



Vocera Incoming VMI Adapter Configuration Guide

Version 1.0.1

Notice

Stryker Corporation or its divisions or other corporate affiliated entities own, use or have applied for the following trademarks or service marks: Stryker, Vocera. All other trademarks are trademarks of their respective owners or holders. The absence of a product or service name or logo from this list does not constitute a waiver of Stryker's trademark or other intellectual property rights concerning that name or logo. Copyright © 2023 Stryker.

Last modified: 2023-02-24 13:56

ADP-incomingvmi-101-Docs build 246

Contents

| | |
|---|----|
| Understanding a Vocera Incoming VMI Adapter Configuration..... | 4 |
| Viewing the Vocera Incoming VMI Adapter Requirements..... | 4 |
| Configuring a Vocera Incoming VMI Adapter..... | 8 |
| Understanding Regular Expressions (Regex)..... | 12 |
| Multiple Choice Response (MCR) Matching..... | 20 |
| Working with the Vocera Incoming VMI Adapter Rules..... | 26 |
| Understanding Vocera Incoming VMI Adapter Operations..... | 28 |
| Integrating the Vocera Incoming VMI Adapter with Vocera Platform..... | 31 |
| Configuring AlertMetaData Dataset..... | 32 |
| Configuring the ResponseOptions Dataset..... | 34 |
| Understanding Adapter Installation..... | 37 |
| Recreating a Repository..... | 37 |
| Installing an Adapter..... | 38 |
| Practicing an Adapter Installation..... | 38 |
| Navigating the Vocera Platform Adapters..... | 40 |
| Editing an Adapter..... | 42 |
| Creating a New Adapter..... | 43 |
| Saving an Adapter..... | 44 |
| Deactivating an Adapter..... | 44 |
| Removing an Adapter..... | 45 |

Understanding a Vocera Incoming VMI Adapter Configuration

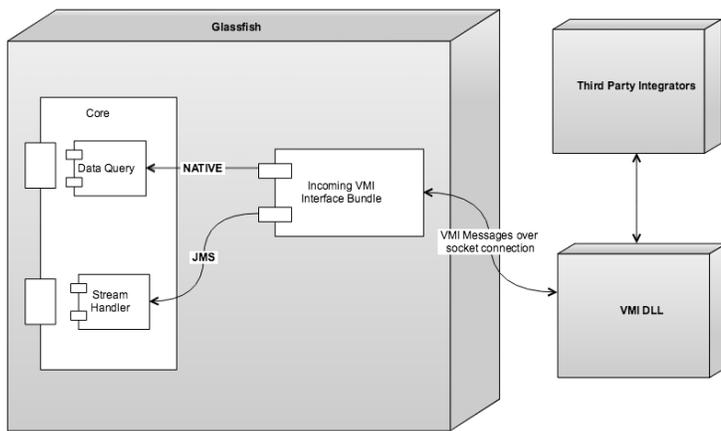
The Vocera Incoming VMI Adapter acts as a server for third party integrators using the VMI protocol.

Adapters send information to and receive information from the Vocera Platform, as well as monitor and collect data. Each adapter is configured to allow the Vocera Platform to communicate with a specific type of resource and any devices that resource may control.

The Vocera Incoming VMI Adapter implements the VMI protocol as a server with the intent of receiving inbound messages and providing responses to the senders. This adapter manages the connection with the VMI client, receives and stores messages based on the message types configured, and provides the ability to receive messages and send responses to both alpha-numeric and multiple choice response (MCR) messages. (See [Multiple Choice Response \(MCR\) Matching](#) on page 20 for additional information.)

This deployment diagram shows how the Vocera Incoming VMI Adapter fits within the Vocera appliance in a hospital environment.

Deployment Diagram



Viewing the Vocera Incoming VMI Adapter Requirements

The minimum requirements for an Vocera Incoming VMI Adapter are described here.

System

The adapter is supported on Vocera Interoperability/Engage server version 5.5.0 and later.

Datasets

An adapter defines a default Dataset structure in order to function. Attributes are organized by Datasets and store the information required by the adapter. Adapters use this data during the process of receiving and sending messages.

Not all adapters require Datasets to function. When an adapter does require Datasets, the system will determine if they already exist. If they do not exist, the system will create the needed Datasets.

When creating or editing an adapter, use the following information to select the appropriate datasets in the Required Datasets section.

- The **ALERT_META_DATA Dataset** stores the Metadata for persisting additional Alert information.
- The **DEVICES Dataset** stores all details of every device registered with the Vocera Platform. Each device to which Vocera can send a message must be listed in this dataset.
- The **GROUPS Dataset** stores all user groups.
- The **RESPONSE_OPTIONS Dataset** stores a response option for sending and replying with custom responses.
- The **USERS Dataset** stores all Vocera users.

ALERT_META_DATA Dataset

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|-------------|--------------|-------|-------------|----------|--------|---|
| Attribute | message_key | N/A | True | N/A | N/A | String | Attribute that stores the locally unique message key for the message. |
| Attribute | interface | N/A | False | N/A | False | String | Attribute that stores the reference name of the processing interface. |
| Attribute | message_id | N/A | False | N/A | False | String | Attribute that stores the external identifier for the message. |
| Attribute | sender_id | N/A | False | N/A | False | String | Attribute that stores the external identifier for the client. |

DEVICES Dataset

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|--------|--------------|-------|-------------|----------|-------------|--|
| Attribute | name | N/A | True | N/A | N/A | String | Attribute that stores the name that identifies the device, often based upon the MAC address of the device. |
| Attribute | status | N/A | False | N/A | True | String | Attribute that stores the current registration status of the device. Possible values are Registered, Disconnected, Virtual, or Unregistered. |
| Attribute | vendor | N/A | False | N/A | True | String | Attribute that stores the vendor of the device. For example, Cisco or XMPP. |
| Link | usr | devices | False | False | N/A | Many-to-one | The DEVICES Dataset is linked to the USERS Dataset, and the link order is n:1 (many devices associated to one user) |

GROUPS Dataset

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|--------|--------------|------|-------------|----------|--------|--|
| Attribute | number | N/A | True | N/A | N/A | String | Attribute that stores the number of the recipient. |

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|---------|-------|--------------|-------|-------------|----------|--------------|---|
| Link | users | groups | False | False | N/A | Many-to-many | The GROUPS Dataset is linked to the USERS Dataset, and the link order is m:n (many groups associated to many users) |

RESPONSE_OPTIONS Dataset

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|----------------|--------------|------|-------------|----------|--------|---|
| Attribute | display_value | N/A | True | N/A | N/A | String | Attribute that stores a response value which may be used to display to the recipient. |
| Attribute | response_index | N/A | True | N/A | N/A | String | Attribute that stores the index of the response in the list of responses. |
| Attribute | response_value | N/A | True | N/A | N/A | String | Attribute that stores the response value to be sent back to the originating system. |

USERS Dataset

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|---------------|--------------|-------|-------------|----------|--------|---|
| Attribute | login | N/A | True | N/A | N/A | String | Attribute that stores the login name of the user. |
| Attribute | presence_show | N/A | False | N/A | False | String | Attribute that stores the current presence show value for the user. |

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|---------|---------|--------------|-------|-------------|----------|--------------|---|
| Link | devices | usr | False | False | N/A | One-to-many | The USERS Dataset is linked to the DEVICES Dataset, and the link order is 1:n (one user associated to many devices) |
| Link | groups | users | False | False | N/A | Many-to-many | The USERS Dataset is linked to the GROUPS Dataset, and the link order is m:n (many users associated to many groups) |

Configuring a Vocera Incoming VMI Adapter

Description of the settings that enable direct communication between the Vocera Incoming VMI Adapter and the Vocera Platform.

Select an empty field and begin typing, or select an existing value and type over it. To keep an existing value, do not edit that field.

1. Access the Vocera Platform Web Console and navigate to the adapters.
See [Navigating the Vocera Platform Adapters](#) on page 40 for instructions.
2. Select **New Adapter** in the Action menu, or select an adapter you wish to configure and then select **Edit**, to display the configuration fields. The configuration fields are the same for new and existing adapters.
3. Navigate to the New Adapter option, or navigate to an existing adapter to edit. See [Creating a New Adapter](#) on page 43 and [Editing an Adapter](#) on page 42 for instruction as needed.

The configuration fields are the same for new and existing adapters.

Adapters > IncomingVMI > Details

IncomingVMI Adapter Edit Remove

Reference Name: IncomingVMI

Component Name: IncomingVMI

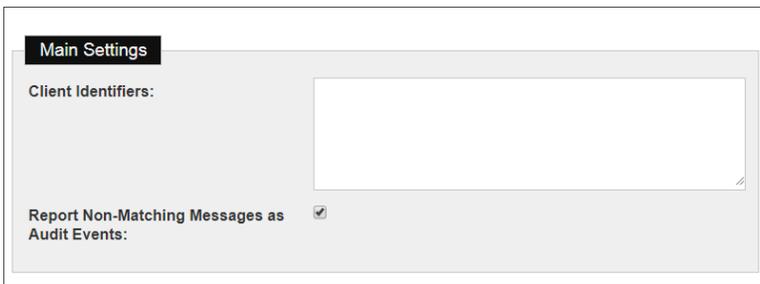
Enabled: true

4. Complete the configuration fields as described in the table.

| Configuration Field | Description |
|---------------------|--|
| Component Name | Click the Component Name field to display a list of the systems and devices that the Vocera Platform currently supports. Select the name of the adapter to create. |

| Configuration Field | Description |
|---------------------|--|
| Reference Name | Enter a short descriptive name in the Reference Name field to uniquely identify an adapter instance. It may demonstrate the adapter function or other information; for example, Production adapter may differentiate a live adapter from a development or "sandbox" adapter. |
| Enabled | Select the Enabled checkbox to allow the Vocera Platform to use the new adapter. The Vocera Platform ignores the adapter if this option is disabled. |
| Required Datasets | If more than one dataset exists that meets the adapter's requirements, select the appropriate datasets for the new adapter to function correctly. The system searches for the datasets that meet the adapters requirements. If the datasets already exist, the system will use them. If the datasets do not exist, the system will create them automatically. Select Create in the drop-down menu to create a new dataset to meet the organization's requirements. |

5. Complete the **Main Settings** configuration fields as described in the table.



| Main Settings | Description |
|--|---|
| Client Identifiers | Enter a list of client identifiers to be matched for incoming messages. If none are specified, then this configuration will handle any message not handled by another active configuration. A configuration can have a maximum of 10 client identifiers. Only one configuration can have no client identifier specified. Client identifiers may contain only the following characters: letter, digit, space, apostrophe, underscore, period, backslash, dash. |
| Report Non-Matching Messages as Audit Events | Select this checkbox to specify whether or not an audit event should be recorded for received messages that do not match a message definition. |

