Presenter(s):
Dave Venier
Greg Hill

Topic
Improve NextGen Integration with Mirth Connect

Level
200
Safe Harbor Provisions/Legal Disclaimer

This presentation may contain forward-looking statements within the meaning of the federal securities laws, including statements concerning future prospects, events, developments, the Company’s future performance, management's expectations, intentions, estimates, beliefs, projections and plans, business outlook and product availability. **These forward-looking statements do not represent a commitment, promise or legal obligation to deliver any material, code or functionality.** The development, release and timing of any features or functionality described for our products remains at our sole discretion. Future products developed beyond what is contemplated by existing maintenance agreements, will be priced separately. **This roadmap does not constitute an offer to sell any product or technology.** We believe that these forward-looking statements are reasonable and are based on reasonable assumptions and forecasts, however, undue reliance should not be placed on such statements that speak only as of the date hereof. Moreover, these forward-looking statements are subject to a number of risks and uncertainties, some of which are outlined below. As a result, actual results may vary materially from those anticipated by the forward-looking statements. Among the important factors that could cause actual results to differ materially from those indicated by such forward-looking statements are: the volume and timing of systems sales and installations; the possibility that products will not achieve or sustain market acceptance; the impact of incentive payments under The American Recovery and Reinvestment Act on sales and the ability of the Company to meet continued certification requirements; the development by competitors of new or superior technologies; the timing, cost and success or failure of new product and service introductions, development and product upgrade releases; undetected errors or bugs in software; changing economic, political or regulatory influences in the health-care industry or applicable to our business; changes in product-pricing policies; availability of third-party products and components; competitive pressures including product offerings, pricing and promotional activities; the Company's ability or inability to attract and retain qualified personnel; uncertainties concerning threatened, pending and new litigation against the Company; general economic conditions; and the risk factors detailed from time to time in the Company’s periodic reports and registration statements filed with the Securities and Exchange Commission.
Introduction

A background on Rosetta, Mirth Connect and how they come together at a high level.
What Is Rosetta?

Rosetta is NextGen’s proprietary interface message handler

Supports HL7 versions 2.3 and 2.5 (2.51)

Rosetta includes support for the HIE Gateway component

Rosetta supports many HL7 message types – below are some common items:

- Demographics – ADT [A04,A08,A18,A28,A31]
- Patient Case – ADT [Z01]
- Scheduling – SIU [S12,S13,S14,S15,S17,S26]
- Charges – DFT [P03]
- Lab/RIS/PACS Orders – ORM [O01]
- Lab Results – ORU [R01]
- Documents/RIS/PACS Results – MDM [T02] & ORU [R01]
- Immunizations – VXU [V04]
- Pharmacy – RDE [O11]
- Master File Notification – MFN
- Syndromic Surveillance – ADT [A08]
What Rosetta Is Not?

Rosetta is not an interface (mapping) engine like Mirth Connect, CloverLeaf, eGate and Siemens OpenLink.

Rosetta expects HL7 messages to adhere to the NextGen specs.

Rosetta cannot manipulate, move, or reformat data so an engine must be used to manipulate messages:

- Any changes to inbound messages must be made before the message is processed by Rosetta.
- Any changes to outbound messages must be made after the message is sent by Rosetta.

Rosetta is not generally used for Equipment Interfaces.
What’s “In the Box” with Rosetta?

Rosetta includes the Windows service, Management console and Holding Tank…but is not the product clients actually purchase

• HIE Gateway component is included with the Rosetta installer

Clients actually purchase individual interface “Agents”

• Lab Orders/Results
• Demographics Exchange
• CCD/C-CDA Exchange

Mirth Connect is **not included** with purchase of Rosetta Agents

• However, Mirth Connect is a free download when needed
What is the HIE Gateway?

