



# Advanced Topics on the Mirth Connect Interface Engine

July 6, 2016



You have been automatically muted. Please use the Q&A panel to submit questions during the presentation

The screenshot shows the Cisco WebEx Event Center interface. The main content area displays a presentation slide with the GALEN Healthcare Solutions logo and the text "MUCH MORE THAN I.T. POSSIBILITY Welcome to Today's Webcast The webcast will begin shortly...". The slide features a background image of a car driving on a dirt road.

Annotations with red arrows point to the following elements:

- Click for Full Screen Mode:** Points to the full screen icon (a square with a diagonal line) in the top toolbar.
- Click to open Q&A Panel:** Points to the Q&A icon (a blue square with a white question mark) in the top right corner.

The Q&A panel is open on the right side of the screen, showing a list of questions under the heading "Q&A All (0)". The panel includes a search bar, a dropdown menu for "Ask: All Panelists", and a text input field for submitting a question. A "Send" button is located at the bottom right of the panel. A red arrow points from the Q&A icon to the text input field.

At the bottom left, the Cisco logo is visible. At the bottom right, the status "Connected" is shown with a green dot.

# PRESENTER



Nate Bessa  
Technical Consultant

- Based in Boston, MA
- Specializing in custom programming projects, HIE connectivity, data migrations, and HL7 interface development
- 2+ years experience with Mirth Connect
- 5+ years in healthcare IT

[Nate.Bessa@galenhealthcare.com](mailto:Nate.Bessa@galenhealthcare.com)

# POLL #1

# TOPICS

- I. Performance tweaks
- II. Security improvements
- III. Programming suggestions
- IV. Maintenance strategies
- V. Q & A



# PERFORMANCE TWEAKS

# CHANNEL METADATA

**Edit Channel - ADT A04 - Patient Registration - Source Transformer**

#	Name	Type
1	Add important data to channel map variables for use as message metadata	JavaScript

Step

```

1 var patientId = msg['PID']['PID.3']['PID.3.1'].toString();
2 var attendingPhysicianLastName = msg['PV1']['PV1.7']['PV1.7.1'].toString();
3 var attendingPhysicianFirstName = msg['PV1']['PV1.7']['PV1.7.2'].toString();
4 var insuranceProvider = msg['IN1']['IN1.4']['IN1.4.1'].toString();
5
6 // Save to channel map variables the data that we want to serve as message metadata
7 channelMap.put('PatientID', patientId);
8 channelMap.put('AttendingPhysician', attendingPhysicianFirstName + " " + attendingPhysicianLastName);
9 channelMap.put('InsuranceProvider', insuranceProvider);
  
```

**Custom Metadata**

Column Name	Type	Variable Mapping	
PATIENT_ID	STRING	▼ PatientID	Add
ATTENDING_PHYSICIAN	STRING	▼ AttendingPhysician	Delete
INSURANCE	STRING	▼ InsuranceProvider	
			Revert

Add clarity to your message logs and minimize data storage needs

**Channel Messages - ADT A04 - Patient Registration**

Start Time: 09:34 AM ☐ All Day ☐ RECEIVED ☐ TRANSFORMED ☐ FILTERED ☐ QUEUED ☐ SENT ☐ ERROR

End Time: 09:34 AM

Text Search:  ☐ Regex

Page Size: 20

Current Search: Max Message ID: 19 Date Range: (any) to (any) Statuses: (any)

Results 1 - 3 of 3 Page 1 of 1

Id	Connector	Status	Received Date	PATIENT_ID	ATTENDING_PHYSICIAN	INSURANCE
19	Source	TRANSFORMED	2016-06-24 13:37:51:387	22333	John Doctor	Medicare
	Send HL7 to external vendor	SENT	2016-06-24 13:37:51:393	22333	John Doctor	Medicare
18	Source	TRANSFORMED	2016-06-24 13:37:24:727	31231	Nurse Test	Blue Cross Blue Shield
	Send HL7 to external vendor	SENT	2016-06-24 13:37:24:737	31231	Nurse Test	Blue Cross Blue Shield
17	Source	TRANSFORMED	2016-06-24 13:36:44:177	1234	John Doctor	Blue Cross Blue Shield
	Send HL7 to external vendor	SENT	2016-06-24 13:36:44:183	1234	John Doctor	Blue Cross Blue Shield

# CHANNEL METADATA

But most of all, speed up your message searches big time.

