versions are over 10 years old and are already listed as End-Of-Life by Oracle, we have made the decision to adjust our system requirements to reflect that 5.6+ is now the minimum supported version for MySQL.

If you still need to connect to an older version of MySQL, do not worry you still can!

Connecting via a Database Reader/Writer or in JavaScript

- Download the appropriate MySQL JDBC Driver JAR for the version of MySQL you need to connect to.

- Place this JAR into a folder of your choice on the system running Mirth Connect.
- Create a new library resource (Settings -> Resources tab) pointing to that folder.
- Link your connector/channel to your new resource (Summary tab -> Set Dependencies).
- Make sure you are using the old driver class name, "com.mysql.jdbc.Driver".

Using an Old Version of MySQL for Your Connect Backend Database

- Download the appropriate MySQL JDBC Driver JAR for the version of MySQL you need to connect to.
- Swap this JAR with the MySQL 8.x driver in the "server-lib/database" folder in your installation directory.
- Set "database.driver = com.mysql.jdbc.Driver" in mirth.properties.
- Make sure that any Database Reader/Writer connectors that are connecting to the old version of MySQL use the old driver class name, "com.mysql.jdbc.Driver".

If you need to use an old version of MySQL for the backend database and also need to connect to newer MySQL 8.x+ databases as well, you can follow the steps in the section above to include the 8.x JDBC driver as a custom resource.

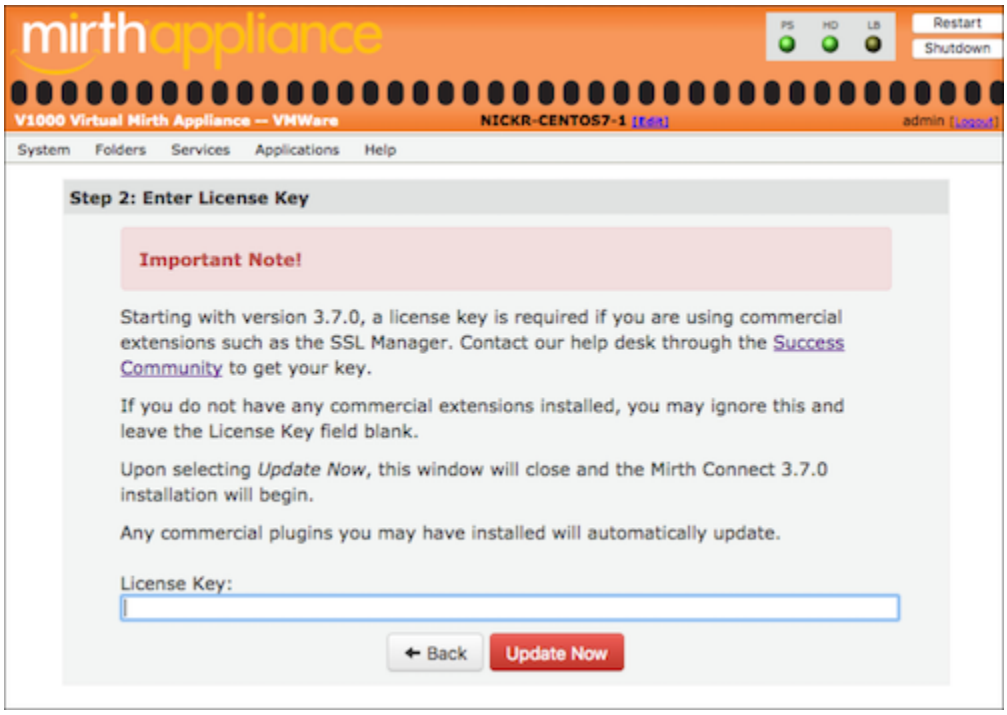
3.7.0 Upgrade Notes

Commercial License Key

As of 3.7 commercial extensions now require a license key to be set in mirth.properties. The machine running Connect also must have outbound internet access so that our licensing server can be reached. Before upgrading, contact the help desk through our NextGen Healthcare Success Community to get your license key.

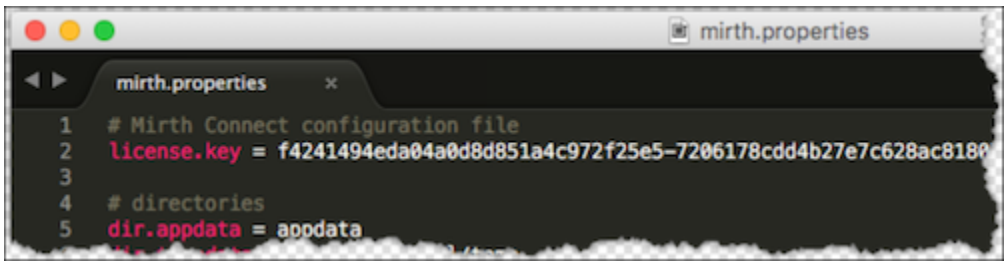
Using Mirth Appliance

When installing the 3.7.0 update, a wizard will prompt you to enter the license key:



Standalone Installation

- Edit conf/mirth.properties
- Add a setting with key "license.key" and your license key for the value



- Restart the Connect service

Database Connection Pools

In previous versions, there was a connection pool dedicated to the Donkey engine (message processing), and a separate pool dedicated to everything else, like Administrator or API actions.

In 3.7, this has been changed somewhat. You have two options:

Option 1: Use two pools, one read/write and one read-only

This is the default in 3.7. To enable, set this in mirth.properties:

```
database.enable-read-write-split = true
```

When this 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.

Option 2: Use one connection pool for everything

To combine all database connections used by Connect into a single pool, set this in mirth.properties:

```
database.enable-read-write-split = false
```

All of the `database-readonly` options will be ignored in this case. If you switch to this mode, note that to may want to increase `database.max-connections` to compensate for the `database-readonly.max-connections` value that will no longer be used.

Keystore Passwords

The default keystore storepass and keypass is `81uWxplDtB`. However when Connect starts up for the first time and the keystore file hasn't yet been created, Connect will automatically replace the default passwords with new auto-generated passwords.

If you are upgrading from a previous version nothing will change, your keystore and passwords will remain the same. This is only done for completely new servers where the keystore hasn't yet been created and the passwords are still set to the default values.

3.5.0 Upgrade Notes

Important: Mirth® Connect by NextGen Healthcare 3.5 and above now require a minimum version of Java Runtime Environment (JRE) 8. Please ensure that you have Java 8 (or later) installed first before attempting to upgrade.

Due to the recent SWEET32 vulnerability, we have disabled cipher suites using Triple DES. If for any reason you need to use one of these cipher suites to communicate with a legacy system, you can re-enable it in mirth.properties with the `https.ciphersuites` setting. If you are using the SSL Manager extension, then you can choose to enable these cipher suites on a specific channel / connector rather than for the entire server. If you have any connectors using the SSL Manager that are not using the server default cipher suites, then you will want to update these connectors and deselect any cipher suites containing "3DES".

To address the secondary "nation state resources" theoretical use-case of the Logjam vulnerability, in 3.5 we're now setting the ephemeral Diffie-Hellman key size to 2048. If for any reason you need to change this, you can do so in mirth.properties with the `https.ephemeraldhkeysize` setting.

Custom attachment handlers that extend MirthAttachmentHandlerProvider will need to provide a constructor that matches and calls this super constructor:

```
public MirthAttachmentHandlerProvider(MessageController messageController)
```

Custom attachment handlers also may need to update method signatures if any of the following methods have been overridden (updated signatures on the second line of each group):

```
public byte[] reAttachMessage(String raw, ConnectorMessage connectorMessage, S
public byte[] reAttachMessage(String raw, ConnectorMessage connectorMessage, S

public String reAttachMessage(ConnectorMessage message)
public String reAttachMessage(ConnectorMessage message, boolean reattach)

public String reAttachMessage(ImmutableConnectorMessage message)
public String reAttachMessage(ImmutableConnectorMessage message, boolean reatt

public String reAttachMessage(String raw, ConnectorMessage message)
public String reAttachMessage(String raw, ConnectorMessage message, boolean re

public String reAttachMessage(String raw, ImmutableConnectorMessage message)
public String reAttachMessage(String raw, ImmutableConnectorMessage message, b

public byte[] reAttachMessage(String raw, ImmutableConnectorMessage connectorM
public byte[] reAttachMessage(String raw, ImmutableConnectorMessage connectorM
```

3.4.0 Upgrade Notes

Mirth® Connect by NextGen Healthcare plugins that implement custom client <-> server communication will need to be updated to use the new HTTP servlet architecture introduced in version 3.4.0.

Specifically, the invoke() method has been removed from the ServicePlugin interface as well as the ClientPlugin abstract class. Prior to 3.4.0, calls to ClientPlugin.invoke() on the client were routed to ServicePlugin.invoke() on the server.

Plugins that previously used these methods must now provide a separate servlet interface on the server side using JAX-RS and Swagger annotations:

ServerLogInterface.java:

```
@Path("/extensions/serverlog")
@Api("Extension Services")
@Consumes(MediaType.APPLICATION_XML)
@Produces(MediaType.APPLICATION_XML)
public interface ServerLogServletInterface extends BaseServletInterface {
    public static final String PLUGIN_POINT = "Server Log";
    public static final String PERMISSION_VIEW = "View Server Log";
```

```
    @GET
    @Path("/")
    @ApiOperation("Retrieves all server log entries.")
    @MirthOperation(name = "getMirthServerLogs", display = "View Server Log",
    public LinkedList<String[]> getServerLogs() throws ClientException;
```

To invoke servlet methods from the client side, use the getServlet() method instead of the previous invokePluginMethod* methods.

ServerLogClient.java - 3.3.x and earlier:

```
try {
    serverLogReceived = (LinkedList<String[]>) PlatformUI.MIRTH_FRAME.mirthCli
} catch (ClientException e) {
```

ServerLogClient.java - 3.4.0 and later:

```
try {
    serverLogReceived = PlatformUI.MIRTH_FRAME.mirthClient.getServlet(ServerLo
} catch (ClientException e) {
```

The servlet interface must also be specified in the plugin's plugin.xml file:

```
<pluginMetaData path="serverlog">
    ...
    <apiProvider type="SERVLET_INTERFACE" name="com.mirth.connect.plugins.serv
    ...
</pluginMetaData>
```

3.2.0 Upgrade Notes

JRE 1.6 (Java 6) is no longer supported as of version 3.2.0. If you are still using JRE 1.6, please ensure you update to at least JRE 1.7 before upgrading to Mirth® Connect 3.2.0+. However future releases in the 2.x, 3.0.x, and 3.1.x branches will still continue to support JRE 1.6 unless otherwise stated.

The default connection pool used for message processing has been updated to HikariCP. We have seen that HikariCP is generally faster and more reliable than DBCP although this change should be imperceptible to most users. Support for DBCP still exists and users can revert back to it if necessary by adding the following line to mirth.properties and restarting the server.

```
database.pool = dbcp
```

A bug in the configuration map was fixed where certain special characters were not being handled correctly. Since this bug affected both the saving and loading of the configuration map, there is a chance your configuration map values will be loaded differently after upgrading Mirth® Connect by NextGen Healthcare. Please verify your configuration map values after upgrading.

3.1.1 Upgrade Notes

Due to the recent POODLE vulnerability, we have disabled SSLv3 for all of our relevant connectors and connections that use SSL. In case you are connecting to some legacy systems that require SSLv3, it is possible to re-enable SSLv3. In your mirth.properties file, you will find the following two properties:

```
https.client.protocols = TLSv1.2,TLSv1.1,TLSv1
https.server.protocols = TLSv1.2,TLSv1.1,TLSv1,SSLv2Hello
```

If you need to use SSLv3, simply add "SSLv3" into the comma separated list for the client or server protocols and restart your server. These two properties also allow you to configure exactly which SSL protocols you wish to use. Similarly, if you need to add or remove cipher suites, you can update the https.ciphersuites property in the mirth.properties file.

Please note that if you are making HTTPS connections programmatically in JavaScript, you will still need to update your code accordingly in order to disable SSLv3.

When upgrading to Mirth® Connect by NextGen Healthcare 3.1.1+ OR Java 8, some users have experienced issues connecting to the Administrator or external SSL endpoints which previously worked fine. This is likely because some servers do not implement forward compatibility correctly and refuse to talk to TLS 1.1 or TLS 1.2 clients. TLSv1.2 and TLSv1.1 are enabled by default in Java 8. As such we have also decided to enabled them by default beginning with 3.1.1 regardless of Java version. If you experience these issues, you should first try disabling TLSv1.2 and TLSv1.1 by updating the https.client.protocols property and restarting the server

```
https.client.protocols = TLSv1
```

If you are still unable to connect to the Administrator, you can try also disabling TLSv1.2 and TLSv1.1 on the server protocols. For security reasons, It is recommended to leave the default settings unless you are explicitly running into issues connecting to SSL endpoints.

3.1.0 Upgrade Notes

A new index has been added to the message metadata table for each channel to improve queue

performance . Channels created in 3.1.0 will automatically create this index, but existing channels will need to have the index created manually:

1. Stop all channels that require the index
2. Navigate to Settings > Database Tasks
3. Select the "Add Metadata Index" task
4. Select Run Task

If you do not see the "Add Metadata Index" task, then the index already exists for all of your channels and there is no action needed. If the task fails to add the index for all of your channels, ensure that the remaining affected channels are stopped or undeployed and run the task again.

Ensure you have enough disk space before adding the new index. The index will require roughly 45 megabytes of disk space per connector for every million messages currently stored in the database. It may take some time depending on how many messages are currently stored, so it is recommended to schedule some downtime for adding the index.

3.0.2 Upgrade Notes

Channel Updates

The Database Reader no longer prepends table name to the column name or alias regardless of how many tables are used in the query. Each element name in the generated XML will be the alias if one exists, otherwise it will be the column name. Users connecting to Postgres or Oracle databases with the Database Reader connector should not be required to make any changes. Users connecting to other databases should update their channels accordingly if needed. It is recommended to always use a unique alias.

Note: Code templates in 3.0.0 and 3.0.1 were not being restricted from certain JavaScript scopes based on their context. Users upgrading from these versions to 3.0.2+ should check their code templates to ensure their contexts are correctly set. See MIRTH-3150 for more information.

