Oracle® Communications Operations Monitor

User's Guide

Release 3.3.92

E74351-01

June 2016

ORACLE

Oracle Communications Operations Monitor User's Guide, Release 3.3.92

E74351-01

Contents

4 Operations

5 Customers

6 Control Plane Monitor

7 Settings

8 Call Merging Algorithms

9 Implementing Apps

Preface

This guide describes how to use Oracle Communications Operations Monitor to monitor, detect, and troubleshoot IP Multimedia Subsystem (IMS), Voice over Long-Term Evolution (VoLTE), and next-generation network (NGN) networks.

This guide applies to both the Oracle Communications Operations Monitor and the Oracle Communications Enterprise Operations Monitor.

The Oracle Communications Session Monitor product family includes the following products:

- Operations Monitor
- Enterprise Operations Monitor
- Fraud Monitor
- Control Plane Monitor

Audience

This guide is intended for network administrators in charge of managing and operating IMS (based on SIP), VoLTE, and NGN multimedia networks and support personnel who identify and solve customer issues.

Downloading Oracle Communications Documentation

Oracle Communications Session Monitor documentation and additional Oracle documentation is available from the Oracle Help Center Web Site:

http://docs.oracle.com

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Document Revision History

The following table lists the revision history for this document:

Version	Date	Description
E74351-01	June 2016	Initial release.

1

Connecting Operations Monitor to Your Network

Oracle Communications Operations Monitor software module includes the operating system, and can be installed on standard x86-64 architecture machines. The number of network ports may vary, although a minimum of two is required.

Operations Monitor distinguishes between two kinds of network interfaces:

Management interfaces

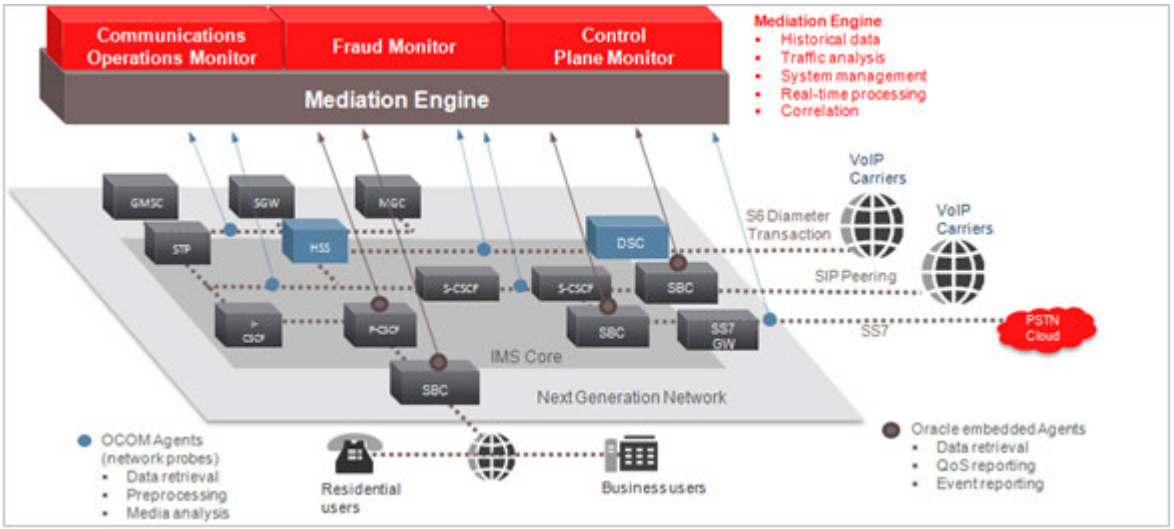
Management interfaces are used for accessing the Operations Monitor web user interface and other management network traffic (including REST, CDR access, SNMP, ICMP, connection to probe).

Monitoring interfaces

Monitoring interfaces are used for passively receiving the signaling and/or media traffic to be analyzed by the Oracle Communications Session Monitor (Session Monitor) family of products. Monitoring interfaces do not require an IP address and are not assigned one by default.

Figure 1–1 represents where network traffic can be sent to Operations Monitor.

Figure 1–1 Connecting Operations Monitor



The Oracle Communications Session Monitor Mediation Engine can receive traffic from multiple locations on the network and provide a single end-to-end view of the

network. The more data that is made available to Operations Monitor, the more comprehensive the results are. The signaling or media-related traffic captured depends on the licensed extensions. Licenses include, but are not limited to:

- SIP
- RTP
- RTCP
- SIGTRAN/ISUP/SS7
- H.248/Megaco
- MGCP
- Diameter
- ENUM

The traffic is copied from the active paths to the monitoring interfaces using mirroring port features of network switches (for example SPAN), remote port mirroring protocols (for example RSPAN), or tapping devices. In addition network data can be retrieved directly by configuring the Operations Monitor probe within Oracle Communications Session Border Controller (Session Border Controller). Using Session Border Controller is the only way to analyze otherwise encrypted signaling traffic.

Note: For information on configuring the network switches, refer to the manual provided by your switch manufacturer.

The number of monitored network segments is not limited by the number of network interfaces of the machine that hosts Operations Monitor. Aggregating traffic from multiple tapping points into one single *monitoring* port works as well as configuring any number of external network probes.

Note: The Session Monitor product family is not compatible with third party probes.

2

Initial Configuration

This chapter guides you through the configuration steps required to get Oracle Communications Operations Monitor started. You must first install an instance of Operations Monitor, connect it to your network, and access the web interface via a web browser. See the referenced sections when changing settings for additional required information.

> **Note:** The web interface requires a browser with Javascript enabled. Refer to "Requirements" for the list of supported browsers. Allowing HTTP cookies is strongly recommended to enable several convenience features.

Logging In

Point the browser to the configured IP address of the management interface to load the login screen.

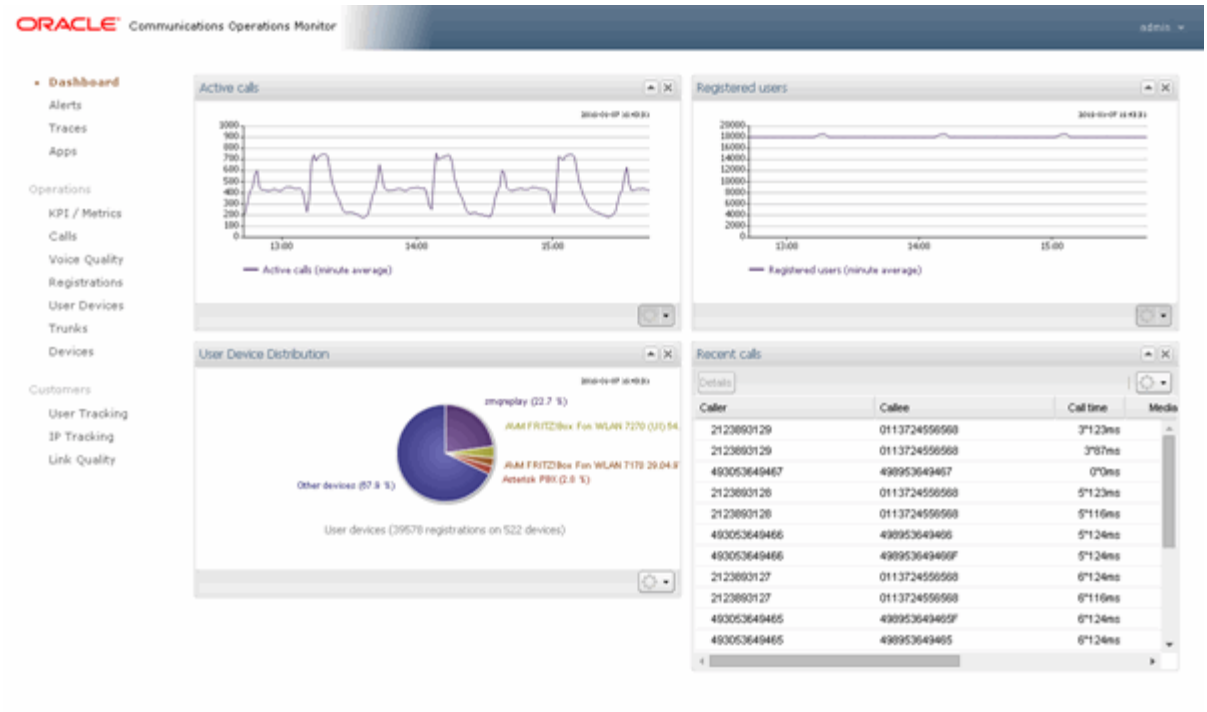
Login to the preconfigured account with username **admin** and password **oracle**. If the **Remember me** check box is marked, Operations Monitor will remember your login for a week on your computer.

During the first login, you will be prompted to choose a unique password for your account.

After login, you should see a web page similar to the one in Figure 2–1. The menu on the left provides general Operations Monitor functions and individual product features. The main features are described in "User Interface".

In the top-right corner, the current logged-in user is displayed.

Figure 2–1 Operations Monitor Web Interface



Changing the Default Administrator Password

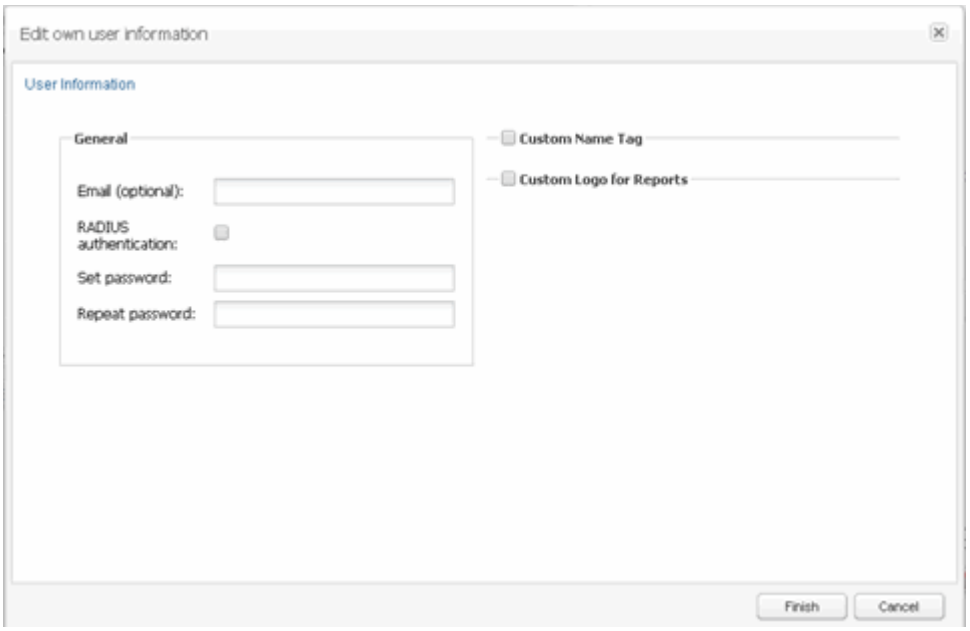
To change the default administrator password:

1. In the top-right corner of the Operations Monitor web interface, select **admin** and then select **My Profile**.

   The Edit own user information dialog box opens, as shown in Figure 2–2.

2. In the **Set password** field, enter a new password.
3. In the **Repeat password** field, re-enter the password used in the **Set password** field, which verifies that the password value was entered correctly.
4. Click **Finish**.

Figure 2–2 User information Page



Important: The default administrator password, which is provided in the documentation is not secure. Oracle strongly recommends that you change it to a secure password, especially before creating sub-users.

Adjusting System Settings

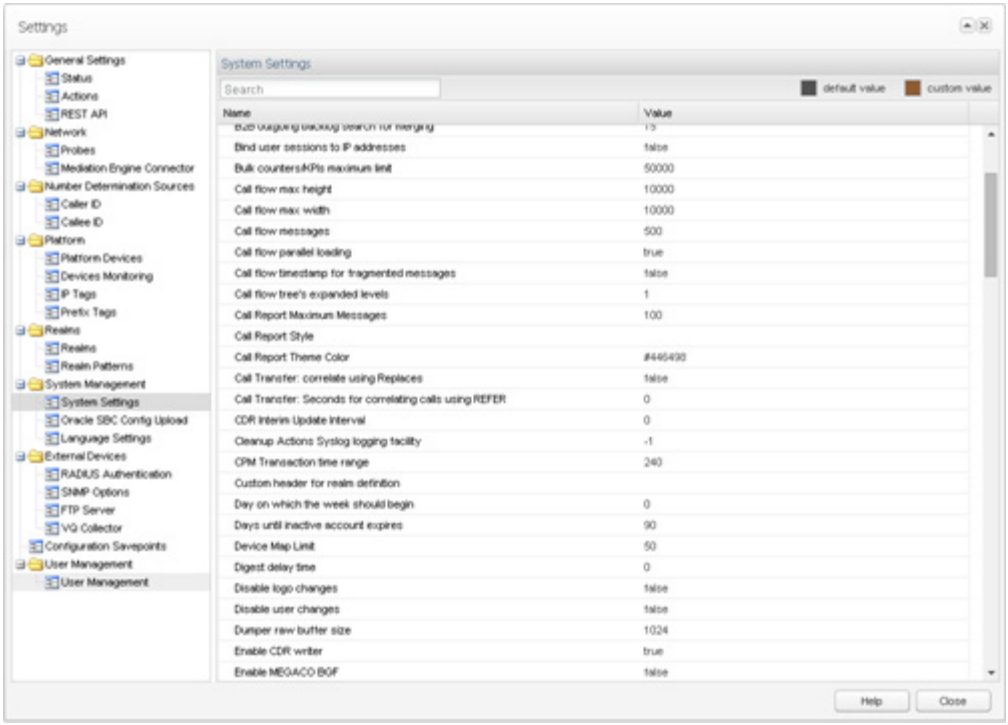
Operations Monitor provides the ability to filter network traffic for relevant IP addresses and ports. By default, Operations Monitor filters SIP messages that use ports 5060 to 6000. If you have SIP messages that use other ports, or other protocols you want to analyze, you can change that in the *Signaling Protocols* section in the Platform Setup Application.

The **Use User Domains** setting allows you to customize how a subscriber is identified. By default, Operations Monitor identifies a subscriber by the user portion of SIP URI. If domain names are significant in your network for identifying the users, you should set this option to **Enabled**. For more information, see "Use User Domains".

These options can be found in the **System Settings** menu entry from the **Settings** window. For more information, see "System Settings".

Figure 2–3 shows the Operations Monitor Settings page.

Figure 2–3 System Settings

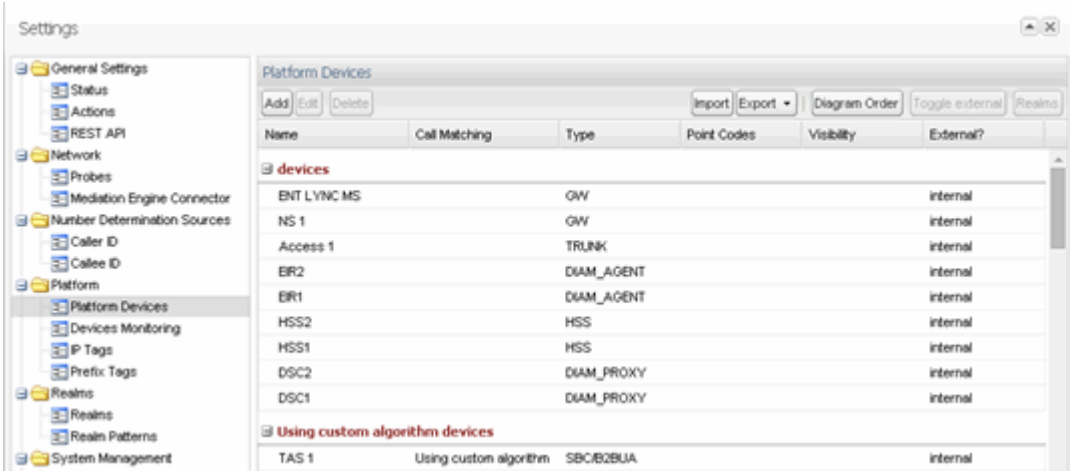


Adjusting Platform Settings and Defining the Network Topology

Operations Monitor can trace calls end-to-end while they traverse your network. This is also true for when SBC and B2BUA devices are in use. Operations Monitor shows comprehensive per-device statistics, including response and transit times. Operations Monitor can also monitor your SIP devices and raise alerts when a device is detected as non-responsive.

Figure 2–4 shows the Platform Devices Settings page.

Figure 2–4 Platform Devices Settings



In order to correlate sessions end-to-end correctly, you need to define your network topology by configuring the devices (network equipment) that process and relay the

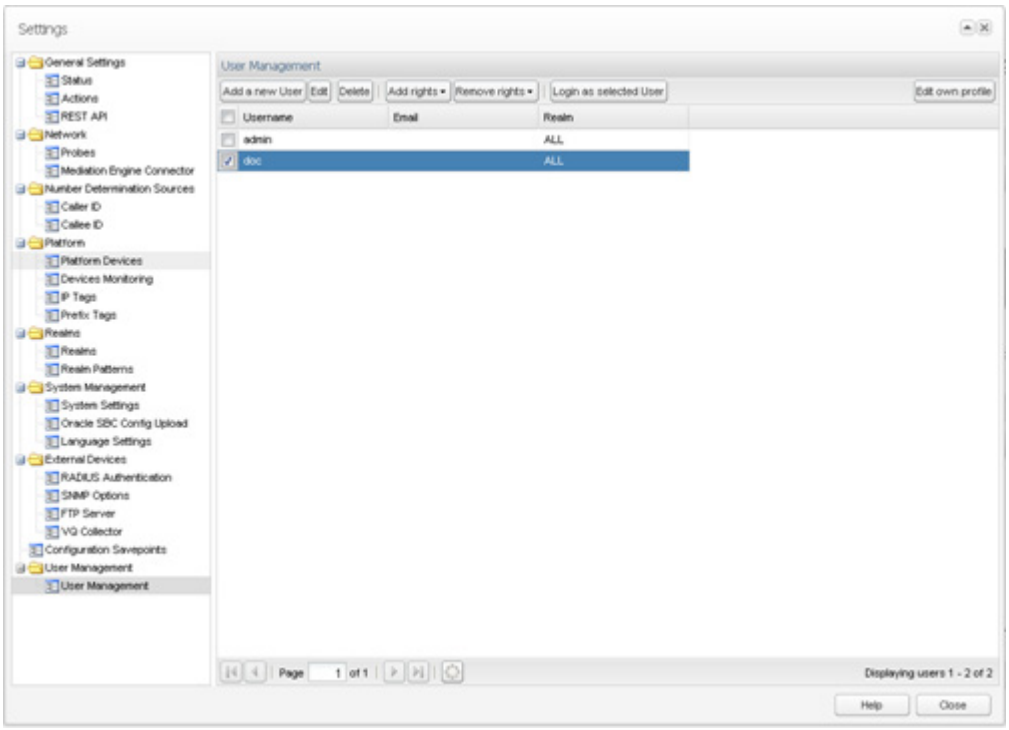
sessions and registrations in your network. For instructions on how to do this, see "Platform Devices".

Defining Web Interface Users

Operations Monitor offers a comprehensive system to manage user permissions that allow you to restrict certain views and functions for users.