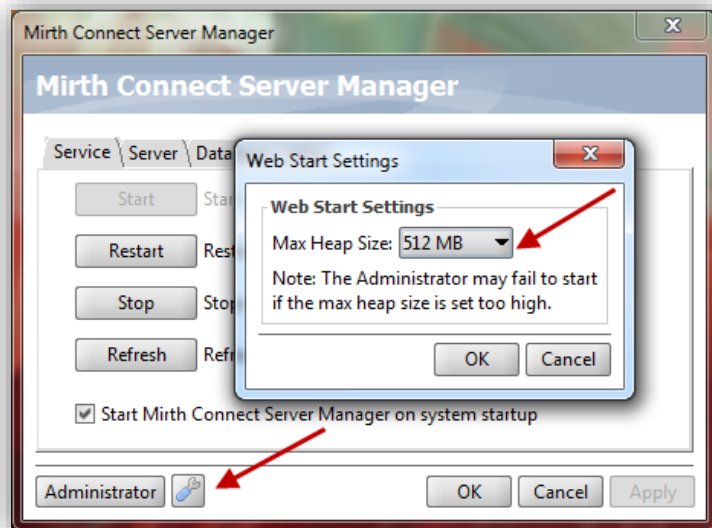
The screenshot shows the Mirth Connect Administrator interface. On the left, the 'Channel Messages - ADT A04' window displays a list of messages. The 'Advanced...' button is highlighted with a red box. On the right, the 'Advanced Search Filter' dialog is open. It features a table of filters with columns 'Id', 'Current Connector Name', and 'Included'. Below this, there are input fields for 'Message Id', 'Original Id', 'Import Id', and 'Server Id'. There are also dropdowns for 'Send Attempts' and checkboxes for 'Has Attachment' and 'Has Error'. A 'Content Type' section contains a 'Contains' table with columns 'Metadata', 'Operator', 'Value', 'Ignore Case', and 'New'. The 'INSURANCE' row is highlighted with a red box. At the bottom, there are 'OK' and 'Cancel' buttons.

Id	Current Connector Name	Included
0	Source	<input checked="" type="checkbox"/>
1	Send HL7 to external vendor	<input checked="" type="checkbox"/>
--	Deleted Connectors	<input checked="" type="checkbox"/>

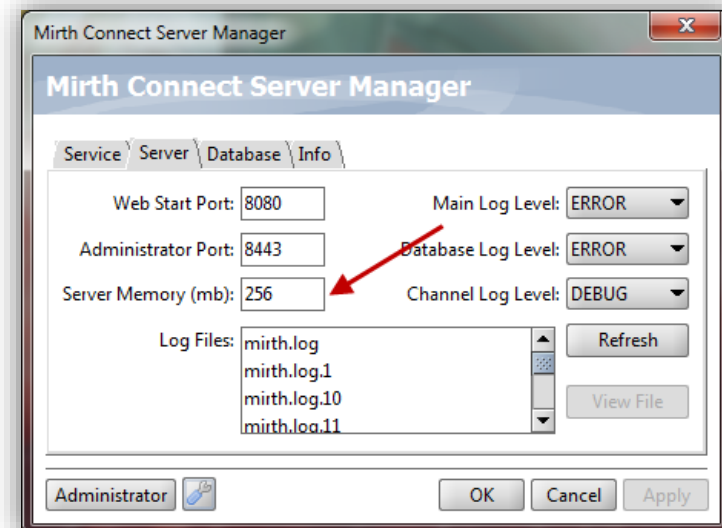
Metadata	Operator	Value	Ignore Case	New
INSURANCE	CONTAINS	Blue Cross	<input type="checkbox"/>	Delete



# INCREASING HEAP MEMORY



The Heap Size is the amount of memory made available for your Mirth Connect front end application. This plays a role especially when searching for messages in channels with lots of traffic.



The Server Memory setting dictates the amount of memory that Mirth Connect as an engine will be able to use in the background. This allows Mirth to process large messages and handle heavy traffic.



# SECURITY IMPROVEMENTS

# ENCRYPT PLAIN-TEXT CONFIG PASSWORDS

Mirth Connect Server Manager

Service Server Database Info

Type: sqlserver

URL: jdbc:jtds:sqlserver://localhost:1433/mirth

Username: sa

Password: .....

Administrator Ok Cancel Apply

```

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

```

```

32
33 # Enabled encryption of properties with passwords.
34 encryption.properties = 1
35

```

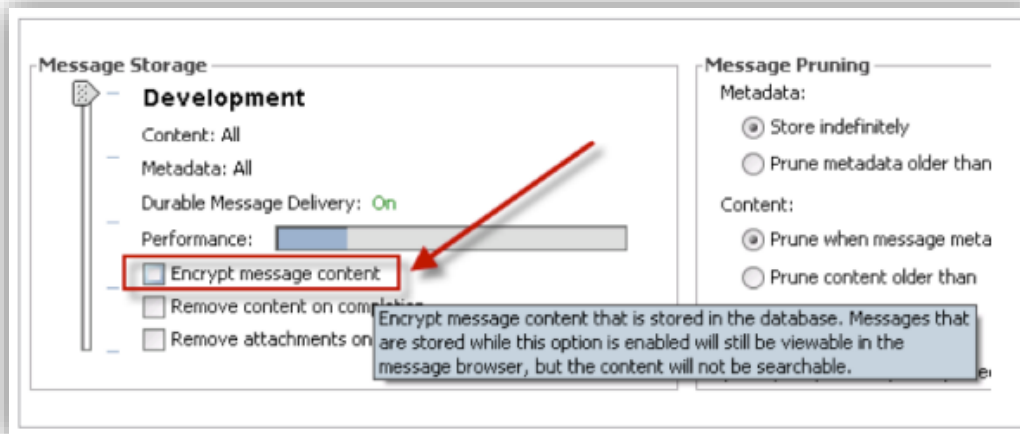
Add this to your mirth.properties file and restart the Mirth Connect service.

```

81
82 # database credentials
83 database.username =sa
84 database.password =(enc)O8HCUSpgBBesW6gwE+nTa2Q8AUNLoYwChecqkjyWpsM=\r\n
85

```

# ENCRYPT MESSAGE STORAGE



For added security, you can encrypt message content stored in the database.