Message Types [Add]

▼ New Message Type, (.+)-(.+)-(.+) Active

Reference Name:

Active:

Discard Message:

Starting Dataset:

Message Regex:

Message Mapping:

bed.bed_number=\$3
 bed.room.room_number=\$2
 bed.room.facility.name=FACILITY
 clinical_id=\$1:\$2:\$3:#{now}
 activity_state=ACTIVE
 alarm_time=#{now.as_iso}

Message Key Path:

Recipient Path:

Priority Path:

Callback Number Path:

Ringtone Path:

Link for Responses :

Response Type Matching

Report Non-Matching Responses as Audit Events :

| Response Regex : | Response Link : | [Add] |
|------------------|---|------------|
| .*(Accept Yes)* | <input type="text" value="accept_response"/> | [Remove] |
| .*(Decline No)* | <input type="text" value="decline_response"/> | [Remove] |

[Clone] [Remove]

▶ **New Message Type (Co..., (.+))** Active

There must be at least one Message Type configured. Message Types can be added and cloned. When the configuration contains multiple Message Types, they can be deleted and reordered via drag and drop. The Message Types are part of the exported settings. Message Types are not available when the adapter is configured for direct forwarding.

| Message Types | Description |
|------------------|---|
| Reference Name | Enter a name to use for the new Message Type, e.g., in audit messages. |
| Active | Select this checkbox to use the configured Message Type. Message Types are ignored by the adapter if not active. |
| Discard Message | Select this checkbox to discard the message without processing it. Matched messages are treated (except for auditing) as if they did not match any message type. Discarding Message Types may be created for the purpose of filtering them from the audit log. |
| Starting Dataset | Select the dataset from which all paths/expressions are calculated and into which the message's data should be stored. Not available for discarding message types; otherwise, this field is required. |

| Message Types | Description |
|---|---|
| Message Regex | <p>Enter a regular expression to capture values from the Message Data of a message received from a VMI endpoint, after any decoding.</p> <p>See Understanding Regular Expressions (Regex) on page 12 for details.</p> <p>This is used to determine which Message Type matches the message. Not available for discarding message types; otherwise, this field is required.</p> |
| Message Mapping | <p>Enter one or more attributes or attribute paths to be filled with data from the Message Regex field configuration, specifying how to store its information relative to the starting dataset.</p> <p>One entry per line. Not available for discarding message types; otherwise, this field is required.</p> |
| Message Key Path | <p>Enter the location into which to store the "global identifier" for the message so that responses and status updates can be sent.</p> <p>Not available for discarding message types; otherwise, this field is required.</p> |
| Recipient Path | <p>Enter the location into which to store the recipient ID specified in the message.</p> |
| Priority Path | <p>Enter the location into which to store the priority specified in the message.</p> |
| Callback Number Path | <p>Enter the location into which to store the callback number specified in the message.</p> |
| Ringtone Path | <p>Enter the location into which to store the ringtone path specified in the message.</p> |
| Link for Responses | <p>Enter the link to the Responses dataset where the matching responses will be stored.</p> <p>This field is optional. If configured, the following fields will display.</p> |
| Report Non-Matching Responses as Audit Events | <p>Check this box to create an audit log entry when a message contains responses but fails to successfully match one or more of the specified response types.</p> <p>This field is only available if Link for Responses is specified. Optional.</p> |
| Response Regex | <p>Enter the regex to match a response type.</p> <p>This field is only available if Link for Responses is specified. Optional.</p> |
| Response Link | <p>Enter the name of the link on the Responses dataset where the response value will be stored.</p> <p>This field is only available if Link for Responses is specified. Optional.</p> |
| Add Message Type | <p>Select Add to create additional message types.</p> |
| Clone Message Type | <p>Select Clone to create a duplicate of the selected message type. The reference name of the cloned message type will automatically be unique, and will be set as inactive by default.</p> |
| Remove Message Type | <p>If one or more message type is created, the ability to remove a message type becomes active. Select Remove to delete the message type from the adapter configuration.</p> |

- Select one of the available options to exit the adapter configuration page. See [Saving an Adapter](#) on page 44 for details.

Understanding Regular Expressions (Regex)

Adapters uses regular expressions (Regex) when parsing incoming messages for storage in a designated dataset in the Data Manager. This page contains an explanation of Regex, describes methods to specify Regex mappings, and provides a quick reference table of operators.

Incoming message data is processed in order to store information from a nurse call system and may later be used by the Workflow Engine to display a message on the end user's device. For example, the Vocera NaviCare Adapter uses Regex mappings configured in the Message Type settings to capture alert data sent by the NaviCare nurse call system.

This document will discuss [Literal Expressions](#), [Statements of Equality](#), and [Global Variables](#).

General Guideline for Regular Expressions

When building a Regex for a facility, be sure to keep the expression as simple as possible while still meeting the needs of the facility. Performance issues, including delays in Alerts, can occur when overly complex Regex strings are written.

The system enforces a maximum processing time limit of five seconds to ensure that CPU loads are not significantly increased by overcomplicated Regex strings. A string that exceeds the threshold will generate an audit event and will not match any results.



Warning: Avoid building extremely complex Regex functions, as performance issues may result.

For example, preprocessing rules might contain inefficient Regex with nested ungreedy matchers such as `(.*?)` that result in excess load on the system.

Below is an example of inefficient Regex for a preprocessor rule:

```
(OBR(\|.*?){7}.?)(\d){4}(\d){2}(\d){2}(\d){2}(\d){2}
```

This example could be expressed more efficiently as the following:

```
"OBR(\|.*){7}.*(\d){4}(\d){2}(\d){2}(\d){2}(\d){2}(\d){2}"
```

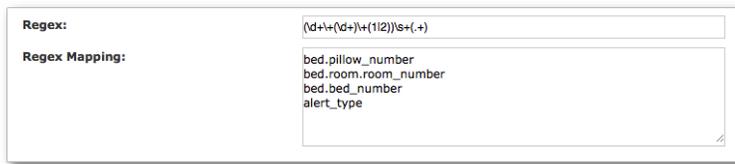
Literal Expressions

Regex describes a search pattern, similar to the way `*.txt` is used to find text files in a file management system. An adapter that uses Message Types to parse incoming message data will define Regex fields to describe the data pattern to match, and a corresponding mapping which describes the attribute expressions to store the data. Each segment in the Regex field corresponds to one line in the Regex Mapping field.

The adapter expects to find data in the defined pattern for processing; if no match is made, the message is not processed. A number of Message Types, with Regex mappings, have to be created to address each and every format or combination of data that the implementation will need to handle.

In this example, the Regex `\d+\+(\d+)\+(1|2)\s+(.+)` maps to the following attribute expressions:

- `bed.pillow_number`
- `bed.room.room_number`
- `bed.bed_number`
- `alert_type`



This Regex mapping results in stored data, which may be used to display on the user device. The Regex `(d+\\(d+\\)+(1|2))s+(.)` may result in a user's device displaying "100:103:1 Code Blue" due to the stored data captured by the following segments.

- "100" is stored by the first segment and mapping, and is the pillow number.
- "103" is stored by the second segment and mapping, and is the room number.
- "1" is stored by the third segment and mapping, and is the bed number. Bed numbers depend on how many beds are in the room.
- "Code Blue" is stored by the fourth segment and mapping, and is the alert type triggered. Alerts are preconfigured to cover conditions ranging from "Code Blue" to "Patient needs water" in a clinical setting.

Literal Expressions Explanation

Regex provides a method of pattern matching where the system expects to find the designated characters in a particular position in the incoming message data. Regex is written in a formal language that can be interpreted by a regular expression processor, which is a program that either serves as a parser generator or examines text and identifies parts that match the provided specification.

Regular expressions have a syntax in which a few characters are special constructs, called metacharacters, and the rest are ordinary. An ordinary character matches that same character and nothing else. The metacharacters are reserved for special search terms and to use one of them as a literal in a Regex, it must be escaped with a backslash (`\`) character.

There are 11 special or metacharacters: the opening square bracket `[`, the backslash `\`, the caret `^`, the dollar sign `$`, the period or dot `.`, the vertical bar or pipe symbol `|`, the question mark `?`, the asterisk or star `*`, the plus sign `+`, the opening round bracket (and the closing round bracket). Any two regular expressions can be concatenated.

The Regex example `(d+\\(d+\\)+(1|2))s+(.)` is described by its concatenated segments:

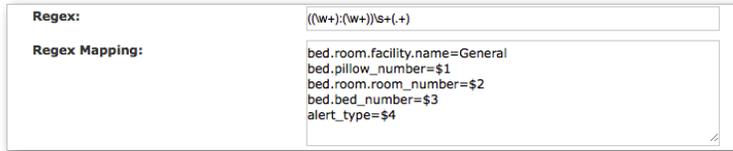
- The `(d+\\)` segment matches the `bed.pillow_number` mapping, which stores digits in the message data that pertain to the pillow number of the patient.
- The `(d+\\)+` segment matches the `bed.room.room_number` mapping, which stores digits in the message data that pertain to the room number of the patient.
- The `(1|2)` segment matches the `bed.bed_number` mapping, which stores digits 1 or 2 in the message data to associate the alert to bed one or two in that room.
- The `s+` tells Vocera Platform to expect any number of spaces after the group, but there must be at least one space.
- The `(.)` segment matches the `alert_type` mapping. This segment catches any other information that is not recognized by the rest of the string.

Statements of Equality

Statements of equality are an alternative way to specify Regex mappings. In a statement of equality mapping, the left-hand side of the equality statement is the attribute path, while the right-hand side is the value of the attribute path. The right-hand side should use numbered captured groups (e.g., `$1`) to reference elements matched, but may also include literal strings. When any item contains an equals sign, then every item is expected to be a statement of equality. If the item is not formatted as a statement of equality (i.e., no equals sign or more than one attribute path or value), then an error message will be written to the audit log indicating that the processing failed for this message type.

In this example, the Regex `((\w+):(\w+))s+(.+)` maps to the following attribute expressions:

- `bed.room.facility.name`
- `bed.pillow_number`
- `bed.room.room_number`
- `bed.bed_number`
- `alert_type`



This Regex mapping represents how the information is sent to Vocera Platform by NaviCare. Using the mapping of `((\w+):(\w+))s+(.+)` the information comes to Vocera Platform as `General301:1 Code Blue`.