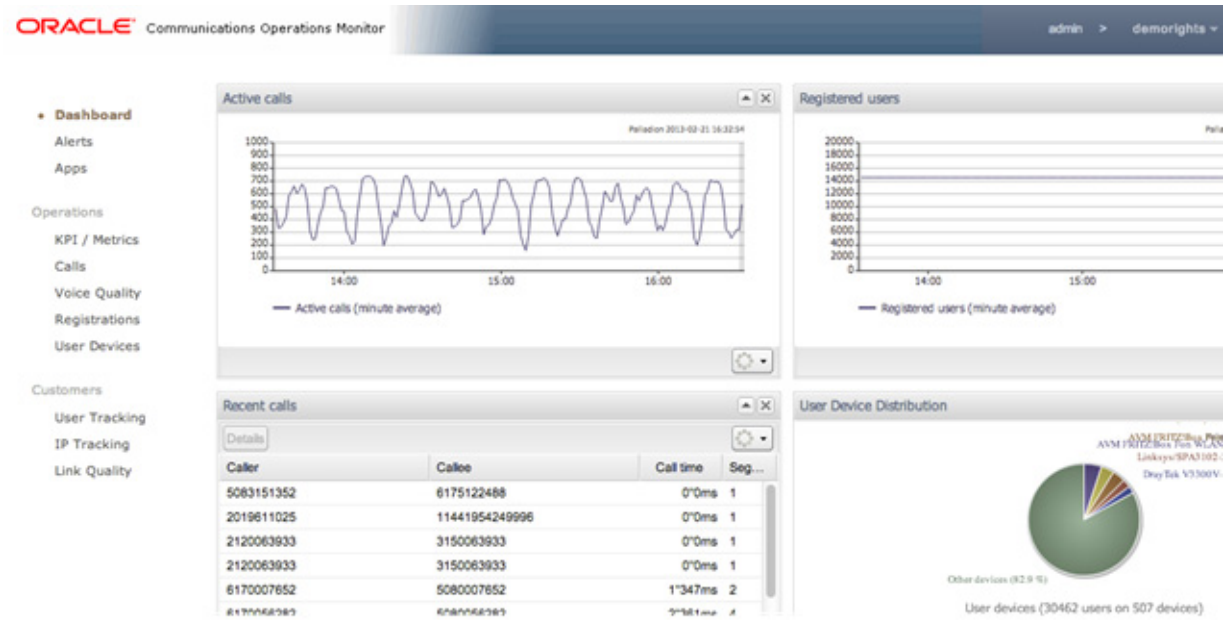
Figure 2–5 shows the User Management page.

Figure 2–5 User Management Page



You can customize restrictions per user for certain pages, tabs, and product features. For example, you can restrict the ability to view raw SIP messages, change settings, and the option to create PDF reports (Figure 2–6 shows an example). For details about how to define users and assign rights, please refer to the "User Management" section of this manual. For more information on restrictions extended to single phone numbers/subscribers, see "Realms Definitions".

Figure 2–6 Restricted interface for a Sub-User



3

User Interface

This chapter describes how to work with the Oracle Communications Operations Monitor web user interface.

General

This section provides an introduction to the web interface, requirements, and frequently used interface elements of Operations Monitor.

Requirements

Operations Monitor runs in any major web browser without extra plug-ins or add-ons.

> **Note:** Operations Monitor does *not* require Flash, Java or Microsoft Silverlight.

The following list contains the web browsers in which we regularly test the web interface:

- **Internet Explorer**: version 8 or higher.
- **Firefox**: 1.5 or higher running on any operating system (Windows, Mac, Oracle Linux).
- **Safari**: any version, including **Safari** for the iPad.
- **Google Chrome**: any version.
- **Opera**: 9 or higher running on any operating system (Windows, Mac, Oracle Linux).

In order to use the web interface of Operations Monitor, you must enable Javascript. It is recommended that you enable HTTP cookies for features such as remembering login credentials and the state of tables.

Interface Arrangement

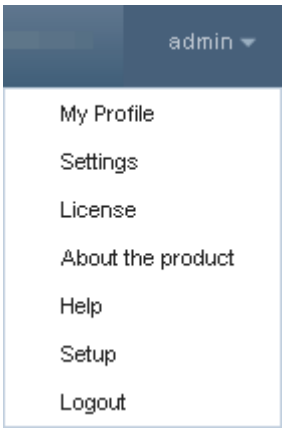
On the left side of the Operations Monitor web user interface is the navigation pane. This navigation bar allows access to many of the features of Operations Monitor.

The right side of the Operations Monitor menu bar displays your username, which contains a list with following options (see Figure 3–1):

- *My Profile* for user settings.
- *Settings* for administrative features.

- *License* for the Oracle Communications Session Monitor license terms.
- *About the product* for the Operations Monitor copyright information.
- *Help* for access to the Operations Monitor online manual.
- *Setup* to access the Session Monitor Platform Setup Application.
- *Logout* to sign out.

Figure 3–1 User Menu



Language Selection

The right side of the Operations Monitor menu bar contains the selected language code, as shown in Figure 3–2. From the locale list, select the language to display in Operations Monitor.

Figure 3–2 Locale Selection

Panels

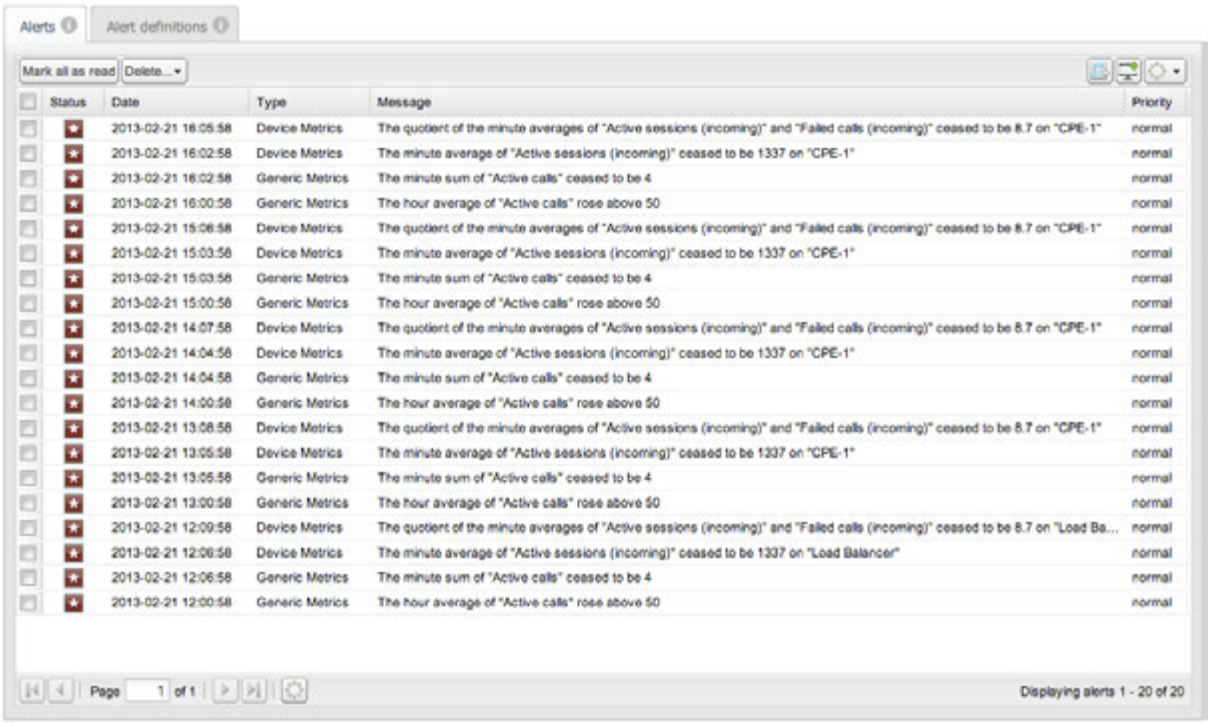
When you click an element in the navigation pane, a panel appears in the center of the browser window. This panel is a user interface element that displays data.

Many panels can be added to your dashboard for at-a-glance viewing. For more information, see "Dashboard".

To collapse panels and display titles only, you can click the up-arrow in the upper-right hand corner.

Another common user interface element is the tab panel. An instance of this element is depicted in Figure 3–3.

Figure 3–3 Panel with Two Tabs



Tooltips and Help

In the title bar of each panel is a gray *i*-symbol. This is the information box. When you hover over the information box, a question mark symbol appears and a tooltip provides a brief description of the panel. When you click the information box, the corresponding section of the user manual appears in a new browser window or tab describing the panel and its functions.

Other user interface elements support *tooltips*. Hovering over an element with the mouse pointer often brings up a small box explaining the element underneath.

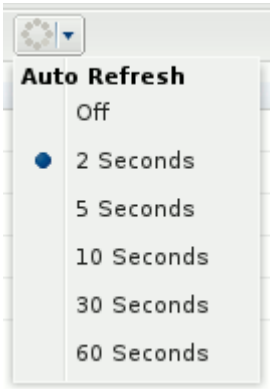
Windows

You may encounter modal windows overlaid on the user interface, similar to the windows of your desktop environment. These windows can be dragged by their title bar, and closed by clicking on the *x*-symbol or pressing the ESC key. These windows also contain **Help** and **Close** buttons at the bottom. The **Help** button opens the online Help in a new window, describing the current page and its functionality.

Refresh Button

Some areas of the interface contain a **Refresh** button that controls the update interval for the content of the page, panel, or section. It is identified by a circle of dots (flashing during refresh). Click the button once to refresh the corresponding content. It also contains a drop-down list to set the interval for the **Auto Refresh** as shown in Figure 3–4.

Figure 3–4 Automatic Refresh Menu



Tables

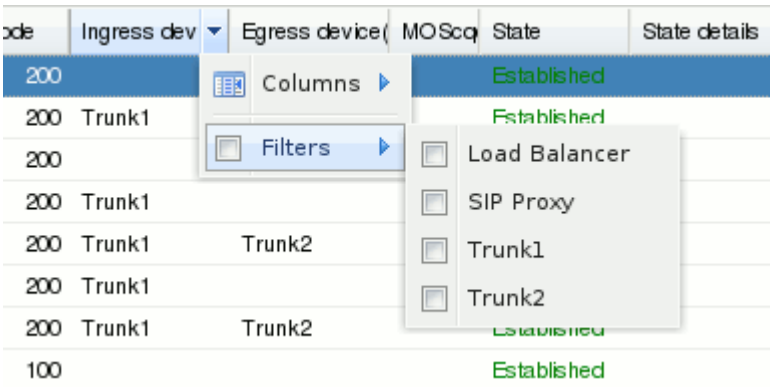
The tables used in the Operations Monitor user interface offer some features to make the data most useful to you.

To adjust tables, hover above a *column header*. A down-arrow appears on the right side of the column header. To access the *table menu*, click the *down arrow*. The table menu options may differ depending on the data presented in the table.

Filters

Some tables support filters. If a table column can be filtered, the column menu contains a *Filter* submenu as illustrated in Figure 3–5.

Figure 3–5 Filters Sub Menu



Below are possible options in the filters sub menu (depending on the column type):

- **String Filter**

  Filters the table to display rows that contain the entered value as a sub-string. The input value is not exclusive. This filter menu provides a text field for entering the string.

- **Numeric Comparison Filter**

  Restricts the table to rows where the comparison of the column value with the entered numeric results is 'true'. This filter menu provides three fields, one for each of the supported operators (>, <, =). The = field cannot be used simultaneously with any of the other fields.

- **Date Comparison Filter**

  Restricts the table to rows where the column value in the selected date range. This filter menu provides three items for choosing a date (*Before*, *After*, *On*). With *Before* and *After* you can select the rows where the timestamp of the column values lies before and/or after the chosen day. With *On* you can select the rows of only the chosen day.

- **List Filter**

  Restricts the table to rows where the column values contains the selected value. This filter menu provides a list of possible values, which you can choose by selecting or deselecting the corresponding check box.

You can choose to hide certain columns in each table. Every column drop-down menu contains a *Columns* submenu that lists all available columns for this table. Select or deselect the check box to the left of each entry to show or hide the corresponding column.

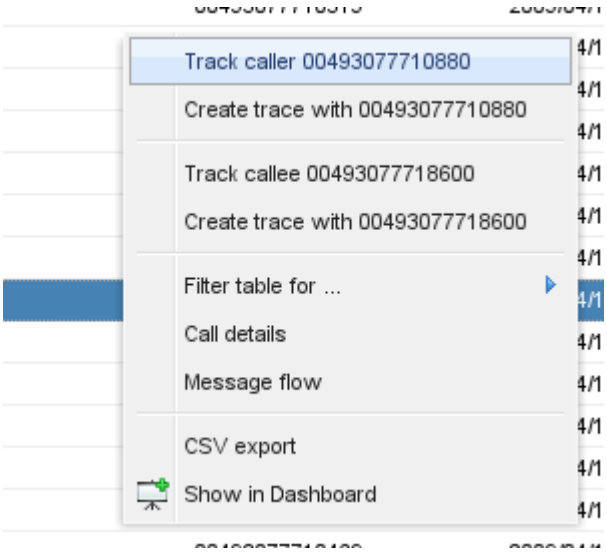
Note the paging toolbar at the bottom of tables as illustrated in Figure 3–6. The presence of this toolbar means that the table offers page-wise access to its data. The paging toolbar displays the total number of pages, and to the right, the total number of items in the table.

Figure 3–6 Paging Toolbar

Quick User Inspection

Tables such as **Recent calls**, **Registrations**, and the user table in **User Devices** have context menu entries to quickly trace or track a user (see Figure 3–7). You can right-click on a table row to track or trace the user (in the **Recent calls** table, you can quickly inspect both the caller and the callee). For more information, see "Recent Calls", "Registrations" and "User Devices".

Figure 3–7 Right-Click Menu



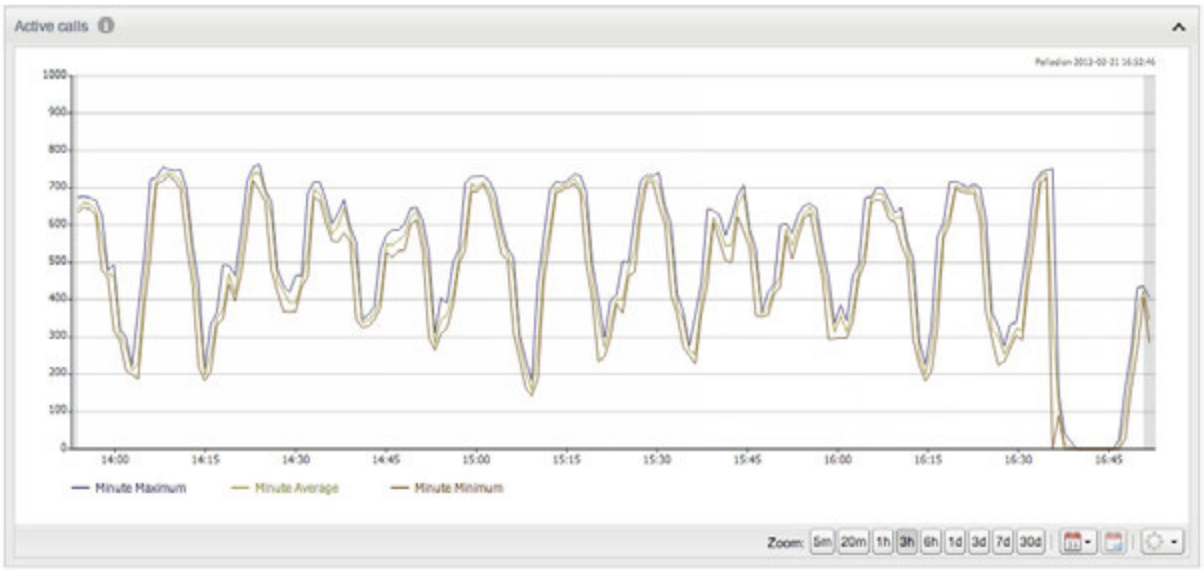
Charts

You may encounter metric charts that display values of a numeric property over a certain time span. Metric charts provide two important features:

- Adjusting the temporal resolution.
- Panning of the visible portion along the time line.

A basic metric chart is shown in Figure 3–8:

Figure 3–8 Basic Metric Chart



Adjusting the temporal resolution can be achieved using the **Zoom** buttons in the bottom-right corner of the chart. The buttons correspond to resolutions of five minutes to thirty days.

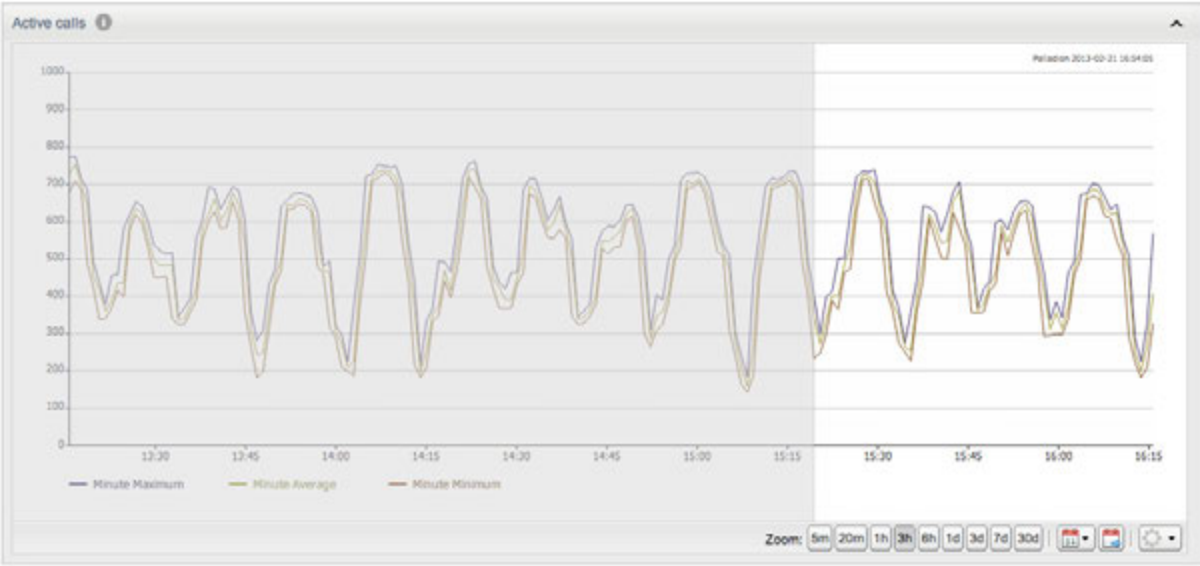
Operations Monitor's time-lined charts can be updated by clicking the **Refresh** button. This button also offers **Automatic Refresh** functionality from the button's drop-down list. For more information, see "Refresh Button".

Panning

To pan the chart, click and drag the mouse pointer inside the chart itself. When you click the chart and drag it to the right, the visible area will be shifted further left (further in the past) on the time line. When you drag it to the left, the visible area is shifted further right (nearer to the present) on the time line. While dragging, a gray overlay indicates the direction of the chart, and how far on the spectrum you are dragging. Panning is shown in Figure 3–9.

Note: Panning stops the automatic refresh of a chart. Automatic refresh is re-enabled after clicking on the **Go to now** button at the bottom of the chart.

Figure 3–9 Panning in Metric Charts



Working with Message Flows

A message flow is a diagram of a call event or a registration event. The message flow displays the network devices involved in the call event or registration event and the protocol messages that have been exchanged between the network devices. For more information on how to configure the hosts on which the network devices run, see "Platform Devices".

Displaying a Message Flow

To display a message flow of a call event or a registration event:

1. In a web browser, log in to Operations Monitor.

   The Operations Monitor window appears.

2. In the navigation pane under **Operations**, do one of the following:

   - To display a message flow for a call event:

     a. Click **Calls**.

        The **Calls** page opens in the main display.

     b. In the **Recent calls** table, right-click the row for which to display the call event's message flow and select **Message flow**.

        The **Message Flow for Call:** *caller and callee* window appears.

        where *caller* is the number that initiated the call and *callee* is the call number that received the call.

     Note: Message flows for a call event can also be displayed by right-clicking on the call row and selecting **Message flow** in the **Calls going through this device** tab on the **Devices** page.

   - To display a message flow for a registration event:

     a. Click **Registrations**.

The **Registrations** page opens in the main display.

b. In the **Registrations** table, select a registration event for which to display a registration message flow.

c. From the **Registrations** table tool bar, click the **Message flow** button.

The **Registration** *registration_number* window appears.

where *registration_number* is the number of the registration event.

3. Press the **ESC** key, which closes the message flow window.

Arranging Message Flows

In order to compare the network devices and protocol messages contained in multiple call events or registration events, you can open multiple message flows to view side-by-side in the same browser window, or you can open multiple message flows to view side-by-side in multiple browser windows.

To adjust the message flow window to fit in your browser screen, view a message flow in a new browser window, resize the message flow window, or view more of the message flow, do one or more of the following:

- To adjust the width of the sequence diagram within the message flow window. In the message flow window's toolbar, click the **Zoom in** or **Zoom out** icons accordingly.
- To resize the message flow window. In the message flow window's title bar, toggle between the **Maximize and Minimize** icon, or expand or contract the message flow window's borders.
- To view more of the message flow without resizing the message flow's window. Inside the message flow window, drag the message flow, or use the message flow window's scroll bar.
- To view the message flow in another browser window. In the message flow window's toolbar, click the **Open in a new window** button.
- To refresh the message flow window. In the message flow window's toolbar, click the **Refresh** button. For more information, see "Refresh Button".