It is a component of Rosetta which supports most processes doing C-CDA, XDS.b, PIX/PDQ and other transactions

Requires Microsoft Internet Information Server

• Leverages Web Services for communication and data transfer (i.e. SSL)

Adds support for XDR (used with NG Share) and XDS.b messages (different from HL7)

Adds support for PIX/PDQ for Master Person Index (MPI) integrations

Used when exchanging a CCD/C-CDA with a third-party HIE (XDS.b)

• No two HIEs are setup the same way, so implementations take time
Rosetta Architecture

Rosetta Management Console
Rosetta Holding Tank Manager
NextGen EHR
NextGen EPM

NextGen Database

Rosetta HIE Gateway
Demographics, Master Files
Scheduling
Labs, Radiology
Documents, Images

Health Information Exchange Gateway
IHE PIX/PDQ
Charges
Medication
Immunizations

Interface Engine*
--Mirth
--Cloverleaf

TCP/IP, File based connectivity

RHIO / HIE / IHE Repository, Registry
Web Services, HTTPS, VPN

External Vendor Systems
--Quest
--Meditech

*The External system may not include an Interface Engine

NEXTGEN HEALTHCARE

NEXTGEN CONFIDENTIAL-NOT FOR DISTRIBUTION Copyright © 2015 NextGen Healthcare Information Systems, LLC
Mirth Corp and QSI

QSI acquired Mirth in 2013 and Mirth is a core part of NextGen’s strategy.

Mirth currently offers a suite of products:

- Mirth Connect (Integration Engine)
- Mirth Results (Clinical Data Repository/Health Information Exchange/Portals)
- Mirth Match (Enterprise Master Patient Index/eMPI)
- Mirth Mail (DirectTrust Certified HISP)
- Mirth Analytics (Population Health Management)
- Mirth Care Enterprise (Chronic Disease Management, Transitions of Care)
What Is Mirth Connect?

A Healthcare Integration Engine

• Cross-platform
• Message & Data Type Agnostic Including Flat files, Batch, Real-Time, HL7, CCD, Etc.

Easy to Use and Powerful

• Graphical Administration Tools
• Simple Channel-Based Interface Development

Flexible

• Plug-in Architecture Extensible via Java and JavaScript
• Multiple Routes, Filters, and Transformations
What Is Mirth Connect?

Cost Effective
- Open Source version is free to download, but does not include support
- Commercial version comes with support and enterprise extensions
- No “Per Interface License Fees”

Certification Training Available
- Mirth Connect Fundamentals
- Mirth Connect Advanced
- Each class is 4 days long and requires passing an exam to become certified
What Mirth Connect Is Not?

Mirth Connect is not typically a direct interface with the NextGen DB.
Mirth Connect is not required to run in Windows.
Mirth Connect does not have per interface fees
Mirth Connect is not tied to the NextGen EHR
Mirth Connect is not a data repository or master patient index
Mirth Connect is not a replacement for Rosetta
Where do Rosetta and Connect Meet?

• When integrating with other systems in your community such as hospitals or labs
• When filtering / modification of messages coming into Rosetta must be performed
• When modifications of messages being sent to system downstream of Rosetta are required
• When outbound messages need to be fanned out to multiple downstream systems
Use Cases

There are 1,000’s of use cases, these are a few examples of how you can use Rosetta and Connect together.

*** These use cases are only intended to be examples. If you attempt to implement them or something like them you may have to perform more research, manipulation and actions than are described in the following section.
Message Filtering
Message Filtering Use Case 1

**Issue:** NextGen client sends ADT messages to their Hospital system. The hospital only wants to see messages for patients which they have seen previously. NextGen has a custom demographic field which says “Seen at Local Hospital” and has “Yes” and “No” entries. This information can be exported in the HL7 message in a ZPD segment.

**Resolution:** Use Connect to create a new channel, in the Source or Destination there is a “Edit Filter” item. Create a new filter to look at the ZPD segment and if the value is not “Yes”, filter the message. *Alternately the filter can be done on the destination if multiple destinations are used.*
Message Filtering Use Case 1 - Simple

Message Filtering Use Case 1 - Advanced
Message Filtering Use Case 2

**Issue:** Local Hospital sends Lab Results for all of their patients to the NextGen clients system. The NextGen client only wants to receive messages for providers which are rendering practices in their system.