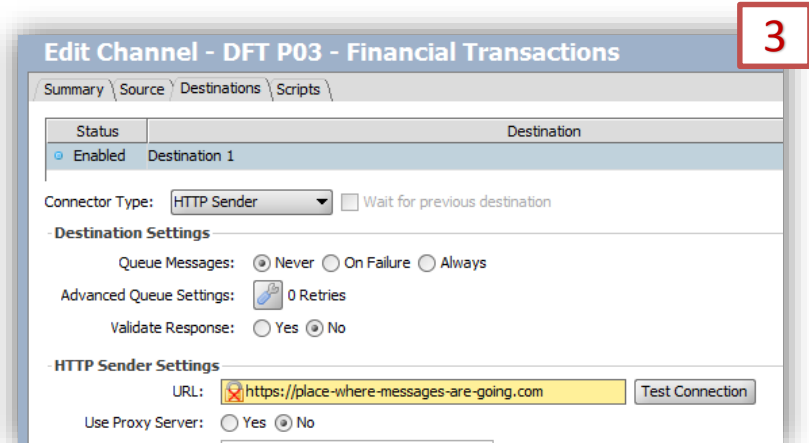
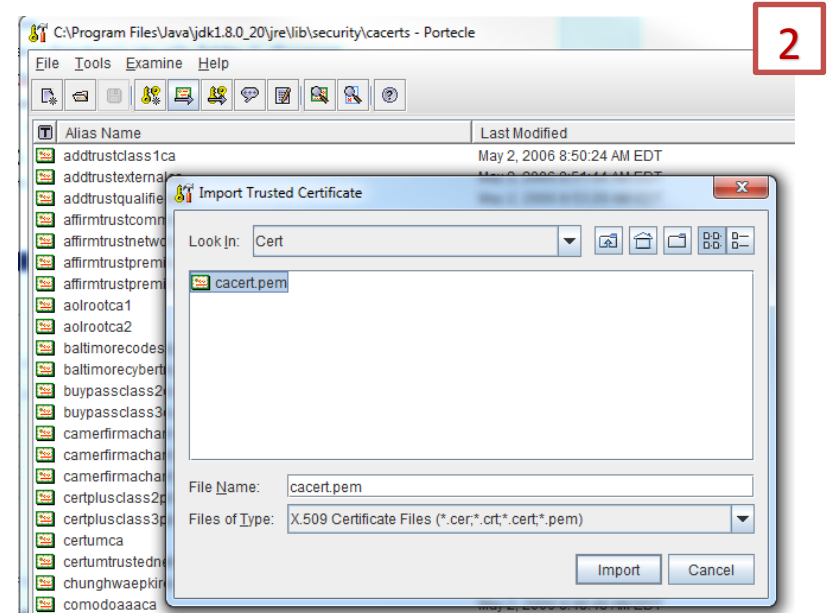
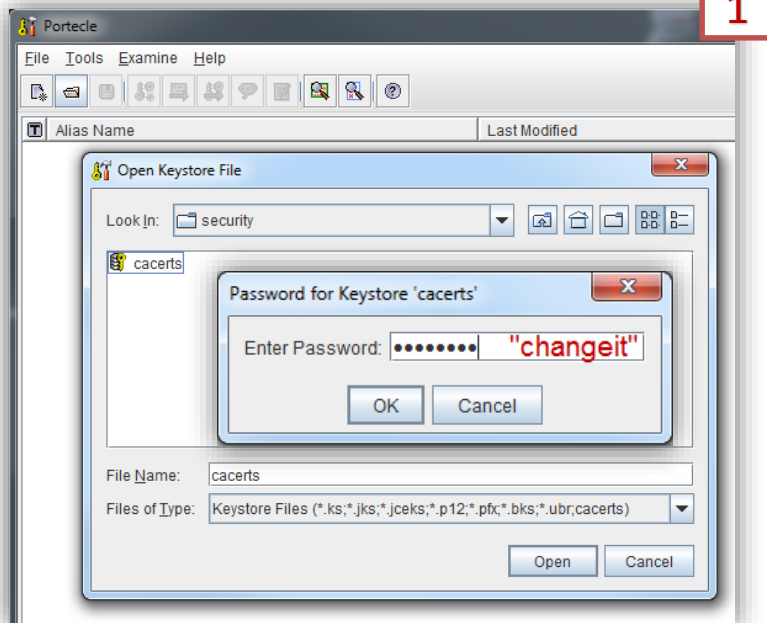
If you choose to use this feature, add these settings to your mirth.properties file and restart the Mirth service.

```

23
24 # The algorithm to use for encryption (DES, AES, etc.)
25 encryption.algorithm = AES
26
27 # The key length
28 encryption.keylength = 256
29
30 # Enables encryption/decryption of export/import through the Administrator.
31 encryption.export = 1
32
33 # The algorithm to use for one-way hashing (MD5, SHA, etc.)
34 digest.algorithm = MD5
35
36 # The security provider to use for all encryption and hashing.
37 security.provider = org.bouncycastle.jce.provider.BouncyCastleProvider
38