Customizing the Display of Contents within Message Flows

The contents displayed in the sequence diagram, protocol messages, and network devices within the message flow are customizable.

To customize the display of contents within message flows:

1. Verify that the message flow is displayed.
2. In the message flow window's toolbar, select the **View** menu.
3. From the **View** menu, select or de-select one or more of the following check boxes:
   - **Pin Devices**

     When enabled, the devices are always visible at the top of the scrolling area.
   - **Unwind messaging**

     By default, each network device appears only once and a transaction arrow is drawn for each message that passes between the hosts. When several messages are exchanged between two particular network devices, it becomes difficult to distinguish the call legs. When enabled, each call leg is drawn

separately, which results in a wider diagram. Hosts that appear multiple times contain a number suffix, which is the number of times the host appears.

- **Resolve IP addresses**

  When enabled, Operations Monitor creates a reverse DNS lookup on the IP addresses of the network devices.

- **Duplicates**

  For retransmissions, only one SIP message appears. Enable this option to display all duplicate messages.

- **Transport protocol**

  When enabled, the message flow displays the transport protocol (TCP or UDP).

- **IP addresses**

  When enabled, the message flow displays the IP addresses for the source and destination of the message.

- **MAC addresses**

  When enabled, this option displays the MAC addresses for the source and destination of the message. In a scenario where one server is realized by several machines, this can be used to determine which machine was used to process the request.

- **OPC/DPC**

  When enabled, this option displays the originating point code (OPC) and the destination point code (DPC) for the source and destination of the message.

- **Codecs**

  When enabled, the message flow displays the media codecs of the caller and the callee. If the codecs are the same for the both the caller and the callee after the INVITE transaction is completed, the codecs are grouped.

- **Absolute Time**

  By default, the information below a transaction arrow displays the time that has passed after the first message was seen. When enabled, the date and time are displayed for each transaction arrow.

- **Media**

  When enabled, protocol messages for media transport, such as RTP, are displayed in addition to the signal.

Note: For some protocols, the raw message is not available.

- **DTMF Events**

  When enabled, protocol messages for RTP events of the type DTMF are visible. The DTMF event will show the name and code of the tone. For tones 0-9 the name and the code are the same.

  Example

```
DTMF Event '2' (2)
DTMF Event '#' (11)
```

Requires the **View DTMF tones** permission to be visible. See "User Permissions".

- **SIP URIs**

  When enabled, the transaction arrows display the recipient's URI.

- **Hide SIP Methods**

  This submenu allows you to hide certain types of SIP messages based on their method. For example, enabling **OPTIONS** will remove all SIP messages from the diagram that have the method OPTIONS.

- **MEGACO**

  This submenu allows you to hide selected MEGACO message properties.

4. Press the **ESC** key, which closes the message flow window.

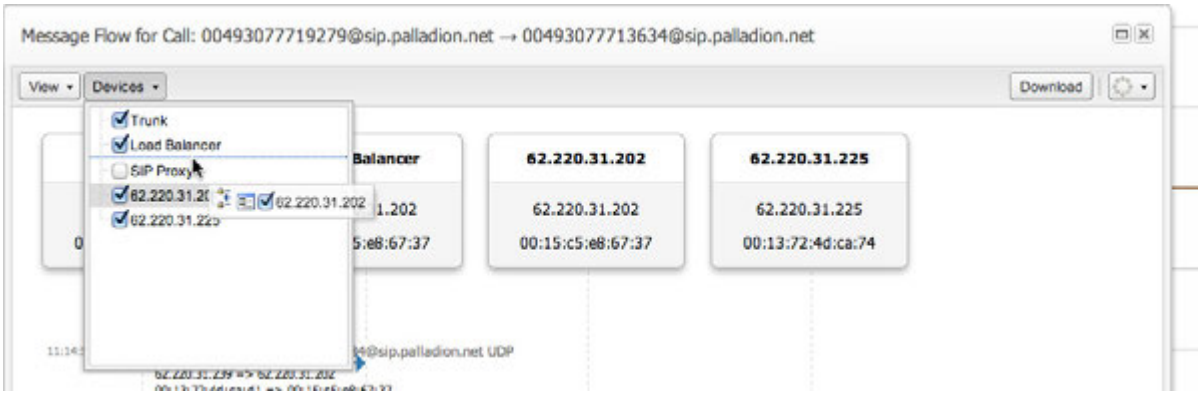
Customizing the Visibility and Position of Network Devices within Message Flows

By default, all networking devices are displayed in the message flow in the order of the call event transaction or the registration event transaction. The visibility and position of a network device is customizable in a message flow.

To customize the visibility and position of network devices within message flows:

1. Verify that the message flow is displayed.
2. In the message flow window's toolbar, select the **Devices** menu.
3. Do one or both of the following:
   - To hide a network device, deselect the check box of the network device you want to hide. Alternatively, select the check box of the network device you want to display.
   - To reposition a device in the sequence diagram, select the check box row of the network device you want repositioned and drag the check box row to the position you require. See Figure 3–10.
4. Press the **ESC** key, which closes the message flow window.

Figure 3–10 Devices Menu



Viewing Individual Protocol Messages

Protocol messages of the sniffed packet can be viewed and saved. For more information on saving a protocol message, see "Saving a Message Flow as an HTML File".

Each protocol message is numbered in brackets in the order of the call event transaction or the registration event transaction between the network devices.

To view individual protocol messages:

1. Verify that the message flow is displayed.
2. In the message flow, click the message type you wish to view, which appears above the message transaction arrow.

   The **Message** *transaction_number* window appears.

   where *transaction_number* is the number associated with the order of the transaction between the network devices.
3. (Optional) Resize, or move the protocol message by dragging the title bar of the protocol message's window to the required position within the message flow.
4. When you have finished viewing the protocol message, click the **Close** icon in the protocol message window's title bar.
5. Press the **ESC** key, which closes the message flow window.

> **Note:** The maximum number of messages for a flow is limited. You can change the limit value in **System Settings**. For more information, see "System Settings".

Viewing Call Event ISUP Protocol Messages

When a call event contains ISDN User Part (ISUP) binary content, you can view the ISUP binary content as human readable text.

To view call event ISUP protocol messages:

1. Verify that the message flow is displayed.
2. In the message flow, click the message type identified by **[1]**, which appears above the message transaction arrow.

   The **Message** *transaction_number* window appears.
3. Scroll down the protocol message until you see the **Content-Type** header. Do one of the following:
   - If the **Content-Type** header's value contains either, **multipart/mixed** and one of the multiple parts is **application/ISUP,** or **application/ISUP**, go to step 4.

     For example:

     ```
     Content-Type:application/ISUP;boundary=A6B35A2329D0F2312C9F4692
     ```

   - If the **Content-Type** header's value does not contain either, **multipart/mixed** and one of the multiple parts is **application/ISUP,** or **application/ISUP**, close the protocol message and click the next message.
4. Scroll down the protocol message until you see the **decode ISUP body** link.
5. Click the **decode ISUP body** link.

The ISUP binary content is decoded into human readable text and an **ISUP message contained in Message** *transaction_number* window appears.

6. When you have finished viewing the ISUP message, click the **Close** icon in the ISUP protocol message window's title bar.
7. When you have finished viewing the protocol messages, click the **Close** icon in the protocol message window's title bar.
8. Press the **ESC** key, which closes the message flow window.

Saving a Message Flow as an HTML File

You can save a message flow as an HTML file on your computer to view at another time or share with others. This is important when diagnosing issues.

To save a message flow as an HTML file:

1. Verify that the message flow is displayed.
2. In the message flow window's toolbar, click the **Download** button.

   The Save As dialog box appears.
3. Browse to the directory in which you want to save the HTML file.
4. (Optional) In the **File name** text box, enter a file name for the HTML file.

   By default, HTML files are named **message-flow.html**.
5. Click **Save**.
6. Press the **ESC** key, which closes the message flow window.

Saving a Call Event Message Flow as a PDF File

You can save a call event message flow as a PDF file.

To save a call event message flow as a PDF file:

1. In the navigation pane under **Operations**, click **Calls**.

   The **Calls** page opens in the main display.
2. In the **Recent calls** table, right-click the row for which to create a call event PDF report file and select **PDF report**.

   The **Create Report** window appears.

   By default, all the **Report Content** options are selected.
3. In the **Report Content** section, deselect the options you do not require in your report. Do not deselect the **Messages** option.
4. (Optional) In the **Additional Info** section's **Report Comment (optional)** text box, enter information about the report.
5. (Optional) In the **Additional Info** section's, **Filename (optional)** text box, enter a file name for the report.

   By default, PDF files are named **CallReport**-*YMDHMS-callid***.pdf**

   where

   - *YMDHMS* is the syntax for year, month, day, hour, minutes, and seconds that the file was generated.
   - *callid* is the call event's identifier.

6. Click the **Create** button.

   The Save As dialog box appears.

7. Browse to the directory in which you want to save the call event PDF file and click **Save**.

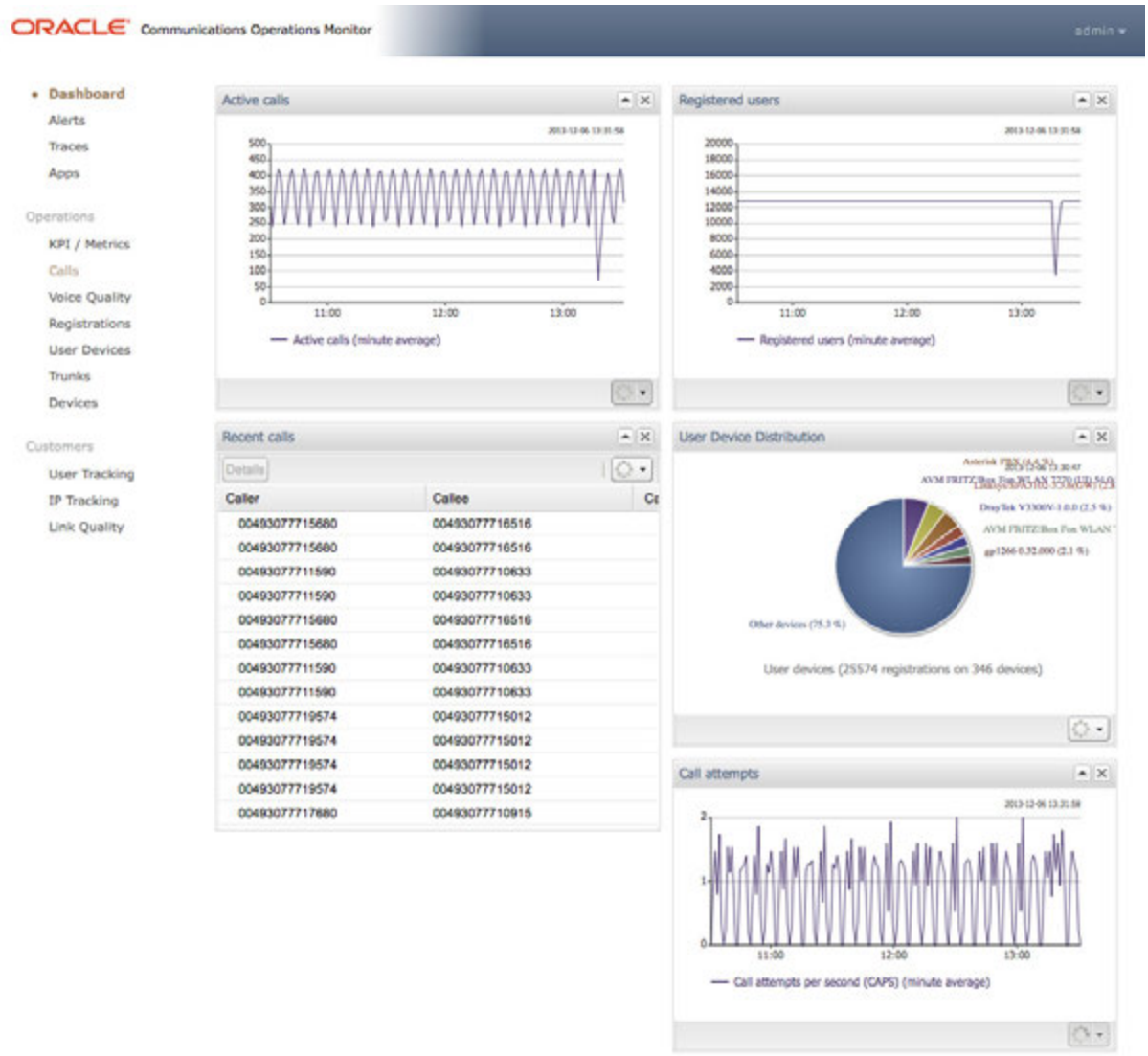
8. Click the **Cancel** button, which closes the **Create Report** window.

> **Note:** Because the decoded binary ISUP content can be lengthy, only two levels of information are displayed in the PDF report.

Dashboard

The **Dashboard** is Operations Monitor's flexible entry page (as illustrated in Figure 3–11). It allows you to view the information at a glance that is most important to you. Reduced panels are defined for many features of Operations Monitor, and can be displayed on the **Dashboard**.

Figure 3–11 Dashboard



Types of panels that may appear on your dashboard are:

- Counter chart (see "KPI/Metrics")
- Recent calls (see "Calls")
- Registrations (see "Registrations")
- User devices pie chart (see "User Devices")
- Voice Quality (VQ) overview chart (see "Voice Quality")
- Alerts (see "Alerts")

These panels come in variants. For example, you can display a **Recent calls** table only for call events passing through a certain platform device.

Configuring Your Personal Dashboard

Each Operations Monitor user can define their own **Dashboard** layout. The default **Dashboard** layout contains four panels:

- Registrations Counter
- Active Calls Counter
- Recent Calls table
- User Devices Chart

Panels can be added, removed, and rearranged.

Adding a Dashboard Panel

You can add Dashboard panels from the **Dashboard** page or from the Operations Monitor pages, which provide information that you would like to display on the **Dashboard**.

To add a panel directly from the **Dashboard** page, right-click on the **Dashboard** to get a context menu, and choose **Add a panel...**. A pop-up window appears that provides dashboard panel options.

To add a dashboard panel from another window in Operations Monitor, click the **Show in Dashboard** icon (shown in Figure 3–12). This button is located throughout the Operations Monitor web interface. For more information, please refer to the documentation sections referenced beside the dashboard panel types listed above.

Figure 3–12 Show in Dashboard Icon

Removing a Dashboard Panel

To remove a dashboard panel, click the **x** button in the upper-right corner of the dashboard panel. A confirmation box appears. Click **Yes** to continue or **N**o to cancel.

Rearranging Dashboard Panels

You can drag and drop dashboard panels to rearrange them. Drag a dashboard panel by its title bar to relocate it on the page.

Alerts

Operations Monitor has a notification system to warn you of important events called **alerts**. You can prioritize alerts and configure them to trigger certain actions. The **Alerts** page displays all alerts that have been raised. The following **alert types** are supported:

- **Generic Metrics**

  Statistics metrics exceed or fall below certain thresholds.

- **Device/Tag Metrics**

  Statistics metrics exceed or fall below certain thresholds.

- **Metric Baseline Derivation**

  Allows comparison between a current KPI value and a previously captured KPI value. An alert is generated based on the difference between the two values.

- **User Devices**

A certain user device is detected by the system.

- **Voice Quality**

  The quality for a percentage of calls falls beyond a MOS threshold.

- **Device Monitoring**

  A device is down.

- **Limits**

  An internal soft limit within Operations Monitor was passed.

- **Phone Number**

  A certain phone number placed or received a call.

The **Alerts** table displays notifications. You can create or edit an **alert definition** in the **Alert Definitions** tab.

Alerts Table

Click **Alerts** in the navigation pane to display the alerts panel, as shown in Figure 3–13.

Figure 3–13 Alerts Table

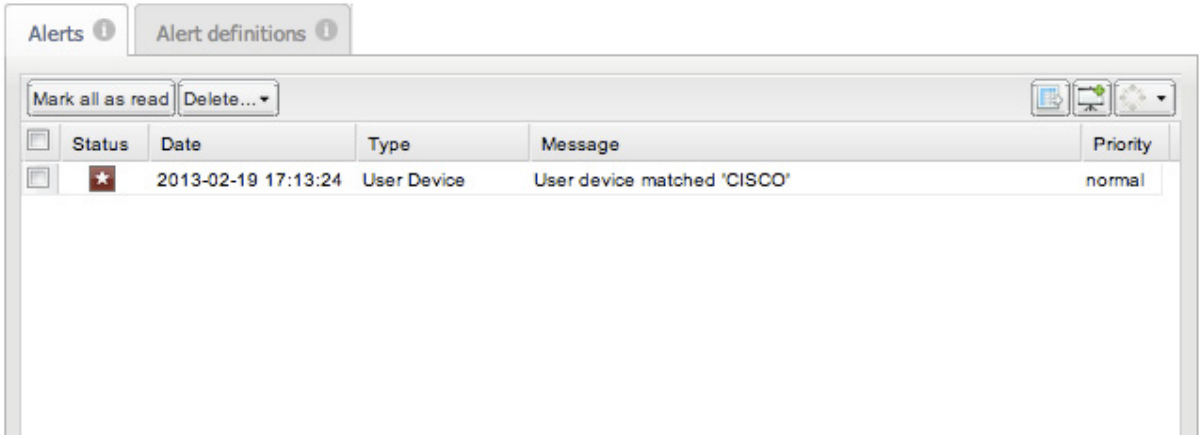


Table 3–1 lists the **Alerts** panel table columns:

Table 3–1 Alerts Panel Columns

Column	Description
Status	Unread alerts are marked with a red icon. Clicking on the icon toggles the status as read/unread.
Date	The date when the alert was raised.
Type	The type of the alert. This columns allows for easy filtering.
Message	The alert message explains why the alert was raised.
Priority	Setting priority ranks the alerts. The alert entries with high priority are highlighted by a red background. Low priority alerts are displayed in a lighter color.

When you double-click an alert, a small window appears with an extended message and an offer to mark the alert as read.

The **Mark all as read** button respects table filters, and can be helpful when treating similar alerts. For example, to mark all statistic threshold alerts, enable **Filters** in the **Type** column and click **Mark all as read**. The **Delete** drop-down menu allows you to delete either:

- Selected alerts (the check box in the first column is marked).
- Alerts marked as read.
- All alerts.

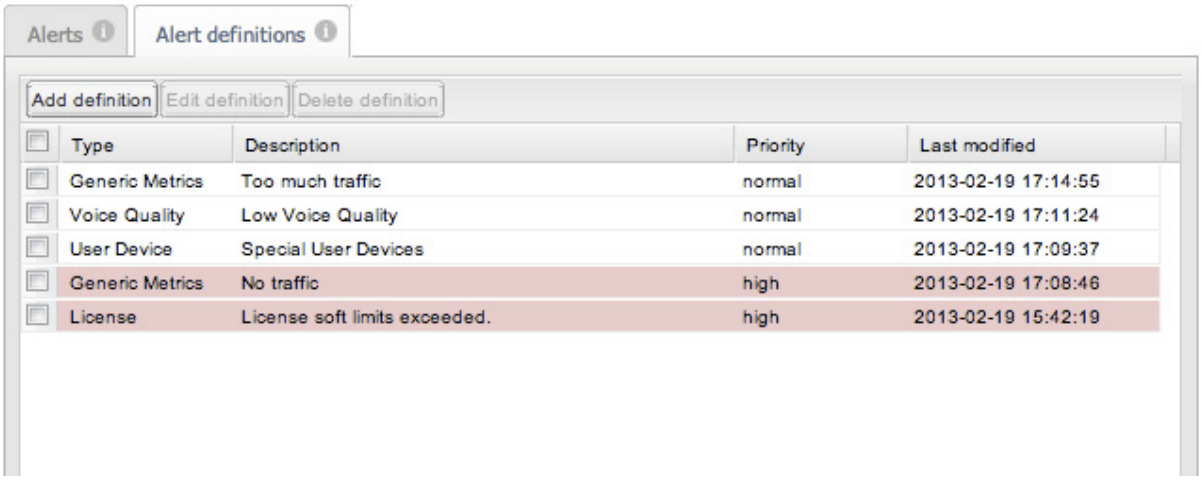
Upon initial installation of Operations Monitor, the only alerts raised are related to licenses. You must create *Alert definitions* to receive further notifications.