**Resolution:** Use Connect to create a new channel, in the Source or Destination there is a “Edit Filter” item. Create a new filter to look at the HL7 provider fields (OBR-16, ORC-12, PV1-7, etc, etc). Check the provider id in those fields and if the provider is not one of the clients rendering providers, filter the message.
Message Filtering Use Case 2 - Simple
Message Filtering Use Case 2 - Advanced

```java
// This script executes once when the mule engine is started
// You only have access to the globalMap here to persist data
var dbConn = DatabaseConnectionFactory.createDatabaseConnection('net.sourceforge.jtds.jdbc.Driver', 'jdbc:sqlserver://localhost:1433/databasehere', 'sa', 'password');
var result = dbConn.executeQuery('select provider_identifier from provider_table');
dbConn.close();
var AllowedValues = new Array();
var i = 0;
while (result.next() != '0') {
    AllowedValues[i] = result.getString(1);
    i++;
}
globalMap.put('AllowedValues', AllowedValues);
return;
```
Message Filtering Use Case 2 - Advanced

```
var globalProviders = globalMap.get("AllowedValues");
var OBRProviderArray = new Array();
var ORCProviderArray = new Array();
var arraycount = 0;
for (var ob in msg..OBR)
    { OBRProviderArray[arraycount] = ob["OBR.16"]['OBR.16.1'].toString();
      arraycount++;
    }
arraycount = 0;
for (var or in msg..ORC)
    { ORCProviderArray[arraycount] = or["ORC.12"]['ORC.12.1'].toString();
      arraycount++;
    }
var result = loopProviders(globalProviders, OBRProviderArray);
if (result == true)
    return true;
var result = loopProviders(globalProviders, ORCProviderArray);
if (result == true)
    return true;
return false;

function loopProviders(AllowedProviders, MessageProviders)
{ var retValue = false;
  for (var i=0;i<MessageProviders.length;i++)
      { for (var proc=0;proc<AllowedProviders.length;proc++)
          { if (MessageProviders[i].toString() == AllowedProviders[proc].toString())
              { retValue = true;
                break;
              }
          }
        if (retValue == true)
            break;
      } return retValue;
}
```

Message Manipulation
Message Manipulation Use Case 1 - Simple

**Issue:** Local Hospital sends Documents to the NextGen client’s system. The document descriptions are not user friendly and are difficult for doctors to decipher what they are looking at.

**Resolution:** Use Connect to create a new channel, in the Source or Destination there is a “Edit Transformer” item. Create a new transformer to look at the HL7 document description field (OBX-4.2, TXA-16 or OBX-3.2). Use the information in that field to map the bad descriptions to user friendly ones.
Message Manipulation Use Case 1 - Simple

```javascript
function GetDocumentName(documentName)
{
    var obr42 = "";
    switch (documentName)
    {
    case "Ort_referral_Ltr":
        obr42 = "Consult";
        break;
    case "master_document":
        obr42 = "Office Visit";
        break;
    case "ssk_np":
        obr42 = "RIS Study";
        break;
    case "thr_1y":
        obr42 = "Lab Report";
        break;
    case "thr_2y":
        obr42 = "Referral Letter";
        break;
    default:
        obr42 = "";
    }
    return obr42;
}
```
Message Manipulation Use Case 2 - Advanced

**Issue:** Local Hospital sends Plain Text Documents to the NextGen client’s system. The document format needs manipulation and conversion to PDF so it cannot be edited.

**Resolution:** Use Connect to create a new channel with two destinations. In the first destination make it a “document writer” and edit the filter to create your document content which is used in the template field. In the second destination, get the document created from the first destination and insert it into the message.
Message Manipulation Use Case 2 - Advanced

<table>
<thead>
<tr>
<th>Status</th>
<th>Destination</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Extract Text to PDF</td>
<td>1</td>
</tr>
<tr>
<td>Enabled</td>
<td>To Rosetta</td>
<td>2</td>
</tr>
</tbody>
</table>

Connector Type: **Document Writer**

**Destination Settings**
- Queue Messages: [ ] Never [ ] On Failure [ ] Always
- Advanced Queue Settings: [ ] 0 Retries
- Validate Response: [ ] Yes [ ] No

