

Bedmaster High Availability Using Mirth Channels



Notice

Copyright © 2002-2018 Vocera Communications, Inc. All rights reserved.

Vocera® is a registered trademark of Vocera Communications, Inc.

This software is licensed, not sold, by Vocera Communications, Inc. ("Vocera"). The reference text of the license governing this software can be found at <http://www.vocera.com/legal/>. The version legally binding on you (which includes limitations of warranty, limitations of remedy and liability, and other provisions) is as agreed between Vocera and the reseller from whom your system was acquired and is available from that reseller.

Certain portions of Vocera's product are derived from software licensed by the third parties as described at <http://www.vocera.com/legal/>.

Microsoft®, Windows®, Windows Server®, Internet Explorer®, Excel®, and Active Directory® are registered trademarks of Microsoft Corporation in the United States and other countries.

Java® is a registered trademark of Oracle Corporation and/or its affiliates.

All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owner/s. All other brands and/or product names are the trademarks (or registered trademarks) and property of their respective owner/s.

Vocera Communications, Inc.

www.vocera.com

tel :: +1 408 882 5100

fax :: +1 408 882 5101

Last modified: 2018-11-27 07:54

VAM-225-Docs build 26



Contents

- Using Bedmaster in a High Availability Environment..... 4**
- Architecture Diagram.....4
- Mirth Channels.....5
 - The BMHL7Receiver Channel..... 5
 - The BMJSONCreator Channel..... 6
 - The BMJSONReceiver Channel.....7
 - The BMMonitor Channel..... 8
 - Deploying a Channel..... 9
- Changing the Load Balancer IP Address..... 10
- Changing the Connector Type to File Writer..... 10
- Specifying the Sending Application Name.....11
- Default Ports..... 11
- Using the Load Balancer..... 12

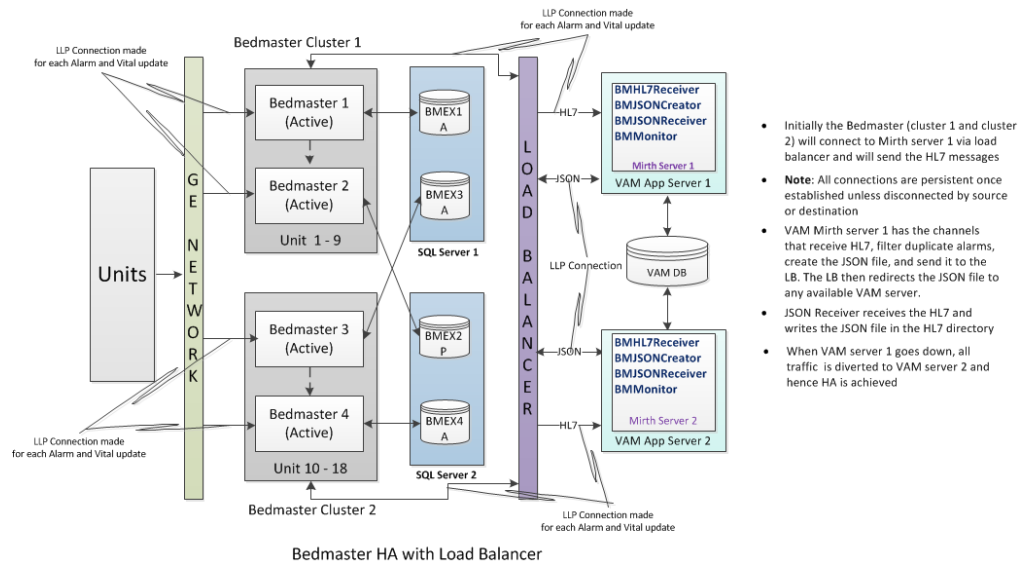
Using Bedmaster in a High Availability Environment

These sections describe how to deploy Bedmaster in a high-availability environment using Mirth channels and data deduplication logic.

In a high-availability environment, each deployed Bedmaster runs in active-active mode. This means that all alarms or vitals messages that are generated are duplicated. To handle these duplicated alarms, deduplication logic is added in the BMH7Receiver Mirth channel, which is one of the four Mirth channels needed in a high-availability environment.