3.0.0 Upgrade Notes

Message Migration

If you are upgrading to Mirth® Connect by NextGen Healthcare 3.x from Mirth® Connect by NextGen

Healthcare2.x, your existing messages will not be upgraded. This includes any queued messages that have not been sent, as well as any historical message records. After upgrading, Mirth® Connect 2.x messages will not be available in the Mirth® Connect Administrator, but will remain in the database in the OLD_MESSAGE table.

Any messages that were queued at the time of upgrade from 2.x will not be processed. If you would like to ensure these messages are processed, we recommend flushing out all queued messages before upgrading:

For Non-Polling Channels

Pause the channel Wait for all queued messages to finish processing

For Polling Channels

1. Edit the channel, export the source connector (to have a backup)
2. Change the source connector type to Channel Reader
3. Redeploy the channel
4. Wait for all queued messages to finish processing
5. Edit the channel again, restore the exported source connector, but do not redeploy

Do the above for all channels until there are no more queued messages. If any queued messages cannot be processed because the external system is down, you can search for the queued messages and export them as XML or Plain Text for manual processing later.

To simply remove all queued messages you can remove them from the Message Browser, which should also remove them from the file system, or simply delete the queuestore folder (or individual channel folders inside of queuestore) in your appdata directory.

Channel Updates

The API in Mirth® Connect by NextGen Healthcare 3.0 has changed dramatically. All old JavaScript references and variables from the Reference List should still work properly, but they may log messages stating the code you are using has been deprecated, and recommend code changes that you should complete. Any deprecated methods may be removed in an upcoming version.

Calls to any unsupported internal Mirth® Connect by NextGen Healthcare code may have changed and may no longer work. This includes any JavaScript that has to import or reference anything in the com.mirth.connect.* package. If you are using any unsupported internal Mirth® Connect classes to

perform actions like starting and stopping channels, please take a look at Mirth® Connect by NextGen Healthcare 3.0's new ChannelUtil API.

CHAPTER 20

Commercial Support/Extensions

A Mirth® Connect commercial license includes:

- **Enterprise-class Support:** Most importantly, you will be safely under our support umbrella.
- **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.

Contact Us if you have any questions.

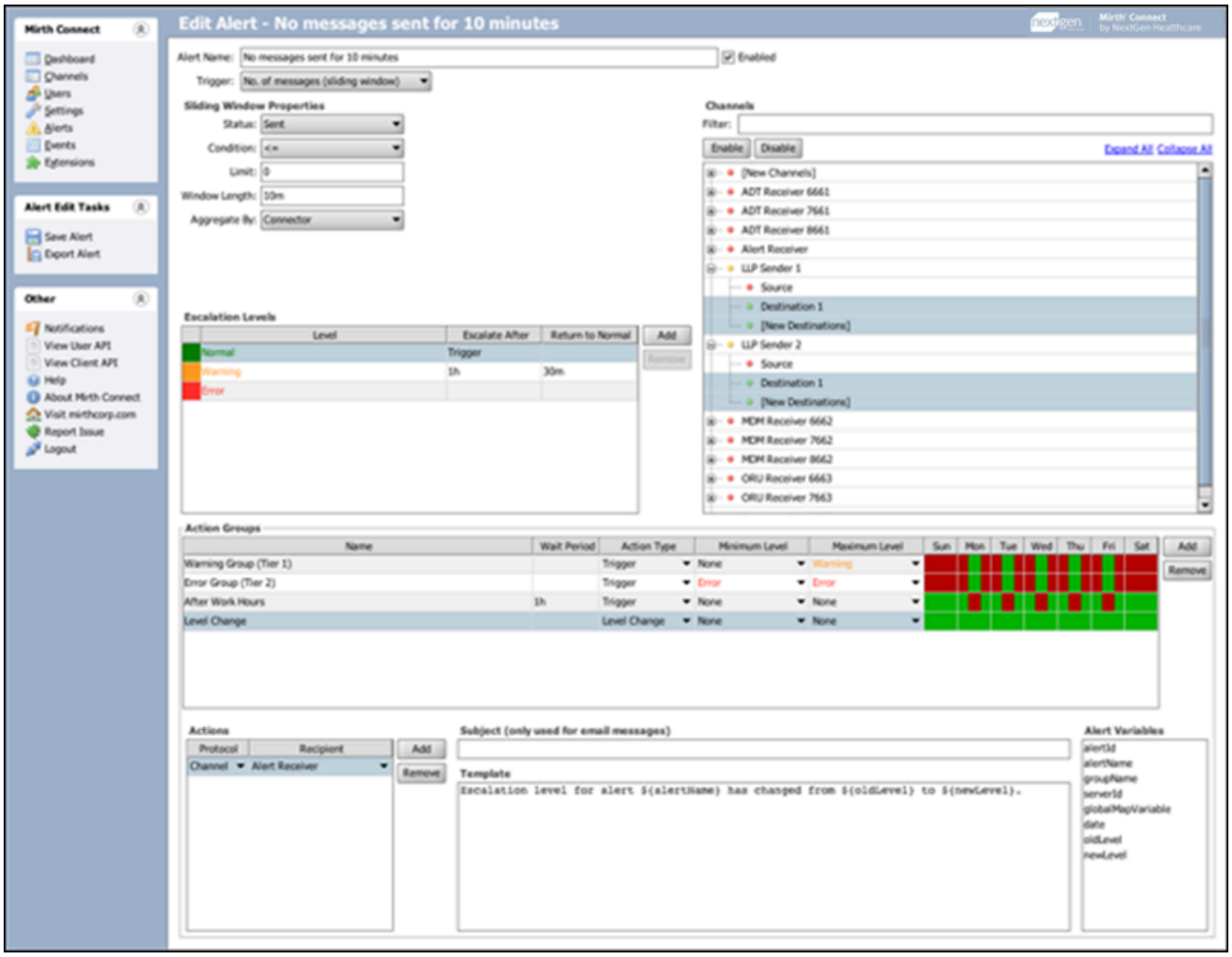
The following extensions are available as part of commercial licenses:

- Advanced Alerting *(on page 583)*
- Advanced Clustering *(on page 584)*
- ASTM E1381 Transmission Mode *(on page 585)*
- ASTM E1394 Data Type *(on page 586)*
- Channel History *(on page 587)*
- Cures Certification Support *(on page 588)*
- Email Reader *(on page 588)*
- FHIR Connector *(on page 590)*
- Interoperability Connector Suite *(on page 592)*
- LDAP Authorization *(on page 594)*

- Message Generator *(on page 595)*
- Multi-Factor Authentication *(on page 596)*
- Serial Connector *(on page 597)*
- SSL Manager *(on page 598)*
- Role-Based Access Control *(on page 599)*
- NextGen Results CDR Connector *(on page 600)*

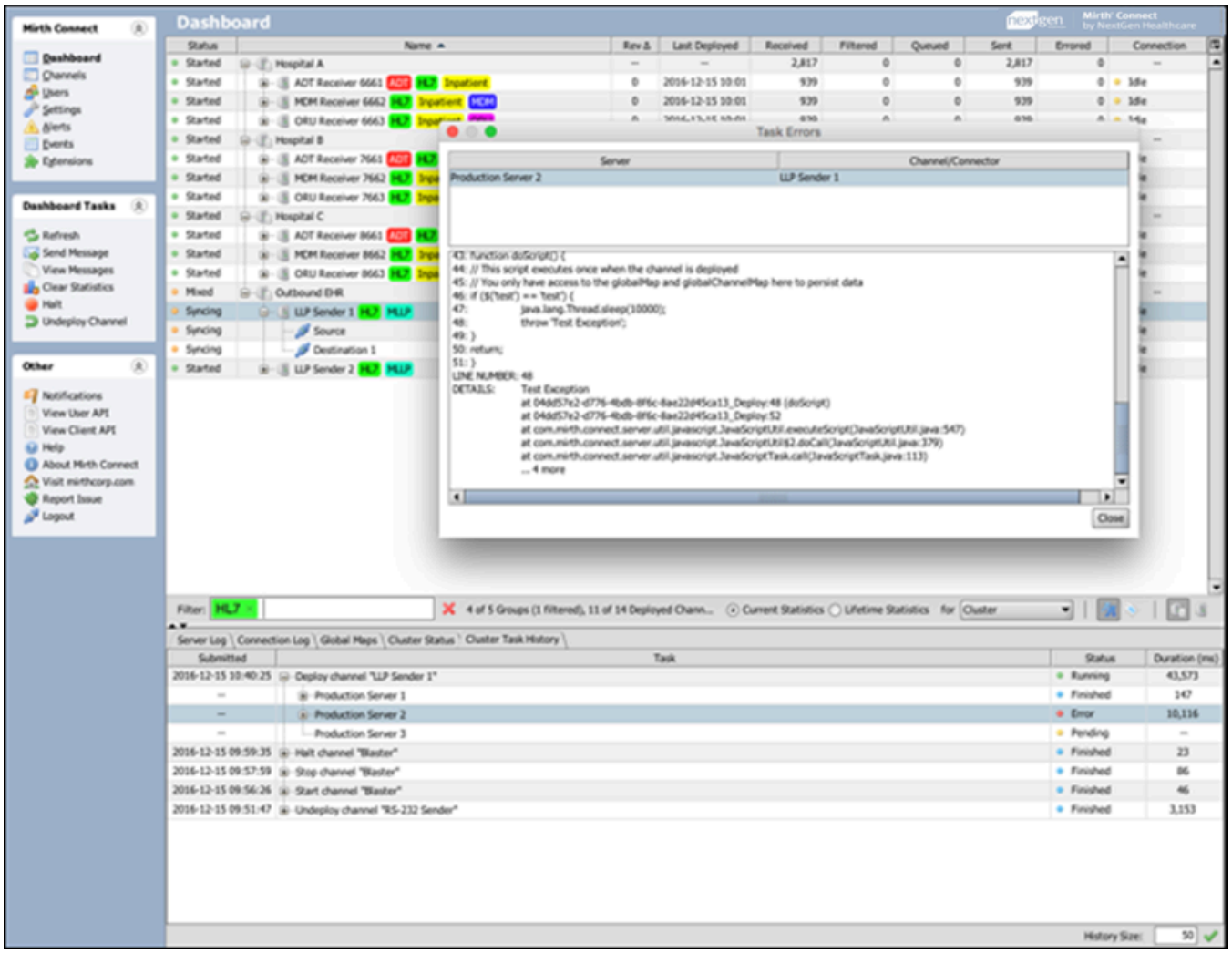
Advanced Alerting

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. Contact Us if you have any questions.



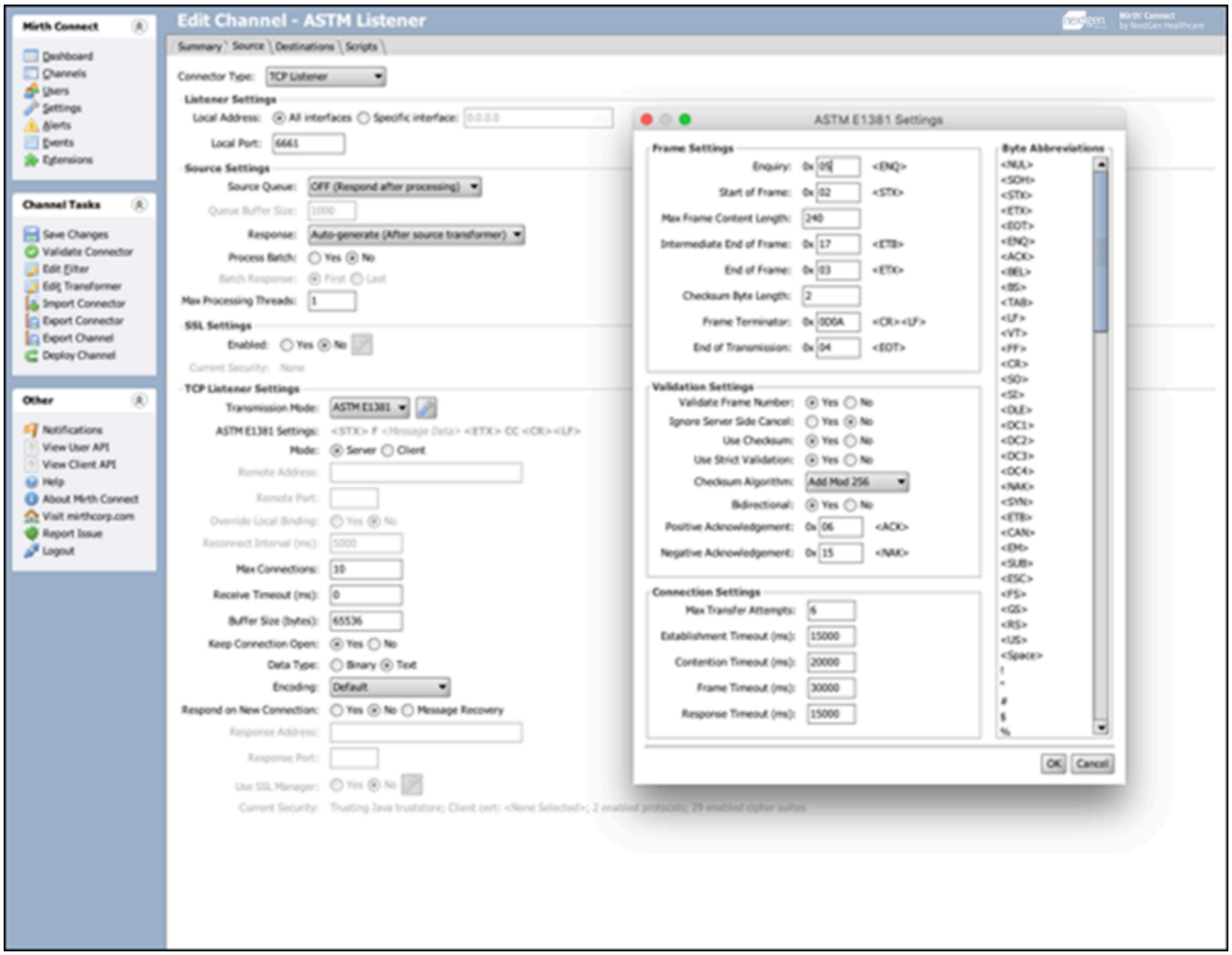
Advanced Clustering

The Advanced Clustering plug-in improves the availability of your Mirth® Connect 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 Mirth® Connect instances using your own load-balancing solution, but also seamlessly integrates with the load-balancing and failover services provided on Mirth Appliances. Contact Us if you have any questions.