**Document Writer Settings**
- Output: [ ] File [ ] Attachment [ ] Both
- Directory: 
- File Name: 
- Document Type: [ ] PDF [ ] RTF
- Encrypted: [ ] Yes [ ] No
- Password: 
- Template: `{docContent}`
Message Manipulation Use Case 2 - Advanced
Message Manipulation Use Case 2 - Advanced

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Map Doc To OBX</td>
</tr>
</tbody>
</table>

```
1   tmp = msg;  
2   delete tmp["OBX"];  
3   createSegmentAfter("OBX", tmp["OBR"]);  
4   tmp["OBX"] = '1|ED|||^AP^PDF^Base64^' + responseMap.get("d1").getMessage().toString() +'FFFFF';
```

Time for a demo!
Feed Splitting/Combining
Feed Splitting/Combining – Use Case

**Issue:** NextGen client needs to send ADT to three local hospitals also receive unsolicited lab results and documents from those hospitals into the NextGen EHR. They would like to accomplish this with one feed of demographics leaving NextGen along with one feed of labs results and one feed of documents coming into NextGen.
Feed Splitting/Combining – Before

Legend

- Demographics
- Discrete Unsolicited Lab Results
- Document Based (ORU) – Discharge Summaries, RIS Results, ETC

NextGen Rosetta

Hospital 1

Hospital 2

Hospital 3
Feed Splitting/Combining – Use Case Before

NextGen Setup – Rosetta Agents:
- Demographic Export - System 1
- Demographic Export - System 2
- Demographic Export - System 3
- Lab Result Import – System 1
- Lab Result Import – System 2
- Lab Result Import – System 3
- Document Import – System 1
- Document Import – System 2
- Document Import – System 3

NextGen Setup – Cross References:
- Demographic Export - System 1
- Demographic Export - System 2
- Demographic Export - System 3
- Lab Result Import – System 1
- Lab Result Import – System 2
- Lab Result Import – System 3
- Document Import – System 1
- Document Import – System 2
- Document Import – System 3
Feed Splitting/Combining – Use Case After

Legend
- Demographics
- Discrete Unsolicited Lab Results
- Document Based (ORU) – Discharge Summaries, RIS Results, ETC
Feed Splitting/Combining – Use Case After

**Resolution:** Use Connect to create a new set of channels to perform the necessary actions. The number of Rosetta agents and setup within NextGen is greatly reduced.

**NextGen Setup – Rosetta Agents:**
- Demographic Export - Mirth Connect
- Lab Result Import – Mirth Connect
- Document Import – Mirth Connect

**NextGen Setup – Cross References:**
- Demographic Export - Mirth Connect
- Lab Result Import – Mirth Connect
- Document Import – Mirth Connect

**Mirth Setup – 9 Channels**
- 1 Demographic Channel
- 4 Lab Result Channels
- 4 Lab Result Channels
Feed Splitting/Combining – Use Case

![Dashboard Table](image-url)

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Rev A</th>
<th>Last Deployed</th>
<th>Received</th>
<th>Filtered</th>
<th>Queued</th>
<th>Sent</th>
<th>Error</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Started</td>
<td>Demographics From Rosetta</td>
<td>0</td>
<td>2015-08-28 13:24</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Hospital 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Hospital 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Hospital 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Documents From Hospital 1</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Documents From Hospital 2</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Documents From Hospital 3</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Documents to Rosetta</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Lab Results From Hospital 1</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Lab Results From Hospital 2</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Lab Results From Hospital 3</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>Started</td>
<td>Lab Results to Rosetta</td>
<td>0</td>
<td>2015-08-28 13:22</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>Idle</td>
</tr>
</tbody>
</table>
Feed Splitting/Combining – Use Case
Food for Thought - Identifiers

All of the different identifier types in the message need to be merged down to one common set for their specific identifier type.

Any provider id sent in the message should be resolved to that it is their NPI (National Provider ID).

Any location id sent in the message should be resolved to a common location identifier.

The visit id’s send in the message should be resolved to a common visit identifier.