```

# IMPORTING A SELF-SIGNED SSL CERTIFICATE



# PROGRAMMING SUGGESTIONS

## POLL #2

# MOVE FUNCTIONS TO CODE TEMPLATES

**Code Templates**

Name	Converts
Miscellaneous	
Format Date For HL7	Converts

1 Library, 1 Code Template

Library: Miscellaneous

Type: Function

Code:

```

1  /**
2   * Converts a date in the YYYY-MM-DD HH:MM:SS format to the HL7 standard (YYYYMMDDHHMMSS)
3   *
4   * @param {String} date - string representing date in YYYY-MM-DD HH:MM:SS format
5   * @return {String} return string representing date in YYYYMMDDHHMMSS format (HL7 standard)
6   */
7  function formatDateForHL7(datetime)
8  {
9      // If the value given is an empty string return an empty string
10     if (datetime.length <= 0)
11         return "";
12
13     // Check that the given timestamp is at least 19 characters long (YYYY-MM-DD HH:MM:SS)
14     if (datetime.length < 19)
15         throw ("Code Template formatDateToHL7 - Error: Given datetime of " + datetime + " does not " +
16             "have at least 19 characters. Format should be YYYY-MM-DD HH:MM:SS[.000]");
17
18     var year = datetime.substr(0,4);
19     var month = datetime.substr(5,2);
20     var day = datetime.substr(8,2);
21     var hours = datetime.substr(11,2);
22     var minutes = datetime.substr(14,2);
23     var seconds = datetime.substr(17,2);
24
25     var formattedDate = year + month + day;
26
27     // Only include the time if there was one provided
28     if (hours != "00" || minutes != "00" || seconds != "00")
29         formattedDate += hours + minutes + seconds;
30
31     return formattedDate;
32 }
33

```

**Edit Channel - DFT P03 - Financial Transactions - Source Transformer**

#	Name
0	Format patient date of birth in HL7 standard

Step

```

1  // Copy the incoming HL7 message to a variable we can manipulate
2  tmp = msg;
3
4  // Change patient date of birth from YYYY-MM-DD HH:MM:SS format to YYYYMMDDHHMMSS
5  var patientDateOfBirth = msg['PID.7']['PID.7.1'].toString();
6  var patientDateOfBirthInHL7Format = formatDateForHL7(patientDateOfBirth);
7
8  // Place newly formatted date of birth in the outgoing HL7 message
9  tmp['PID.7']['PID.7.1'] = patientDateOfBirthInHL7Format;
10

```

**Edit Channel - ADT A04 - Patient Registration - Source Transformer**

#	Name
0	Format insurance expiration date in HL7 standard

Step

```

1  // Copy the incoming HL7 message to a variable we can manipulate
2  tmp = msg;
3
4  // Change insurance expiration date from YYYY-MM-DD HH:MM:SS format to YYYYMMDDHHMMSS
5  var insuranceExpirationDate = msg['IN1.13']['IN1.13.1'].toString();
6  var insuranceExpirationDateInHL7Format = formatDateForHL7(insuranceExpirationDate);
7
8  // Place newly formatted insurance expiration date in the outgoing HL7 message
9  tmp['IN1.13']['IN1.13.1'] = insuranceExpirationDateInHL7Format;
10

```

Allows you to edit a function in one location instead of having to make changes in every channel where you are using.

Also lets you keep your channels simple by moving long and complicated code out of them and into the code templates screen.



# MODIFY AN HL7 MESSAGE WITH DATABASE DATA

Edit Channel - ADT A04 - Patient Registration - Source Trans

#	Name
0	Find the patient's MRN from our database and add it to the HL7 message

Step \

```

1 // Open a database connection
2 var dbConn = GetDatabaseConnection();
3
4 // Pull the patient ID in the incoming HL7 message
5 var patientId = msg['PID']['PID.3']['PID.3.1'].toString();
6
7 // Build a query to find this patient's MRN in our table
8 var sqlQuery =
9     "SELECT PatientMrn " +
10     "FROM dbo.PatientRegistration " +
11     "WHERE id = '" + patientId + "'";
12
13 // Execute the query
14 var queryResults = dbConn.executeCachedQuery(sqlQuery);
15
16 // Check if a row was found containing the patient
17 if (queryResults.next())
18 {
19     // Pull the MRN (a string) from the query results
20     var patientMrn = queryResults.getString('PatientMrn');
21
22     // Save the incoming HL7 message to the transformation variable
23     tmp = msg;
24
25     // Add the MRN to PID 4.1
26     tmp['PID']['PID.4']['PID.4.1'] = patientMrn;
27 }
  
```

Source Transformer Javascript Writer

SELECT Id, PatientMrn FROM dbo.PatientRegistration

	Id	PatientMrn
1	15159465-B820-4D5F-8BE2-07A1AE24956C	1234

The row in our table we want to query

Code Templates

Name	Returns a
Get Database Connection	DatabaseConnection

1 Library, 2 Code Templates

Library: Miscellaneous

Type: Function

Code:

```

1 /**
2     Returns a database connection object used for querying data
3
4     @return {DatabaseConnection} returns a DatabaseConnection object
5 */
6 function GetDatabaseConnection() {
7     // These variables are defined in Mirth's (Deploy) Global Script
8     return DatabaseConnectionFactory.createDatabaseConnection(
9         $('sqlServerDriver'),
10        $('databaseUrl'),
11        $('databaseUsername'),
12        $('databasePassword'));
13 }
  
```

Using a code template for database connections

MSH|^~\&|Mirth Connect|Galen Healthcare Solutions|Your Hospital|General Pr  
PID|||15159465-B820-4D5F-8BE2-07A1AE24956C||Test^Patient^L||19911006|M|||7  
PV1|1|Outpatient|General Practice||||Doctor^John|||||598545|||||  
IN1|1|001||Blue Cross Blue Shield|101 Huntington Avenue^^Boston^MA^02199||