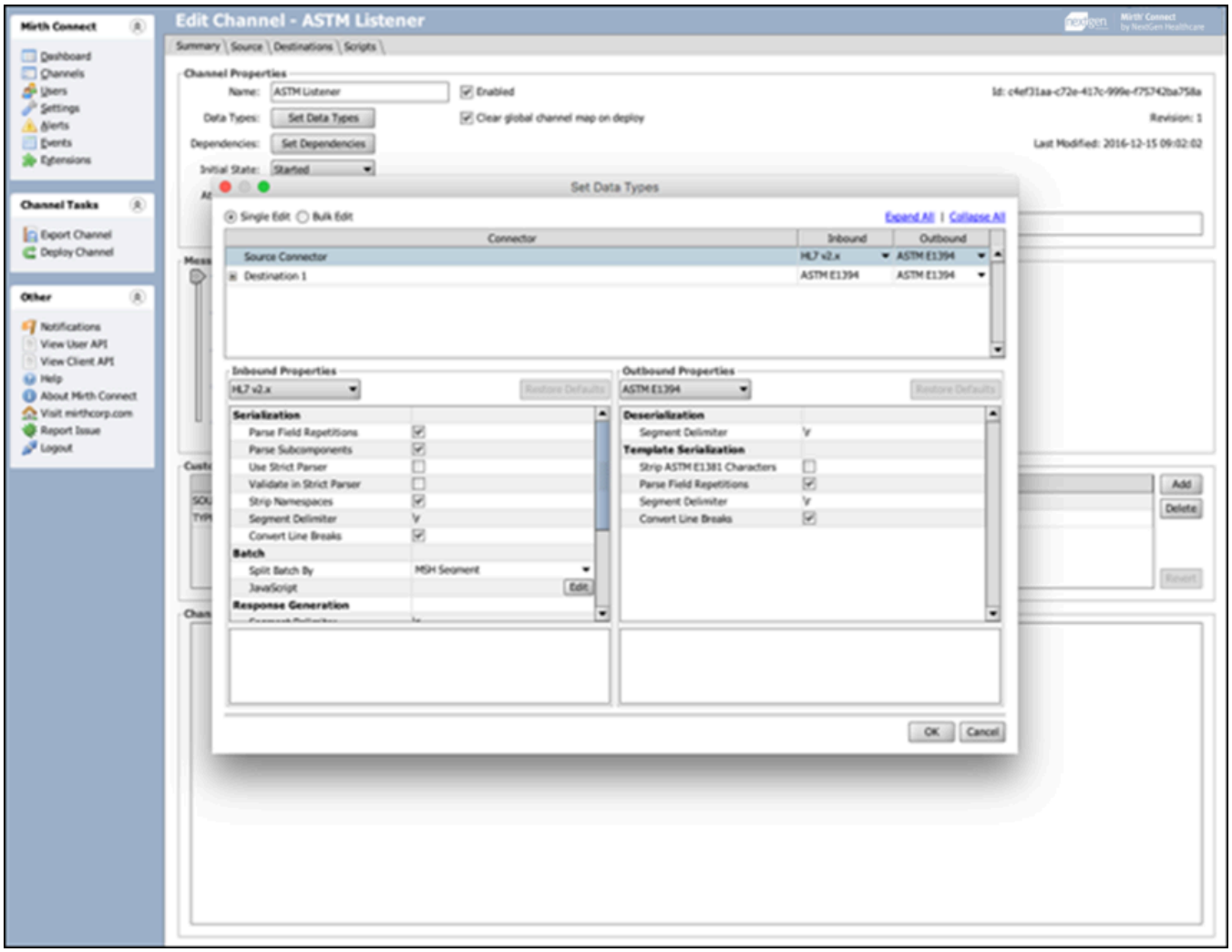
ASTM E1381 Transmission Mode

The ASTM E1381 transmission mode for Mirth® Connect enables 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. Contact Us if you have any questions.



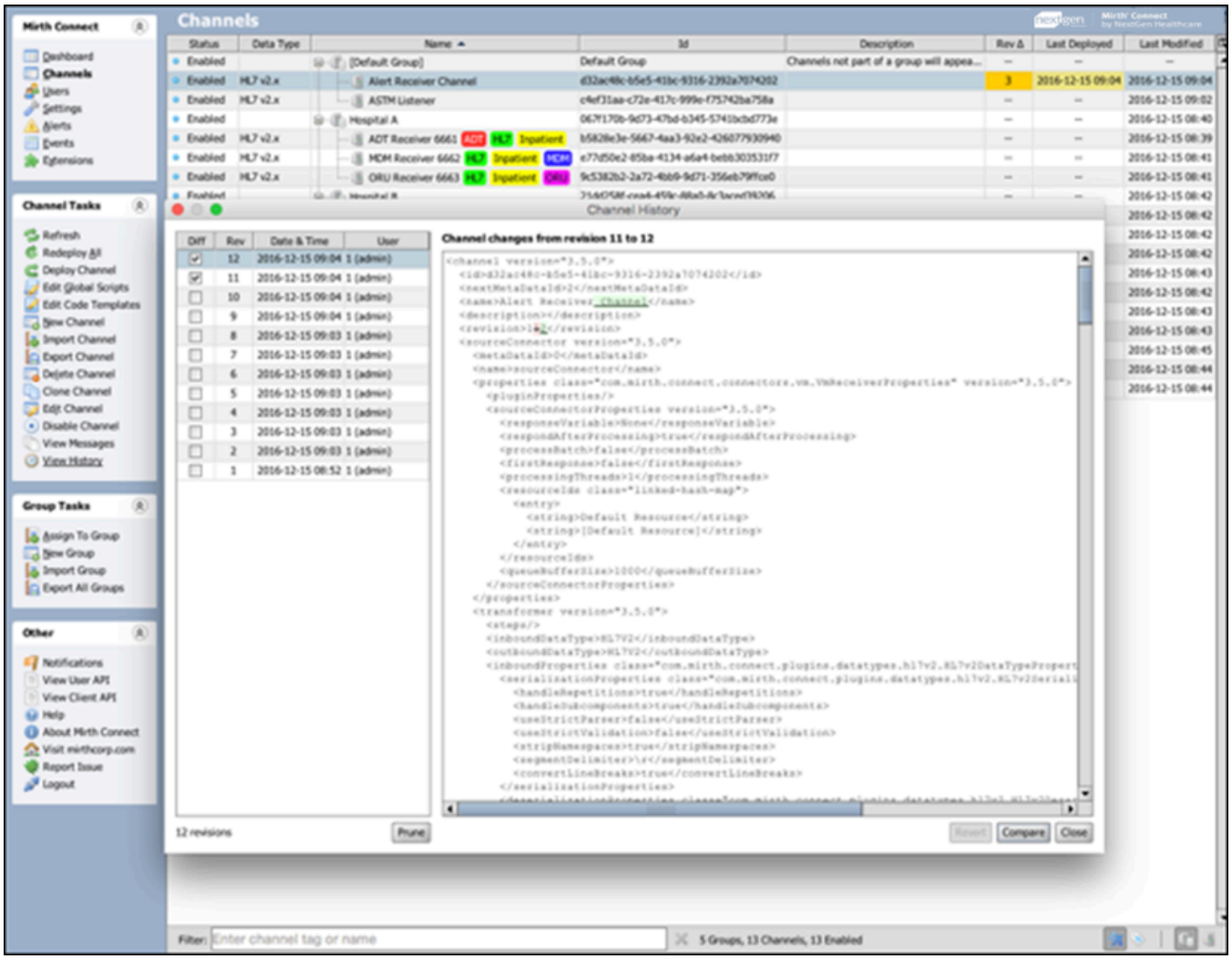
ASTM E1394 Data Type

The ASTM E1394 data type for Mirth® Connect enables 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. Contact Us if you have any questions.



Channel History

You can 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. You can also revert to a past revision from the embedded viewer. Contact Us if you have any questions.



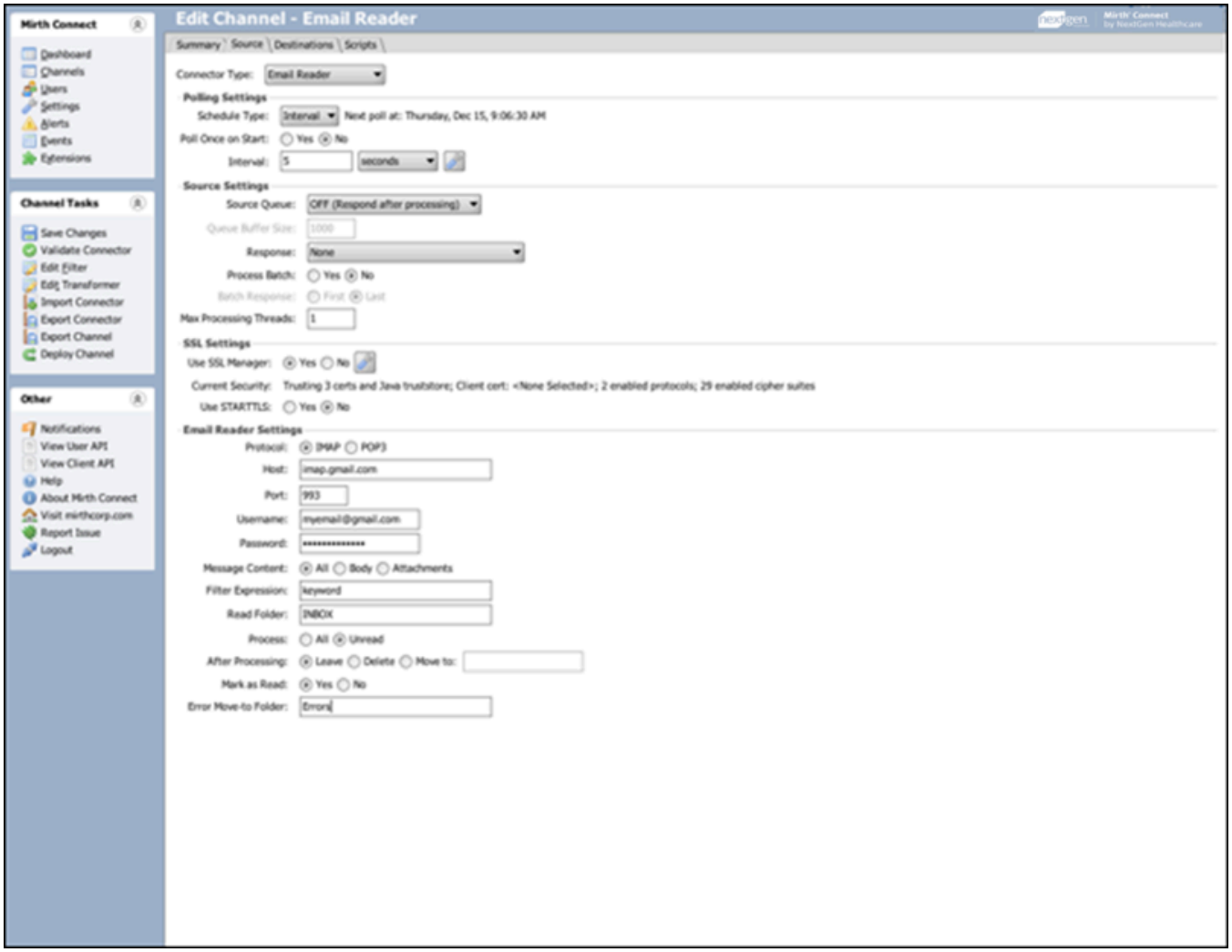
Cures Certification Support

Cures Certification Support provides additional features to support Mirth® Connect's Cures certification. This includes automatic detection of audit log altercations. Contact Us if you have any questions.

Email Reader

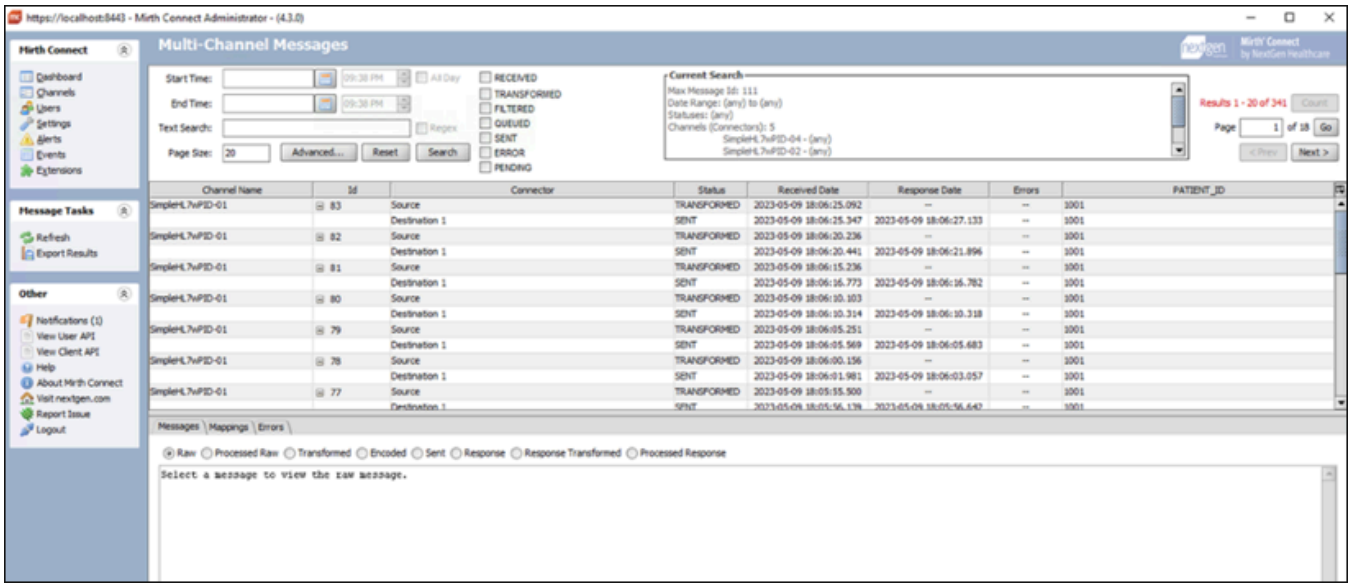
You can securely connect to a POP3 or IMAP email server and download email messages for processing in a channel using the Email Reader. With the Message Content parameter, you can specify if an individual email message should be read as XML, including the metadata and the body, as the body, or

as a set of attachments. You can also access numerous options for specifying behavior after a message is read. Contact Us if you have any questions.