A diagram of the expression is: `((\w+):(\w+))s+(.+)`

Pillow Number is equal to `bed.pillow_number=$1`

Room Number is equal to `bed.room.room_number=$2`

Bed Number is equal to `bed.bed_number=$3`

Alert Type is equal to `alert_type=$4`

- "General" is the name of the facility. In the example above General has been hardcoded into the mapping. The string from NaviCare must include General or the message will not be parsed correctly by Vocera Platform.
- The first section of the mapping is the group `((\w+):(\w+))` and represents the Pillow Number. The Pillow Number is derived by taking the Room Number and the Bed Number and separating them with a colon ":". Using the example above, the Pillow number is `301:1`
- Contained within the group of `((\w+):(\w+))`, the first segment is `(\w+)` which is the Room Number. The example above tells Vocera Platform that the room number is 301.
- The colon ":" is used to separate the Room Number from the Bed Number.
- The second part of the group is located after the colon ":" and is `(\w+)` which is the Bed Number. The example above tells Vocera Platform that the bed number is 1. Bed numbers depend on how many beds are in the room.
- The `s+` tells Vocera Platform to expect any number of spaces after the group, but there must be at least one space.
- The final segment of the mapping is `(.+)`. This tells Vocera Platform to expect any number of characters, but there must be at least one character. This is the alert type that is triggered. In our example, the alert type is Code Blue. Alerts are preconfigured to cover the clinical conditions needed by the facility.

Statements of Equality Explanation

Regex provides a method of pattern matching where the system expects to find the designated characters in a particular position in the incoming message data. Regex is written in a formal language that can be interpreted by a regular expression processor, which is a program that either serves as a parser generator or examines text and identifies parts that match the provided specification.

Regular expressions have a syntax in which a few characters are special constructs, called metacharacters, and the rest are ordinary. An ordinary character matches that same character and nothing else. The metacharacters are reserved for special search terms and to use one of them as a literal in a Regex, it must be escaped with a backslash (`\`) character.

There are 11 special or metacharacters: the opening square bracket [, the backslash \, the caret ^, the dollar sign \$, the period or dot ., the vertical bar or pipe symbol |, the question mark ?, the asterisk or star *, the plus sign +, the opening round bracket (and the closing round bracket). Any two regular expressions can be concatenated.

The Regex example `((\w+):(\w+))s+(.+)` is described by its concatenated segments:

- The `((\w+):(\w+))` segment matches the `bed.pillow_number=$1` mapping, which stores the Room Number:Bed Number.
- The `(\w+)` segment matches the `bed.room.room_number=$2` mapping, which stores digits in the message data that pertain to the room number of the patient.
- The `(\w+)` segment matches the `bed.bed_number=$3` mapping, which stores digits in the message data to associate the alert to bed number one or two in that room.
- The `(.+)` segment matches the `alert_type=$4` mapping. This segment stores the alert type information. In our example, the alert type is Code Blue. Alerts are preconfigured to cover the clinical conditions needed by the facility.

Global Variables

A Global Variable represents a capture group that has a fixed set of modifiers. These variables define ways to transform and format data that is stored in the Vocera Platform appliance.

Vocera Platform has created two global variables; `now` which evaluates to the current time of the systems time zone, and `today` which evaluates to the current date. To utilize these variables, select a modifier from the list below.

To format these variables, the modifier must be to the right side of the global variable. For example, selecting the global variable of `now` will add the current time of the system time zone. To format that time as HH:mm, use the modifier of `as_mil_time`. The full global variable is: `#{now.as_mil_time}`.

The modifiers below are also defined and may be used to format or augment the base value of the date/time group, now by adding ".modifier", for example `#{now.as_mil_time}`.

| Modifier | Definition |
|------------------------------|---|
| <code>as_date</code> | The date portion of a date/time returned in the format MM/DD/YYYY. |
| <code>as_iso</code> | A time formatted using ISO8601. The format is YYYY-MM-DD'T'hh:mm:ss.sssTZD |
| <code>as_iso_date</code> | The date portion of a date/time using ISO8601. The format is YYYY-MM-DD. |
| <code>as_mil_time</code> | The time portion of a date/time in the form 'HH:mm' (where HH is 0-23). |
| <code>as_mil_time_sec</code> | The time portion of a date/time in the form 'HH:mm:ss' (where HH is 0-23). |
| <code>as_time</code> | The time portion of a date/time in the form 'hh:mm AM/PM' |
| <code>as_time_sec</code> | The time portion of a date/time in the form 'hh:mm:ss AM/PM'. |
| <code>as_weekday</code> | Returns the name of the week part of a date. |
| <code>with_increment</code> | Returns the succeeding day of the value being modified; may be used multiple times. |
| <code>with_decrement</code> | Returns the preceding day of the value being modified; may be used multiple times. |

Only a subset of the above modifiers may be used to format or augment the base value of the date group, {today} by adding ".modifier", for example #{today.as_date}.

| Modifier | Definition |
|----------------|---|
| as_date | The date portion of a date/time returned in the format MM/DD/YYYY. |
| as_iso_date | The date portion of a date/time using ISO8601. The format is YYYY-MM-DD. |
| as_weekday | Returns the name of the week part of a date. |
| with_increment | Returns the succeeding day of the value being modified; may be used multiple times. |
| with_decrement | Returns the preceding day of the value being modified; may be used multiple times. |

Global Variables Example

The global variables and modifiers that are defined above are used to store a date, time, or date and time stamp on any alert. The following example shows how to use the literal expression, the statement of equality, and the global variables.

The Regex example ((\w+):(\w+))\s+(.+) maps to the following attributes:

- bed.room.facility.name
- bed.pillow_number
- bed.room.room_number
- bed.bed_number
- alert_type
- clinical.alarm_time
- clinical_id

| | |
|----------------|--|
| Regex: | ((\w+):(\w+))\s+(.+) |
| Regex Mapping: | bed.room.facility.name=General bed.pillow_number=\$1 bed.room.room_number=\$2 bed.bed_number=\$3 alert_type=\$4\$ clinical.alarm_time=#{now} clinical_id=MR:General\$1\$2\$3\$4#{now.as_iso} |

This Regex mapping represents how the information is sent to Vocera Platform by the vendor. Using the mapping of ((\w+):(\w+))\s+(.+), the information comes to Vocera Platform as General301:1 Code Blue.

- "General" is the name of the facility. In the example above General has been hardcoded into the mapping. The string from the vendor must include General or the message will not be parsed correctly by Vocera Platform.
- The first section of the mapping is the group ((\w+):(\w+)) and represents the Pillow Number. The Pillow Number is derived by taking the Room Number and the Bed Number and separating them with a colon. Using the example above, the Pillow number is 301:1.
- Contained within the group of ((\w+):(\w+)), the first section is (\w+) which is the Room Number. The example above tells Vocera Platform that the room number is 301.
- The ":" is used to separate the Room Number from the Bed Number.
- The second part of the group is located after the ":" and is (\w+) which is the Bed Number. The example above tells Vocera Platform that the bed number is 1. Bed numbers depend on how many beds are in the room.
- The \s+ tells Vocera Platform to expect any number of spaces after the group, but there must be at least one space.

- The next piece to the mapping is (.+). This tells Vocera Platform to expect any number of characters, but there must be at least one character. This is the alert type that is triggered. In our example the alert type is Code Blue. Alerts are preconfigured to cover conditions ranging from "Code Blue" to "Patient needs water" in a clinical setting.
- The \s+ tells Vocera Platform to expect any number of spaces after the alert, but there must be at least one space.
- The clinical.alarm_time will store the time the alert was received by Vocera Platform, according to the time zone set in the appliance.
- The clinical.id will store the type of device from which the alert was received, (MR), the name of the facility, (General), all of the alert information, (\$1\$2\$3\$4), and transform the time to a date/time formatted using ISO8601, ({now.as_iso}).

Regular Expression Quick Reference

Use the following list of Regular Expression operators to help create useful mappings in a Vocera Platform implementation.

Regular Expressions Anchors

| | |
|----|---|
| ^ | Start of string, or start of line in multi-line pattern |
| \A | Start of string |
| \$ | End of string, or end of line in multi-line pattern |
| \Z | End of string |
| \b | Word boundary |
| \B | Not word boundary |
| \< | Start of word |
| \> | End of word |

Regular Expressions Character Classes

| | |
|----|-------------------|
| \c | Control character |
| \s | White space |
| \S | Not white space |
| \d | Digit |
| \D | Not digit |
| \w | Word |
| \W | Not word |
| \x | Hexadecimal digit |
| \O | Octal digit |

Regular Expressions POSIX

| | |
|-------------|--------------------|
| [[:upper:]] | Upper case letters |
|-------------|--------------------|

| | |
|------------|--------------------------------|
| [:lower:] | Lower case letters |
| [:alpha:] | All letters |
| [:alnum:] | Digits and letters |
| [:digit:] | Digits |
| [:xdigit:] | Hexadecimal digits |
| [:punct:] | Punctuation |
| [:blank:] | Space and tab |
| [:space:] | Blank characters |
| [:cntrl:] | Control characters |
| [:graph:] | Printed characters |
| [:print:] | Printed characters and spaces |
| [:word:] | Digits, letters and underscore |

Regular Expressions Assertions

| | |
|------------|--------------------------|
| ?= | Lookahead assertion |
| ?! | Negative lookahead |
| ?<= | Lookbehind assertion |
| ?!= or ?<! | Negative lookbehind |
| ?> | Once-only Subexpression |
| ?() | Condition [if then] |
| ?() | Condition [if then else] |
| ?# | Comment |

Regular Expressions Quantifiers

Add a ? to a quantifier to make it ungreedy.

| | |
|-------|-----------|
| * | 0 or more |
| + | 1 or more |
| ? | 0 or 1 |
| {3} | Exactly 3 |
| {3,} | 3 or more |
| {3,5} | 3, 4 or 5 |

Regular Expressions Escape Sequences

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

| | |
|----|----------------------------|
| \ | Escape following character |
| \Q | Begin literal sequence |
| \E | End literal sequence |

Regular Expression Common Metacharacters

The escape character is usually the backslash - \.

| | | |
|----|---|---|
| ^ | [| . |
| \$ | { | * |
| (| \ | + |
|) | | ? |
| < | > | |

Regular Expressions Special Characters

| | |
|------|---------------------|
| \n | New line |
| \r | Carriage return |
| \t | Tab |
| \v | Vertical tab |
| \f | Form feed |
| \xxx | Octal character xxx |
| \xhh | Hex character hh |

Regular Expressions Groups and Ranges

Ranges are inclusive.

| | |
|---------|------------------------------------|
| . | Any character except new line (\n) |
| (a b) | a or b |
| (...) | Group |
| (?:...) | Passive (non-capturing) group |
| [abc] | Range (a or b or c) |
| [^abc] | Not a or b or c |
| [a-q] | Letter from a to q |

| | |
|-------|-------------------------------|
| [A-Q] | Upper case letter from A to Q |
| [0-7] | Digit from 0 to 7 |
| \n | nth group/subpattern |

Regular Expressions Pattern Modifiers

| | |
|---|---|
| g | Global match |
| i | Case-insensitive |
| m | Multiple lines |
| s | Treat string as single line |
| x | Allow comments and white space in pattern |
| e | Evaluate replacement |
| U | Ungreedy pattern |

Regular Expressions String Replacement

Some Regex implementations use \ instead of \$.

| | |
|-----|---------------------------------|
| \$n | nth non-passive group |
| \$2 | "xyz-" in / ^ (abc(xyz))\$/ |
| \$1 | "xyz-" in / ^ (? : abc)(xyz)\$/ |
| \$` | Before matched string |
| \$' | After matched string |
| \$+ | Last matched string |
| \$& | Entire matched string |

Ruby Editor Reference

Use the Ruby Regular Expression Editor for troubleshooting: <http://rubular.com/>

Conclusion

This concludes the discussion of Regular Expressions that may be used in a Vocera Platform implementation.