Incoming HL7 Message (note PID 4 is blank)

MSH|^~\&|Mirth Connect|Galen Healthcare Solutions|Your Hospital|General Pr  
PID|||15159465-B820-4D5F-8BE2-07A1AE24956C|1234|Test^Patient^L||19911006|M  
PV1|1|Outpatient|General Practice||||Doctor^John|||||598545|||||  
IN1|1|001||Blue Cross Blue Shield|101 Huntington Avenue^^Boston^MA^02199||

Transformed HL7 Message (PID 4 now has the PatientMrn)

# COMMON CHANNEL SETTINGS

## Global Scripts

Script: **Deploy**

```
1 globalMap.put('databaseUrl', 'jdbc:jtds:sqlserver://mirthserver:1433;databaseName=mirthdb');
2 globalMap.put('databaseUsername', 'mirthuser');
3 globalMap.put('databasePassword', 'mirthpassword');
4
```

## Edit Channel - CCD - Patient History

Summary | Source | Destinations | Scripts

Connector Type: **Database Reader**

### Polling Settings

Schedule Type: **Interval** Next poll at: Thursday, Jun 23, 1:01:10 PM

Poll Once on Start: ☐ Yes ☒ No

Interval: **5** seconds

### Source Settings

Source Queue: **OFF (Respond after processing)**

Queue Buffer Size: **1000**

Response: **None**

Process Batch: ☐ Yes ☒ No

Batch Response: ☐ First ☒ Last

Max Processing Threads: **1**

### Database Reader Settings

Driver: **SQL Server/Sybase** **Insert URL Template**

URL: **\${databaseUrl}**

Username: **\${databaseUsername}**

Password: **..... \${databasePassword}**

Use JavaScript: ☐ Yes ☒ No

Keep Connection Open: ☒ Yes ☐ No

Cache Results: ☒ Yes ☐ No

## Edit Channel - DFT P03 - Financial Transactions

Summary | Source | Destinations | Scripts

Connector Type: **Database Reader**

### Polling Settings

Schedule Type: **Interval** Next poll at: Thursday, Jun 23, 1:06:35 PM

Poll Once on Start: ☐ Yes ☒ No

Interval: **5** seconds

### Source Settings

Source Queue: **OFF (Respond after processing)**

Queue Buffer Size: **1000**

Response: **None**

Process Batch: ☐ Yes ☒ No

Batch Response: ☐ First ☒ Last

Max Processing Threads: **1**

### Database Reader Settings

Driver: **Please Select One** **Insert URL Template**

URL: **\${databaseUrl}**

Username: **\${databaseUsername}**

Password: **..... \${databasePassword}**

Use JavaScript: ☐ Yes ☒ No

Keep Connection Open: ☒ Yes ☐ No

Cache Results: ☒ Yes ☐ No

# ENVIRONMENT STRATEGIES

## Global Scripts

Script:

```
1 // Set the names for the production and test Mirth environments
2 var productionMirthServerName = 'PRODSERVER';
3 var testMirthServerName = 'TESTSERVER';
4
5 // Get the name of this server running Mirth
6 var serverName = Packages.java.net.InetAddress.getLocalHost().getHostName();
7
8 // General database connection information
9 globalMap.put('sqlServerDriver', 'net.sourceforge.jtds.jdbc.Driver');
10
11 // Set server specific variables
12 if (serverName == productionMirthServerName)
13 {
14     globalMap.put('systemEnvironment', 'PROD');
15     globalMap.put('databaseUrl', 'jdbc:jtds:sqlserver://prodserver:1433;data
16     globalMap.put('databaseUsername', 'mirthuser');
17     globalMap.put('databasePassword', 'mirthpassword');
18 }
19 else if (serverName == testMirthServerName)
20 {
21     globalMap.put('systemEnvironment', 'TEST');
22     globalMap.put('databaseUrl', 'jdbc:jtds:sqlserver://testserver:1433;data
23     globalMap.put('databaseUsername', 'mirthuser');
24     globalMap.put('databasePassword', '1Ac#jkas710z');
25 }
26 else
27 {
28     throw "FATAL ERROR: The server launching Mirth is unrecognized. Look at
29 }
30
```

## Edit Channel - Run Nightly Processes

#	Operator
0	▼ Filter Destination in TEST Environment
▲▼	
Rule	
1	if (\$('systemEnvironment') == 'TEST')
2	return false;
3	else
4	return true;

- Programmatically dictate desired behavior based on server names rather than manually making changes to channels between development and production environments.
- This eliminates potential for human error when exporting channels from development environment to production.