> **Note:** To avoid being inundated with alerts (for example *Number of active calls exceeded 5000*), Operations Monitor creates duplicate alerts only when the previous one has been deleted, marked as read, or when 1 day has passed (recurrence threshold adjustable via recurrent alerts threshold. For more information, see "Recurrent Alerts Threshold").

Alert Definitions Tab

Alert definitions establish which events on the system raise a notification. On the **Alerts** page, click the **Alert Definitions** tab to see a table of all definitions that have been created.

Figure 3–14 Alert Definitions Table



Creating Alert Definitions

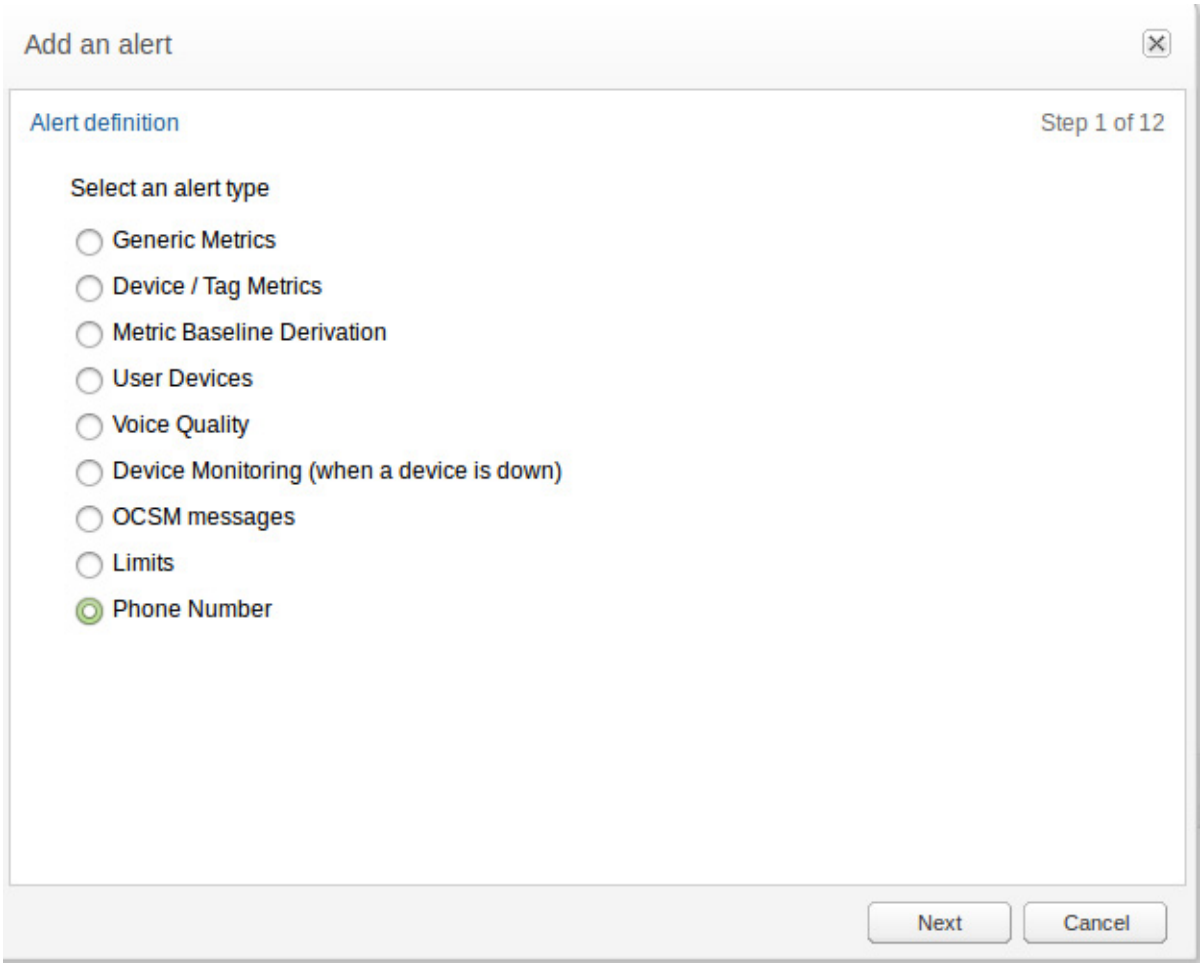
To create an alert definition:

1. Click **Add definition**, which starts the Alert Definition wizard.
2. In the **Add an alert** page, select an alert type (see Figure 3–15). This also determines if parameters are required.
3. Optionally, *parameters* have to be provided that tell Operations Monitor when to raise an alert and of that type. Some alert types do not require parameters.
4. If desired, set actions to execute when the alert is raised, for example, to send an e-mail. For more information see, "Actions".

5. Assign an alert name and priority for the notifications raised according to this definition. For more information, see "Alert Name and Priority".
6. Click **Finish** to save the alert definition.

   Operations Monitor raises an alert whenever the specified event occurs.

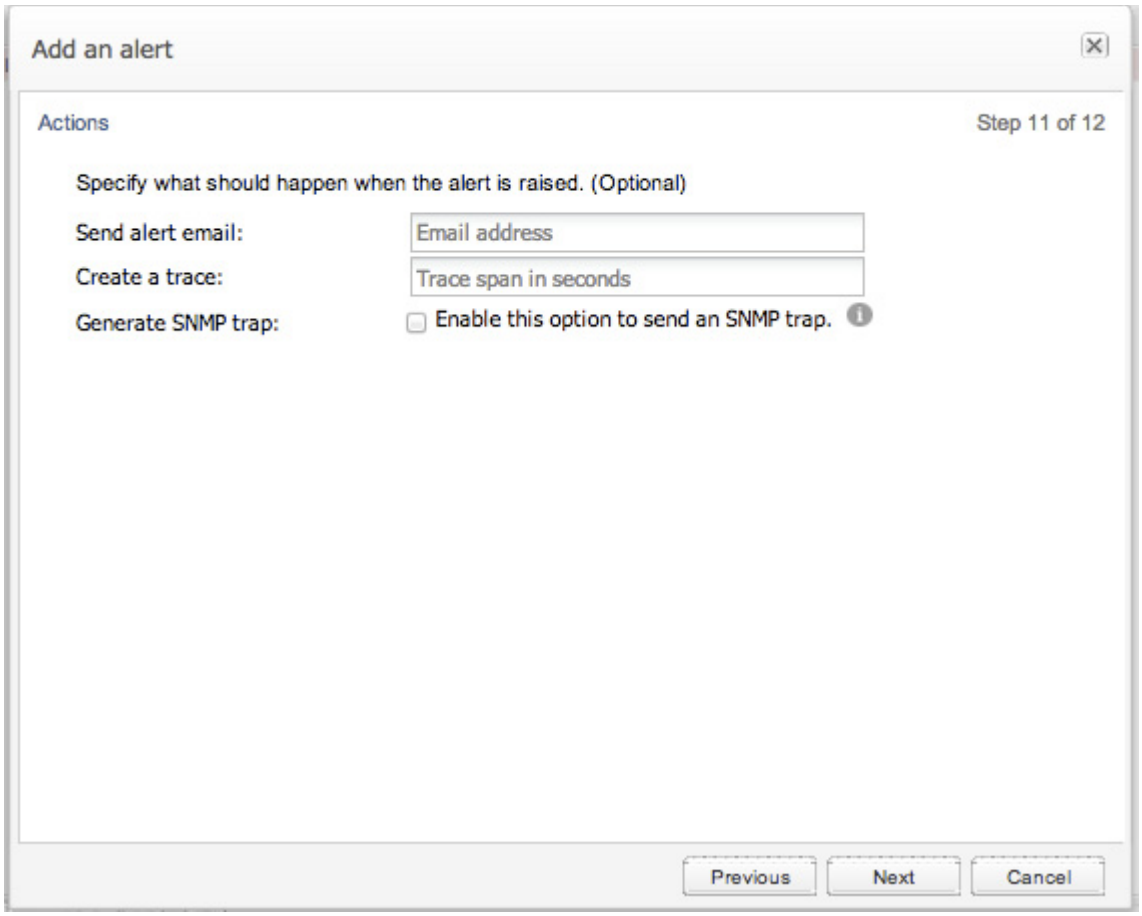
Figure 3–15 Alert Definition Wizard



Actions

For all alerts additional actions can be configured (see Figure 3–16). All actions are optional and can be combined.

Figure 3–16 Specifying Alert Actions



- **Send alert e-mail**

  Sends an e-mail about the alert to the specified address. An SMTP access needs to be configured before. Please check the PSA manual for more details about configuring SMTP.

  For most alert types, a deep link to the source of the alert can be provided in the alert e-mail. To receive this link, you must first configure the External IP/hostname. For more information, see "External IP/hostname."

- **Create a trace**

  Creates a trace of the current SIP traffic when the alert is raised. Enter an appropriate time span (in seconds) for packet capture to tell Operations Monitor how far back in time the trace should go.

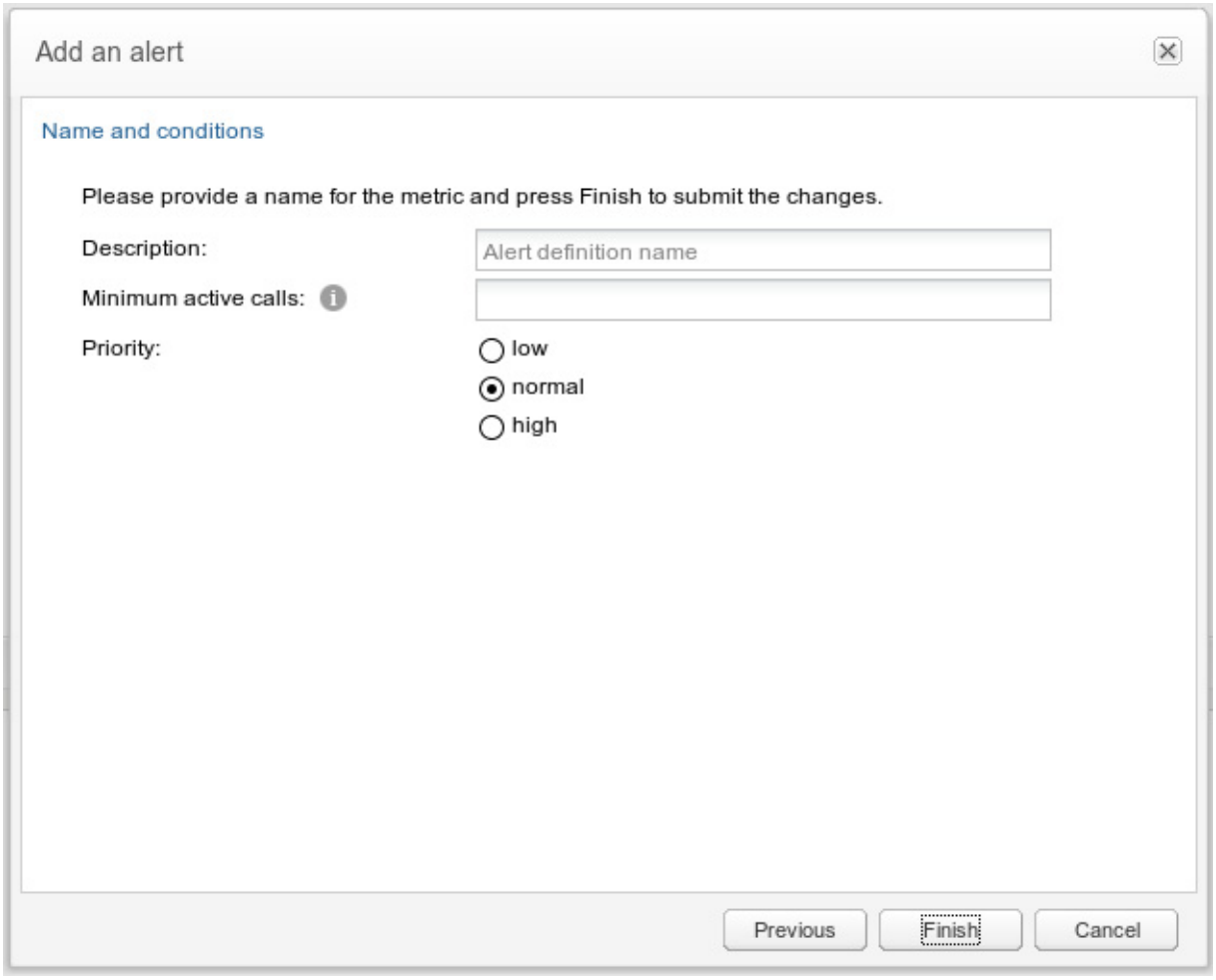
- **Generate SNMP trap**

  Sends an SNMP trap to a configured SNMP target. When this is enabled, Operations Monitor alerts can be tracked and analyzed by an SNMP manager in the network. For more information, see "SNMP Options".

Alert Name and Priority

For all alerts a name and priority needs to be provided (see Figure 3–17). It is also possible to define a minimum number of active calls to trigger the alert.

Figure 3–17 Adding an Alert Name



Editing and Deleting Alert Definitions

To edit a definition, perform one of the following actions:

- Select the entry and click Delete.
- Double click the entry.

To delete a definition, select the entry and click **Delete definition**. Once an alert has been deleted, Operations Monitor no longer raises a notification for this definition.

Parameters

The following sections describe the various parameters for each alert type.

Statistics Metrics

Alerts can be defined with either one or two metrics. The metrics are compared to a threshold value using a comparison operator (for example **=**, **<**, **>** or **not**).

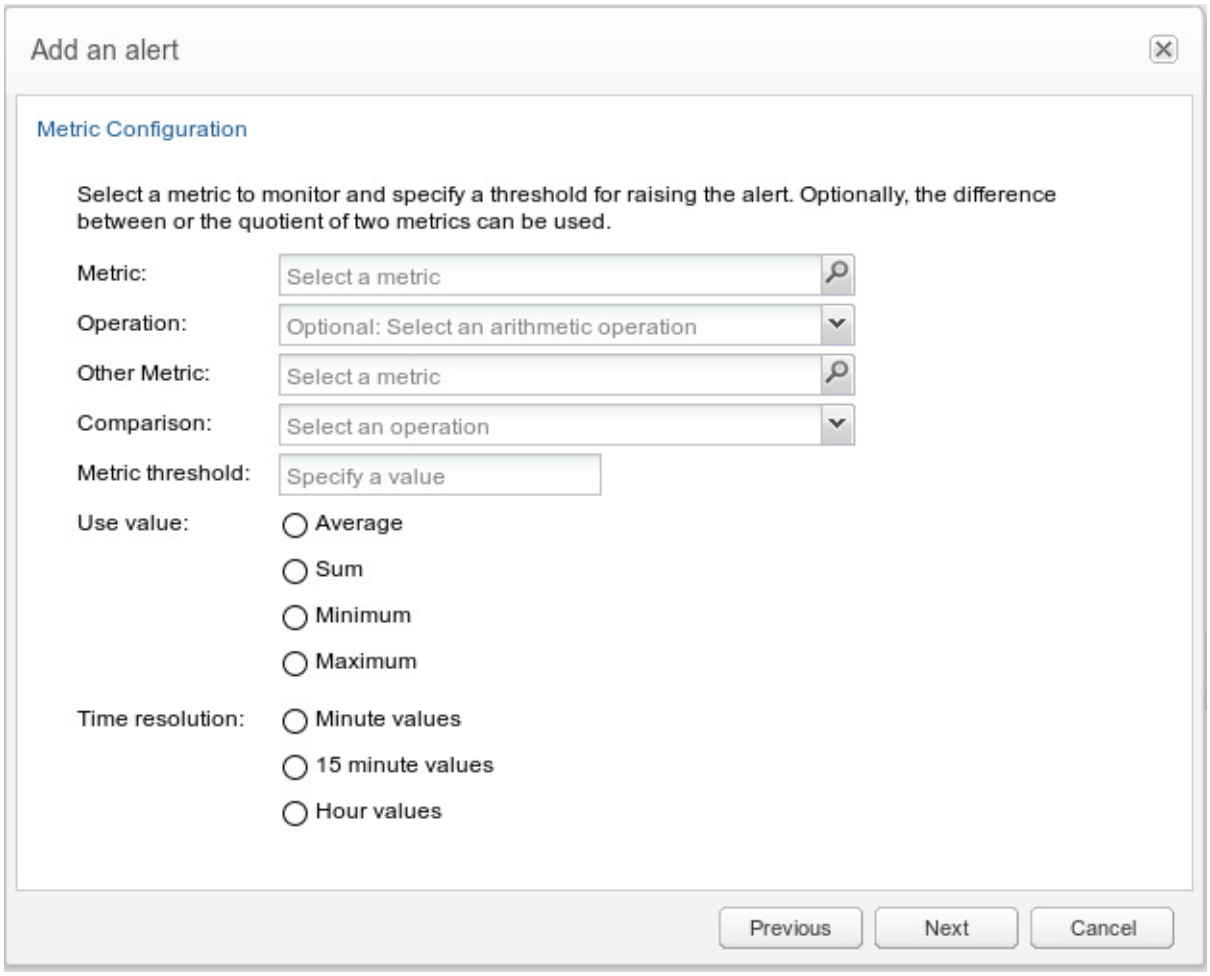
When you input one metric, the alert is triggered if the metric exceeds or falls below the given threshold value, depending on the comparison operator provided. This is suited for alerts such as 'The number of registered users exceeds 20,000.' In this example, an alert is sent when the number of registered users exceeds 20,000. If you

would like a second alert to notify you if the value falls below 20,000, you must create a second definition.

When you provide two metrics, Operations Monitor considers the difference of the two metric values to a given threshold value, and therefore allows for more sophisticated alerting scenarios. For example, a statistics metric value deviates too much from its average value.

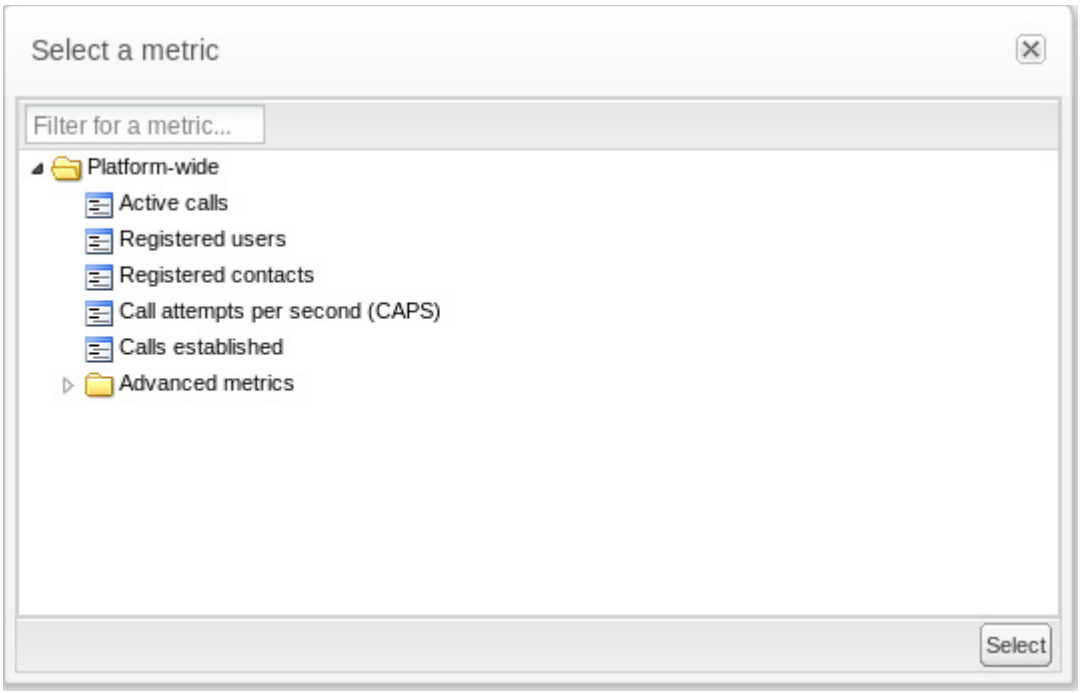
The example in Figure 3–18 shows the metric configuration of the alert scenario 'The number of registered users fell by 1,000 in the last week'. For using this alert definition, you must set the regular value for the metric as well as its average metric. For more information, see "KPI/Metrics".