Multiple Choice Response (MCR) Matching

This page describes the MCR functionality at a high level for your understanding, and highlights the necessary configurations.

In some integrations, some adapters, such as Vocera Incoming WCTP Adapter or Vocera Incoming VMI Adapter, can be configured to handle an external system's multiple and arbitrary possible responses to an alert. These adapters, which use protocols that support multiple choice response (MCR) messages, may also be configured to allow the end user to select among multiple possible responses to an alert on their device. This functionality requires manual integration in the standard Vocera solution.

This page describes the MCR functionality at a high level for your understanding, and highlights the necessary configurations. It shows how the adapters for those protocols define the message types and rules to support responding to messages processed as events. It also shows how display values for responses received from a remote system can be displayed in events. Finally, it shows storing responses to support message forwarding where the remote message and all specified options can be forwarded to the user's device.

The standard Vocera Platform EMDAN alerting process provides Accept and Decline responses to an alert. An external system may require a different response option (such as Yes and No) or even a dynamically generated response. Furthermore, a Vocera Platform EMDAN integration may be required to provide the end user with the responses specified by the external sending system. This may occur when an existing Vocera Platform Voice Service integration is upgraded to Vocera Platform EMDAN and that customer's VMI-based user experience must be maintained, which includes responses other than Accept and Decline.

The MCR functionality in this page includes description of:

- AlertMetaData and ResponseOptions datasets which are used to store, send, and reply, using the responses provided by the originating external system.
- Message forwarding where the Vocera Platform simply forwards the message, including the pre-defined responses, to the user's device, and then forwards the response back to the originating system, in order to provide a more message-oriented solution.
- Workflows which present to the end user the response options that were originally provided by the external source system.

Design Overview

The following features must be created on the Vocera system if they do not exist in the customer's implementation.

A ResponseOptions dataset on the Vocera system contains each option for each message received by the MCR-capable adapters. The dataset contains the value to send for that response (stored in the response_value attribute), the display text for that value (stored in the display_value attribute), and an index for the response within the list of responses for the message (stored in the response_index attribute).

Message types are configured on the MCR-capable adapters. The message type configuration requires a path relative to the starting dataset which stores a "message key" that uniquely identifies the message. The message key is used to identify the message so that data for the response can be retrieved.

The message type configuration also allows matching responses to a regular expression. The match process takes a path to a to-one link attribute for the ResponseOptions dataset which is used to link the matching response to the record created by the message type mapping.

The standard Vocera Platform EMDAN solution design requires some additional configuration for the MCR-capable adapters to function. You will have to create a one-to-one link attribute from the Alerts dataset to the AlertMetaData dataset, and name the link meta_data. The message type will store the message key value in the meta_data.message_key path, so the alert record is linked to the adapter's message data. You will also need to create two many-to-one link attributes from the Alerts dataset to the ResponseOptions dataset, and name these links accept_response and decline_response. These link attributes are used by message types in the matching process to identify the response to use for Accept and the response to use for Decline. You can follow this pattern to create additional link attributes for other responses.

Rules are configured on a dataset to deliver event responses back to the originating external system. These rules should be conditioned on the user's response and the existence of the link for that response. When the response is Accept, the rule should send the accepted_response.response_value for the linked alert back to the external system. When the response is Decline, the rule should send the declined_response.response_value for the linked alert back to the external system.

In order to allow end user devices to use the display value specified by the external originating system for event responses, follow the same paths on the rule or workflow that sends the alert to the device to find the correct display value.

In order to support the "message forwarding" scenario functionality:

- Create a many-to-many link attribute named `response_options` from the Alerts dataset to the ResponseOptions datasets. This link stores all responses received by the MCR-capable adapter. This is useful for sending a group of responses to an adapter that can accept a list of responses, such as the outbound Vocera VMI adapter, as well as useful for displaying received responses in a workflow page.
- Vocera adapters used for message forwarding must be configured to display one value and store another value, so that the actual `response_value` attribute can be stored in the response.
- The originating system will be sent a response containing the value that was stored in the ResponseOptions dataset.
- An MCR-capable adapter will have a configurable link to store all of the responses for a message type, and that link can be re-used by any adapter which allows for the sending of a list of response options to an endpoint.

Event Handling Design

The MCR-capable functionality is based on the existing Vocera Platform EMDAN solution. It uses the standard alert processing design, including the existing rules, adapter configurations, and workflows for escalating alerts, sending alerts, and receiving responses.

The primary changes are in the processes that interface with the new MCR-capable adapters:

- Initial message processing (message type definition in the adapters)
- Sending the response back to the originator (rule definition on the datasets)

An exception to these primary changes occurs when the actual values requested by the originating source system are to be sent as responses to the endpoint device. See **Using Requested Display Values** below for details.

The examples in this event handling design assume the following scenario:

- The originating system is sending messages via the Vocera Incoming WCTP Adapter.
- The incoming message represents a "nurse call event".
- The escalation is controlled by the originating system.
- The event recipient is controlled by the originating system.
- The recipient has a Cisco workflow phone.

Other scenarios should work in a similar manner. However, there are a couple of special cases that need further mention:

- If Vocera Platform EMDAN is escalating the alert rather than the originating system, a decision needs to be made at which point(s) to send response(s) to the originating system. Following are the most likely decision points:
 - Sending an accept response immediately after the request is received. This is likely triggered by the creation (or escalation to primary) of the alert record, but the rule configuration is much the same as described below. See **Returning a Response** below for details.
 - Sending an accept response for each accept response received from an end user device. This can be implemented using the same rule(s) as the accept response for an alert escalated by the originating system. (Decline responses on declines and escalations could also be sent, but this seems less likely to be useful.)
 - Sending a decline response when the last escalation level declines or times out. This would trigger as if it were a final escalation, using the same rule configuration as below. See **Returning a Response** below for details.

- If there is a desire to use the "display" responses specified by the originating system, this can be done using the `accept_response` and `decline_response` link attributes. See **Dataset Support** below for details.

Dataset Support

In order to support sending responses after a failover, each adapter will utilize a table in which to store critical information about received messages needed to send responses and status updates. The `AlertMetaData` and `ResponseOptions` datasets are required by the MCR-capable adapters and must be created if they do not exist in the customer implementation. The Alerts links must be manually added to the `AlertMetaData` and `ResponseOptions` datasets.

The **AlertMetaData** dataset is used by all MCR-capable adapters to store additional alerting options that should be persisted across a failover. The `AlertMetaData` dataset must have a locally unique key, a single string "message key", which is required to begin with the interface reference name followed by a ":".

The **ResponseOptions** dataset is used by the adapters to store response options. The `ResponseOptions` dataset provides the following attributes:

- `response_value` attribute: contains the value to be sent back to the originating system.
- `display_value` attribute: contains the value that the originating system requested be displayed to users.
- `response_index` attribute: contains the index of the response within the list of responses as specified by the originating system. The first response is assigned 1, the second 2, etc.

The **Alerts** dataset requires the following additional link attributes be created:

- `meta_data` link attribute: a one-to-one link with the `AlertMetaData` dataset indicating the message that triggered the alert.
- `accept_response` link attribute: a many-to-one link with the `ResponseOptions` dataset indicating the response to be used if the alert is accepted.
- `decline_response` link attribute: a many-to-one link with the `ResponseOptions` dataset indicating the response to be used if the alert is declined.
- `response_options` link attribute: a many-to-many link with the `ResponseOptions` dataset indicating all the responses provided with the message.

Adapter Message Type Definition

Message type definition is similar to the configurations defined on other adapters receiving alerts, with the following significant differences:

- In order to link to the `AlertMetaData` record for the triggering message, the "message key" for the message should be stored in the `meta_data.message_key` field of the alert. In the example here, it is stored in the `NurseCallMessages` dataset for the particular alert, even though `NurseCalls` is the starting dataset, since the `NurseCallMessages` dataset is the triggering record for the alert.
- "Response type" matches must be configured for the `accept_response` attribute, and, if needed, the `decline_response` attribute for the record. Again, this example actually links to the created `NurseCalls` record.

Message Types [Add]

▸ **clinicals, (\w+)** Not Active

▼ **Alert Match, (.+)** Active

Reference Name:

Active:

Discard Message:

Starting Dataset:

Message Regex:

Message Mapping:

Message Key Path:

Sender ID Path:

Recipient ID Path:

Priority Path:

Link for Responses :

Response Type Matching

Report Non-Matching Responses as Audit Events :

| Response Regex : | Response Link : | [Add] |
|---|---|------------|
| <input type="text" value=".*ACCEPT.*OKAY.*"/> | <input type="text" value="accept_response"/> | [Remove] |
| <input type="text" value=".*DECLINE.*NO.*"/> | <input type="text" value="decline_response"/> | [Remove] |

[Clone] [Remove]

This example is effectively a copy of the standard TAP externally escalated Nurse Call message, with the additional support for the multi-response messages. Note the presence of the "Link for Responses" configuration option. In a "message forwarding" scenario, this is used to specify all of the Alert record's "responses".