Architecture Diagram

This diagram shows the typical layout in a Bedmaster high-availability environment. This shows the Mirth channels that are required.



The connection between the Bedmaster and the BMHL7Receiver channel is persistent. This means that HL7 messages are sent to VAM in the order specified in the load balancer.

The connection between the BMJSONCreator channel and the load balancer can be persistent or non-persistent. If the connection is persistent, the JSON messages are sent to only the VAM server to which it is connected, or in the order specified in the load balancer. If the connection is non-persistent, JSON messages are sent to any of the VAM servers.



Note: If the HL7 service is stopped on a Bedmaster server and then restarted, you may receive duplicate alarms.

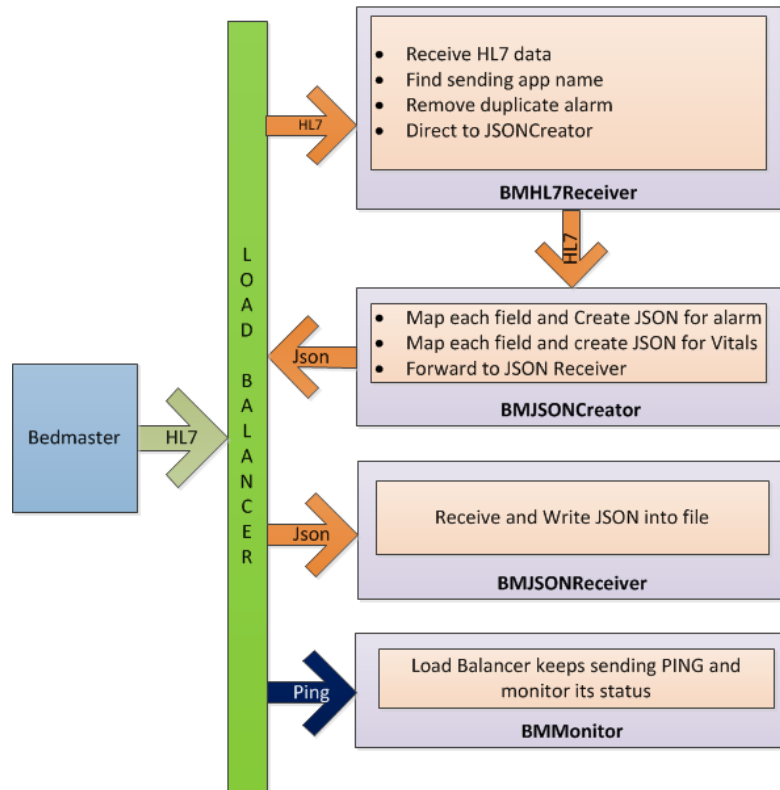
Mirth Channels

In a VAM load balancer environment, four channels are required to receive and process patient monitor HL7 data from Bedmaster.

- BMHL7Receiver
- BMJSONCreator
- BMJSONReceiver
- BMMonitor

In a load balancer environment, all four channels must be deployed on each VAM server.

This diagram shows the relationship between the Mirth channels, the load balancer, and the Bedmaster:



Mirth Channels required in a VAM server in LB Environment

Details on how these channels are configured can be found in the following sections. For information on how to deploy these channels after they have been created, see [Deploying a Channel](#) on page 9.

The BMHL7Receiver Channel

For the BMHL7Receiver channel, you must make these modifications.

Update the Source tab as shown:

Edit Channel - BMHL7Receiver

Summary \ Source \ Destinations \ Scripts \

Connector Type: **LLP Listener**

LLP Listener

LLP Mode: ☒ Server ☐ Client

Listener Address:

Listener Port:

Reconnect Interval (ms):

Receive Timeout (ms):

Buffer Size (bytes):

Process Batch: ☐ Yes ☒ No

LLP Frame Encoding: ☐ ASCII ☒ Hex

Start of Message Char: End of Message Char:

Record Separator Char: End of Segment Char:

Use Strict LLP Validation: ☒ Yes ☐ No

Wait for End of Message Char: ☐ Yes ☒ No

Encoding: **Default**

Send ACK: ☒ Yes ☐ No ☐ Respond from: **None**

Successful ACK Code: Message:

Error ACK Code: Message:

Rejected ACK Code: Message:

MSH-15 ACK Accept: ☐ Yes ☒ No

ACK on New Connection: ☐ Yes ☒ No

ACK Address:

ACK Port:

Update the Destination tab as shown:

Edit Channel - BMHL7Receiver

Summary \ Source \ Destinations \ Scripts \

| Status | Destination |
|--|-------------|
| <input checked="" type="radio"/> Enabled | Alarms |

Connector Type: **Channel Writer**

Channel Writer

Channel Name: **None**

Wait for Channel Response: ☐ Yes ☒ No

Template: `${message.encodedData}`

The BMJSONCreator Channel

For the BMJSONCreator channel, you must make these modifications.

Update the Source tab as shown:

Edit Channel - BMJSONCreator

Summary \ Source \ Destinations \ Scripts \

Connector Type: **Channel Reader**

Channel Reader

Respond from: **None**

Update the Destination tab as shown:

Edit Channel - BMJSONCreator

Summary \ Source \ **Destinations** \ Scripts

| Status | Destination |
|---------|-------------|
| Enabled | Vitals |
| Enabled | Alarms |

Connector Type: **LLP Sender**

LLP Sender

Host Address: 172.30.17.209 Test Connection

Host Port: 6666

Send Timeout (ms): 1000

Buffer Size (bytes): 65536

Keep Connection Open: ☐ Yes ☒ No

LLP Frame Encoding: ☐ ASCII ☒ Hex

Start of Message Char: 0x0B End of Message Char: 0x1C

Record Separator Char: 0x0D End of Segment Char: 0x0D

Maximum Retry Count: 2

Reconnect Interval (ms): 1000

Use Persistent Queues: ☐ Yes ☒ No ☐ Rotate Messages in Queue

Queue Poll Interval (ms): 200

Queue on ACK Timeout: ☒ Yes ☐ No

ACK Timeout (ms): 5000 ☐ Ignore ACK

Process HL7 ACK: ☐ Yes ☒ No

Encoding: Default

The BMJSONReceiver Channel

For the BMJSONReceiver channel, you must make these modifications.

Update the Source tab as shown:

Edit Channel - BMJsonReceiver

Summary \ Source \ Destinations \ Scripts \

Connector Type: **LLP Listener**

LLP Listener

LLP Mode: ☒ Server ☐ Client

Listener Address:

Listener Port:

Reconnect Interval (ms):

Receive Timeout (ms):

Buffer Size (bytes):

Process Batch: ☐ Yes ☒ No

LLP Frame Encoding: ☐ ASCII ☒ Hex

Start of Message Char: End of Message Char:

Record Separator Char: End of Segment Char:

Use Strict LLP Validation: ☒ Yes ☐ No

Wait for End of Message Char: ☐ Yes ☒ No

Encoding: **Default**

Send ACK: ☒ Yes ☐ No ☐ Respond from: **None**

Successful ACK Code: Message:

Error ACK Code: Message:

Rejected ACK Code: Message:

MSH-15 ACK Accept: ☐ Yes ☒ No

ACK on New Connection: ☐ Yes ☒ No

ACK Address:

ACK Port:

Update the Destination tab as shown:

Edit Channel - BMJsonReceiver

Summary \ Source \ Destinations \ Scripts \

| Status | Destination |
|---|-------------|
| <input checked="" type="checkbox"/> Enabled | Alarms |

Connector Type: **File Writer**

File Writer

Method: **file**

Directory:

ftp:// /

File Name:

Anonymous: ☒ Yes ☐ No

Username:

Password:

Timeout (ms):

Secure Mode: ☒ Yes ☐ No

Passive Mode: ☒ Yes ☐ No

Validate Connection: ☒ Yes ☐ No

Append to file: ☒ Yes ☐ No

File Type: ☐ Binary ☒ ASCII

Encoding: **Default**

Template:

The BMMonitor Channel

For the BMMonitor channel, you must make these modifications.