Figure 3–18 Defining Alerts Based on Metrics



Clicking on the **Choose** button for a metric displays the metrics dialog box (see Figure 3–19), where you can choose a metric:

Figure 3–19 Metrics Dialog



Device Metrics for All Trunks

It is possible to define an alert that triggers when the value of a metric exceeds or falls below a given threshold on any trunk. You must select the alert type **Device Metrics for All Trunks**. The metric in question needs to be added to all trunks first. For more information, see "Adding Metrics to All Trunks".

If the metric exceeds or falls below the threshold on multiple trunks, the alert is triggered for each trunk and will thus have multiple independent entries on the alerts table.

User Agent

Operations Monitor keeps track of all user devices that are registered on the platform and raises an alert depending on the desired option:

Highlighted Devices

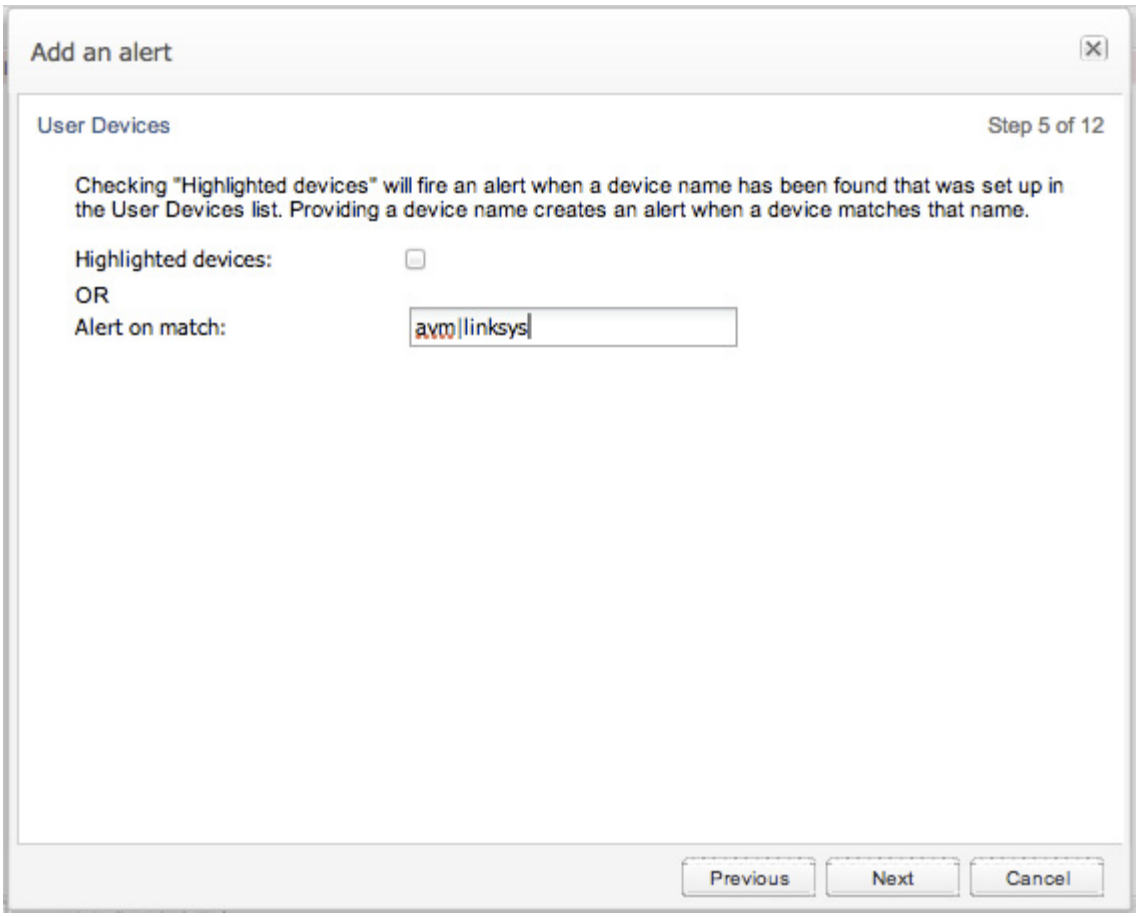
If a list of disapproved devices has been configured in User Devices, and Operations Monitor notices a user device from that list, an alert is sent. This option is recommended if User Devices is frequently updated so that the list can be centrally managed there. For more information, see "User Devices".

Alert on match

This option is intended for a more ad-hoc tracking of user devices. A regular expression can be entered to raise an alert when a matching user device is found. For every expression to track you need to add another alert definition.

Figure 3–20 shows an example configuration for detecting any AVM or Linksys devices.

Figure 3–20 Defining Alerts Based on Devices



Device Monitoring

This alert type refers to Device Monitoring and does not need additional parameters. Whenever a monitored device is down, an alert is raised. Only one alert definition of this type is needed. For more information, see "Device Monitoring".

Phone Number Alerting

The Phone Number type provides the possibility to alert on a specific phone number. The alert will be triggered when a call to or from a configured number is detected. To use this alert type the user needs to have the **Number Alerting** permission (see Figure 3–21). Without this permission the **Phone Number** option will not be shown.

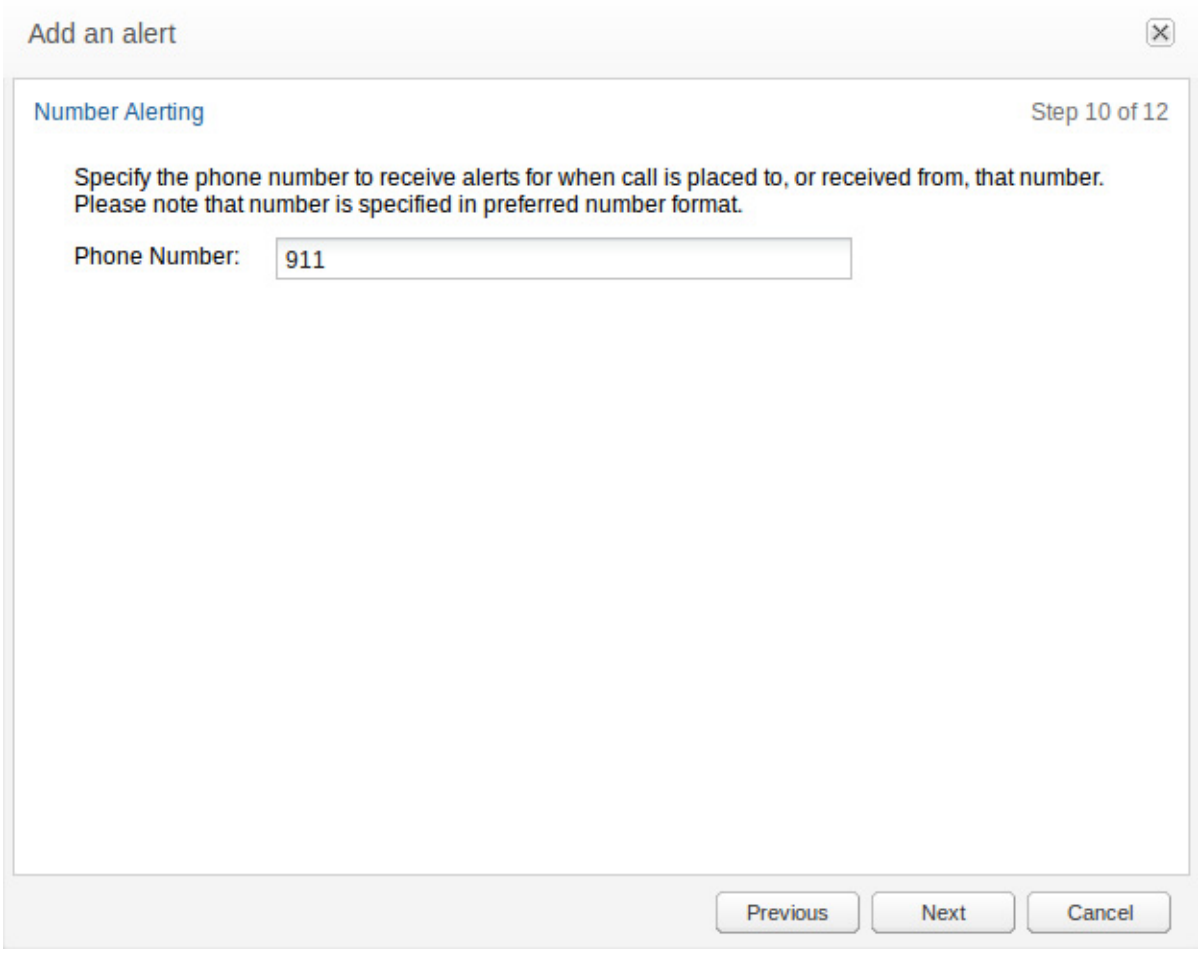
Phone numbers are configured in preferred number format. If the Use User Domains system setting is set to **true** then a phone number should be provided as a SIP user, that is, with a domain name: for example, 00403077811-call@example.com. If not, the phone number should be provided without domain (and without the @ sign). For more information, see "Use User Domains".

Note: Only one number per alert can be defined. No wild cards are allowed.

A good strategy for setting up a number alert is to place a phone call to the desired number and look in the calls grid for the callee to get the exact SIP address/phone number to use for the alert definition.

The search for predefined phone numbers happens two times per minute, with calls that are at least 10 seconds old (in the early stages of a call, not all information needed is available). This means that a call in best case is detected after 10 seconds, in worst case after 40 seconds.

Figure 3–21 Configuring the Phone Number for Alerts



Traces

The **Traces** page allows you to capture, store and download packet traces of the SIP traffic. Downloads are done in the industry-standard PCAP format, which contains the complete messages as monitored on the network.

The PCAP file format can be interpreted by most of the available protocol analyzers, including open source tools. Any stored message contains TCP/UDP headers, an IP header, and layer 2 headers, as well as the time stamp at which Operations Monitor received it. All messages are written to the disk before any processing is performed. The filter defining, which traffic the traces, can be set in the **Signaling Protocols** section in the Platform Setup Application.

Packets are captured from the moment a trace is started. With Operations Monitor, you can also trace events in the *recent past*. For more information, see "Packet Buffer".

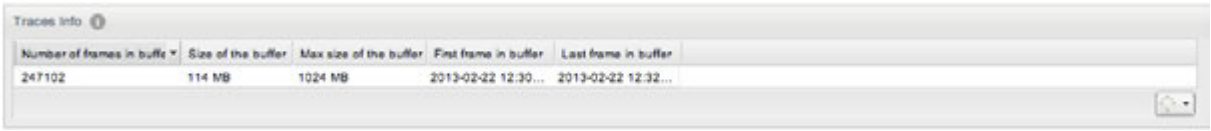
You can restrict traces to a single SIP user or an IP.

> **Note:** If you restrict the traces to a SIP user, only UDP packets will be captured.

Packet Buffer

In order to create traces with packets received prior to the current time, Operations Monitor keeps an in-memory, rotating **raw packet buffer** with all the messages captured. The default size of this buffer is 1 GB including the message overhead. The buffer size system option allows you to change the size of this buffer.

Figure 3–22 Raw Packet Buffer Details



The **Traces** page contains three panels: the first shows information about the raw packet buffer, the second panel can be used for creating a new trace, and the third panel shows the currently running and finished traces.

Traces Info Panel

The **Traces Info** panel displays details about the state of the raw packet buffer:

- **Number of frames in buffer**

  The total number of frames that are currently in the in-memory buffer.

- **Size of the buffer**

  The total size in bytes of the raw packet buffer, including the overhead needed for indexing the messages.

- **Max size of the buffer**

  The maximum size in bytes to which the buffer can grow. When this limit is reached, old packets are discarded.

- **First frame in buffer**

  The timestamp of the oldest packet still in the buffer. Traces cannot capture messages before this point in time.

- **Last frame in buffer**

  The timestamp of the newest packet in the buffer. This information is provided in order to approximate the capacity of the buffer in seconds.

Signaling Traffic Capture Panel

The **Signaling Traffic Capture** panel allows you to capture signaling traffic and save the data to a **PCAP** file. If you click **Start Capture** immediately, a trace begins with default settings. This provides a trace for activity in the last five minutes including non-call related messages such as OPTIONS and PUBLISH.

This panel allows you to configure time ranges, and filter SIP traffic by user or IP.

You can specify the **Start time**, **End time**, and the **Filters** used for a new trace in the **Signaling Traffic Capture** panel. Start and end times can either be set relative to the current time or by using absolute times. If the start time lies *before* the current time, the trace contains packets from the raw buffer. If the end time lies *after* the current time, the trace is started and write the packets received until the selected end time or until the user stops the trace.

Note: Traces are limited to durations up to 24 hours. Set traces to a duration shorter than 24 hours. After 24 hours of capturing, no further packets are traced.

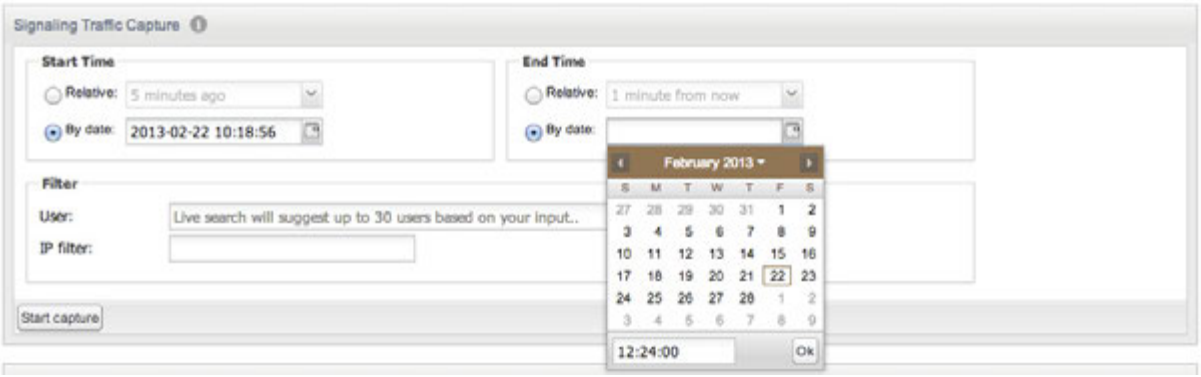
A *Filter* restricts captured messages to a single user, a single Operations Monitor network port, an IP address, or a combination of these. The **User** field has a live auto-complete feature: while typing the first characters of a user name or number, Operations Monitor suggests user names and numbers containing those characters from the set of known users.

The number of traces running simultaneously is limited to 3 by default and are configurable depending on the license. If the limit is exceeded, an error message is displayed.

Examples

Figure 3–23 shows a new trace created for a past transaction using absolute times (from 10:18:56 to 12:24:00). No filter is selected: all messages received by Operations Monitor in the given time interval are included in the trace.

Figure 3–23 Trace Example 1



In Figure 3–24, a new trace is created for all messages from user '00493077718485' from the last 10 minutes. Here, we use a relative start and end time.

Figure 3–24 Trace Example 2

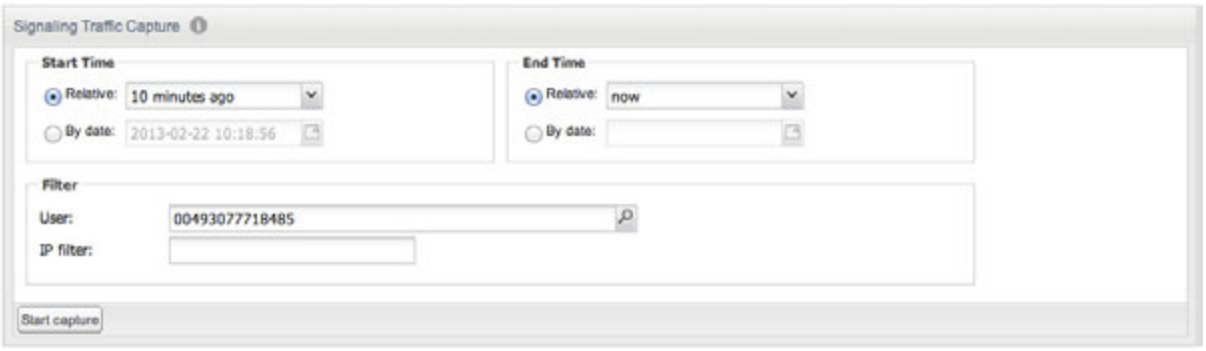
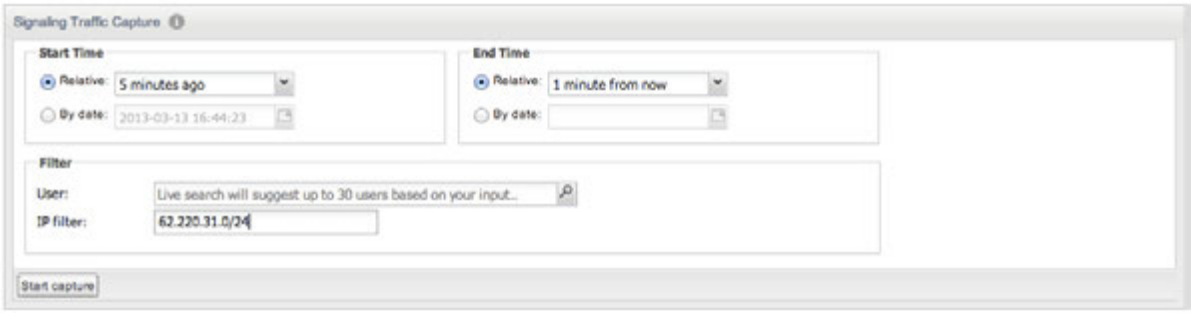


Figure 3–25 illustrates a trace created for all messages, which have either the source or destination IP address in the 62.220.32.0/24 network range. The IP filter field accepts both a host IP address in dotted format (*a.b.c.d*) or a sub-network address with the net mask appended(*a.b.c.d/e*).

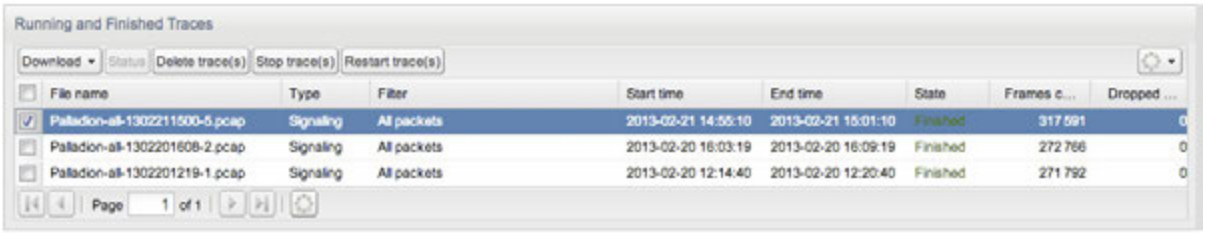
Figure 3–25 Trace Example 3



Running and Finished Traces Panel

The last panel on the page lists all traces that are stored on the disk and offered for download. Traces that are currently in progress are also listed and may be stopped before their scheduled end time. Figure 3–26 illustrates the panel.

Figure 3–26 Traces List



The following information is presented for every trace:

- **File name**

  The name of the PCAP file on disk. This will be also used as the proposed filename when downloading the trace.

- **Filter**

Short description of the filter used for the trace.

- **Start time**

  The date and time the packet trace began. This can lie before the time at which the user has requested the trace.

- **End time**

  The date and time when the trace was stopped, or the time it is scheduled to stop in the case of running traces.

- **State**

  The current state of the trace is one of the following:

  – Running: Trace is in progress.

  – Finished: Trace has finished normally because its end time was reached.

  – Stopped: Trace was stopped by the user before its end time was reached.

  – Error: Trace failed to complete, possibly due to an internal error or an Operations Monitor core restart.

- **Frames captured**

  The number of packets stored in the trace. For running traces, this value is the number of packets written thus far.

- **Size**

  The size in bytes of the capture file.

All columns of in the *Running and Finished Traces* can be sorted. For example, you can click the **Size** column to sort the traces by size.

By default, the table is refreshed every five seconds, which can be changed using the **Refresher's** drop-down menu at the top right corner of the table.

When you select a trace from the list, the following actions are available from the toolbar above the table:

- **Download**

  Downloads the selected trace. This only works when selecting one trace at a time. Traces are compressed in a **.gzip** format in order to speed up the download.