Enhancement Bundle

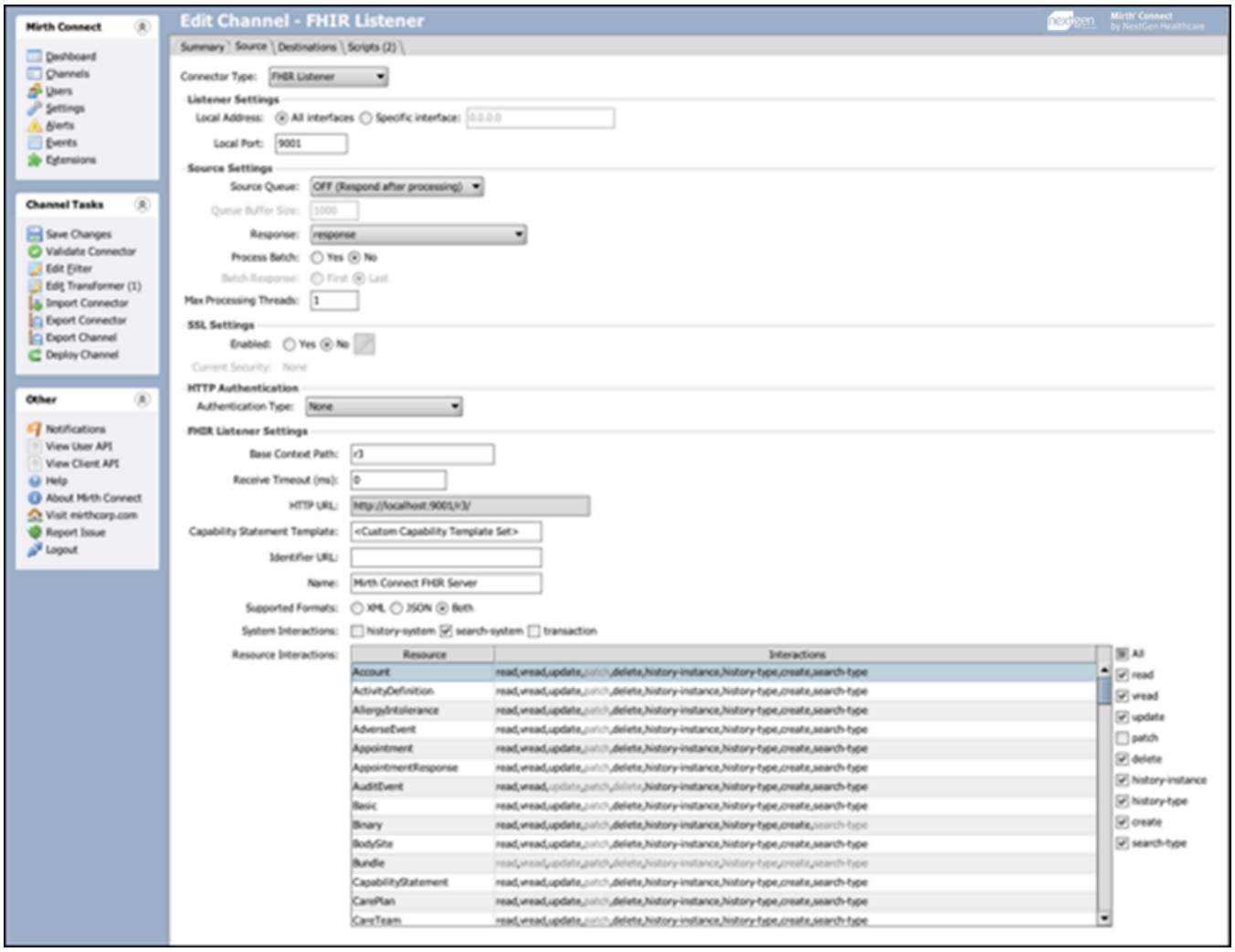
Enhancement Bundle provides additional support to enable users to search for messages across multiple channels in the dashboard and channel panels. Contact Us if you have any questions.

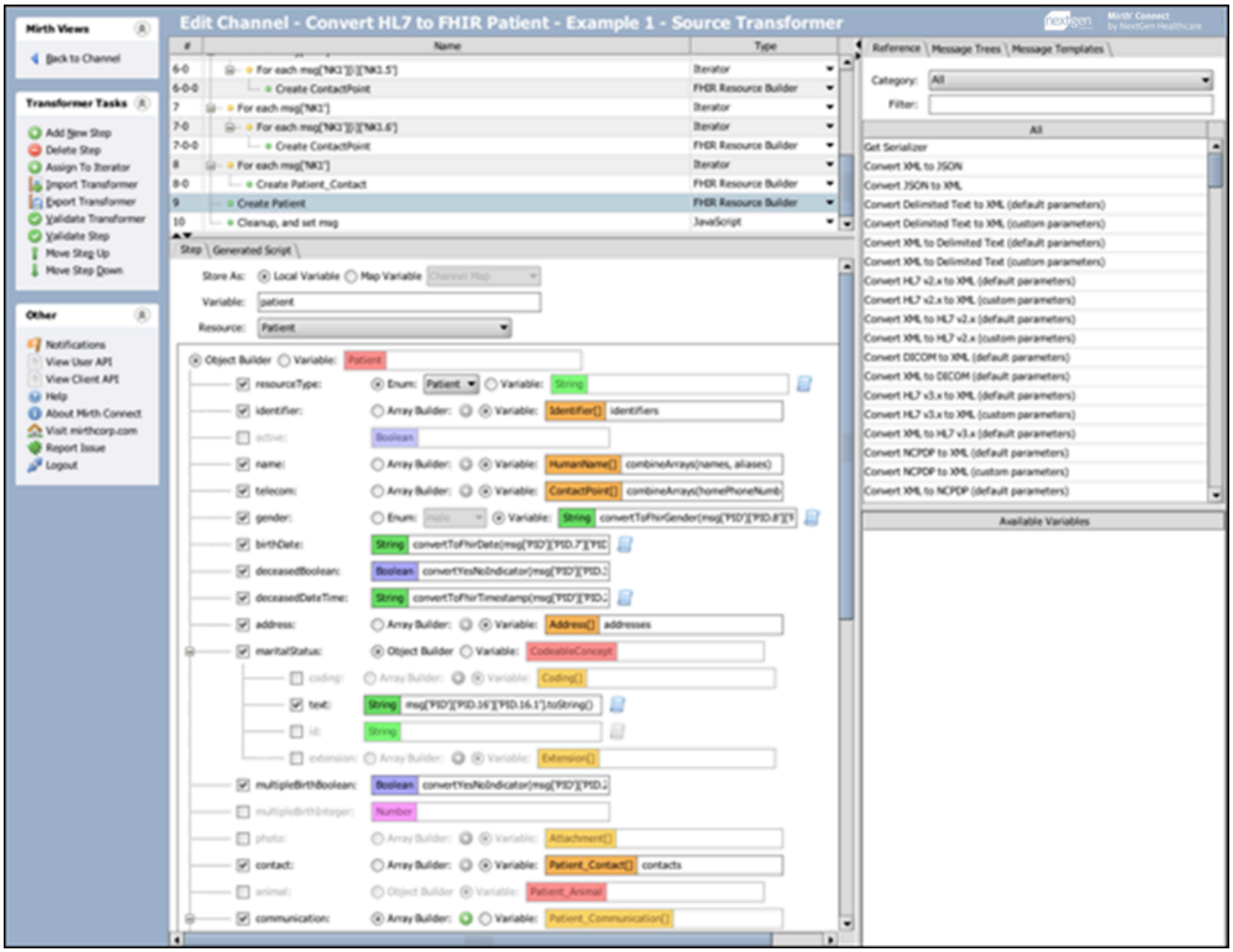


FHIR Connector

Fast Healthcare Interoperability Resources (FHIR) 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 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, view the user guide on the public wiki: https://www.mirthcorp.com/community/wiki/display/mirth/FHIR+Connector+Extension+(3.6)


Mirth Connect
Dashboard
Channels
Users
Settings
Alerts
Events
Extensions
Channel Tasks
Save Changes
Validate Connector
Edit Filter
Edit Transformer (1)
Import Connector
Export Connector
Export Channel
Deploy Channel
Other
Notifications
View User API
View Client API
Help
About Mirth Connect
Visit mirthcorp.com
Report Issue
Logout
Edit Channel - FHIR Listener
Summary
Source
Destinations
Scripts (2)
Connector Type:
FHIR Listener
Listener Settings
Local Address:
All interfaces
Specific interface:
Local Port:
9001
Source Settings
Source Queue:
OFF (Respond after processing)
Queue Buffer Size:
1000
Response:
response
Process Batch:
Yes
No
Batch Response:
First
Last
Max Processing Threads:
1
SSL Settings
Enabled:
Yes
No
Current Security:
None
HTTP Authentication
Authentication Type:
None
FHIR Listener Settings
Base Context Path:
r3
Receive Timeout (ms):
0
HTTP URL:
http://localhost:9001/r3/
Capability Statement Template:
<Custom Capability Template Set>
Identifier URL:
Name:
Mirth Connect FHIR Server
Supported Formats:
XML
JSON
Both
System Interactions:
history-system
search-system
transaction
Resource Interactions:
Resource
Interactions
Account
ActivityDefinition
AllergyIntolerance
AdverseEvent
Appointment
AppointmentResponse
AuditEvent
Basic
Binary
BodySite
Bundle
CapabilityStatement
CarePlan
CareTeam
read,vread,update,patch,delete,history-instance,history-type,create,search-type
All
read
vread
update
patch
delete
history-instance
history-type
create
search-type



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

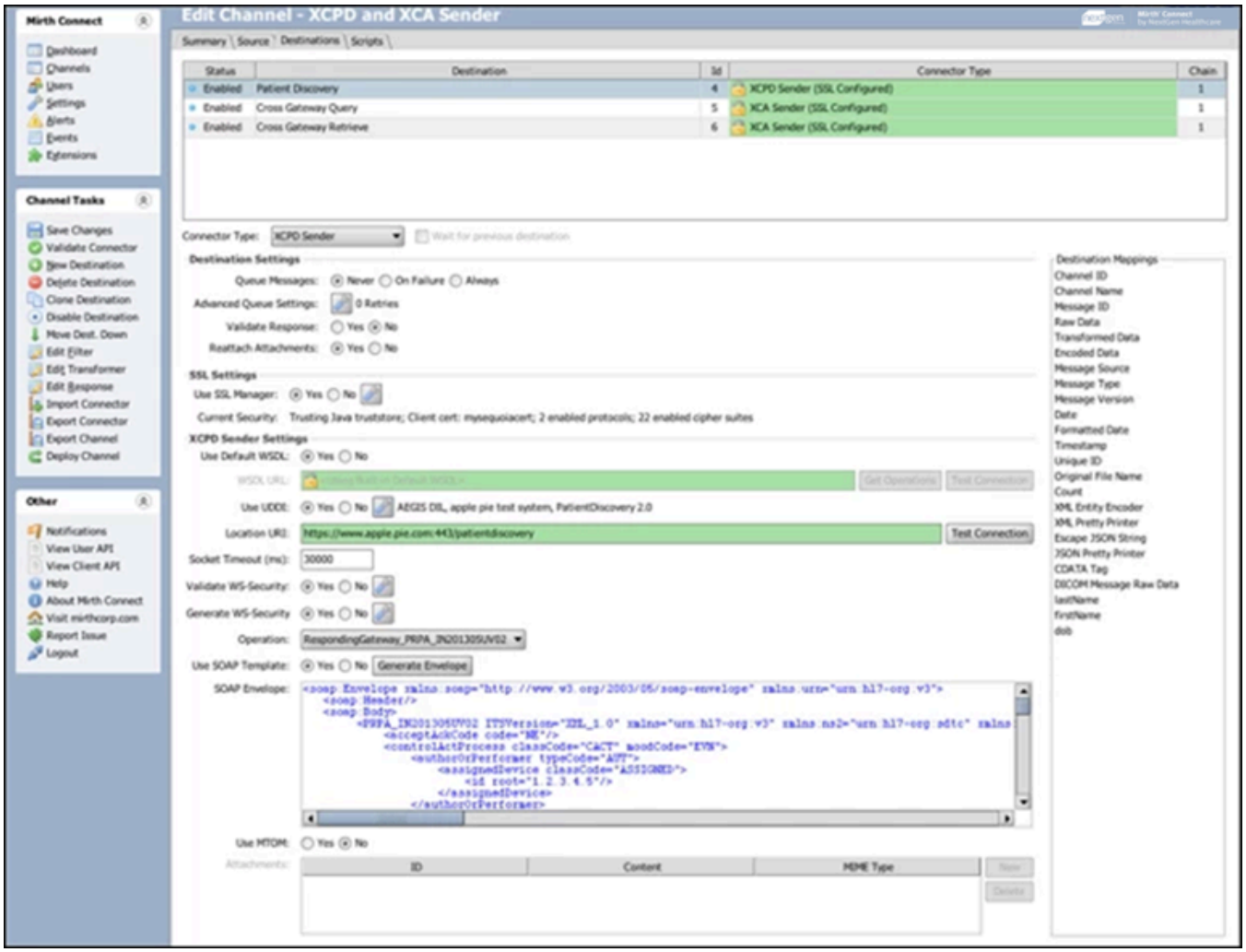
- **PDQ** (v2 and v3)
    - PRPA_IN201305UV02
    - QUQI_IN000003UV01_Cancel
    - QUQI_IN000003UV01_Continue

- **XDS.b**
    - RegistryStoredQuery
    - RegisterDocumentSet-b
    - RetrieveDocumentSet
    - ProvideAndRegisterDocumentSet-b