# ROUTING A MESSAGE TO OTHER CHANNELS

## Channels

Status	Data Type	Name	Id
Enabled		Message Routing Example	70c0a01e-7d26-4c3a-b748-3e48cd24ecbd
Enabled	HL7 v2.x	Update Patient	92b11d53-f9d5-455a-b614-da38be986405
Enabled	HL7 v2.x	Get Patient Status	2593837b-02f7-46dd-b621-7f255c43baa9
Enabled	HL7 v2.x	Create Patient	c6eef31d-0006-4108-9105-3a49beb65c9c

## Edit Channel - Get Patient Status

Summary | Source | Destinations | Scripts

Status	Destination
Enabled	Get Patient Status

Connector Type: **Web Service Sender** ☐ Wait for previous destination

**Destination Settings**

Queue Messages: ☒ Never ☐ On Failure ☐ Always

Advanced Queue Settings:  0 Retries

Validate Response: ☐ Yes ☒ No

**Web Service Sender Settings**

WSDL URL:  {some http address here}

Service:  {some service address here}

Port / Endpoint:  {some port here}

Location URI:  {some URI here}

Socket Timeout (ms):  30000

Authentication: ☐ Yes ☒ No

Username:

Password:

Invocation Type: ☒ Two-Way ☐ One-Way

Operation: **Press Get Operations**

SOAP Action: **GetPatientStatus**

SOAP Envelope: 

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns:SearchClient_Input>
      <mes:MessageContextInput ProgramID="" />
    </ns:SearchClient_Input>
  </soapenv:Body>
</soapenv:Envelope>
```

## Edit Channel - Get Patient Status - Get Patient Status Response Transformer

#	Name	Type
0	Route message to correct channel based on response	JavaScript

Step

```
1 // Check Web Service response for indication if patient exists (this is a hypothetical example)
2 if (msg['ResponseMessage'] == "Patient Already Exists")
3 {
4   // Since patient exists, route message to the Update Patient channel
5   router.routeMessageByChannelId('92b11d53-f9d5-455a-b614-da38be986405', connectorMessage.getRawData());
6 }
7 else
8 {
9   // Patient does not already exist, so route message to Create Patient channel
10  router.routeMessageByChannelId('c6eef31d-0006-4108-9105-3a49beb65c9c', connectorMessage.getRawData());
11 }
```

# KNOW THE REASON FOR FILTERING

If you use a lot of filters, Mirth Connect currently does not tell you which filter was triggered. If that information is useful, here is a custom way of showing that.

### Edit Channel - ADT A04 - Patient Registration - Source Filter

#	Operator	Filter
0		Filter message if patient has no social security number
1	AND	Filter message if patient has no date of birth

**Rule**

```

1 var dateOfBirth = msg['PID']['PID.7']['PID.7.1'].toString();
2
3 if (dateOfBirth == "")
4 {
5     channelMap.put("FilterReason", "DOB Missing");
6     return false;
7 }
8 else
9 {
10    return true;
11 }
12

```

### Custom Metadata

Column Name	Type	Variable Mapping
PATIENTINTERNALID	STRING	▼ PatientInternalId
FILTERREASON	STRING	▼ FilterReason

Buttons: Add, Delete, Revert

### Channel Messages - ADT A04 - Patient Registration

**Start Time:** 09:34 AM ☐ All Day ☐ RECEIVED ☐ TRANSFORMED ☐ FILTERED ☐ QUEUED ☐ SENT ☐ ERROR  
**End Time:** 09:34 AM  
**Text Search:**  ☐ Regex  
**Page Size:** 20

**Current Search**  
 Max Message Id: 7  
 Date Range: (any) to (any)  
 Statuses: (any)  
 Connectors: (any)

**Results 1 - 7 of 7**   
 Page 1 of 1

Id	Connector	Status	Received Date	Response Date	PATIENTINTERNALID	FILTERREASON
7	Source	TRANSFORMED	2016-06-23 14:00:37:620	2016-06-23 14:00:37:650	2E639DE2-96B9-425E-8...	--
	Save Information to Clinical Database	SENT	2016-06-23 14:00:37:630	2016-06-23 14:00:37:647	2E639DE2-96B9-425E-8...	--
6	Source	FILTERED	2016-06-23 13:58:46:653	2016-06-23 13:58:46:663	--	DOB Missing
5	Source	FILTERED	2016-06-23 13:57:55:633	2016-06-23 13:57:55:643	--	SSN Missing
4	Source	TRANSFORMED	2016-06-23 10:35:04:420	2016-06-23 10:35:04:440	72DF6696-6219-4D63-B...	--
	Save Information to Clinical Database	SENT	2016-06-23 10:35:04:437	2016-06-23 10:35:04:437	72DF6696-6219-4D63-B...	--

# USING A JAVASCRIPT LIBRARY TO ENCRYPT FILES

**Code Templates**

Name
Save Message To Encrypted File
Load CryptoJsAes Library

1 Library, 4 Code Templates

Library: Miscellaneous

Type: Function

Code:

```

1 function saveMessageToEncryptedFile(xmlMessage, globalMessageId, folder)
2 {
3     // Name the file after the Message ID
4     var fileName = globalMessageId + '.xml';
5
6     // Create and map the encrypted, transformed (XSLT reduced) message.
7     var CryptoJS = LoadCryptoJsAes();
8     var plaintextMessage = xmlMessage.toString();
9     var password = "ASDSADS78786SDXKAKSDOWE99061";
10    var encrypted = CryptoJS.AES.encrypt(plaintextMessage, password);
11
12    // Create and write encrypted byte array to file
13    var pathToWriteGetMfResponseTo = folder + "/";
14    var finalAbsoluteFinalPath = pathToWriteGetMfResponseTo + fileName;
15
16    FileUtil.write(finalAbsoluteFinalPath, false, encrypted.toString());
17 }
18
  
```

**Code Templates**

Name
Save Message To Encrypted File
Load CryptoJsAes Library

1 Library, 4 Code Templates

Library: Miscellaneous

Type: Function

Code:

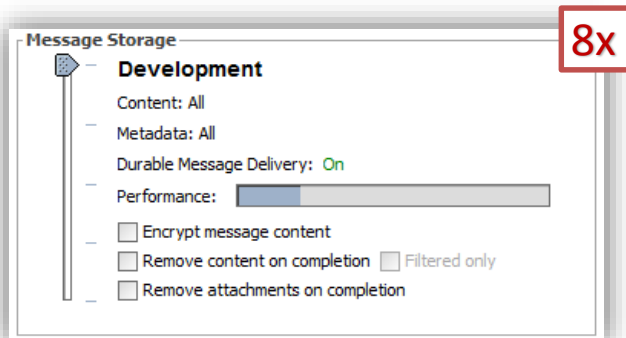
```