Returning a Response

To return a response to the originating system, a dataset rule configured for the MCR-capable adapter is used that fires when:

- The correct condition for sending the given response exists. If the response is being triggered by an end user response, this may be a "Response Is Accept" or "Response Is Decline" condition on the Responses dataset.
- The appropriate response link (alert.accept_response or alert.decline_response) is non-null.
- The appropriate meta_data link (alert.meta_data.message_key) is non-null.
- Optionally, for instance if there are multiple multi-response adapter instances, there might need to be a condition to check the interface reference name stored in the metadata (alert.meta_data.interface), or some other mechanism to fire the rule for the correct adapter instance.

The following shows an example configuration for a rule on the Responses dataset. The important values are the Message Key and Response fields; the other values shown are more specific to the adapter, in this example the Vocera Incoming WCTP Adapter.

| Rule Settings | |
|---------------------|---|
| Rule Action: | Send Response |
| Message Key: | #{alert.meta_data.message_key} |
| Response: | #{alert.accept_response.response_value} |
| Response Timestamp: | #{now} |
| Responding User: | #{usr.login} |

Using Requested Display Values

The `accepted_response` and `declined_response` links (or similarly defined links) can also be used to display the values requested by the originating system, instead of the standard values (e.g., Accept and Decline) currently used in the Vocera Platform EMDAN solution. The implementation details depend on whether the display is controlled by the rule (e.g., Badges with XMPP) or by a workflow page (e.g., Cisco devices). In either case, the value to be stored if the response is selected cannot be changed (to prevent breaking other parts of the Vocera Platform EMDAN workflow for event escalations), so the rules to send the responses do not change.

Note that the current outbound Vocera VMI integration cannot display the values requested by the originating system; VMI cannot specify a value to store that is separate from the value to display. Also, using the Vocera XMPP Adapter, this will not work in the Mobile App which does not allow the display value to be overridden.

Workflow

In the standard Vocera Platform EMDAN solution, the display value for the "Accept" button is embedded in the code, in part to prevent breaking the Vocera Platform EMDAN workflow for event escalations. However, the accept response's display value can be used instead.

In this example, a link in the workflow is updated to use a display value for the "Accept" button that is specified by the originating system:

Enter a label for this link:

#{NurseCallMessages.accept_response.display_value}

Enter the url for this link (e.g. http://www.google.com):

/ei/wu/AlertsNCM/accept?Alerts.id=#{NurseCallMessages.id}&Alerts.responses.action=Accepted&Alerts.responses.line=#{line_ap}

Save changes or Cancel

Note that there is a potential issue because the workflow does not allow much conditional display (e.g., use the `display_value` if set, otherwise use "Accept"). This is generally an issue if alerts of a particular type are coming from multiple systems, in which case, this is not recommended.

Rules

The following example shows how to do this, in this case using the Vocera XMPP Adapter and passing the requested value to the badges:



Accept Badge Phrases:

For rules, if there are cases where the `display_value` would not be available, then separate rules would be needed for the default value, probably differentiated by conditions where the value is available and where the value is not available.

Working with the Vocera Incoming VMI Adapter Rules

This page describes the fields available to configure dataset rules for the Vocera Incoming VMI Adapter.

Rules can be configured on a dataset to send a message back to the third party integrator, based on the parameters and rule action provided in the Adapter Settings. When triggered, the Vocera Incoming VMI Adapter rules allow for sending status updates and handling responses.

See the [Vocera Platform Administration Guide](#) for information about working with rules. See [Configuring a Vocera Incoming VMI Adapter](#) on page 8 for information about the adapter settings.

In the Adapter Settings, configure the Rule Settings fields to manage the selected Rule Action. This adapter defines two rule actions, each with its own parameters, plus two parameters common to both rule actions. The configuration fields are described in the table that follows each displayed rule action below.

Send Response Rule Action

Select the Send Response rule action to indicate a response to send to the original message. This rule action will be triggered on the Responses dataset when a condition configured to accept or decline the alert is matched.

| Adapter Setting | Description |
|-----------------|--|
| Rule Action | Send Response |
| Message Key | This field specifies the global message ID of the original message. This field is required. |
| Responding User | This field specifies the user for the message response. This field is required. |
| Response | Specify the response content in the provided text field. This field is required. There is no attempt to enforce the use of an MCR choice. |

Send Status Update Rule Action

Select this rule action to indicate a delivery status change for the original message. This rule action will be triggered by an event that describes the state of a delivery which matches a configured rule on the DeliveryHistory dataset.

Adapter Settings

The information provided is either invalid or incomplete.

Rule Settings

Rule Action:

Message Key:

Affected User:

Status:

| Adapter Setting | Description |
|-----------------|--|
| Rule Action | Send Status Update |
| Message Key | This field specifies the global message ID of the original message. This field is required. |
| Affected User | This field specifies the user for the message update. This field is required. |
| Status | Specify the changed delivery status to return for the original message. This field is required. Status options provided in the dropdown list are Read, Delivered, Callback started, Callback ended. |

Understanding Vocera Incoming VMI Adapter Operations

The Vocera Incoming VMI Adapter acts as a server for third party integrators using the VMI protocol.

As a server, the Vocera Incoming VMI Adapter provides:

- Connection managements
- Receiving messages and sending responses, including:
 - Simple alpha-numeric
 - MCR messages
- Handling of unsupported operations, including optional requests like QueryUser and AddToGroup
- Heartbeat messages

The Vocera Incoming VMI Adapter receives and stores messages according to the message types configured, and Vocera Platform EMDAN handles the delivery of the alerts.

For the operation, generally third party integrators are expected to open a connection and then send a client identification request. Once the client identifier has been received, the connection will be considered valid. Each VMICConnection will have a single client identifier, so there will be as many VMICConnections as necessary. After that vmi requests can be made. The processing of requests or messages work using pipelines created for every VMICConnection. These pipelines take the incoming message and run it through a sequence of handlers, each with their own function. In general, they can be grouped as shown here:

- Encoding/Decoding
- Client Identification Request
- Verifying Client Identification has been sent
- Checking for valid and supported messages
- Error handling

The inbound message might hit several handlers before either being passed to the DatabaseMessageProcessor or being rejected as invalid for any reason. The way messages are processed differs, depending on the mode the Vocera Incoming VMI Adapter is configured for, and is discussed in the **Incoming Message** information in this documentation.

Additional Vocera Messaging Interface documentation is provided in the Vocera Documentation Portal. Access the current version of the Vocera Voice Server documentation in the Portal, and locate the [Vocera Messaging Interface Guide](#) in the list under API Guides.

Connection Management

There are essentially two operations for managing connections:

- Open
- ClientIdentification

Open

Open simply establishes a socket connection. No messages can be sent until a ClientIdentification request is sent. Any connection trying to send a message before the ClientIdentification will be closed after sending an rcFailure response. A VMICConnection is created for each opened socket connection.

ClientIdentification

The client identification request sends the client id. Based on the client id provided, either a processor will be selected based on the configuration, or the connection will be rejected if no such connection is found.

Closing Connections

There is no official method for closing the connection in terms of a request of any kind. Third part integrators just close the connection without any notification for IncomingVMI, and that will be caught as an exception in the pipeline.

Heartbeat

Every 5 seconds, a heartbeat will be sent to each existing connection.

Unsupported Requests

Unsupported operations (including the operation DeleteMessage) will result in a failure confirmation message using the response code rcFailure.

The following table contains a list of requests and indicates whether or not they are supported.

| Request | Operation Support |
|---------|-------------------|
| Open | Supported |

| Request | Operation Support |
|----------------------|-------------------|
| ClientIdentification | Supported |
| Message | Supported |
| DeleteMessage | Not Supported |
| AddToGroup | Not Supported |
| LogEvent | Not Supported |
| QueryGroup | Supported |
| QueryUser | Supported |
| RemoveFromGroup | Not Supported |

Message Support

The message support is divided into three separate operations:

- Handling the incoming messages and requests
- Handling rules to send status updates
- Handling rules to send responses

Incoming Messages

The Vocera Incoming VMI Adapter supports request/response communication for the message operations defined in the VMI protocol.

It does not support:

- Receiving status for and responses to responses it has sent.
- Sending messages other than responses.

If the Vocera Incoming VMI Adapter receives an unsupported message payload, it treats the message as an unsupported request (see above).

Otherwise the message request is processed as follows:

1. The adapter finds an active configuration whose client identifier(s) match the message's client id, or if that fails an active configuration with no configured client identifiers. If no such configuration can be found, a response code of rcFailure is sent and the message is ignored.
2. The message text is extracted from the message and compared against each of the configurations message types until one is found that matches and (if not a discard) its messages content mapping is successfully applied:
 - a. If a message type successfully matches the message, but the message mapping fails, a message is audited and attempts to match continue.
 - b. If not discarding, an audit is generated for the matched message type.
 - c. If not discarding, the mapping data from the matched message type (including message mapping and the paths for the recipient, priority, callback number, ringtone, and MCR choices) is used to build a record and store it in the configured dataset.
 - d. If the message type is a discard or no message type is matched, a response code of rcFailure is sent. An audit is also generated if no message type matches. Otherwise a success confirmation is sent (rcSuccess).

Note that a global message key is calculated for each stored message from the messageID field of the message and the client identifier from the third party integrator. This is stored in the configured message key path and should be passed to the adapter's rules in the message key field.

A successfully received message also has the following data stored in the database, keyed by the global message key:

- The message id.
- The client identifier to which responses and status updates should be sent.

Status Updates

For a status update, a MessageStatus signal is built and sent containing:

- The original message's id as the message id based on the rule supplied message key.
- The rule supplied affected user as the recipient.
- The rule supplied response as the MessageStatusCode for the message.

Responses

Message responses are built as follows and sent:

- The original message's id as the message id based on the rule supplied message key.
- The rule supplied responding user as the recipient.
- The rule supplied status as the response for the user for a message.

Requests

There are other requests that are supported by the Vocera Incoming VMI Adapter; QueryUser, and QueryGroup.

QueryUser

The Vocera Incoming VMI Adapter supports requests for UserInfo based on a user login.

Depending on the user, the following information is returned:

- If the user does not exist, an rcInvalidLoginId is returned.
- If the user does exist, an rcSuccess is returned along with a UserInfo signal. The signal contains:
 - The user id
 - The status of the user
 - If the user is available and connected to a registered device the status scOnline is returned.
 - If the user is linked to a device but not online, the status scNotOnline is returned.
 - Otherwise the status scNotLoggedIn is returned.

QueryGroup

The Vocera Incoming VMI Adapter supports requests for GroupInfo based on a group name. The adapter will ignore any Site qualifiers for groups.