- **XCA**
    - CrossGatewayQuery
    - CrossGatewayRetrieve

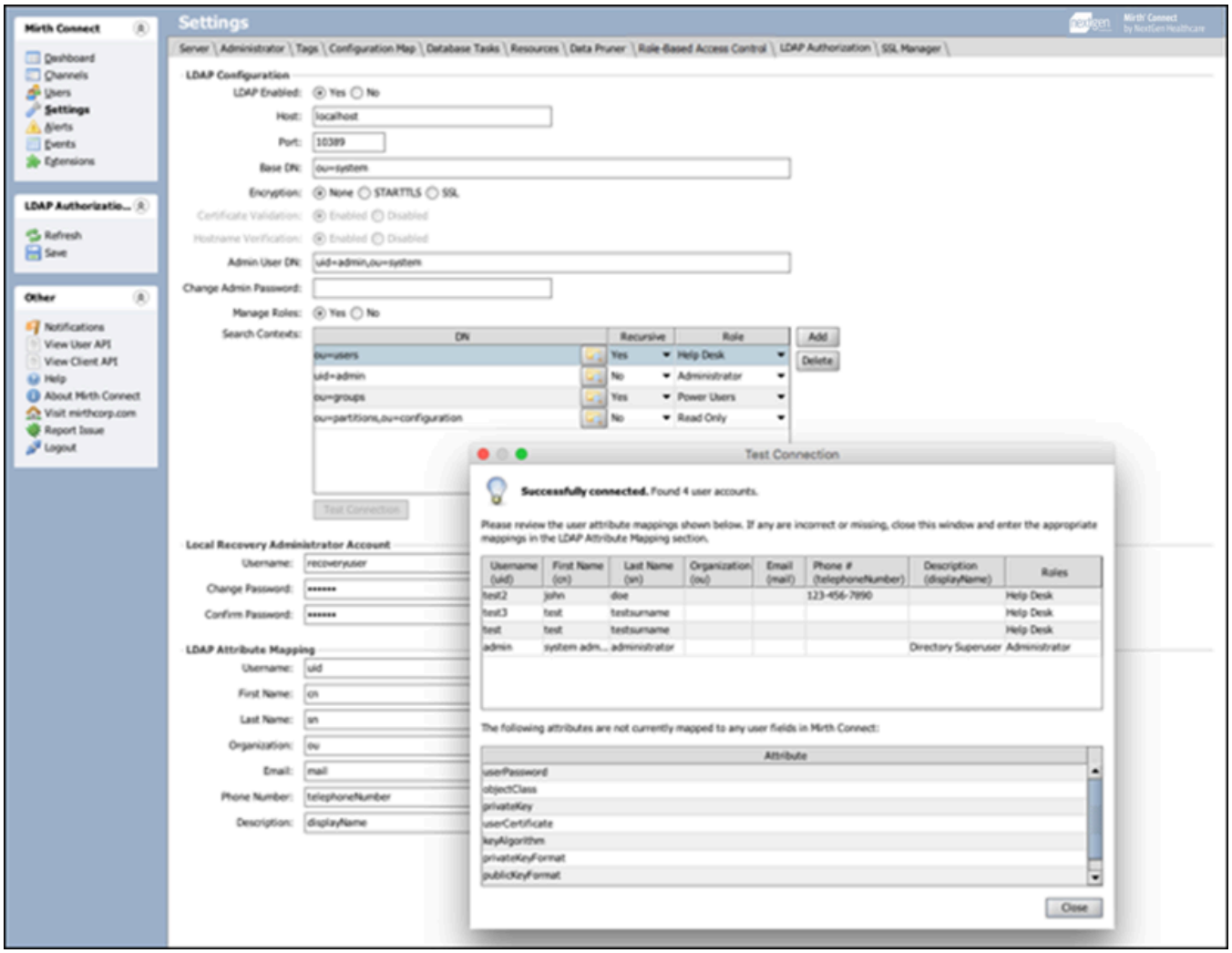
- **XCPD**
    - PRPA_IN201305UV02

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.



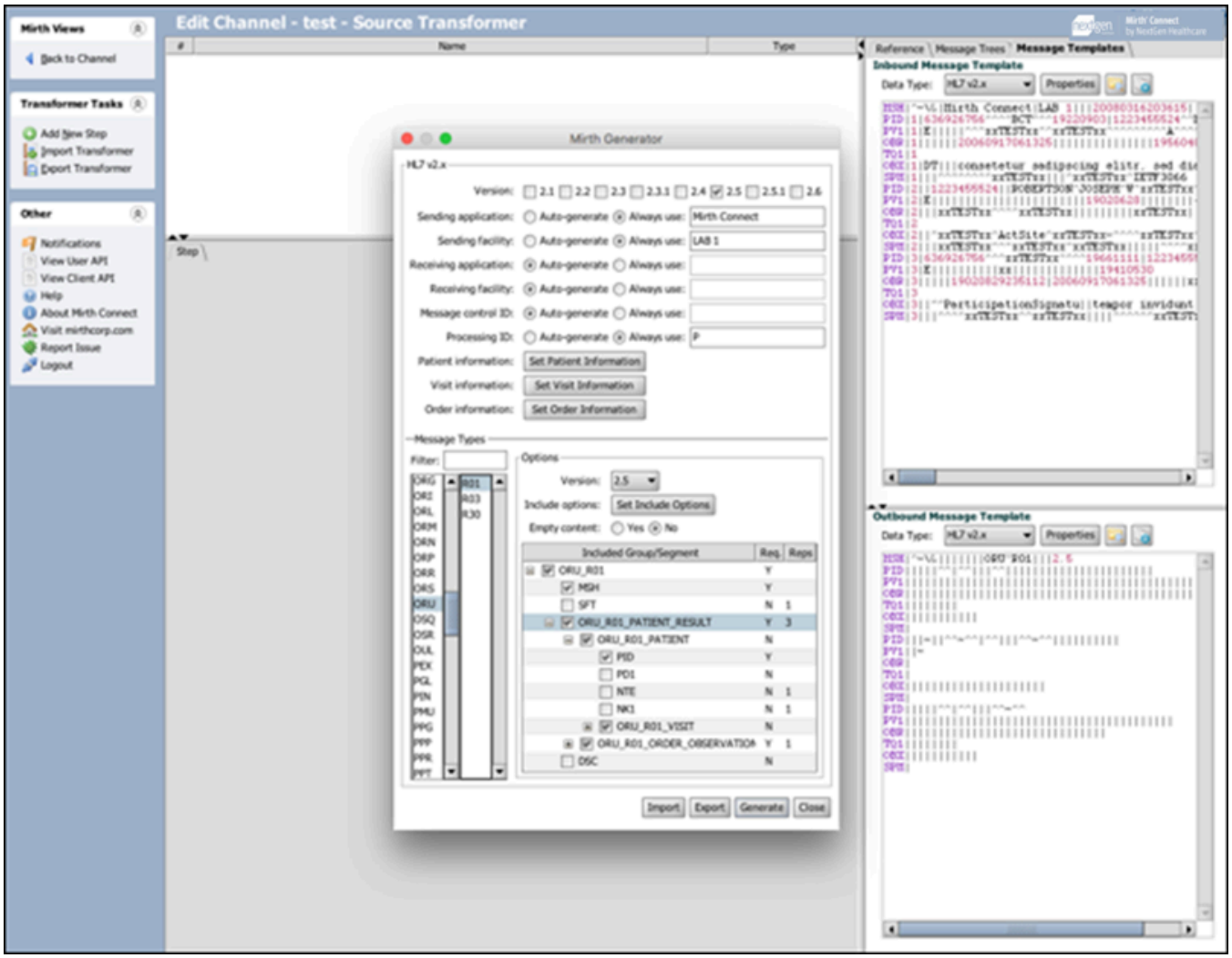
LDAP Authorization

The LDAP Authorization extension 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 Mirth® Connect. When a user logs in, Mirth® Connect copies the user's attributes from the LDAP server. The connection to the LDAP server can optionally use SSL or STARTTLS encryption. Contact Us if you have any questions.



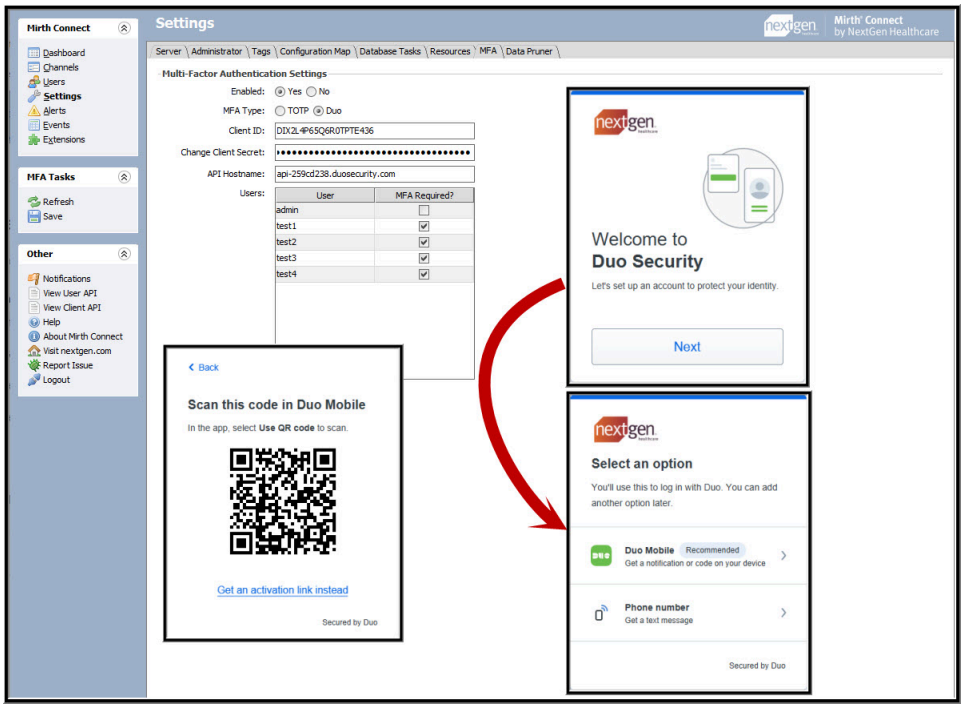
Message Generator

Using the Message Generator, you can 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, 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. Contact Us if you have any questions.



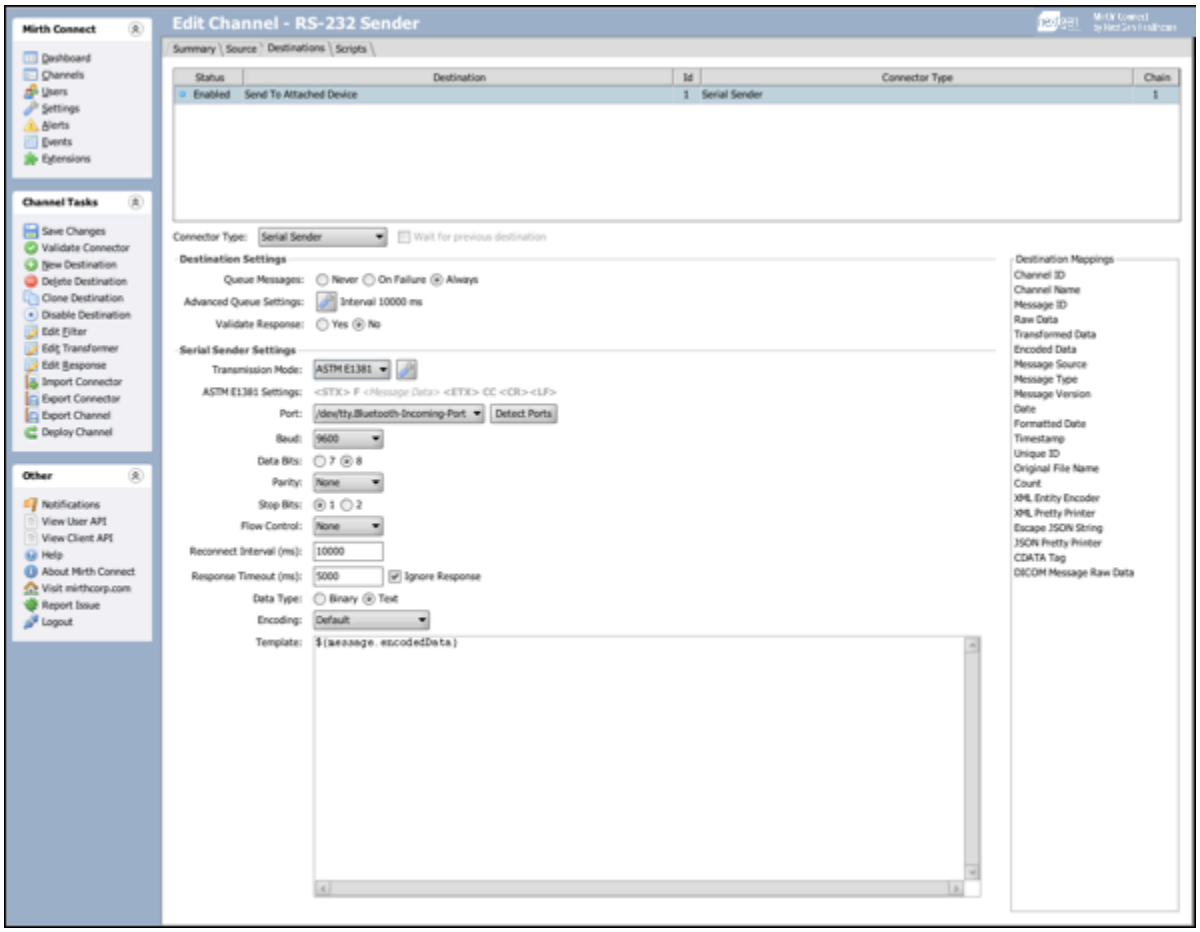
Multi-Factor Authentication

The Multi-Factor Authentication 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 log on, 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. Contact Us if you have any questions.