Update the Source tab as shown:

Edit Channel - BMMonitor

Summary \ Source \ Destinations \ Scripts \

Connector Type: **TCP Listener**

TCP Listener

Listener Address: 127.0.0.1

Listener Port: 5000

Receive Timeout (ms): 1000

Buffer Size (bytes): 65536

Keep Connection Open: ☐ Yes ☒ No

Encoding: Default

Data Type: ☐ Binary ☒ ASCII

Respond from: None

Response on New Connection: ☐ Yes ☒ No

Response Address:

Response Port:

Update the Destination tab as shown:

Edit Channel - BMMonitor

Summary \ Source \ Destinations \ Scripts \

| Status | Destination |
|---|---------------|
| <input checked="" type="checkbox"/> Enabled | Destination 1 |

Connector Type: **File Writer**

File Writer

Method: file Test Write

Directory: C:/vocera/mVisum Alerts

ftp:// /

File Name: \$ {UUID}.txt

Anonymous: ☒ Yes ☐ No

Username: anonymous

Password:

Timeout (ms): 10000

Secure Mode: ☒ Yes ☐ No

Passive Mode: ☒ Yes ☐ No

Validate Connection: ☒ Yes ☐ No

Append to file: ☒ Yes ☐ No

File Type: ☐ Binary ☒ ASCII

Encoding: Default

Template: \$ {message.rawData}

Deploying a Channel

After you have created a channel, you must deploy it.

1. In the Mirth Connect Administrator, in the Mirth Connect pane at the top left of the screen, select Channels. A list of configured channels appears.
2. Right-click on the channel that you want to deploy. From the popup menu that appears, select Deploy Channel.

The Dashboard now displays the deployed channel.

Changing the Load Balancer IP Address

In the BMJSONCreator channel, the load balancer IP address is set to a default value. Follow these steps to change the load balancer IP address.

1. In the Mirth Connect Administrator, in the Mirth Connect pane, select **Channels**. The list of channels is displayed.
2. Right-click BMJSONCreator and select **Edit Channel**.
3. In the Edit Channel screen, click the **Destinations** tab.
4. In the **Host Address** field, type the IP address of your load balancer.

5. Click **Save Changes** to save the changes.
6. Right-click on the channel and select **Deploy Channel** to redeploy the changed channel.

Changing the Connector Type to File Writer

If a File Writer connection is needed instead of an LLP Sender connection, you can update the BMJSONCreator channel to reflect this.

1. In the Mirth Connect Administrator, in the Mirth Connect pane, select **Channels**. The list of channels is displayed.
2. Right-click BMJSONCreator and select **Edit Channel**.
3. In the Edit Channel screen, click the **Destinations** tab.
4. In the **Connector Type** dropdown list, select **File Writer**.

5. In the **Directory** field, type `<Vocera>\mVisumAlerts\MVisum Message Generator\HL7`, where `<Vocera>` is the folder in which VAM is installed. This is where the message generator obtains the JSON files.
6. In the **File Name** field, type `${UUID}.json`. This generates a unique ID for each file name, and ensures that the file extension is `.json`.
7. From the **Append to file** radio buttons, select **Yes**.
8. In the **Template** field, add the following:

```
{
  "VitalData": {
    "SendingApplication": "${SendingApplication}"
    , "SendingFacility": "${SendingFacility}"
    , "DateTimeOfMessage": "${DateTimeOfMessage}"
    , "MessageControlID": "${MessageControlID}"
    , "ProcessingID": "${ProcessingID}"
    , "PatientIDExternalID": "${PatientIDExternalID}"
    , "PatientInternalId": "${PatientInternalId}"
    , "PatientGivenName": "${PatientGivenName}"
    , "FamilyName": "${FamilyName}"
    , "PatientClass": "${PatientClass}"
    , "PointOfCare": "${PointOfCare}"
    , "Bed": "${Bed}"
    , "AdmissionType": "${AdmissionType}"
  }
}
```

```

    , "UniversalServiceIdentifier": "${UniversalServiceIdentifier}"
    , "ObservationDateTime": "${ObservationDateTime}"
    , "ObservationEndDateTime": "${ObservationEndDateTime}"
    , "RelevantClinicalInformation": "${RelevantClinicalInformation}"
    , "DiagnosticServiceSectionID": "${DiagnosticServiceSectionID}"
    , "ParentResult": "${ParentResult}"
    , "AdmitDateTime": "${AdmitDateTime}"
    , "Vitals": ${Vitals}
  }
}

{
  "AlarmInfo": {
    "SendingApplication": "${SendingApplication}"
    , "SendingFacility": "${SendingFacility}"
    , "DateTimeOfMessage": "${DateTimeOfMessage}"
    , "MessageControlID": "${MessageControlID}"
    , "ProcessingID": "${ProcessingID}"
    , "PatientInternalID": "${PatientInternalId}"
    , "PatientGivenName": "${PatientGivenName}"
    , "FamilyName": "${FamilyName}"
    , "DateOfBirth": "${DateOfBirth}"
    , "PatientClass": "${PatientClass}"
    , "PointOfCare": "${PointOfCare}"
    , "Bed": "${Bed}"
    , "AdmissionType": "${AdmissionType}"
    , "UniversalServiceIdentifier": "${UniversalServiceIdentifier}"
    , "ObservationDateTime": "${ObservationDateTime}"
    , "ObservationEndDateTime": "${ObservationEndDateTime}"
    , "RelevantClinicalInformation": "${RelevantClinicalInformation}"
    , "DiagnosticServiceSectionID": "${DiagnosticServiceSectionID}"
    , "ParentResult": "${ParentResult}"
    , "QuantityTiming": "${QuantityTiming}"
    , "AlarmLevel": "${AlarmLevel}"
    , "AlarmReason": "${AlarmReason}"
    , "ObservResultStatus": "${ObservResultStatus}"
    , "userDefinedAccessChecks": "${AttachmentDate}"
    , "AttachmentType": "${AttachmentType}"
    , "ControlId": "${ControlId}"
    , "MessageType": "${MessageType}"
    , "VersionId": "${VersionId}"
    , "Waveform": "${Waveform}"
    , "Vitals": ${Vitals}
  }
}