Depending on the group the following information is returned:

- If the group does not exist or two groups with the same name exist, rcInvalidGroupName is returned.
- If the group does exist, an rcSuccess is returned along with a GroupInfo signal. The signal contains:
 - The number of users in the group.
 - Up to 100 members in the group.

Integrating the Vocera Incoming VMI Adapter with Vocera Platform

The Vocera Incoming VMI Adapter has the unique ability to process alerts and store responses dynamically. These responses can be used later in rules to send messages with response options.

The multiple choice response (MCR) matching behavior is controlled by the alerts and responses stored in Vocera datasets. See [Multiple Choice Response \(MCR\) Matching](#) on page 20 for information about how an external alerting system's multiple arbitrary responses can be handled.

The adapter operates in two different modes; one mode provides direct forwarding only and supports the Vocera Badge, while the other mode utilizes the full solution functionality.

When using the Vocera Platform EMDAN mode, the Vocera Incoming VMI Adapter has the ability to process alerts and store responses dynamically. These responses can be used later on in rules to send messages with response options. There are also IncomingVMI rules to send responses and status updates to the third party client.

The other mode is direct forwarding, and supports only Vocera Badge. In this mode, messages from a third party client are sent directly to the Vocera Voice Server. When using this mode, some meta data for alerts is stored, but otherwise no alert data is stored, and no rules are needed. For direct forwarding, only the AlertMetaData dataset and links is necessary.

Configuring AlertMetaData Dataset

The Vocera Incoming VMI Adapter must store additional information about alerts in the standard Vocera data model, because of the nature of the response mechanism and the need to persist data across a failover. This data will be persisted in a separate dataset, the AlertMetaData dataset.

The AlertMetaData dataset is tied to the original alert by a locally unique identifier, referred to as the Message Key, which is stored on the dataset as the message_key attribute.

The Vocera Incoming VMI Adapter is configured with a message type for matching incoming messages; the starting dataset is selected when creating that message type for matching.

ALERT_META_DATA Dataset Attributes

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|-------------|--------------|-------|-------------|----------|--------|---|
| Attribute | message_key | N/A | True | N/A | N/A | String | Attribute that stores the locally unique message key for the message. This key is generated by the adapter, it uniquely identifies the message, and is the key for the AlertMetaData dataset. |
| Attribute | interface | N/A | False | N/A | False | String | Attribute that stores the reference |

| | | | | | | | |
|-----------|------------|-----|-------|-----|-------|--------|---|
| | | | | | | | name of the processing interface which was used to process the message. |
| Attribute | message_id | N/A | False | N/A | False | String | Attribute that stores the external identifier for the message. |
| Attribute | sender_id | N/A | False | N/A | False | String | Attribute that stores the sender identifier. |

Configuring the AlertMetaData Dataset

This is the dataset that will store all additional alerting options that should be persisted across a fail over. First create a one-to-one link between either the Alerts dataset or other Starting dataset, and the AlertMetaData dataset. Then configure the Vocera Incoming VMI Adapter, specifying this link when providing the message_key attribute in the Message Type settings.

- message_key - The "natural key" for the dataset, which uniquely identifies the message. It will be generated by the adapter. This is also the key for the AlertMetaData dataset.
- message_identifier - The message identifier supplied by the source system
- sender_id - The identity of the sender
- interface - The id of the interface which was used to process the message

Create Links on the Starting Dataset to Connect with the AlertMetaData Dataset

When configuring the Vocera Incoming VMI Adapter, it is necessary to create a one-to-one link between either the Alerts or Starting dataset and the AlertMetaData dataset. This link is then specified when providing the message_key in the adapter configuration.

Create the link attribute on the starting dataset that is specified in the rule on the dataset.

| Name | Reverse Name | Linked Dataset | Type | Key | Reverse Key | index |
|-----------|--------------|----------------|------------|-------|-------------|-------|
| meta_data | alerts | AlertMetaData | one-to-one | false | false | false |

Specify the Message Key and Link in the Adapter Configuration

Use the link attribute that connects the starting dataset to the AlertMetaData dataset (meta_data) and join it with the message_key attribute on the AlertMetaData dataset to configure the Message Key Path field in the Message Type settings in the Vocera Incoming VMI Adapter.

Configuring the ResponseOptions Dataset

Because an MCR message may contain multiple responses, the Vocera Incoming VMI Adapter must be able to store each response type and have a default Accept response and a default Decline response.



Note: This information does not apply when the implementation uses Direct Forwarding mode; do not modify the ResponseOptions dataset in Direct Forwarding mode.

Each MCR type may vary in response options between adapter configurations and message types, so each message type should be configured with a regex to match the correct Accept response, and a separate regex to match the correct Decline response. The message type that the message was matched to is configured with the Starting Dataset.

RESPONSE_OPTIONS Dataset Attributes

| Element | Name | Reverse Name | Key | Reverse Key | Required | Type | Description |
|-----------|----------------|--------------|------|-------------|----------|--------|---|
| Attribute | display_value | N/A | True | N/A | N/A | String | Attribute that stores a response value which may be used to display to the recipient. |
| Attribute | response_index | N/A | True | N/A | N/A | String | Attribute that stores the index of the response in the list of responses. |
| Attribute | response_value | N/A | True | N/A | N/A | String | Attribute that stores the response value to be sent back to the originating system. |

ResponseOptions dataset will be the dataset where all MCR responses are stored.

- response_value - The actual response value that should be sent back.
- display_value - This is an optional separate "display value" to be shown to the user when choosing the response.
- response_index - This is the numerical order of in which the responses were received, or some order determined by the adapter.

Configuring the ResponseOptions Dataset

When configuring the Vocera Incoming VMI Adapter to receive responses, it is necessary to create a many-to-many link between either the Alerts or Starting dataset, and the ResponseOptions dataset.

While other adapters may have a distinction between the displayed value and the sent value for responses, IncomingVMI only has one possible value. For IncomingVMI, that value will be stored in both `display_value` and `response_value`. Attribute transformations may be used to further make a distinction for these values if a different `display_value` is desired.

First create a many-to-many link between either the Alerts dataset or other Starting Dataset, and the ResponseOptions dataset. Then configure the Vocera Incoming VMI Adapter, specifying this link when providing the Link for Responses field in the Message Type settings. Finally, create many-to-one links between the Starting Dataset and the ResponseOptions dataset so that the specific response can be accessed in a rule.

Create Links on the Starting Dataset to Connect with the ResponseOptions Dataset

Create the link attribute on the starting dataset that is specified in the rule on the dataset. Create the `response_options` attribute to link to the ResponseOptions dataset from the Alerts dataset in this example.

| Name | Reverse Name | Linked Dataset | Type | Key | Reverse Key | index |
|-------------------------------|--------------|-----------------|--------------|-------|-------------|-------|
| <code>response_options</code> | Alerts | ResponseOptions | many-to-many | false | false | false |

Specify the Link in ResponseOptions

Use the link attribute created to connect with the ResponseOptions dataset in the Vocera Incoming VMI Adapter configuration. Provide the new link attribute in the Link for Responses field in the Message Type settings.

Create Links on Starting Dataset for Sending Rules

Create many-to-one links between the StartingDataset dataset and the ResponseOptions when matching message types. These links provide the specific response to be accessed in a rule to send to the user or to send back to the originating system.

| Name | Reverse Name | Linked Dataset | Type | Key | Reverse Key | index |
|-------------------------------|-----------------------------|-----------------|-------------|-------|-------------|-------|
| <code>accept_response</code> | <code>accepted_alert</code> | ResponseOptions | many-to-one | false | false | false |
| <code>decline_response</code> | <code>declined_alert</code> | ResponseOptions | many-to-one | false | false | false |

Specify the Links in the Adapter Configuration

Use the link attributes in the Response Type Matching settings in the Vocera Incoming VMI Adapter configuration. These many-to-one links can then be supplied in a rule to send.

Use the Links in a Send Rule on a Dataset

These many-to-one links can then be supplied in a rule to send:

These links can then be supplied in a rule to send a message to a user or to send back to the originating system. Configure the Response List field in the Rule Settings on a dataset rule that uses the Vocera adapter in the example below.

Example using Vocera Send Rule

Example for IncomingVMI Response Rule

Handling Message Transformations

In the original VMI protocol, there is a transformation done by the Vocera Voice Server or other systems. This cannot be done now that the messages are stored in the database directly for Vocera Platform EMDAN. If a third party integrator is using something that needs transformed, adding an attribute transformation to the message or other values is required. The notation [CR] is used to indicate a line break.

For example, we can receive a message like 'Hello[CR]Hi again'. If no transformation is added [CR] will be displayed directly. It is necessary to replace the [CR] with a line break the adapters will recognize. For example, if using XMPP then the attribute transformation should replace the [CR]with \n, as that will indicate a new line for XMPP.

Understanding Adapter Installation

Adapters are installed on the Vocera Platform in a solution package, or individually as needed by the customer.

The Vocera Platform uses adapters to integrate with external systems and devices. Each adapter is configured by the user to include information that will allow the Vocera Platform to communicate and interact with a specific type of resource and, depending on the adapter, devices that resource may control. Adapters can allow the Vocera Platform to monitor and collect data, as well as send data out, when triggered manually or automatically.

When implementing Vocera Platform at a customer site, use this document to install an adapter that is not supplied in the Gold Image. Otherwise, you will install a needed adapter when instructed in the solution package installation process described in the [Vocera Platform Installation Guide](#).

Recreating a Repository

In the event that the repository reference file has been compromised, you can re-create the platform repository.

This information should be specified on the related adapter's Release Information page in the wiki. See **Releases** and navigate to the needed adapter.

1. Verify that the adapter resides in a repository which is in `/etc/yum.repos.d/`.
2. If the **repolist** or **yum** commands fail, verify that the file exists and try again. For example, use the following code to verify the repository exists on the Vocera Platform appliance:

```
[tpx-admin@engage log]$ cat /etc/yum.repos.d/vocera.repo
```

3. Verify the output appears as shown.

```
#-----  
# NOTICE: Only use the General Availability (platform-6.X-ga) repository for customer  
# deployments.  
# Use of Controlled Release (platform-6.X-cr) or Software Quality Assurance  
# (platform-6.X-sqa) in  
# accordance to process QOP-75-01 Production Work Order and History Record, contact  
# your  
# manager for questions.  
#-----  
[Platform-6.0]  
name=Platform-6.0  
baseurl=https://box.voceracommunications.com/Platform-6.0-GA  
enabled=1  
gpgcheck=0
```

Installing an Adapter

Install or uninstall a Vocera Platform adapter at a customer site on a Vocera system for a customer.