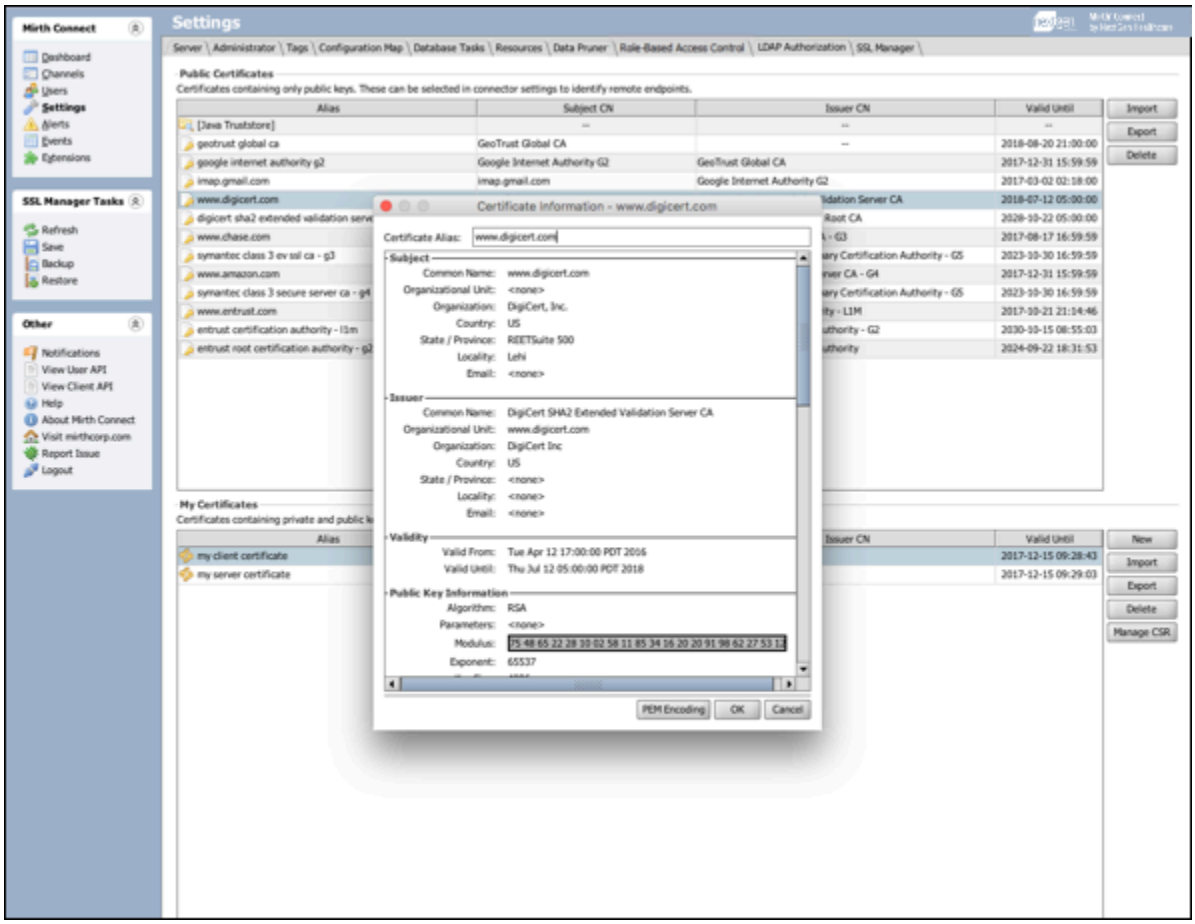
Serial Connector

The Serial Connector enables Mirth® Connect 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, enabling for fully customizable communication. This extension comes with both a Serial Listener source connector and a Serial Sender destination connector. Contact Us if you have any questions.



SSL Manager

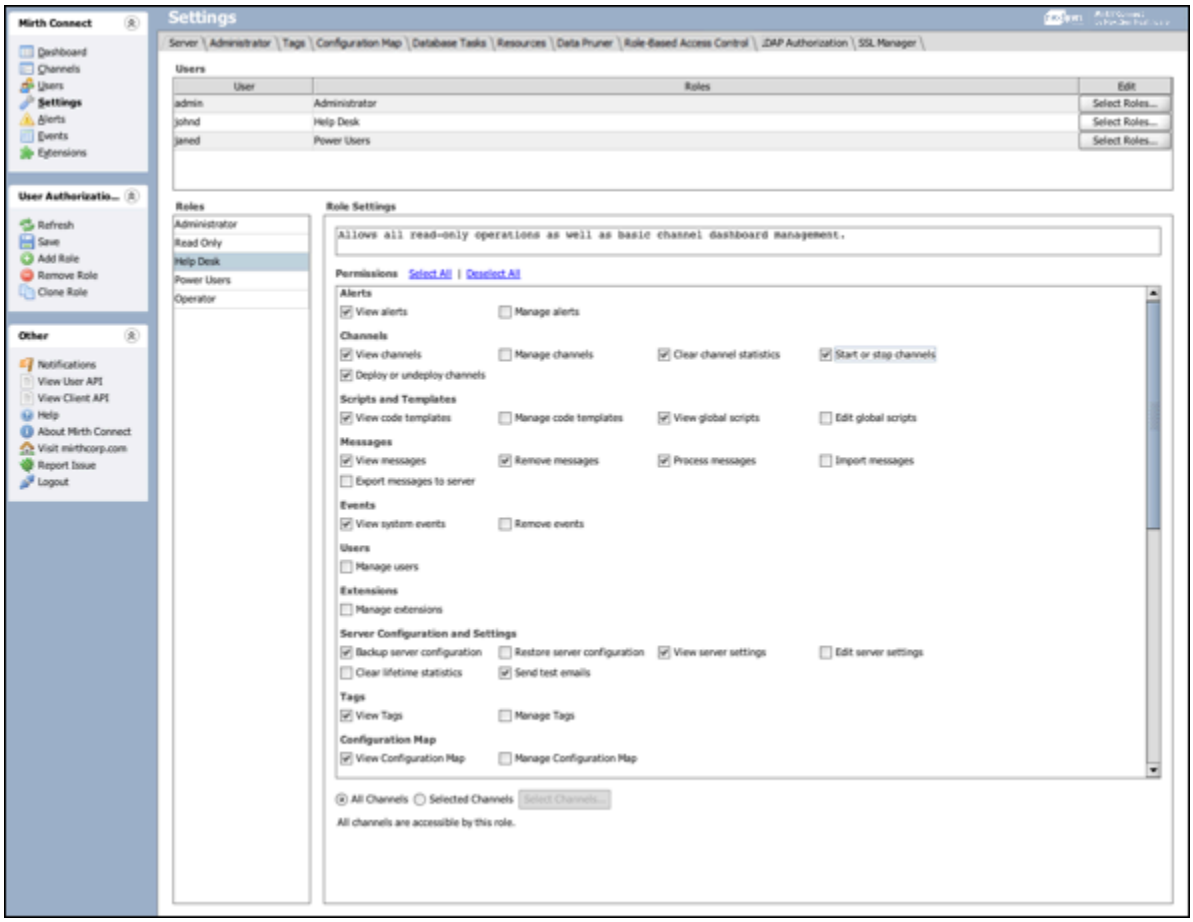
Use the SSL Manager to quickly enable and configure certificate-based SSL connectivity for socket-based connectors such as the HTTP Listener *(on page 362)* / Sender *(on page 400)*, Web Service Listener *(on page 381)* / Sender *(on page 422)*, and FTP Reader *(on page 351)* / Writer *(on page 395)*. 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. Contact Us if you have any questions.



Role-Based Access Control

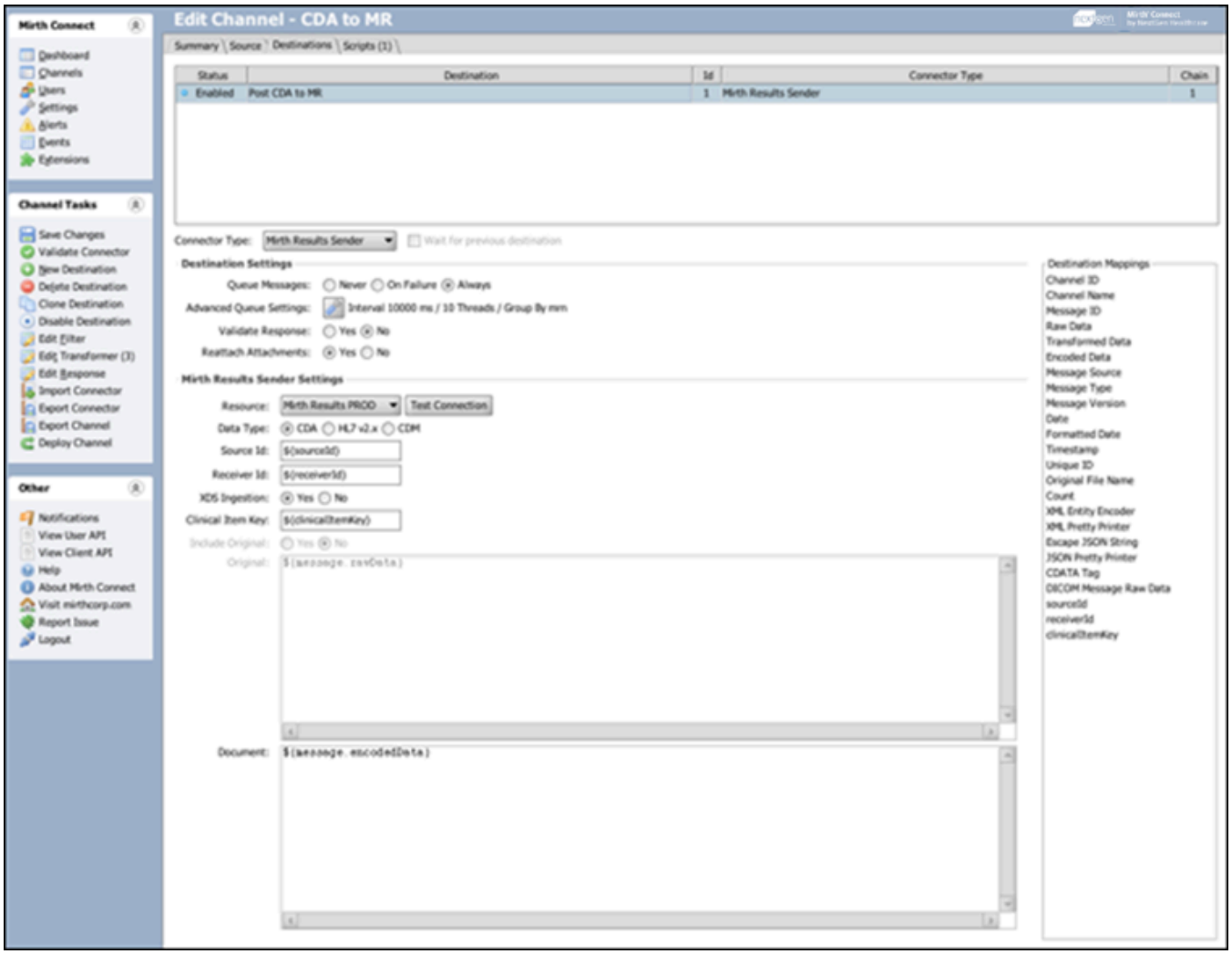
Role-Based Access Control, previously named User Authorization, provides the ability to control all aspects of the Mirth® Connect Administrator. You can create new roles with specific permissions to areas such as channel management or message browsing. You can assign any number of roles to users.

Use this process to manage access to sensitive channel and messaging data across your enterprise. Contact Us if you have any questions.



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. The extension supports posting data in a variety of formats, and comes with a built-in connection pool for persistent and highly available processing. Contact Us if you have any questions.


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 (3)
Edit Response
Import Connector
Export Connector
Export Channel
Deploy Channel
Other
Notifications
View User API
View Client API
Help
About Mirth Connect
Visit mirthcorp.com
Report Issue
Logout
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 mm
Validate Response:
Yes
No
Reattach Attachments:
Yes
No
Mirth Results Sender Settings
Resource:
Mirth Results PROD
Test Connection
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
sourceId
receiverId
clinicalItemKey

CHAPTER 21

Training

NextGen Healthcare offer two levels of Mirth® Connect Certification Training:

In Mirth® Connect **Fundamentals Certification Training**, we will cover everything from basic installation to NextGen Healthcare Connectors. You will leave with a solid grasp on Mirth® Connect. It's designed to be your first Mirth® Connect class and certification.

During Mirth® Connect **Advanced Certification Training**, for users who have completed the Mirth Fundamentals class, you will 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 Mirth® Connect? 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

Contact Us if you have any questions.

CHAPTER 22

Cures Certification

ONC Health IT Certification promotes transparency, modern standards, and enhanced health IT capabilities by fostering innovation in the health care technology ecosystem to deliver better information to patients, clinicians, and other users.

Mirth® Connect by NextGen Healthcare has received ONC Heath IT Certification in the areas described in this section.

The Health IT modules are compliant with the ONC Certification Criteria for Health IT and has been certified by an ONC-ACB in accordance with the applicable certification criteria adopted by the Secretary of Health and Human Services. This certification does not represent an endorsement by the U.S. Department of Health and Human Services.

Holds Certificate No: 15.05.05.2054.NGM2.02.01.1.231213, Certification Date: December 13, 2023

Modules Tested: 170.315 (b)(10); (d)(1-3); (d)(5-10); (d)(12-13); (g)(4,5)

Clinical Quality Measures tested: None

Additional software used: Role-Based Access Control extension, Cures Certification Support extension, Enhancement Bundle extension, SSL Manager extension, and Multi-factor Authentication (DUO, TOTP) extension

Mandatory Disclosure: Mirth® Connect extensions use a tiered annual subscription bundle called Premium Bundles (Gold or Platinum). These standardized bundles benefit customers using our cost-benefit model defined as a "per server installation" of the extensions in production with attributed 3 non-production instances. In comparison, other integration services that charge based on the total number of interfaces and qualified message counts.

All extensions required for ONC Health IT Certification can be added to the Gold or Platinum Premium tiers.

This document includes information about required extensions, features in Mirth® Connect that support certified functionality, and required actions by the user as they apply to each function.

The use of the below Mirth® Connect extensions is required to comply with certification:

- Cures Certification Support *(on page 588)*: to audit tampering of events.

- Enhancement Bundle *(on page 589)*: allows users to search for messages across multiple channels and export results.