- **Delete trace(s)**

  Deletes the selected traces from Operations Monitor storage.

- **Stop trace(s)**

  Stops the selected traces if they are currently running.

- **Restart trace(s)**

  Starts a trace with the same parameters as the selected trace, but with the time shifted to present. This only works for traces with relative start and end times.

Note: The total size of storage space, which all traces can use is limited. When the space is full, the oldest trace is automatically deleted. The default value of this limit is set to 40 GB on a standard Operations Monitor server, with the traces being stored in a compressed form.

Packet Inspector

The **Packet Inspector** page allows you to examine and download arbitrary network traffic captured on the probes. Downloads are done in the industry-standard PCAP file format, that contains the complete messages as monitored on the network.

Most of the available protocol analyzers can interpret the PCAP file format, including open- source tools. Any stored message contains the full raw frame including IP and TCP/UDP headers, as well as the timestamp when the message was received.

The filter for Packet Inspector determines the traffic that is captured and available for searching. You set this filter in the **Signaling Protocols** page in Platform Setup Application on the Probe. For increased system performance, it is important that this filter restrict the traffic that is captured and is comparable to the default filter. The Probe captures the packets from the moment the it is set up. See *Session Monitor Installation Guide* for more information about signaling protocol filters.

If you use Packet Inspector for recording media, you need to include media packets in the filter. Media packets are initially stored on the Probe machine until the Probe sends the packets to the Mediation Engine when a user downloads the media to a PCAP file. You need to ensure that there is sufficient disk space on the Probe machine for storing media.

Packet Inspector Page

The **Packet Inspector** page consists of a form where you specify the search parameters and a panel listing the preview of the results of a search.

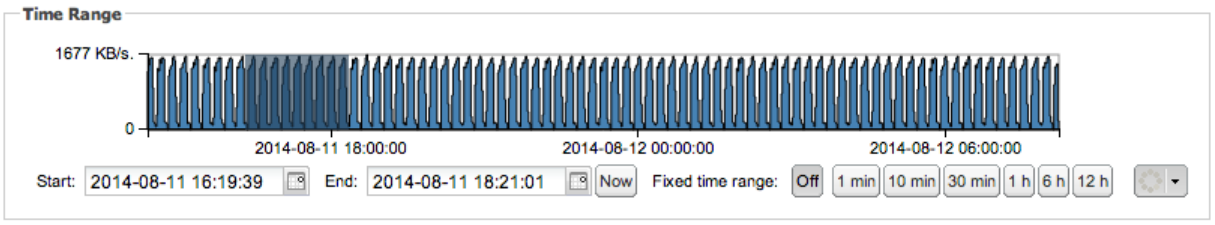
Query Panel

The **Query** panel displays a chart that shows the amount of traffic received every two seconds. You can use the chart for selecting a time range. Operations Monitor searches the packets captured during the selected interval of time.

Time Range Selection

To select a time range, you can select, move, or resize a time range selection in the chart or use the controls to select a specific start and end date or a fixed time range (see Figure 3–27). If you select a start and end date, the selection on the chart changes accordingly. If you select a fixed time range, the start and end dates and the selection on the chart changes accordingly.

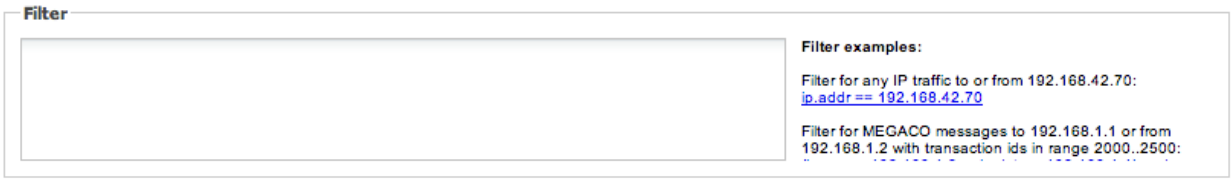
Figure 3–27 Time Range Selection



Filter

To filter the result set further, you can enter filters in the **Filter** text field. On the right of the **Filter** text field (see Figure 3–28) are some filter examples. See "Query Examples" for additional usage examples.

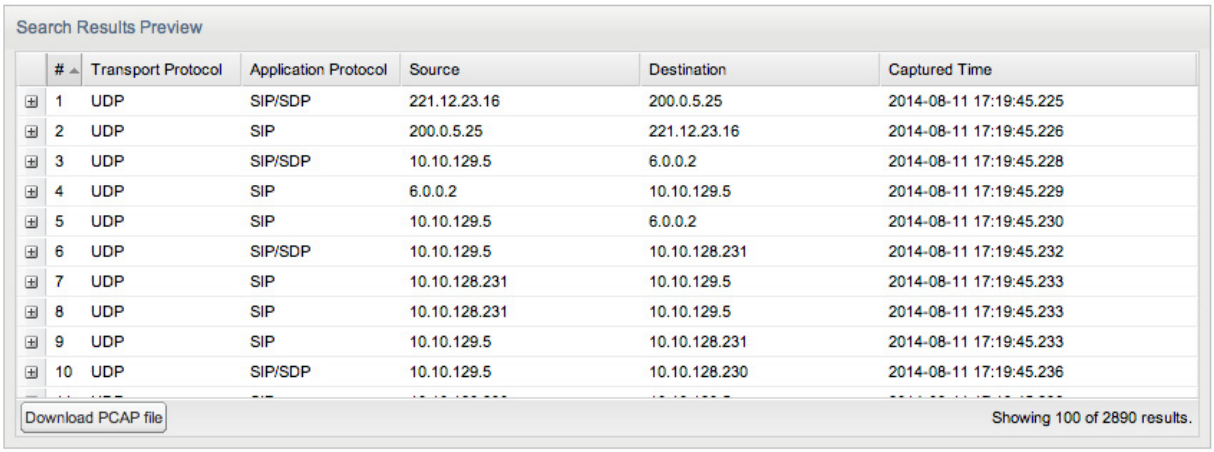
Figure 3–28 Filter Box



Search Results Preview Panel

The **Search Results Preview** panel shows a preview of the search results (see Figure 3–29). You can sort the preview using each column. To browse the packet data details, expand its corresponding row. To download the complete set of search results, click the **Download PCAP file** button on the bottom of the panel.

Figure 3–29 Search Results Preview Panel



Query Examples

Query examples are Wireshark syntax.The following are examples of queries that can be used in the **Filter** text field:

- **Filter for any IP traffic to or from 192.168.42.70:**

  ```
  ip.addr == 192.168.42.70
  ```

- **Filter for MEGACO messages to 192.168.1.1 or from 192.168.1.2 with transaction ids in range 2000..2500**:

  ```
  (ip.src == 192.168.1.2 or ip.dst == 192.168.1.1) and megaco.transid > "2000"
  and megaco.transid < "2500"
  ```

- **Filter for ISUP to the device with point-code 1111 calling number 1234567:**

  ```
  mtp3.dpc == 1111 and isup and e164.called_party_number.digits == "1234567"
  ```

Not all filter expressions work with fragmented packets. To get all packets, use an expression like:

```
sip.Call-ID == "91082042014050560561693 2@172.19.235.6" || ip.flags.mf==1
```

Using a query to filter for media packets can decrease system performance; therefore, it is not recommended.

Note: The total size of the storage space that Packet Inspector can use is limited. The default value of this limit is 90% of the disk space for each configured probe. Furthermore, there is a limit of three concurrent queries for each *product* instance running.

To improve the speed of Packet Inspector queries, use the following syntax to search for IP or ports:

```
tcpdump: [bpf filter]
```

You can use this syntax to search only for IP or ports and without any message content. For examples of BPF filters, see the topic on Filter Syntax in the *Oracle Communications Session monitor Installation Guide*.

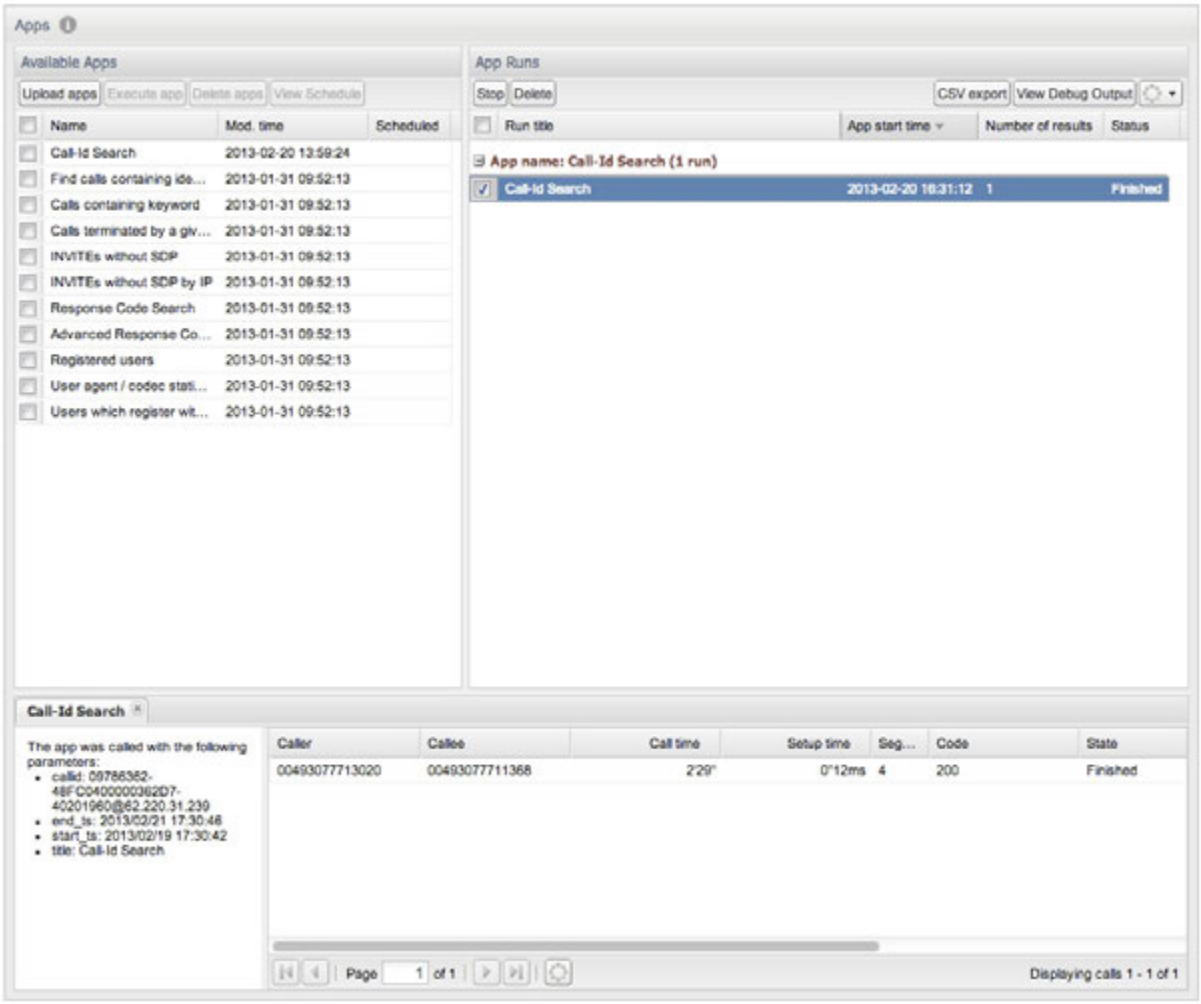
Note: Searches combining tcpdump/BPF and Wireshark filter language are not supported.

Apps

Apps allow you to extend Operations Monitor's reporting capability. You can upload an app to customize the queries and report information provided by Operations Monitor. Operations Monitor provides a number of default apps, which can be accessed on this page. For information on how to develop new extension apps for Operations Monitor, see "Implementing Apps".

Note: Apps searches are not guaranteed to be real-time without advanced hardware sizing.

Figure 3–30 Apps Page Overview



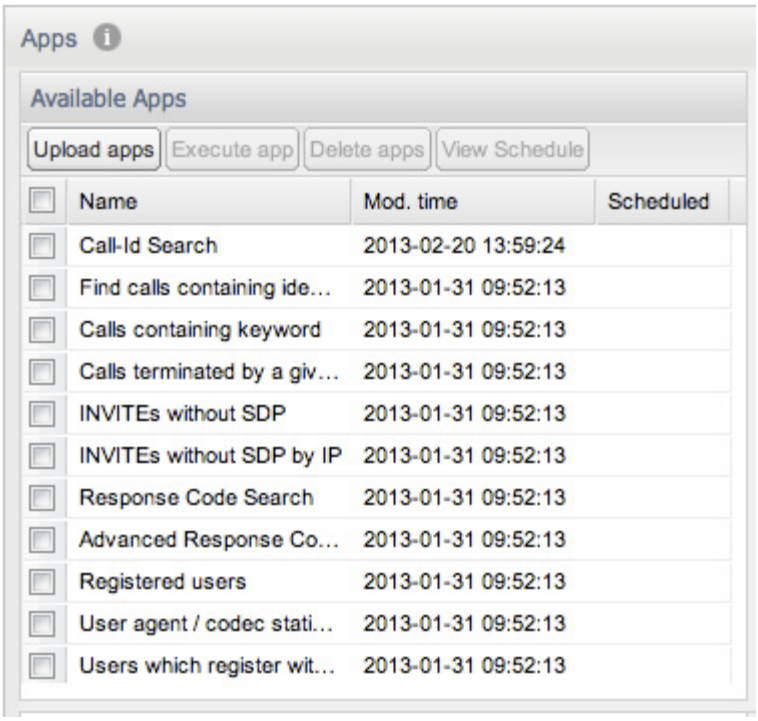
The **Apps** page (see Figure 3–30) is divided into three sections:

- The upper-left area lists *Available Apps*.
- The upper-right area displays outputs of previous *App Runs*.
- The bottom area displays the results of the selected *App Run*. It is empty when you first access the page.

Available Apps

The **Available Apps** table (see Figure 3–31) contains the columns **Name** and **Mod. time** by default. The first column displays a name for each application, the second column displays each apps modification time. The **Scheduled** column marks apps that are scheduled to run at a certain point in time. Additionally, the **Id** column can be enabled, which displays a unique identifier for each app. This is useful when you want to access the application using the remote procedure call functionality described in section "Remote App Procedure Calls" of this manual.

Figure 3–31 Available Apps Table



Uploading Apps

You can upload new apps to Operations Monitor in the **Available apps** table. Click **Upload app** to open a dialog box to search for an app archive on your computer and upload it to Operations Monitor. The procedure for creating an archive is described in the section "Implementing Apps".

Caution: When creating your own applications, or using third-party applications, test your scripts in a test environment to ensure they are safe before uploading them to your production environment.

Applications approved by Oracle are safe to use in your environments. However, non-approved applications could cause security and performance issues. Oracle is not responsible for any loss, costs, or damages incurred from using your own applications, or third-party applications.

Running Apps

The **Available apps** section also allows you to start any app that is already present on the system. Starting an app requires you to enter any parameters that the app defines.

To start an app, select it in the table and click on the **Execute app** button. A dialog appears that allows you to enter the app parameters. Every app has a **title** parameter. The sole purpose of the **title** parameter is to distinguish between different app runs of the same app in the **App runs** table. Click **Execute** in the parameter entry dialog to go to the **App runs** table. Execution also starts when a user double-clicks one of the available apps.

There are limits for simultaneous execution of scripts and for the maximum debug output that an script can generate. They are defined by the Maximum Simultaneous Script Runs and the Maximum Script Output Size system settings. For more information, see "Maximum Simultaneous Script Runs" and "Maximum Script Output Size".

Scheduling Apps

You can also schedule a repeated execution of a given app. For more information, see "Scheduling App Execution".

The table entitled **Available apps** allows you to delete apps from Operations Monitor. To do this, select the app you want to delete in the table and click the **Delete app** button. After confirmation, the selected app is deleted from the system.

The **View Schedule** button displays the scheduling configuration for a given app: the cron expression which determines when this app is triggered, the e-mail address to notify, and the option to attach the result as a **csv** file.

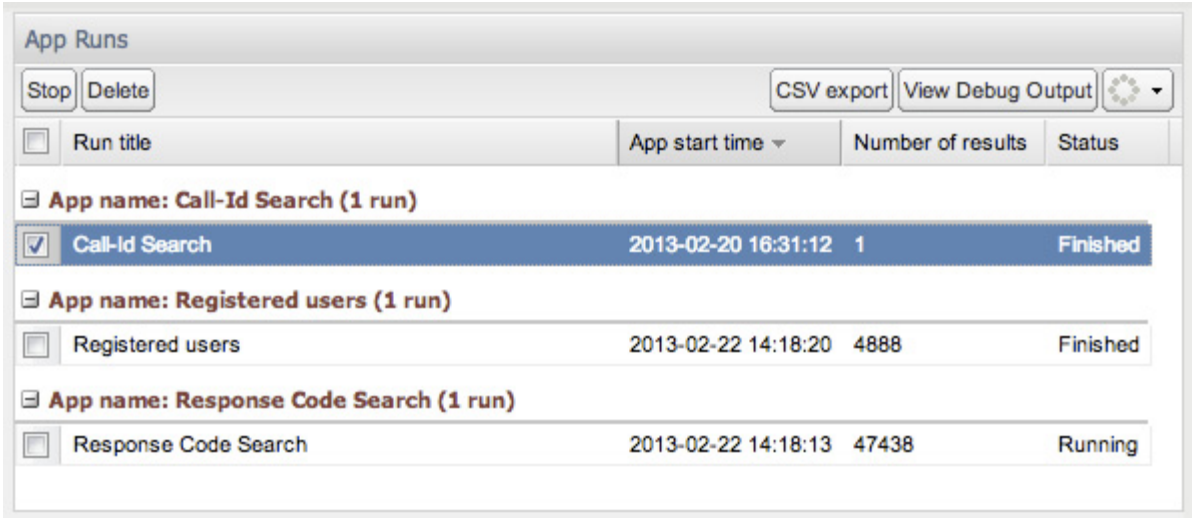
View App Runs

The **App Runs** table (see Figure 3–32) displays four columns by default:

- The **Run title** column displays the **Run title** parameter that was entered when starting the app.
- The **App start time** column displays the start time of the app.
- The **Number of results** column displays the number of results that the app reported. This column updates as the app run proceeds.
- The **Status** column displays either **Starting**, **Running** or **Finished** depending on whether the app run is still in progress or already finished.

Two other columns can be enabled: the **Id** column to display the ID of each app run, and **App name** to show the unaltered name of the app. The **Id** column is useful when you want to access the results of an app using the remote procedure call functionality. For more information, see "Remote App Procedure Calls".

Figure 3–32 App Runs Table



- **Exporting the App Runs to CSV**

The **CSV export** button lets you download the displayed items as a **CSV** file.

- **Debug App Run**

  The *View output* action opens a window in which the console output of the selected app is shown.

- **Stopping and Removing App Runs**

  The **App Runs** table allows you to stop an app run that is still active. To do this, select the app run you want to stop in the table and click the **Stop** button.

  Note: You can only stop an app run when it is in the **Running** state.

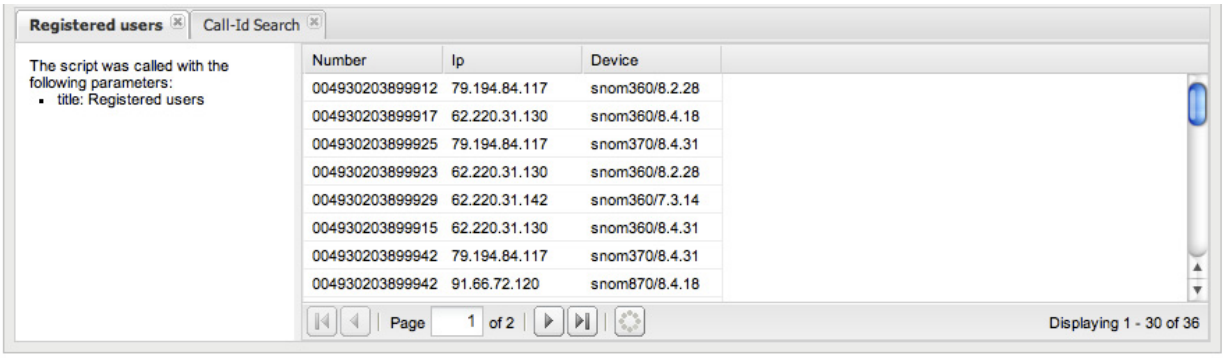
The **App Runs** table can also be used to remove app runs from the system. Select an app run in the table and click the **Delete...** button. You can only delete app runs that are in a **Finished** state.