1 function LoadCryptoJsAes()
2 {
3     /*
4     CryptoJS v3.1.2
5     code.google.com/p/crypto-js
6     (c) 2009-2013 by Jeff Mott. All rights reserved.
7     code.google.com/p/crypto-js/wiki/License
8     */
9     var CryptoJS=CryptoJS||function(u,p){var d={},l=d.lib={},r=l.WordArray=t.extend({init:function(a,c){a=this.words=32-8*(c%4);a.length=u.ceil(c/4)},clone:function(){var a=2),16)<<24-4*(j%8);return new r.init(e,c/2)}},b=w.Latinq=1.BufferedBlockAlgorithm=t.extend({reset:function(){a._data=this._data.clone();return a},_minBufferSize:0});a._data=this._data.clone();return a},_minBufferSize:0);e)).finalize(b)});var n=d.algo={};return d}(Math);(function){var u=CryptoJS,p=u.lib.WordArray;u.enc.Base
  
```

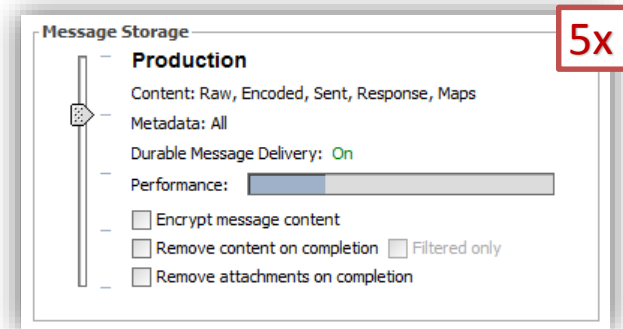


# **MAINTENANCE STRATEGIES**

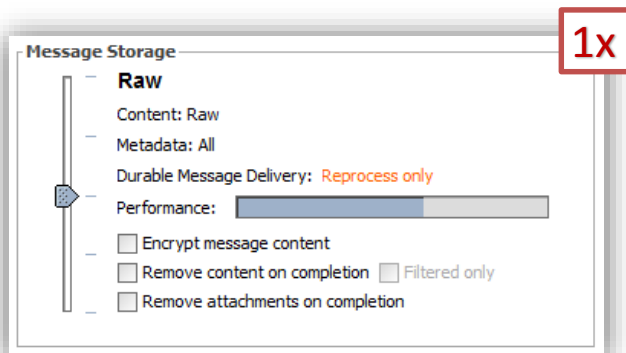
# MESSAGE STORAGE IMPACT



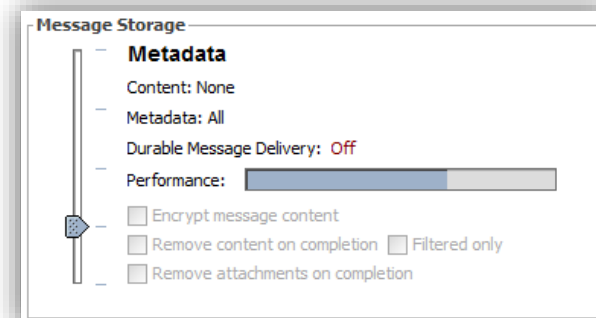
Uses the most disk space which also slows processing time.



Preserves essential troubleshooting data. Pair with pruning options to keep disk usage in check.



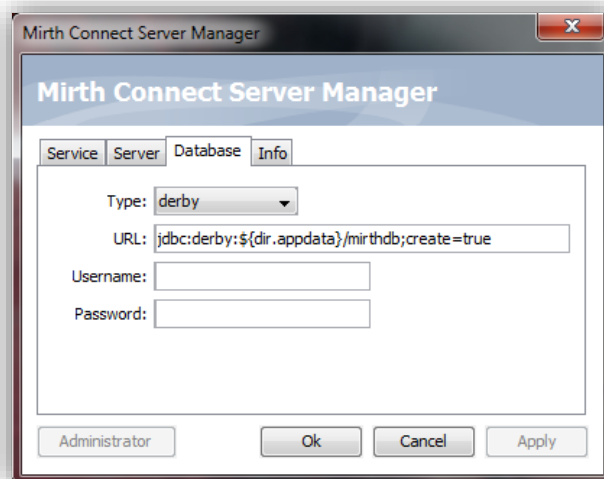
Stores just the original source message. No mapped variables. Reprocess a message when you need to troubleshoot one.



Deletes everything except what you see on the channel message log.  
*Cannot reprocess a message.*



# DATABASE BACKEND



Default Setting

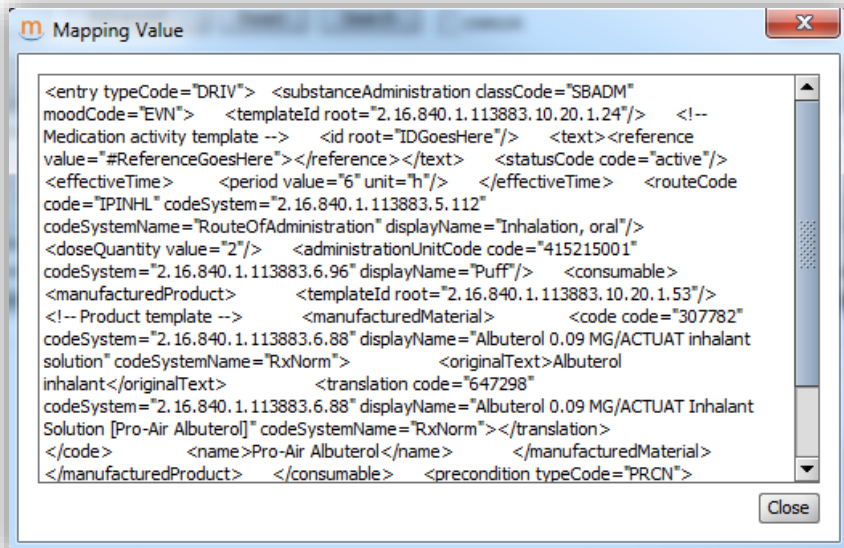


Preferred Configuration

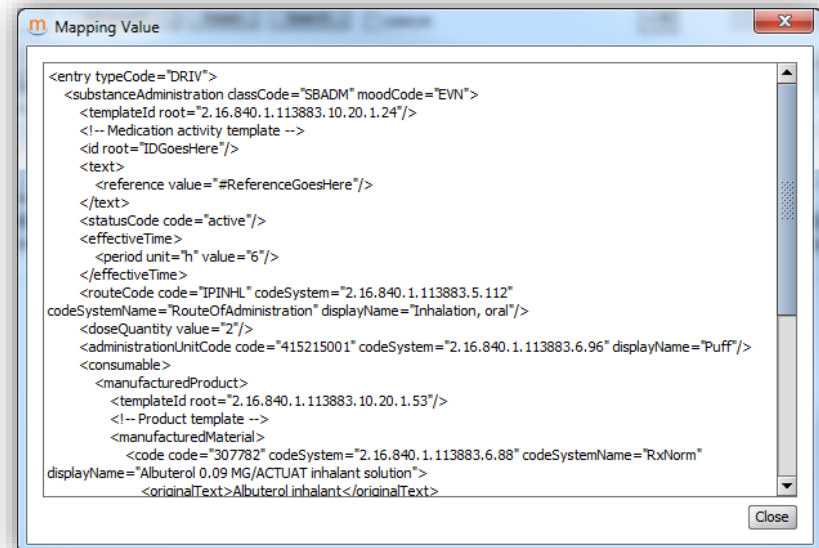
- The default Apache Derby option is for quick deployment and testing.
- Mirth Corp recommends PostgreSQL, MySQL, Oracle, or SQL Server.
- These offer better performance, disk space efficiency, backup options, security, etc.

# PRETTYPRINTXML