- Multi-Factor Authentication *(on page 596)*: adds a layer of security with the option to use Duo or TOTP authentication methods.

- Role-Based Access Control *(on page 599)*: an administrator uses this extension to create roles and assign roles to users. Roles restrict system access to authorized users based on their role within the organization.

- SSL Manager *(on page 598)*: used to enable and configure certificate-based SSL connectivity.

Summary

The following list gives you a snapshot of the current certifications. Select the link in the **Criterion Name** column to see more details about the certification criterion.

Criterion Name	Required Extension
Electronic Health Information (EHI) Export (b)(10) *(on page 605)*	Enhancement Bundle *(on page 589)* Role-Based Access Control *(on page 599)*
Authentication, access control, Authorization (d)(1) *(on page 607)*	Role-Based Access Control *(on page 599)*
Auditable events and tamper-resistance (d)(2) *(on page 607)*	Cures Certification Support *(on page 588)* Role-Based Access Control *(on page 599)*
Automatic access time-out (d)(5) *(on page 609)*	None
Emergency access (d)(6) *(on page 609)*	Role-Based Access Control *(on page 599)*
End-user device encryption (d)(7) *(on page 616)*	None

Criterion Name	Required Extension
Integrity (d)(8) *(on page 616)*	None
Trusted connection (d)(9) *(on page 617)*	SSL Manager *(on page 598)*
Auditing actions on health information (d)(10) *(on page 607)*	Cures Certification Support *(on page 588)*
Encrypt authentication credentials (d)(12) *(on page 617)*	None
Multi-factor authentication (d)(13) *(on page 618)*	Multi-Factor Authentication *(on page 596)*
Quality Management System (g)(4) *(on page 619)*	None
Accessibility-Centered Design (g)(5) *(on page 619)*	Not applicable

§170.315(b)(10) Electronic Health Information (EHI) Export

(b)(10) Electronic Health Information (EHI) Export gives the user the capability to efficiently export single and multi-patient EHI in a secure and timely manner.

Note: Mirth® Connect by NextGen Healthcare is an interoperability engine and is not exposed to full patient data; instead it passes messages from one entity to another and saves only the information in the message.

Note: The CURES(b)(10) criteria requires that no psychotherapy data is included in the export. Since Mirth® Connect only passes data, it is the user's responsibility to exclude all psychotherapy data from the export.

Required Extensions:

- Role-Based Access Control *(on page 599)*
- Enhancement Bundle *(on page 589)*

Features that Support the Certification

1. Use of the Role-Based Access Control extension to limit access to protected health information

(PHI).

2. Ability to view and search for messages on multiple channels.

Required Actions

Administrator needs to create roles and assign them to each user based on the company's policies.

Single Patient Export

Patient PHI is stored in the messages that come from several entities. Each entity could have the same patient with different patient IDs. Because of this, the user must search for the PATIENT_ID and the SOURCE for each entity, resulting in multiple searches. To comply with CURES (b)(10) criteria, the user can export the XML serialized messages and include attachments during the export.

Multi-Patient Export

Patient PHI is stored in the messages that come from several entities. To comply with CURES (b)(10) criteria for searching and exporting all patients, the user must either search by SOURCE or Channel Name to get a list of all messages that have been processed by a channel. Then they can export the XML serialized messages and include attachments during the export.

Viewing Exported Attachments

The attachments extracted from HL7 can be decoded by using decode64 twice to restore the original image. The second decoding should be converted to the file type, as described below.

1. Copy the data in the XML message file between `<content>` and `</content>` after the attachment ID.
2. Use a webpage like https://www.base64decode.org to decode the data.
3. Copy the results.
4. To convert the results to the correct file format (for example PDF), paste the results in a webpage like https://base64.guru/converter/decode/pdf.

Exported DICOM messages are separated using the dmc4che library. They are separated so the data embedded in the DICOM messages can be transformed if needed. To read the DICOM message, it will need to be restored by using the dmc4che library.

§170.315(d)(1) Authentication, Access Control, Authorization

(d)(1) Authentication, Access Control, and Authorization includes limiting access by user name and password, ensuring unique user IDs, and enabling role-based access control.

Required Extensions

Role-Based Access Control *(on page 599)*

Features that Support the Certification

- Use of the Role-Based Access Control Extension
- Validation of username and passwords
- Prevention of duplicate usernames
- Creation of user roles that limit/enable viewing and managing different areas of the application
- Disable user accounts as needed

Required Actions

- Administrator needs to maintain/update valid/invalid users
- Administrator needs to create roles and assign them to each user based on company's policies

§170.315(d)(2) Auditable Events and Tamper-Resistance, §170.315(d)(3) Audit Report(s), and §170.315(d)(10) Auditing Actions on Health Information

These criteria are interrelated and therefore addressed together.

(d)(2) Auditable events and tamper-resistance establishes which actions must be audited, what must be included in the audit records, and required protections around the audit log, including detecting if the audit log has been changed outside of normal processes.

(d)(10) Auditing actions on health information establishes which actions must be audited, what must be included in the audit records, and required protections around the audit log, including detecting if the audit log has been changed outside of normal processes.

(d)(3) Audit Report(s) requires the ability of health information systems to sort the audit logs.

Note: In Mirth® Connect by NextGen Healthcare, the audit log is referred to as Event Log/ Browser.

Required Extensions

- Role-Based Access Control *(on page 599)*
- Cures Certification Support *(on page 588)*

Features that Support the Certification

1. Use of the Role-Based Access Control Extension to limit access to PHI, and the Cures Certification Support Extension to audit tampering of events.
2. An event is created when any of the following occurs in a channel that contains metadata PATIENT_ID (see the Required Actions section).
    - A message that contains PHI is viewed. Event name is “Accessed PHI”.
    - Messages that contain PHI are queried. Event name is “Queried PHI”.
3. In addition, an event is created when any of the following occurs in any channel:
    - A single message is removed. Event name is “Remove message”.
    - All messages are removed. Event name is “Remove all messages”.
    - Messages are pruned. Event name is “Data Pruner”.
4. Events are also created when modifications are made to privileges in the Role Based Access Control tab.
5. Events Browser:

- The following columns have been added: **Channel ID – Message ID**, **Channel Name**, and **Patient ID**.
- Events can be sorted by any column in the Events Browser in ascending and descending order.
- The functionality “Remove all events” is no longer available.

6. With the use of the Cures Certification Support Extension, every 24 hours event logs are audited. If issues are found, notifications are displayed in the Events browser, the Server Log tab, and in the Mirth log file.

Required Actions

- Administrator needs to create roles and assign them to each user based on company policies.
- To fully use the auditing functionality for PHI related audits, a channel must include a **PATIENT_ID** metadata.

§170.315(d)(5) Automatic Access Time-out

Required Extensions

None

Features that Support the Certification

In the Administrator Server settings there is an option to enable auto logout.

Required Actions

Use the “Enable Auto Logout” option in the Server Settings tab. For additional information, see Server Settings Tab *(on page 167)*.

§170.315(d)(6) Emergency Access

(d)(6) Emergency access is intended to allow a pre-determined set of users to have ‘admin’ or increased

access/permissions during an emergency.

Required Extensions

Role-Based Access Control

Features that Support the Certification

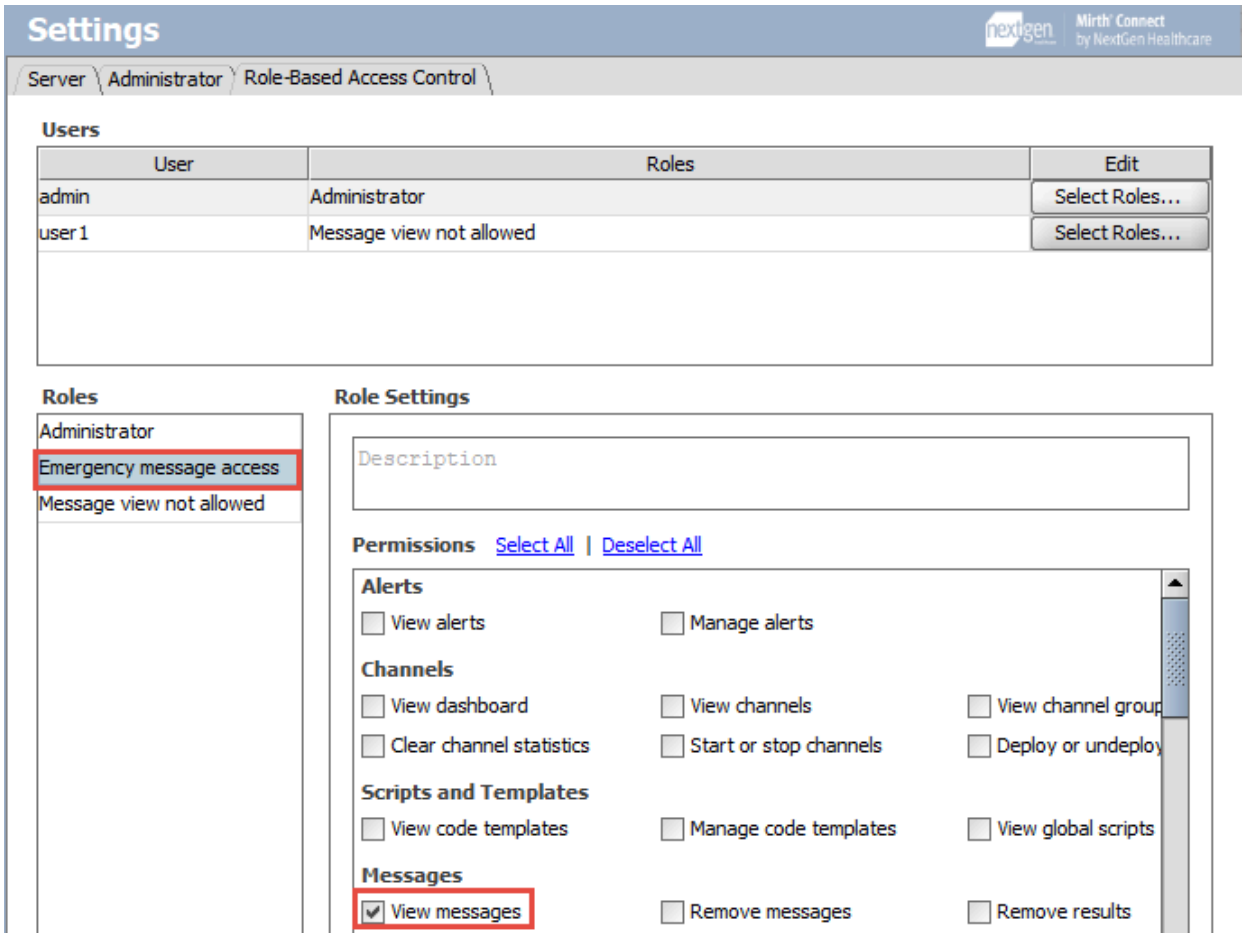
Role-Based Access Control Extension

Required Actions

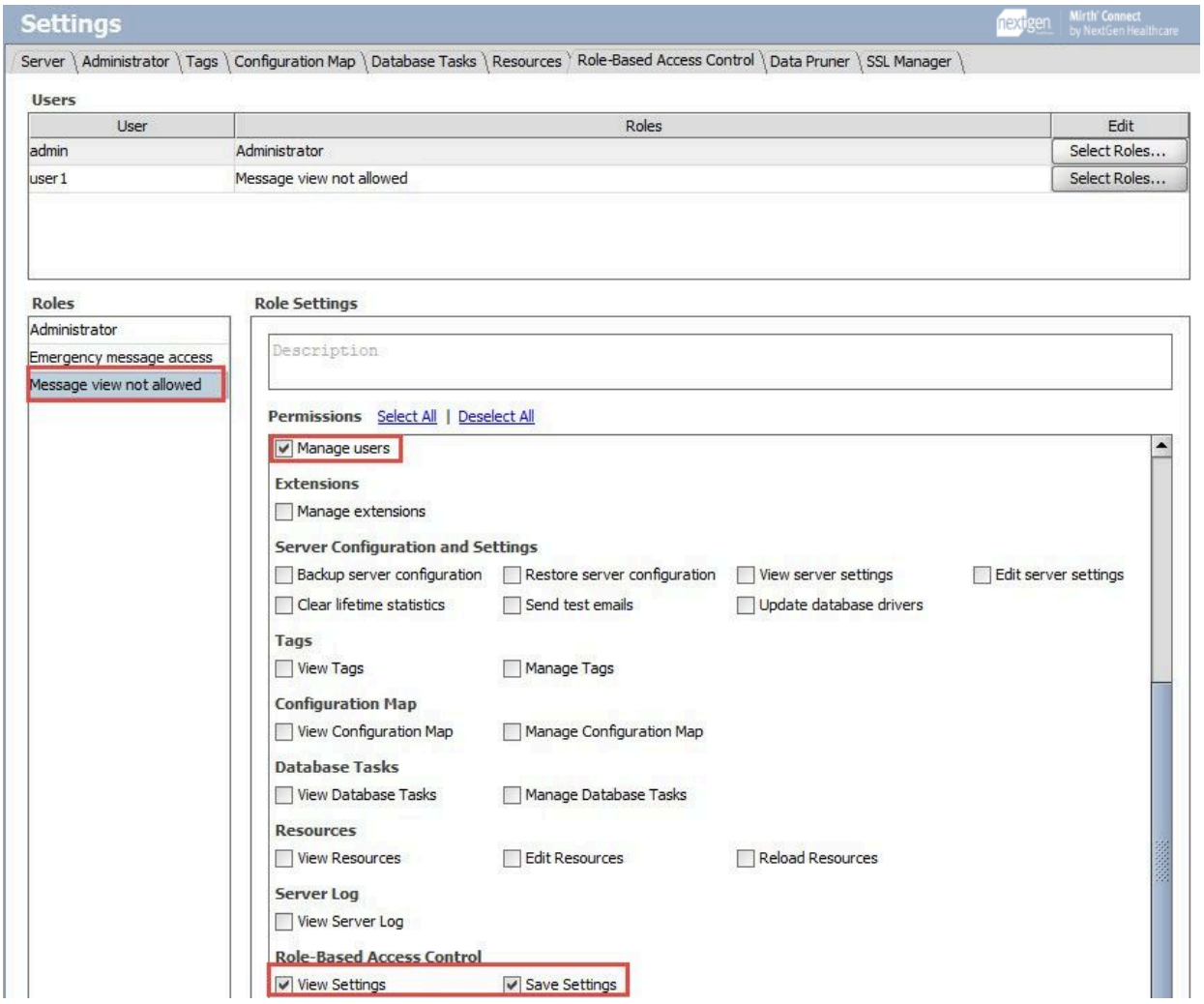
Create an “emergency” role that specific users can temporarily assign to themselves so that they can view otherwise restricted information.