```

9. Click **Save Changes** to save the changes.

10. Right-click on the channel and select **Deploy Channel** to redeploy the changed channel.

Specifying the Sending Application Name

In the HL7 file, use the sending application name as shown in this table.

| Bedmaster Server Name | Sending application name in HL7 |
|-----------------------|---------------------------------|
| Bedmaster 1 (active) | BedMasterEx1A |
| Bedmaster 2 (active) | BedMasterEx1P |
| Bedmaster 3 (active) | BedMasterEx2A |
| Bedmaster 4 (active) | BedMasterEx2P |



Note: The sending application names must be configured in Bedmaster.

Default Ports

The following default ports are configured in the Mirth channels.

| Channel | Port | Comment |
|----------------|------|--|
| BMHL7Receiver | 5001 | To receive HL7 data from Bedmaster |
| BMJSONCreator | 6666 | To send data to BMJSONReceiver via the load balancer |
| BMJSONReceiver | 6666 | To receive data from the BMJSONCreator channel |
| BMMonitor | 5000 | The load balancer sends pings to this port. |

Using the Load Balancer

In the load balancer, these steps must be followed to achieve high availability.

1. Create a monitor with TCP and port 5000. One monitor must be created for each VAM server.
2. Create services for port 5001 (Bedmaster to load balancer) and the IP address of the VAM server. Create one service for each monitor.
3. Create a primary load balancer virtual service with port 5001. Add the primary VAM services to it. This sends all traffic to the primary server.
4. Set up a secondary load balancer virtual service that sends traffic to the secondary VAM server. This virtual server can be created without an IP address.
5. On the primary load balancer virtual service, under Protection Features, set the secondary load balancer service to be the backup virtual server.
As long as the primary backend server is up, the primary load balancer virtual service sends traffic to it. If this server fails, traffic is passed to the secondary load balancer virtual service, and then to the secondary backend server.
6. Create services for port 6666 (BMJSONCreator to load balancer) and the IP address of the VAM server. Create one service for each monitor.
7. Create a virtual service with port 6666, and add both of the services that you have just created. Verify that the service order is correct (VAM 1, VAM 2).
8. Create a service with port 80 for HTTP or port 443 for HTTPS with the IP address of the VAM server. Bind the HTTP or HTTPS monitor (this monitor is present by default in NetScaler) to access the VAM Console. One service needs to be created for each node.
9. Create a virtual service with port 80 or 443 and bind the services that you have just created.

The total required is four monitors, six services, and three virtual servers.

For more details, consult the configuration document for your load balancer.