View App Results

The lower portion of the **Apps** page displays result pages for app runs. The **Apps** page is empty when you first access it. When you double click an app run in the **App Runs** table, a new tab containing an app results view appears in the lower area.

App results view consists of a text display, which shows the parameters provided when starting the app, and a table containing the data rows reported by the app. The information in the table depends on the type of app that generated the results. Apps that search for calls use a similar results table to the calls table described in the section "Calls". Similar to the **Recent calls** table, you can double click on a row to get more detailed call information. This type of results table is shown in Figure 3–33.

Figure 3–33 Calls Results Page



The results page varies for each app. Multiple app results can be displayed simultaneously, and a new tab is opened in the lower portion for each.

Scheduling App Execution

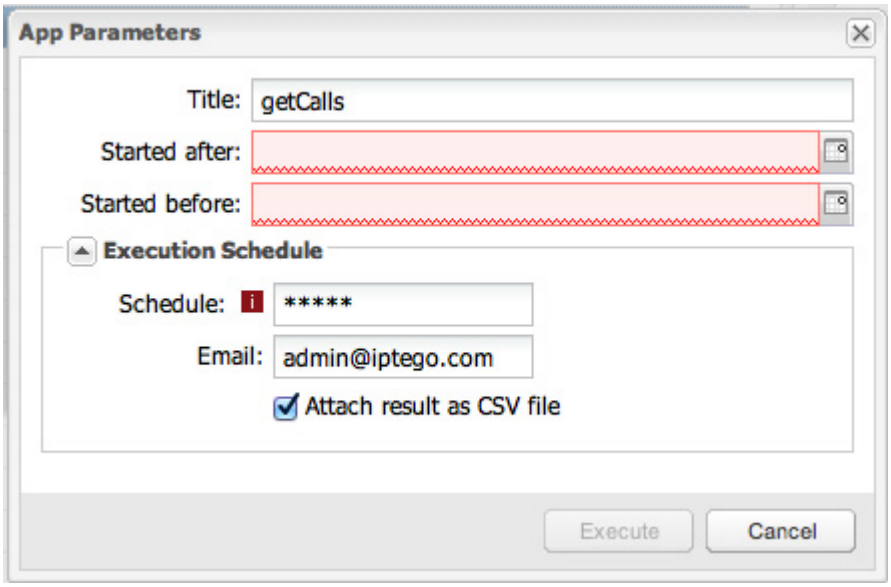
Apps can be scheduled to run periodically.

To set up an app schedule:

1. Select the app from Available Apps and click the **Execute App** button.
2. In the App Parameters dialog box, click on the **Execution Schedule** arrow to expand and show the scheduling options (see Figure 3–34):

- The Schedule itself must be supplied as a cron expression. For more information, see "Cron Expressions".
- Optionally, an email address can be provided for notification of app completion. The results of the execution may be attached to this email as a **CSV** file. Each execution of a scheduled app will also always appear in the View App Runs. For more information, see "View App Runs".

Figure 3–34 Scheduling Apps

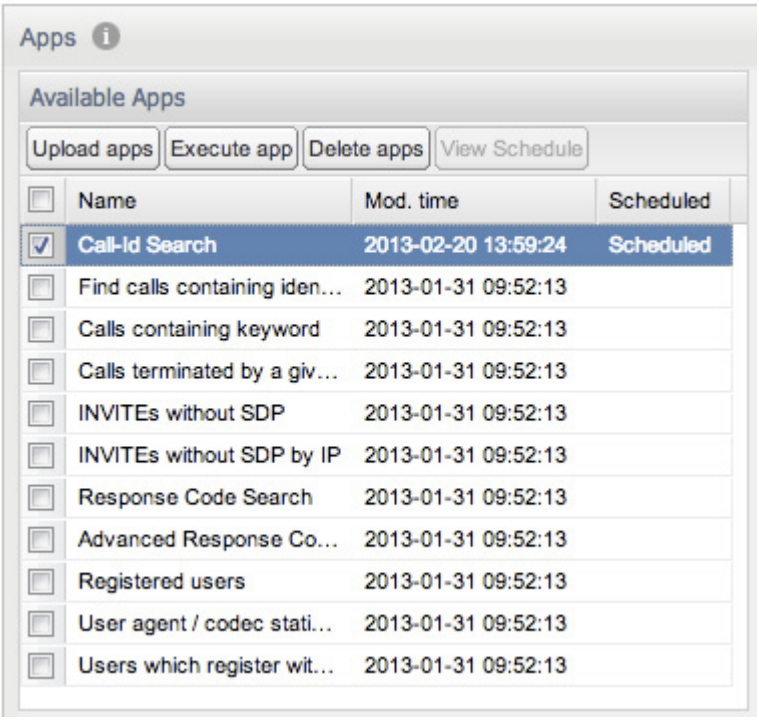


Multiple schedules can be defined for each app. To avoid schedules with a high frequency monopolizing resources, the system limits the number of stored app runs. This is defined by the Maximum Stored Script Runs system setting. For more information, see "Maximum Stored Script Runs".

Removing an App Schedule

Once a schedule has been set it can be deleted from the schedule list view. Select the app and click the **Delete apps** button in the Available Apps. Scheduled apps also have the string *Scheduled* visible in the **Scheduled** column of this table. For more information, see "Available Apps".

Figure 3–35 Scheduled App



Cron Expressions

Operations Monitor supports cron expressions as a means of defining your app schedules. An expression consists of 5 fields separated by white space that represent a set of times. The first field represents minutes, followed by hours, the day of the month, month, and day of the week. For example:

- To run an app every minute use this expression:

  ```
  * * * * * *
  ```

- To run an app every 30 minutes (on the 0th and 30th minute of every hour):

  ```
  0,30 * * * *
  ```

- And below, every Monday (2nd day of the week) at 1am:

  ```
  0, 1, 2 * * *
  ```

> **Note:** App results that include data for calls (result type "calls") remain valid and accessible only as long as the calls are accessible in the system. Once the call(s) fall out of Operations Monitor's scope, the app results which reference them will no longer be available.

4

Operations

This chapter describes how to work with the Operations voice and video monitoring features of Oracle Communications Operations Monitor.

KPI/Metrics

The **KPI/Metrics** page provides standards of measurement by which efficiency, performance, progress or quality of the platform are assessed. Such measurements are called metrics or Key Performance Indicator (KPI).

A metric is a measurement of a numeric property on the monitored platform. Operations Monitor evaluates the metric every second, and aggregates the values into minute and hour averages, minima and maxima. Additionally, it keeps a history of the following increments: one hour, minute and hour averages, and minimum and maximum values.

Operations Monitor offers a wide range of predefined metrics, and allows you to create new ones. Thus, each realm defined in Operations Monitor can have its own personalized list of defined metrics. Operations Monitor users associated to a realm are able to see and change only the metrics defined for that realm. For more information about realms, see "Realms Definitions". Operations Monitor users can save their most used metrics into a **Favorites** list.

You can create an *average* of a metric in Operations Monitor. For example, this is useful if a provider wants to be notified when the number of active calls drops below a certain threshold, but not taking into account the low number of calls during night or weekends. In this case, an average of the active calls during the working days is useful. For information on how to configure this feature, see "Average Metrics/KPIs".

A KPI is a combination of metrics, and is usually defined as ratio of other metrics. Operations Monitor supports the IETF KPIs defined in the draft ietf-pmol-sip-perf-metrics-04.

The evolution of any metrics or KPI is presented in the chart in the upper area.

KPI/Metrics Monitoring Chart

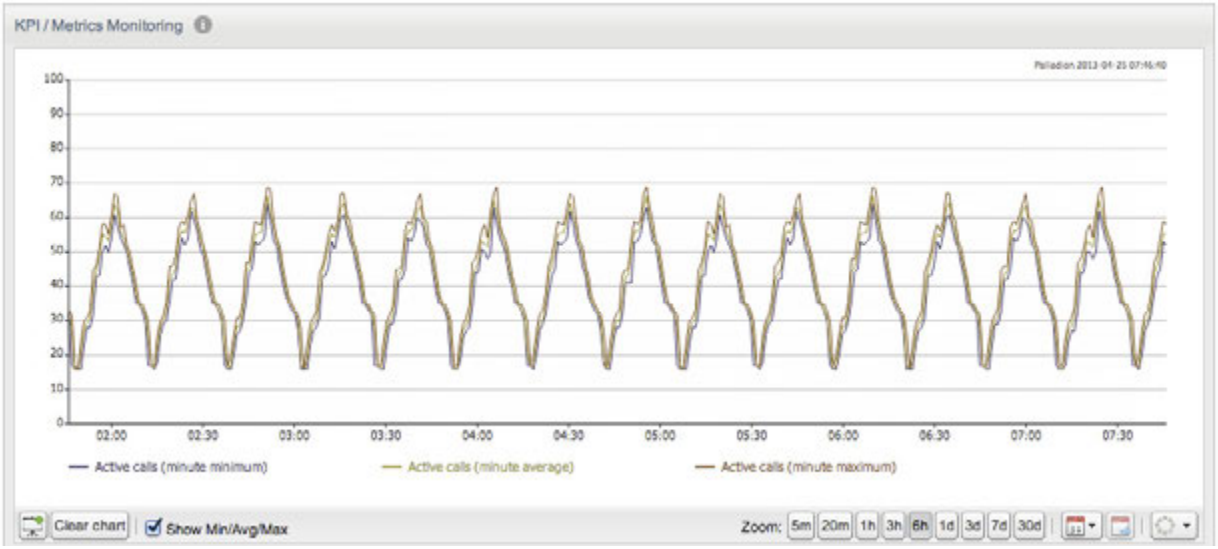
The **KPI /Metrics Monitoring** chart (see Figure 4–1) displays the evolution and history of up to five measurements, including the average values as well as minimum and maximum values.

This chart operates like others in Operations Monitor: you can zoom out to see the evolution through the past, zoom in and scroll to the present, or pan to a specific time. For more information, on how to use Operations Monitor charts, see "Charts".

To display the minimum, average and maximum values of the metrics on the chart, mark the **Show Min/Avg/Max** box at the bottom of the panel. Note that min, average, and max only show meaningful differences when using larger temporal resolutions.

To display the **KPI/Metrics in the Monitoring** chart, you must mark the metric in the **Library: Platform-Wide Metrics and KPIs** section. For more information, see "Library: Platform-Wide Metrics and KPIs".

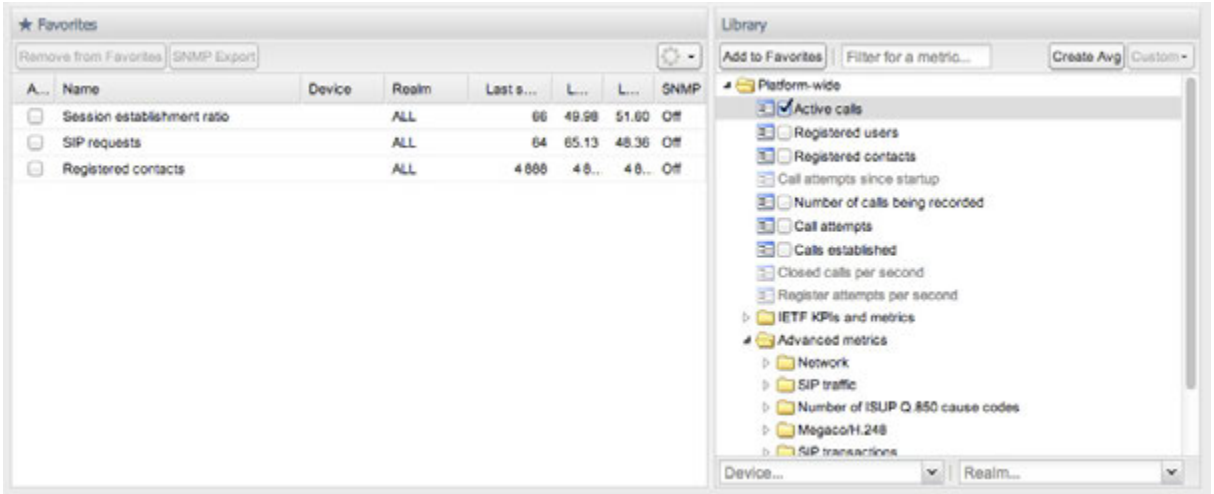
Figure 4–1 KPI/Metrics Monitoring Chart



KPI/Metrics Monitoring Grid

The **KPI/Metrics Monitoring** panel contains a **Favorites** list created by the Operations Monitor user that displays the metrics they are most interested in (see Figure 4–2). The **Library** provides available metrics and KPIs to select, view, mark as favorite and customize.

Figure 4–2 KPI/Metrics Monitoring Library and Favorites



Library: Platform-Wide Metrics and KPIs

The metrics measured on the entire platform are presented in the *Platform-wide* tree under the following three sections:

- *Defined metrics* is the first, unnamed section and where the most common metrics appear.
- *IETF KPIs and metrics* contains the KPIs as defined in draftietf-pmol-sip-perf-metrics-04.
- *Advanced metrics* is the section where you can configure customized metrics.

Pre-Defined Metrics

Operations Monitor adds the most frequently used metrics at the top of the list:

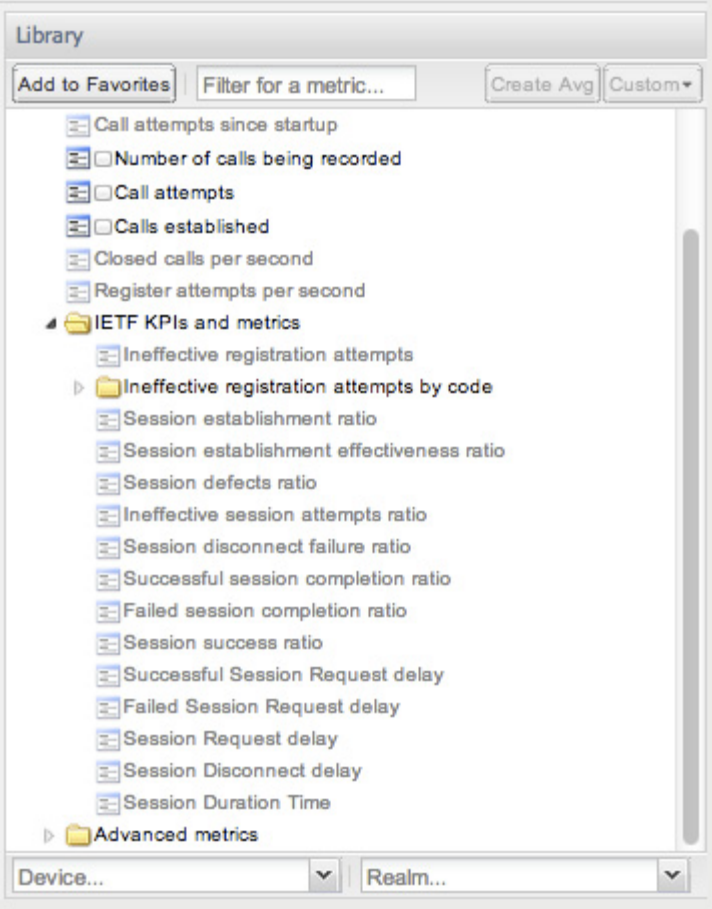
- Active calls.
- Registered users.
- Registered contacts.
- Call attempts.
- Call established.
- Closed calls per second.

To view a metric from the library in the chart above, you must mark the check box. By default, the *Active calls* metric is displayed.

IETF KPIs and Metrics

In the **IETF KPIs and metrics** section are the KPIs defined in the draft ietf-pmol-sip-perf-metrics-04. A list of IETF KPIs is shown in Figure 4–3.

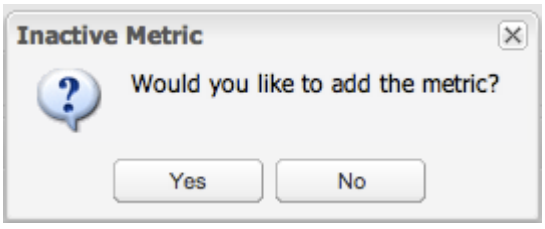
Figure 4–3 IETF KPIs



Enabling Metrics

The metrics displayed in *gray* text are *inactive*. Operations Monitor measures these metrics when you activate them. To enable a metric, click on the gray metric. The **Inactive Metric** window appears as shown in Figure 4–4. Click **Yes** to confirm.

Figure 4–4 Activate a Metric



After a metric is activated, the metric text is displayed in *black*. This signifies that Operations Monitor is providing values for this metric, and you can view its evolution in the chart by marking its check box.

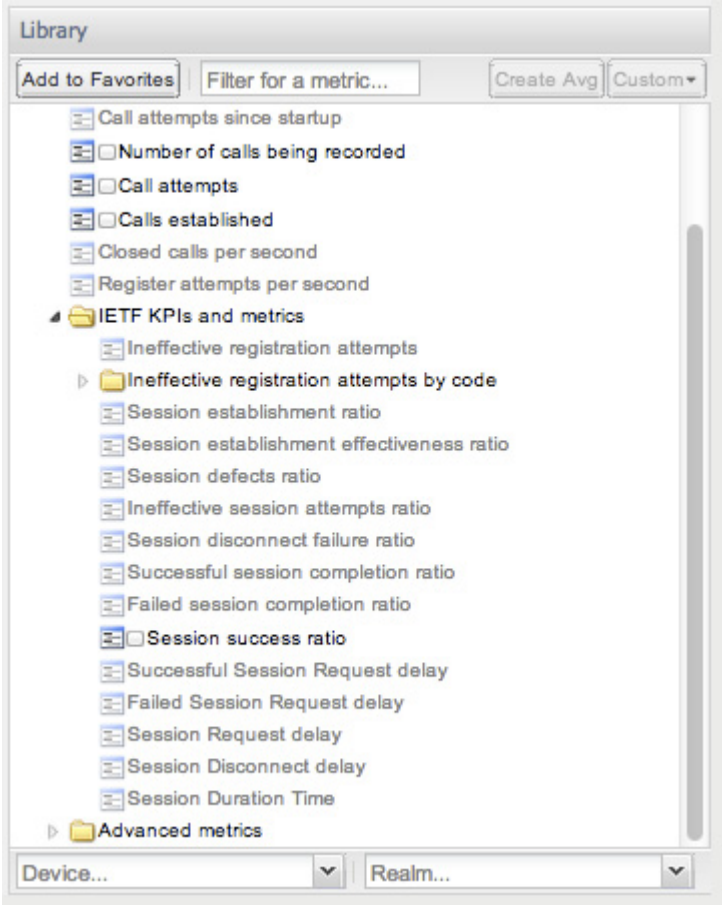
Some KPIs will not provide values for users in a realm. These KPIs are calculated on the transport level when the Ethernet frame is decoded. It is not possible to provide these values for users in a realm. These KPIs are:

- Bandwidth of the UDP packets

- Bandwidth of the TCP packets
- Bandwidth of the SCTP packets

These KPIs appear as disabled. Enabling them does not provide any value to a user in a realm.