Example

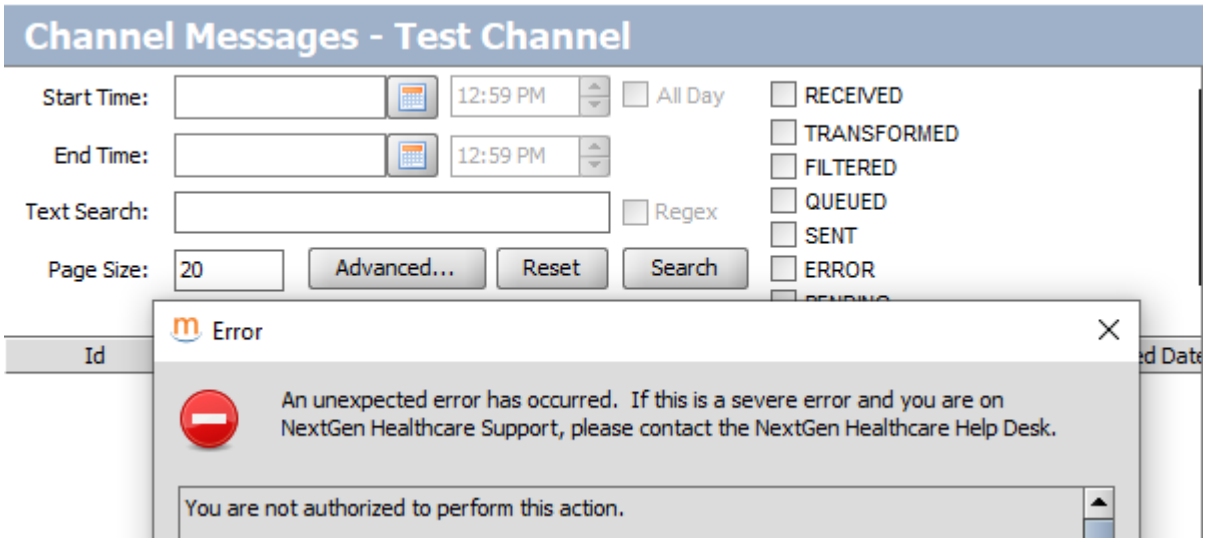
1. We created an **Emergency message access** role, with **View Messages** as the only permission.



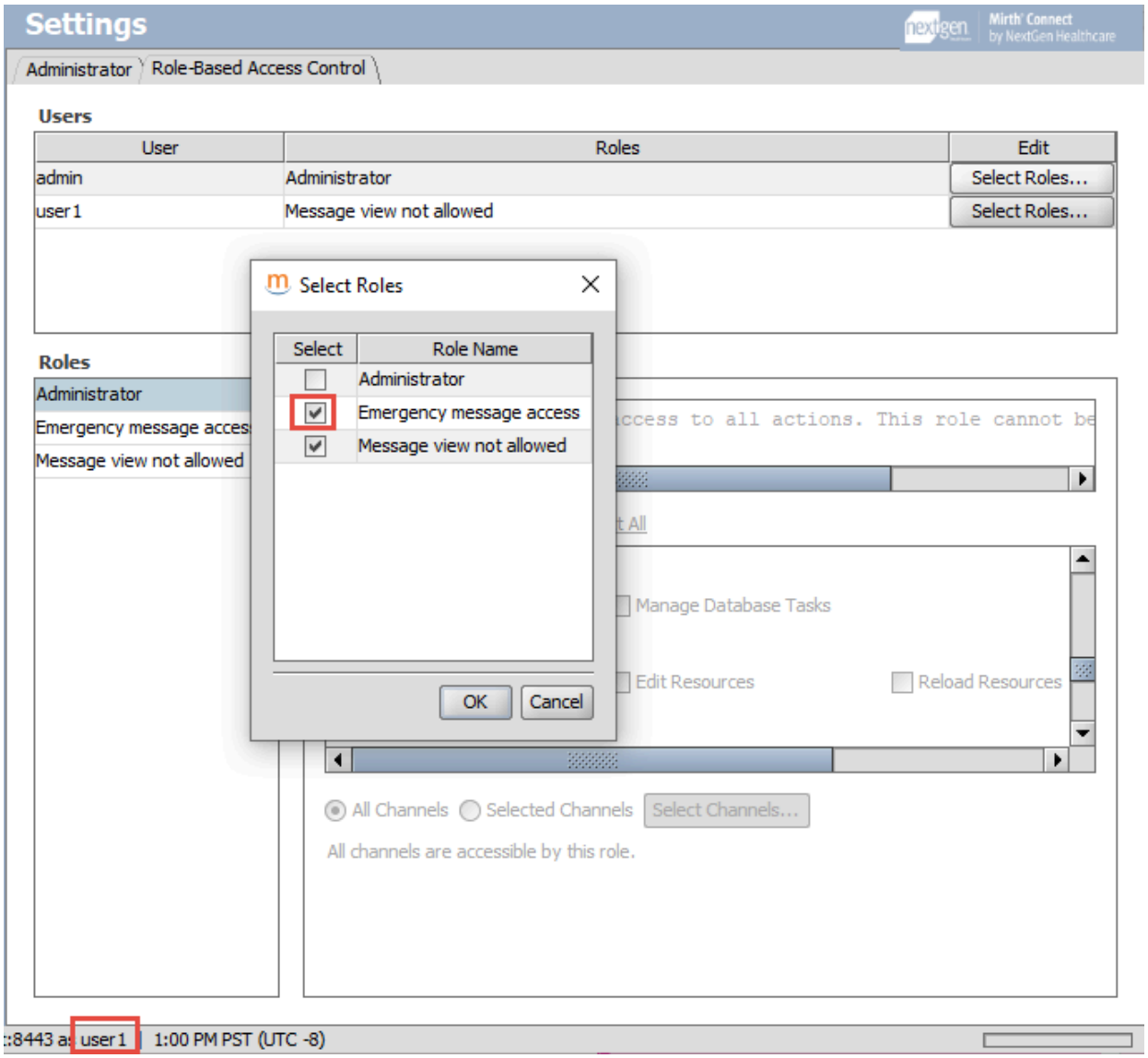
2. We have **user1** to whom the role **Message view not allowed** is assigned. Notice that this role does not have any Messages-related permission. It does have permissions to **Manage users** and to **Role-Based Access Control** settings.



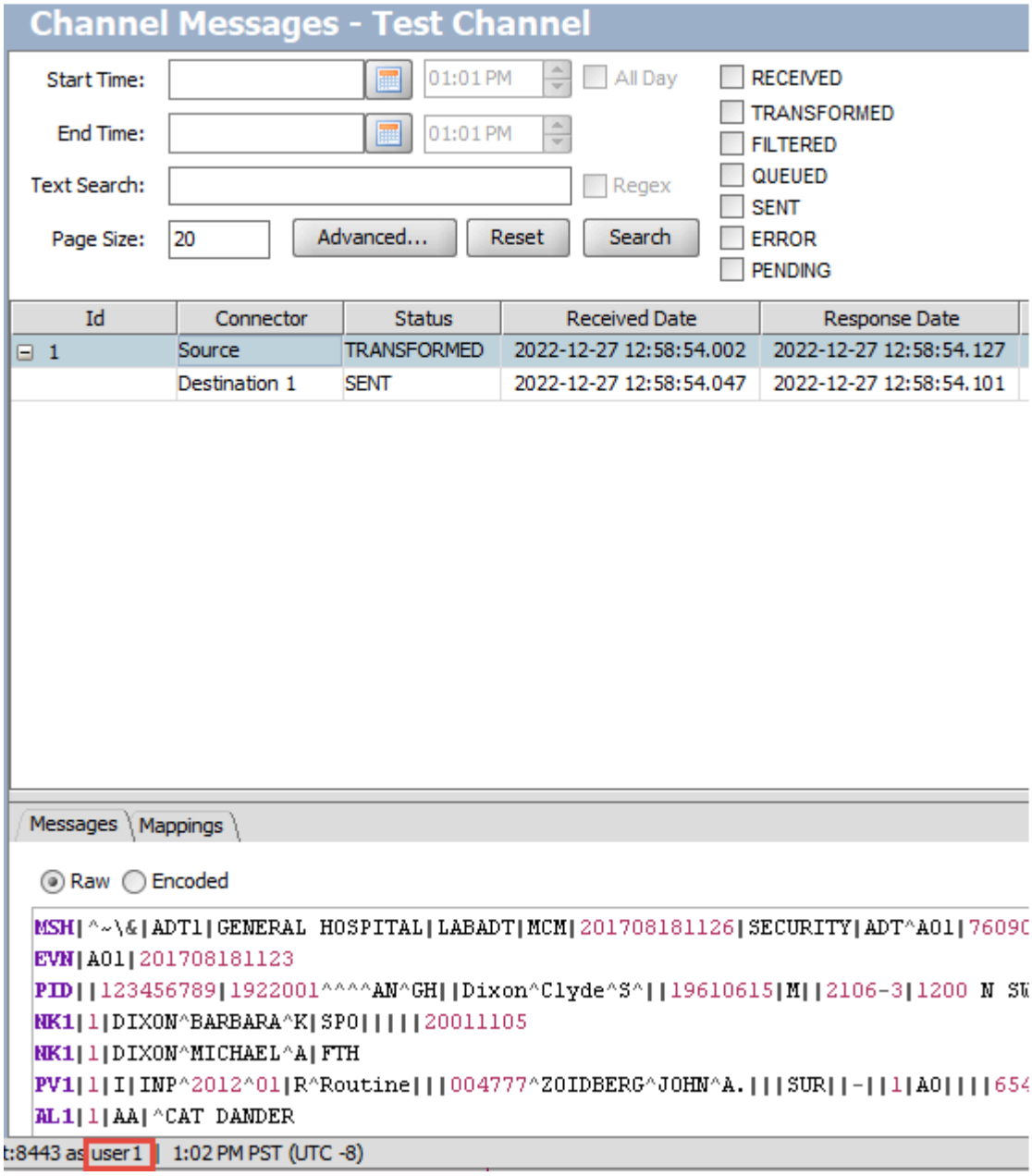
3. When user1 attempts to view messages, the application displays an error.



4. In the Role-Based Access Control, user1 assigns to themselves the role **Emergency message access** in addition to their current role:



5. When the setting is saved, user1 is automatically logged out and prompted to log in again. After logging on, user1 is allowed to view messages.



Note: Changes in role assignments are listed in the Events browser.

§170.315(d)(7) End-User Device Encryption

(d)(7) End-user device encryption ensures that data is not stored in an non-secure manner on local devices.

Required Extensions

None

Features that Support the Certification

Mirth® Connect does not store message information in local cache.

Required Actions

None

§170.315(d)(8) Integrity

(d)(8) Integrity requires that a message digest can be created and sent using a hashing algorithm of SHA-2 security strength or greater.

Required Extensions

None

Features that Support the Certification

- A SHA-256 hash variable (named **Message Hash**) is automatically generated for each message sent out by a destination connector. If sending the hash, the recipient of the message and the hash can validate the content of the message. For additional information, see Destination Mappings *(on page 253)*.
- New utility functions have been added to create a message hash, supporting the filtering of messages either in a source or destination connector. They are:

- HashUtil.generate(object);
- HashUtil.generate(string, encoding, algorithm);
- HashUtil.generate(byte[], algorithm);

Required Actions

- Use of the Message Hash variable.
- Use the hash HashUtil.generate functions to compare received with generated hash.

See examples in the **Message Integrity** section of Channel Development Best Practices and Tips *(on page 509)*.

§170.315(d)(9) Trusted Connection

(d)(9) Trusted connection requires the ability to encrypt messages and send through a trusted connection.

Required Extensions

SSL Manager *(on page 598)*

Features that Support the Certification

The SSL Manager extension provides options that ensure encryption and connectivity validation through the use of SSL certificates.

Required Actions

In your Mirth® Connect channels, enable SSL to establish trusted connections between endpoints.

§170.315(d)(12) Encrypt Authentication Credentials

(d)(12) Encrypt authentication credentials requires encryption of stored authentication credentials.

Required Extensions

None

Features that Support the Certification

- SHA256 is the default algorithm used by Mirth® Connect by NextGen Healthcare that creates salted hash values for user passwords. This value is stored in the database, not the user password. SHA256 is approved for Cures, as are SHA384 and SHA512.
- If a different algorithm is desired, the mirth.properties file can be modified using the digest.algorithm property. For example: digest.algorithm = SHA384

Required Actions

None

§170.315(d)(13) Multi-Factor Authentication

(d)(13) Multi-factor authentication requires that the system mandates access with multi-factor authentication.

Required Extensions

Multi-Factor Authentication *(on page 596)*

Features that Support the Certification

Multi-Factor Authentication Extension

Required Actions

Using the Mirth® Connect Multi-Factor Authentication Extension with Duo or TOTP, set up all users to be required to use the selected authentication method.

§170.315(g)(4) Quality Management System

(g)(4) Quality Management System (QMS) is a set of policies, processes, and procedures required for planning and executing in the core business area of an organization (that is, areas that can impact the organization's ability to meet customer requirements). All certified products are required to adhere to this requirement. Mirth® Connect uses CFR 21 Part 820, a recognized federal government QMS, in the development, testing, implementation, and maintenance for all certified criteria.

Mirth® Connect is certified to meet the (g)(4) Quality Management System requirement.

§170.315(g)(5) Accessibility-Centered Design

(g)(5) 170.315 Accessibility-Centered Design (ACD) is an optional requirement that focuses on certified products using ACD standards in developing, testing, implementing, and maintaining certified functionality.

Currently, Mirth® Connect does not use accessibility-centered design for certified functions.