Execute the following steps using the system's command prompt.

1. Verify that the adapter resides in a repository which is in `/etc/yum.repos.d/`.
2. Run the following commands:

```
sudo yum clean all
sudo yum check-updates
```

3. Verify that the rpm package to be installed is available using the following command:

```
sudo yum list available | grep extension
```

4. Install the adapter by specifying its rpm package name in place of `<package-name>` in the code below. (This information should be specified on the related Release Information page in the wiki; see **Release Notes**.)

```
sudo yum install <package-name>
```

5. Uninstall an adapter by specifying its rpm package name in place of `<package-name>` in the code below. (This information should be specified on the related Release Notes page; see **Release Notes**.)

```
sudo yum remove <package name>
```

Practicing an Adapter Installation

Replicate these steps using the needed adapter package, in order to install adapters other than the example given here.

1. Verify the repo file contains the repos up to and including the release of interest.

```
[tpx-admin@engage log]$ cat /etc/yum.repos.d/vocera.repo
#-----
# NOTICE: Only use the General Availability (platform-6.X-ga) repository for customer
# deployments.
# Use of Controlled Release (platform-6.X-cr) or Software Quality Assurance
# (platform-6.X-sqa) in
# accordance to process QOP-75-01 Production Work Order and History Record, contact
# your
# manager for questions.
#-----
[Platform-6.0]
name=Platform-6.0
baseurl=https://box.voceracommunications.com/Platform-6.0-GA
enabled=1
gpgcheck=0
```

2. Execute the following commands:

```
[tpx-admin@engage log] $ sudo yum check-updates
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Quartz
(1/2): Quartz/group_gz | 3.6 kB 00:00:00
(2/2): Quartz/primary_db | 483 B 00:00:00
| 29 kB 00:00:00
```

3. Verify the package is available, using the following command:

```
[tpx-admin@engage log] $ sudo yum list available | grep extension
extension-навicare-interface.x86_64      1.3.6-0      Platform 5.0
```

4. Install the needed adapter; in this example, install the Navicare adapter:

```
[tpx-admin@engage log] $ sudo yum install extension-навicare-interface
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Resolving Dependencies
--> Running transaction check
---> Package extension-навicare-interface.x86_64 0:1.3.6-0 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package                               Arch                               Size
Version                               Repository                         Size
=====
Installing:
extension-навicare-interface          x86_64                             59 k
1.3.3-0                                Quartz
```

Transaction Summary

Install 1 Package

Total download size: 59 k

Installed size: 62 k

Is this ok [y/d/N]: y

Downloading packages:

```
extension-навicare-interface-1.3.6-0.x86_64.rpm
| 59 kB 00:00:00
```

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

```
Installing : extension-навicare-interface-1.3.6-0.x86_64      1/1
Verifying  : extension-навicare-interface-1.3.6-0.x86_64      1/1
```

Installed:

```
extension-навicare-interface.x86_64 0:1.3.6-0
```

Complete!

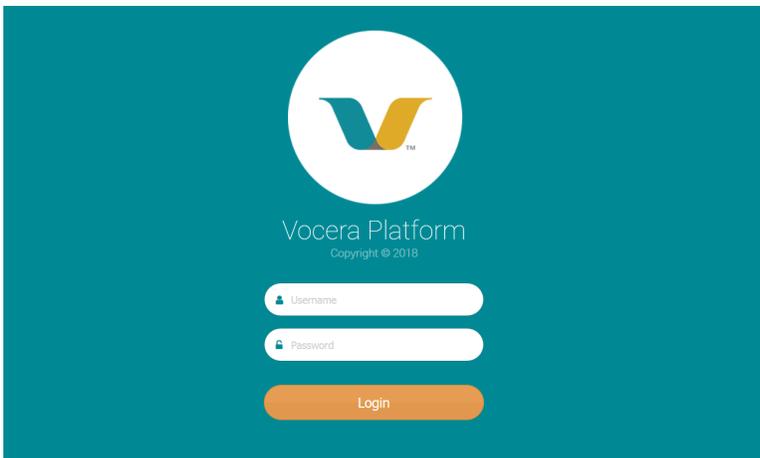
5. This completes the steps to install an adapter.

Navigating the Vocera Platform Adapters

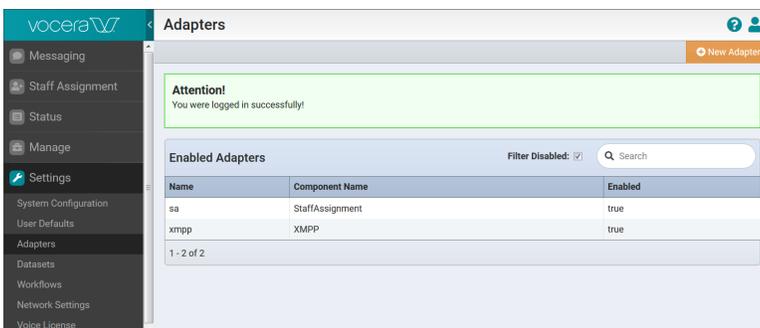
Access the Adapters tab and use the filter or search tools to display a specific adapter.

This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

1. Access the Vocera Platform Web Console and sign in with your system credentials.



2. Select **Settings > Adapters** in the navigation menu.

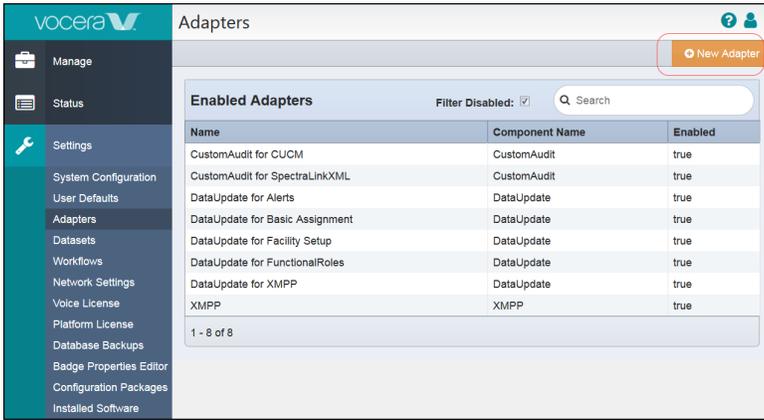


The **Adapters** page displays.

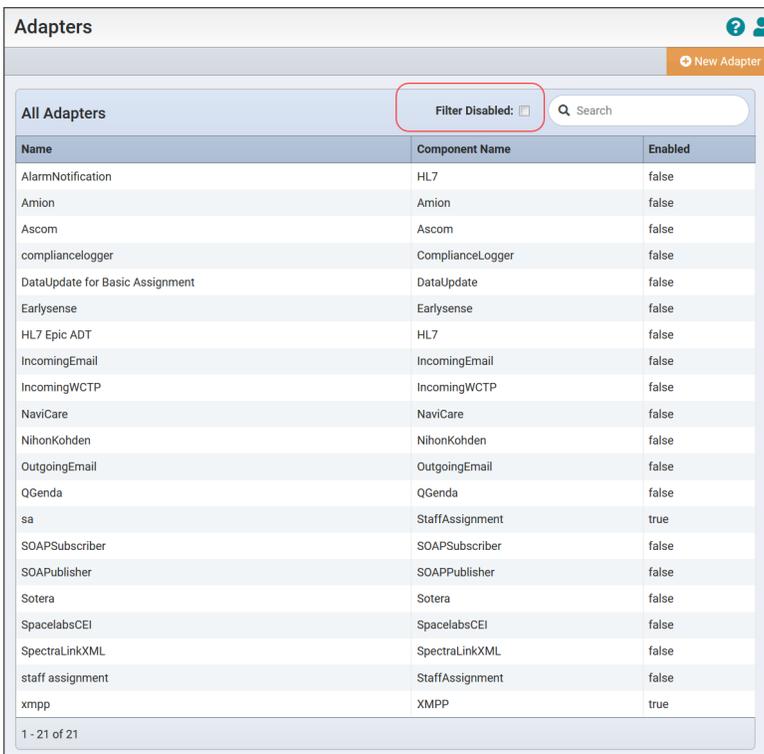
3. Select an adapter to work with from the list displayed in the grid, or select the **New Adapter** Action option to create a new adapter.

On the **Adapters** page you can identify adapters by their name or component name. The Enabled column (displaying a true or false status) indicates whether the adapter is active on the system, or disabled.

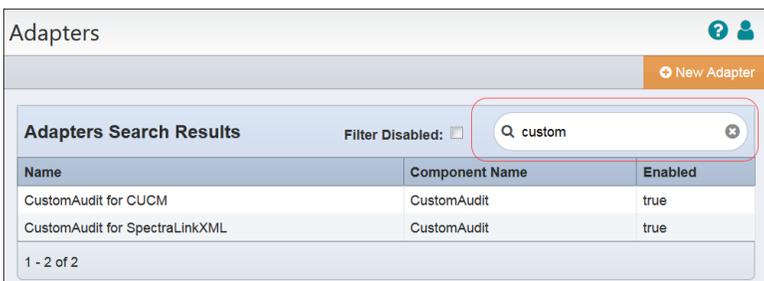
The bottom row of the grid reports the number of adapters displayed, of the available adapters. The Filter Disabled box is checked by default, and displays only the enabled adapters that are configured on the Vocera Platform.



- Uncheck the **Filter Disabled** box to display all the adapters that have been installed, including those that are not currently enabled. The column title now displays **All Adapters**. The Filter Disabled box is checked by default.



- Enter a term in the **Search** field to locate a needed adapter on the system. The search field is identified by a text field with a magnifying glass icon. The search is performed on the Name and Component Name columns. When results are returned, the column header displays **Adapters Search Results** and an **x** icon allows you to clear the search field.



Editing an Adapter

Edit an adapter that has been installed on the Vocera Platform.

This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

1. Access the Vocera Platform Web Console and navigate to the adapters.
See [Navigating the Vocera Platform Adapters](#) on page 40 for instructions.
2. Select the adapter to edit in the **Adapters** list.

| Name | Component Name | Enabled |
|----------------------|------------------|---------|
| AlarmNotification | HL7 | false |
| Ascom | Ascom | false |
| ComplianceLogger | ComplianceLogger | false |
| CUCM | CUCM | false |
| CustomAudit for CUCM | CustomAudit | true |

3. Select **Edit** in the adapter's menu.

The **Update Adapter** page for the adapter displays.