Any other fields in the message which are treated as identifiers need to have resolution performed on Connect.
Feed Splitting/Combining – Use Case
Food for Thought – Message Structure

Create one common structure for the incoming message (see the NextGen specification for information on the specific message you are trying to map) which is going to be imported by Rosetta.

For example if the Document Import interface is setup to pull the description of the document from OBR-4.2 in the message, all messages will pull the description from OBR-4.2 on that feed. Connect must put the document name in that field or default a value in the field for the users to see in the application.
Feed Splitting/Combining – Use Case

Food for Thought – Demographics

Demographic exports are easy to put through Connect.

- Let Connect manage message manipulation along with taking one export feed and sending it to multiple systems.

When importing demographics NextGen it is not recommended to combine multiple feeds into one feed coming into Rosetta.

- There should be one source of truth for demographics across all systems and that system should feed demographics to the child systems.

- When trying to combine multiple demographic feeds the interface engine would need to act as an EMPI system. This could be accomplished through Mirth Match.

- If trying to do this yourself or without sufficient testing you have the possibility of introducing patient safety issues.
Feed Splitting/Combining – Use Case
Food for Thought – Lab Results

Lab Result feeds can often be combined into a common feed for unsolicited lab results. The order numbers (OBR-3 usually) returned by the lab system need to be unique prior to receipt by Rosetta. This identifier change applies to Document interfaces as well.

• For example if you are combining feeds from Hospital 1 and Hospital 2 for unsolicited lab results. Hospital 1 may have an order for patient John Smith with identifier 1234 and Hospital 2 may have an order for patient Jane Doe with identifier 1234.

We combining the result feeds in this manner the results received by NextGen are imported to one NextGen external system.

• For example when doing direct interfaces with Reference Lab 1 and Reference Lab 2 you would have a “Reference Lab 1” external system and a “Reference Lab 2” external system in NextGen. If using Connect to combine feeds there will be an “Interface Engine” external system setup which the results will be imported to.
Mirth Connect Commercial Extensions

Advanced functionality available inside Connect designed to maximize your enterprise deployment.
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 allowing for alerts to be sent to different user groups based on the current escalation level, time, and day.
- Alert dashboard provides a view of all alert statistics and logs.
SSL Connector

- Allows you to quickly enable and configure certificate-based SSL connectivity for socket-based connectors such as the HTTP Listener/Sender, Web Service Listener/Sender, and FTP Reader/Writer.

- Includes keystore-based certificate management allowing you to store your certificates in one location and apply them to your existing connectors.
User Authorization

• Provides role-based access control to all aspects of the Mirth Connect Administrator.

• Create new roles with specific permissions to areas such as channel management or message browsing.

• Assign any number of roles to users.

• Use this to manage access to sensitive channel and messaging data across your enterprise.
Channel History

- Allows for configuration management for your critical channels
- Provides abilities to view and compare past revisions of channel configurations—and identify the user who made the changes.
- Allows for reversion to a past revision from the embedded viewer.
Commercial License Benefits

24x7 Emergency Production Support (phone or email)
Access to Mirth Exchange
Access to Online Tutorials
Access to monthly Developer Q&A Webinars
A channel reporting tool for documenting your channels
In Conclusion

Rosetta and Mirth Connect can be used in concert to maximize your interface needs while minimizing your interface costs.

Mirth Connect compliments Rosetta by allowing you to filter and transform messages and split and/or combine feeds.

Mirth Connect commercial extensions enhance the capabilities of the open source version by providing advanced alerting and role based authorization, enabling SSL-based interfaces, and channel history. In addition, the commercial version comes with support and access to additional resources.
Session Survey

Please take a moment to complete a brief survey regarding this session.

1. Open your ONE UGM Mobile App (please note: you must have already logged in and accepted the “Terms of Use” to access this feature)
2. Click the Navigation Button at the top left of the screen
3. Select “Sessions”
4. Search for and select this session
5. From the sessions details screen, select “Survey” at the bottom right of the screen
6. Remember to hit “Save” at the bottom of the survey once you have answered the questions
Any Questions?