```
1 var xmlData = connectorMessage.getRawData();
2 var xmlDataWithPrettyPrintFormatting = XmlUtil.prettyPrint(xmlData);
3
4 channelMap.put('SourceXmlUnformatted', xmlData);
5 channelMap.put('SourceXmlPrettyPrintFormat', xmlDataWithPrettyPrintFormatting);
```



SourceXmlUnformatted



SourceXmlPrettyPrintFormat

# POLL #3



## **WRAPPING UP**

# ADDITIONAL RESOURCES

## 10 Tips and Tricks to Make Mirth™ Connect Work for You

By Nate Bessa. Published on May 12, 2015.

2

In our [last blog post](#) in this series, we introduced the Mirth™ Connect interface engine and discussed it up against the competition in the healthcare IT space. What we love most about Mirth™ is the breadth of customization options that are offered, and today we want to give you a deeper look inside the application tips and tricks segment. For a deeper dive into Mirth™ Connect, [download our whitepaper](#).

- **Add channel metadata to troubleshoot faster:** You may already be storing useful information in incoming messages in channel variables, such as the MRN of a patient, an identifier for a hospital, or an HTTP response code of a message you are POSTing via Mirth™. By adding these channel variable metadata to a channel, you can view the values for these variables on the message log screen, which will speed up your searches when using the Advanced search filter and specifying the metadata you have defined.

Custom Metadata	Column Name
HTTP_Response_Status	
MRN_ID	

**“10 Tips and Tricks To Make Mirth Connect Work For You”** blog post  
Google *“Mirth Connect Tips and Tricks Galen”*



**“Best Practices and Vulnerabilities in Mirth Connect”** white paper  
Google *“Mirth Connect Security White Paper Galen”*



**“Mirth Connect - Introduction”** webcast  
Search *“Mirth Connect Galen”* on Youtube

Thank you for joining us today.  
To access the slides from today's presentation, please visit:

<http://wiki.galenhealthcare.com/Category:Webcasts>

For additional assistance or to request information about our many products and services, including help with Mirth Connect, please contact us through our website:

[www.galenhealthcare.com](http://www.galenhealthcare.com)