4. Edit the adapter's settings to revise the configuration as needed. See the adapter-specific configuration page for details on working with settings for this adapter.
Select an empty field and begin typing, or select an existing value and type over it. To keep an existing value, do not edit that field.

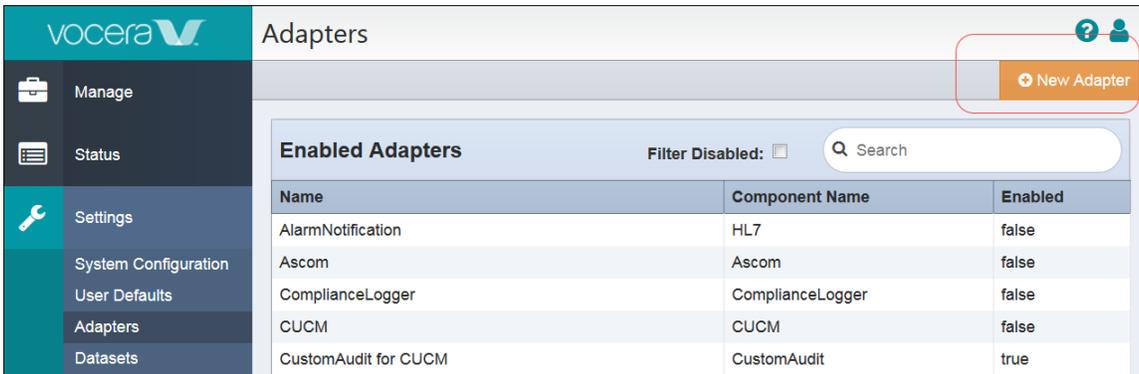
5. Select one of the options to exit the **Update Adapter** page. See [Saving an Adapter](#) on page 44 for details.

Creating a New Adapter

Access the Vocera Platform Web Console to work with adapters, or create a new adapter when prompted in the package import process.

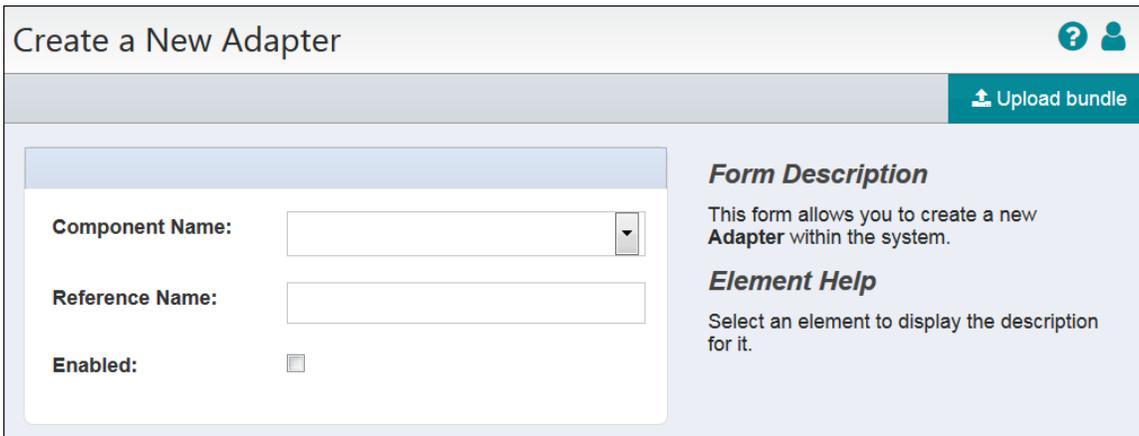
This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

1. Access the Vocera Platform Web Console and navigate to the adapters.
See [Navigating the Vocera Platform Adapters](#) on page 40 for instructions.
2. Select **New Adapter** in the Action menu on the Adapters page.



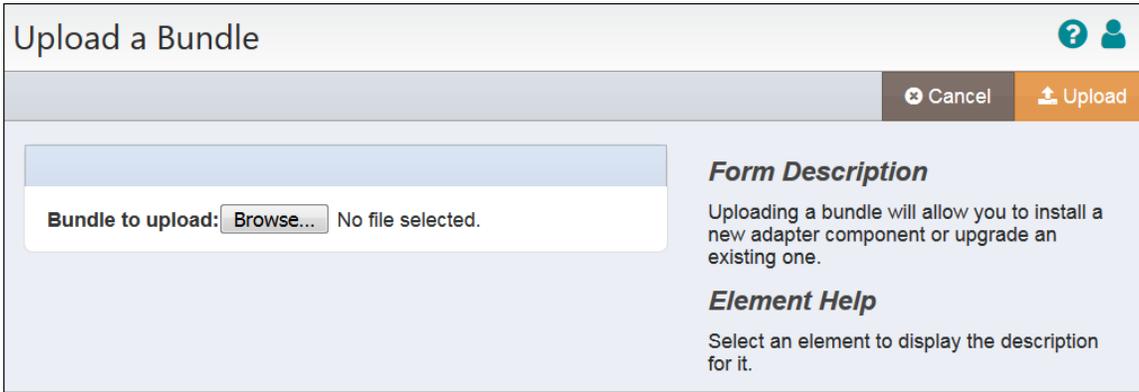
The **Create a New Adapter** dialog displays.

3. Complete the configuration fields.



| Name | Description |
|-------------------------|--|
| Component Name * | Select the Component Name field dropdown arrow to display a list of the systems and devices that Vocera currently supports. Select the name of the adapter to create. |
| Reference Name | Enter a short descriptive name in the Reference Name field to uniquely identify an adapter instance. It may demonstrate the adapter function or other information; for example, Production adapter may differentiate a live adapter from a development or "sandbox" adapter. |
| Enabled | Select the Enabled check box to allow Vocera Platform to use the new adapter. Vocera ignores the adapter if this option is disabled. |

4. Select **Upload Bundle** in the Action menu to install a package on a Vocera Platform.
Use the Upload Bundle feature to install when the adapter is not available in the Component Name dropdown list, and you have downloaded the needed adapter bundle to a storage location.
5. Click on **Browse** to navigate to the bundle to install.



6. Select one of the Action options to exit from the Upload a Bundle dialog.
 - **Upload:** Upload the selected bundle to the appliance.
 - **Cancel:** Close the Upload a Bundle dialog without making a change to the system.

Saving an Adapter

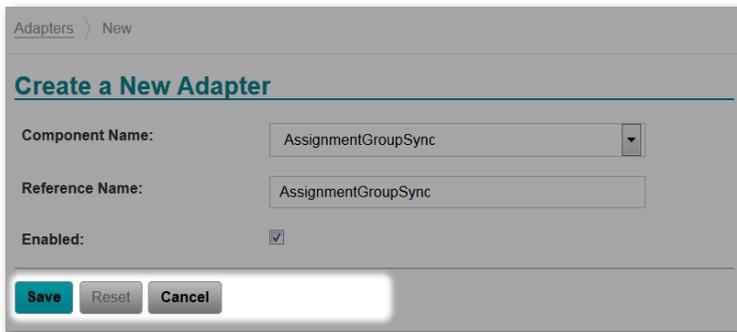
Close an adapter configuration dialog using the Save, Reset, or Cancel options.

This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

When creating a new adapter, the options at the bottom of the adapter configuration page are Save, and Cancel.

When editing an existing adapter, the options are Save, Reset, and Cancel.

Choose an option to close the dialog:



| Option | Description |
|---------------|--|
| Save | Select Save to store the adapter configuration in the system, when the fields are set to desired specifications. |
| Cancel | Select Cancel to close the configuration window without saving your changes to the system. |
| Reset | Select Reset to clear all fields without closing the window, in order to select other specifications for the adapter's settings. |

Deactivating an Adapter

Temporarily deactivate an adapter to avoid unintentional use of it in an implementation.

This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

1. Access the Vocera Platform Web Console and navigate to the adapter to deactivate.
See [Navigating the Vocera Platform Adapters](#) on page 40 for instructions.
2. Select **Edit** in the Actions menu to access the Update page for the adapter.

XMPP Adapter

Remove Edit

Reference Name: XMPP
Component Name: XMPP
Enabled: true

Main Adapter Settings Version: 4.0.0.175

It Might Help to Know...
You are currently viewing the primary details for this adapter. You may also choose to edit or remove it from the system by selecting the appropriate action on the page. Areas that are highlighted with a red background indicate that there is missing information.

3. Un-check the **Enabled** box to temporarily deactivate the adapter.
When deactivated, the Vocera system will ignore the adapter. You can easily enable or disable the adapter at any time.

Update Adapter

Reference Name: XMPP

Enabled:

Required Datasets

Actors: Actors

Assignments: Assignments

Form Description
Edit the details for the 'XMPP' adapter.

Element Help
Select an element to display the description for it.

4. Select one of the options to exit the **Update Adapter** page. See [Saving an Adapter](#) on page 44 for details.

Removing an Adapter

Permanently remove an adapter from the Vocera system.

This page is used by all the adapter guides, and therefore, the adapter used as an example here may not be the adapter that you are working with currently.

Use the remove function to permanently delete the adapter from the system. Alternatively, you can **disable** an adapter and the Vocera system will ignore it.



Warning: Remove cannot be undone. If any system features use this adapter, removing the adapter prevents the features from functioning.

1. Access the Vocera Platform Web Console and navigate to the adapter to remove.
See [Navigating the Vocera Platform Adapters](#) on page 40 for instructions.
2. Select **Remove** in the Actions menu to permanently delete the adapter.

XMPP Adapter

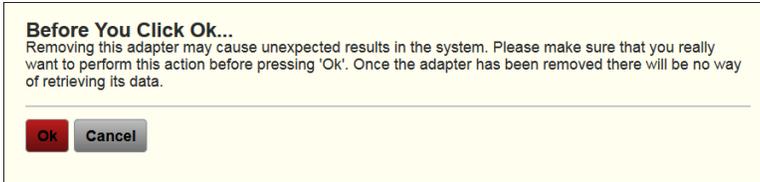
Remove Edit

Reference Name: XMPP
Component Name: XMPP
Enabled: true

Main Adapter Settings Version: 4.0.0.175

It Might Help to Know...
You are currently viewing the primary details for this adapter. You may also choose to edit or remove it from the system by selecting the appropriate action on the page. Areas that are highlighted with a red background indicate that there is missing information.

3. Click **Ok** in the confirmation window.



- **Ok:** Confirm the choice to remove the adapter from the system.
- **Cancel:** Return to the adapter page without making a change.

4. Confirm that the adapter no longer displays in the Adapters list view, when a success message displays.