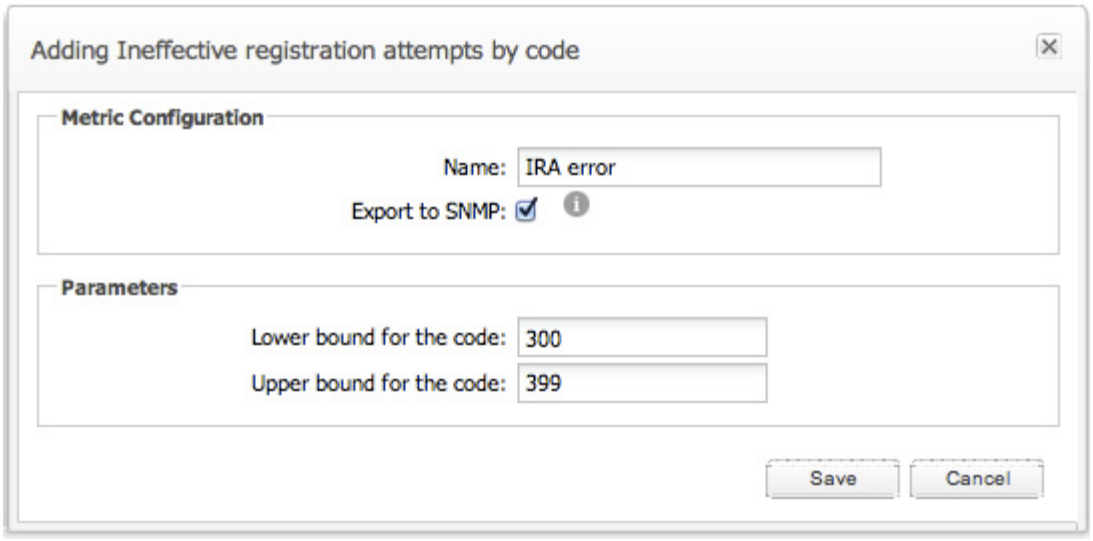
Figure 4–5 IETF KPIs with Session Success Ratio KPI Activated



Configuring KPIs

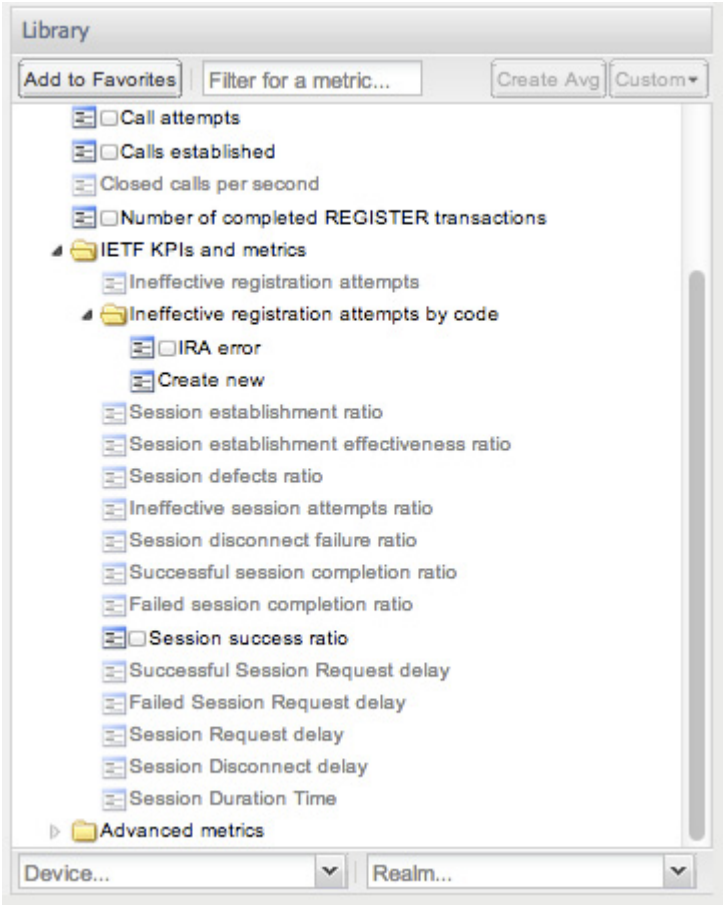
The **Ineffective registration attempts by code** KPI is configurable. It requires an interval of codes to be specified in order to be activated. Thus, you can customize its own KPI by adding the lower code, the upper code and the name of the new created metric as shown in Figure 4–6.

Figure 4–6 Define Ineffective Registration Attempts (IRA)KPI



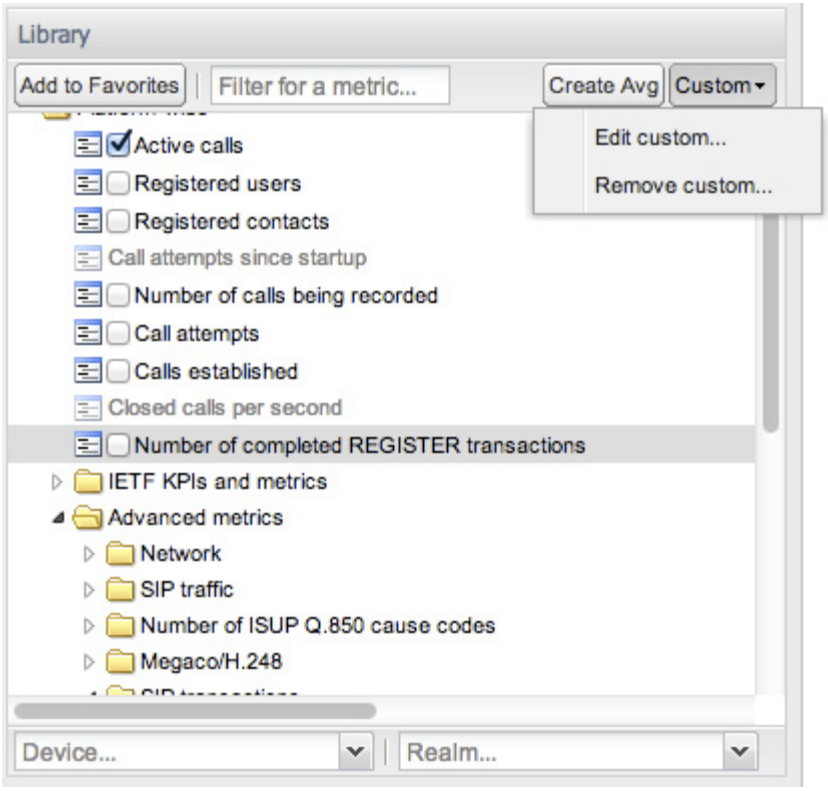
The newly defined KPI is added in the **Ineffective registration attempts by code** folder as shown in Figure 4–7.

Figure 4–7 Personalized IETF KPI



The user defined metrics can be later edited or deleted by clicking on the **Custom** button as shown in Figure 4–8.

Figure 4–8 Custom Menu



Exporting to SNMP

You can access each core metric through an external application via SNMP. To enable exporting to SNMP, select the metric in the library and click the **Custom** button. Select **Edit custom** and mark the **Export to SNMP** check box. For so-called compound metrics, an export is not possible and the check box is not available.

Advanced Metrics

Metric types are grouped into the following categories (see Figure 4–8):

- **Number of ISUP Q.850 cause codes**

  Measures the sniffed ISUP traffic with certain Q.850 cause codes.

- **Network**

  Measures transport layer properties, such as the number of TCP streams or the number of SCTP packets.

- **SIP traffic**

  Measures the sniffed SIP traffic, such as the number of SIP requests or the number of certain SIP replies.

- **Megaco/H.248**

  Measures the amount of Megaco legs that are active, created, or finished.

- **MGCP**

  Measures the amount of MGCP legs that are active, finished, or completed.

- **SIP transactions**

Measures the SIP transactions with certain properties like the number of INVITE transactions that end with an error or the number of failed registrations.

- **Calls**

  Measures the calls with special properties, such as the number of calls with a call length smaller than 30 seconds.

- **Registrations**

  Measures the registrations with certain properties, such as the number of unauthorized registrations in the last second.

- **RTCP**

  Measures the amount of RTCP or RTCP-XR packets and their jitter, delay, and MoScq values.

- **RTCP Usage**

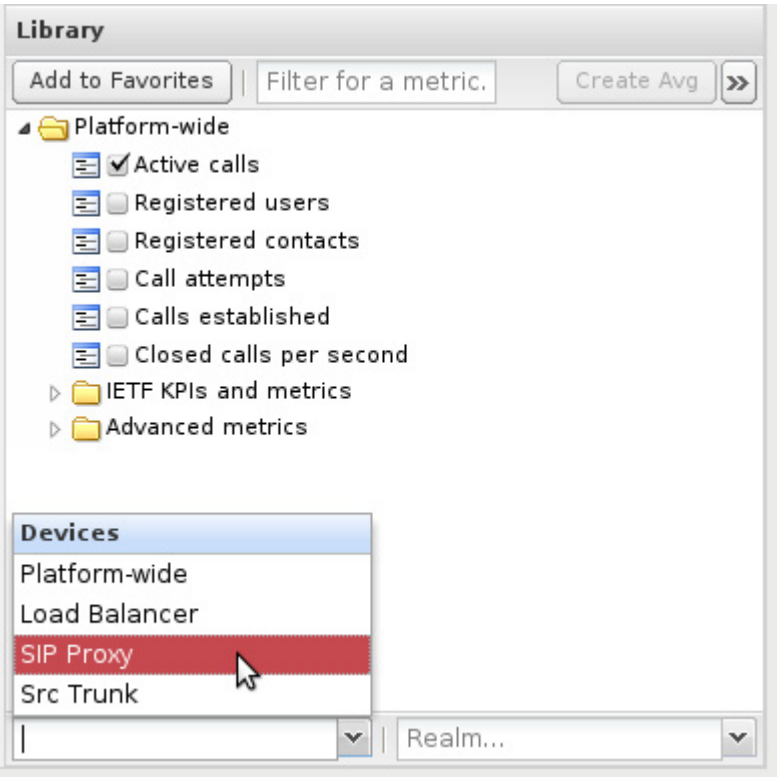
  Measures the RTCP or RTCP-XR streams.

Library: Per-Device Metrics and KPIs

Operations Monitor can measure metrics for a platform device or specified trunk. In order to enable this feature, you must configure the platform devices (or trunks) in the platform settings. For more information, see "Platform Devices".

In the **Library** section of the **KPI/Metrics** page, you must select the device or trunk from the **Devices** field below the tree of metrics as shown in Figure 4–9.

Figure 4–9 Selecting a Device



Once you select a device, the metrics are presented in the Library tree.

When you are creating a metric for a device, you must specify whether the metric represents incoming traffic, outgoing traffic, or traversing. For each of the calls metrics, three values are stored:

- *Incoming* counts the call legs which have the IP address of the selected device as a destination.
- *Outgoing* counts the call legs which have the IP address of the selected device as the source.
- *Traversing* counts the calls that are relayed by the selected device. In other words, the calls that have one incoming and one outgoing call leg to/from the device.

If the device is a trunk, *incoming* is replaced by *egress*, and *outgoing* is replaced by *ingress*, as trunks are always viewed from the platform point of view. For more information, see "Trunks/Prefixes".

Pre-Defined Metrics

Operations Monitor adds the most commonly used metrics at the top of the list:

- Active calls (incoming).
- Active calls (outgoing).
- Active calls (traversing).
- Call attempts (incoming).
- Call attempts (outgoing).
- Call attempts (traversing).
- Registered users.
- Registered contacts.
- Response time (msec).
- Transit time (msec).

Click the check box from the front of metrics name to see the evolution of one of these metrics in the chart.

- **Transit time**

  The time difference, in milliseconds, between the first INVITE message from the incoming call leg and the first INVITE message from the outgoing call leg. Only the calls that are relayed by the selected device are counted. To represent these values over time, if more calls are relayed in the same second, the maximum transit time is saved as value for that second. The web interface can then display average and maximum values per minute and per hour.

- **Response time**

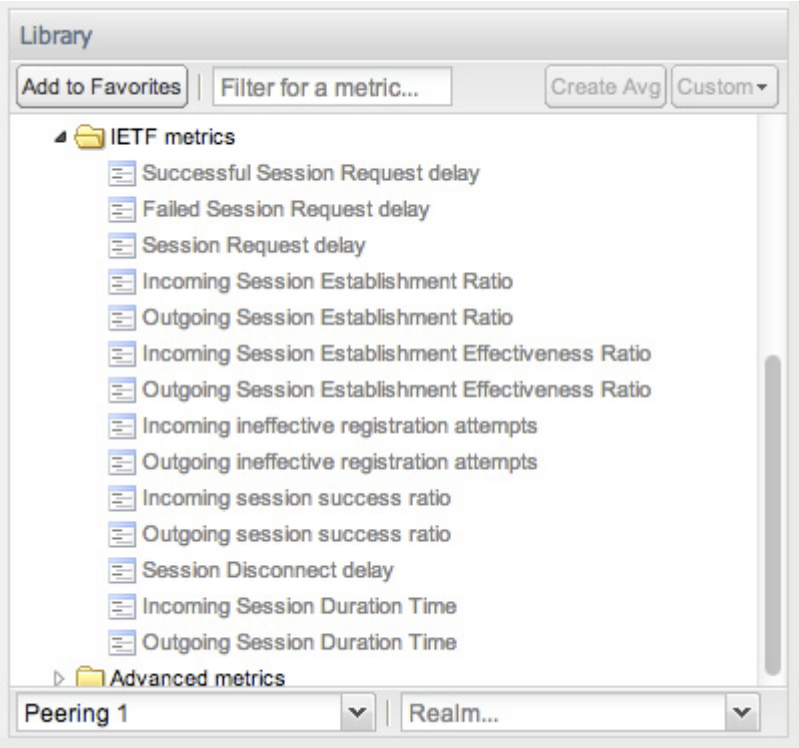
  The time difference, in milliseconds, between the initial request of a non-INVITE transaction and the final answer. Only the transactions, in which the destination IP address is one of the selected device's addresses, are counted.

  Note: The INVITE transactions are not considered here. The INVITE -> 100 time is always small, while the INVITE ->200 time is irrelevant, because it depends on when the callee answers.

IETF Metrics

A list of per device performance metrics (KPIs) is displayed in the **IETF Metrics** section of the library as shown in Figure 4–10.

Figure 4–10 IETF Device KPIs



Advanced Metrics

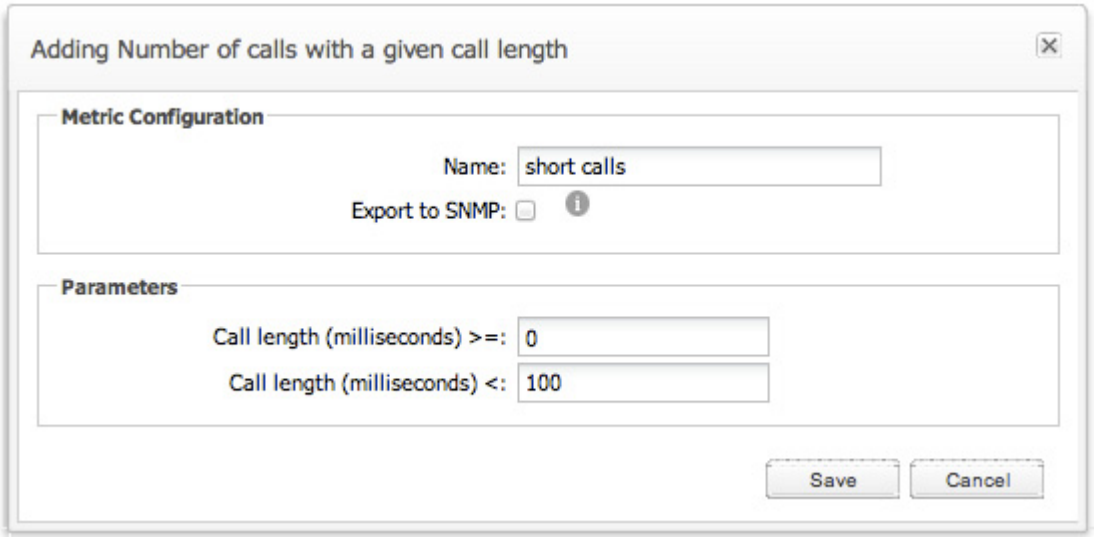
The web user can customize his own metrics under the categories presented in the advanced metrics tree.

Metrics displayed in a *gray color* are inactive and can be activated by clicking on them to bring up a window, as shown in Figure 4–10.

Once activated, their color becomes *black* and Operations Monitor starts providing measurements for them.

The metrics that share the same properties are grouped in folders and can be easily customized by clicking on the **Create new** link from the folder. An example of this group might be *Number of calls with a given call length*. This metric requires a couple of parameters to be configured, such as name, if it is exported via SNMP, direction of the call and the interval of the reply codes (see Figure 4–11).

Figure 4–11 Configure a New Metric



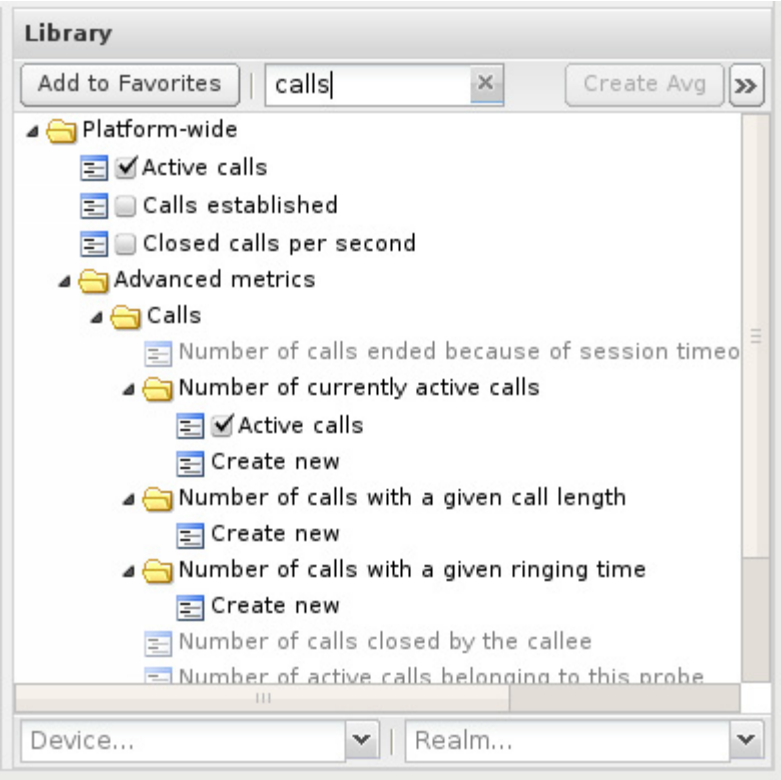
The user defined metrics can be edited or deleted by clicking on the **Custom** button, as shown in Figure 4–8.

The same tree of metrics is displayed under the **Devices** page as they are configurable for a platform device (or trunk). For more information, see "Devices".

Filtering in the Metrics/KPIs List

Operations Monitor offers a search box for filtering the metrics by name. For example, if you want to filter the metrics for those that contain 'calls', enter **calls** into the search box as shown in Figure 4–12.

Figure 4–12 Filter the Metrics

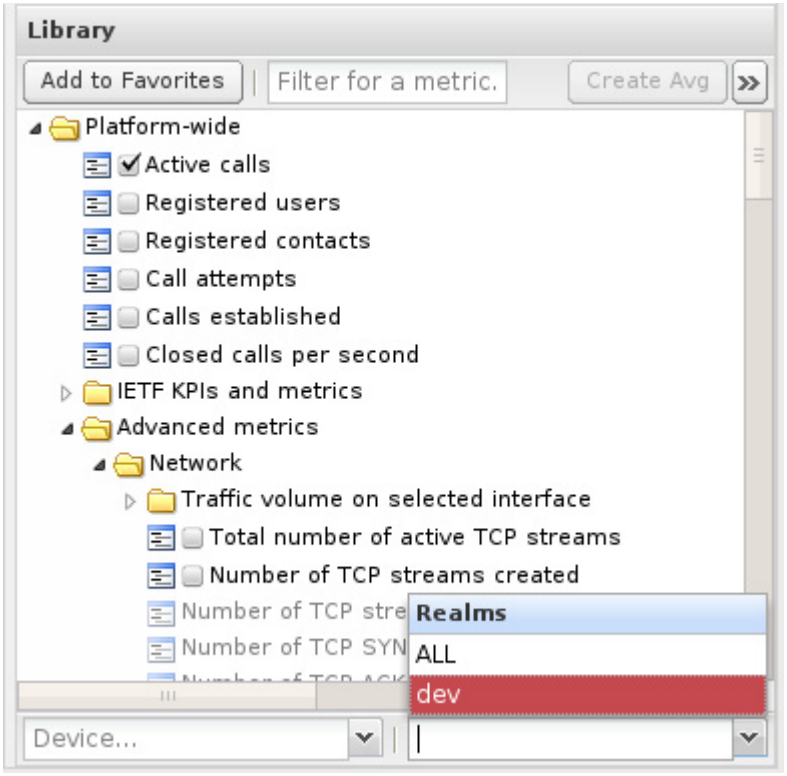


Restricting the Scope of Metrics/KPIs

You can restrict metrics/KPIs to a subset of the subscriber base using **Realms**. Operations Monitor users defined in the same realm share the same metrics, but another Operations Monitor user from a different realm is not able to see the metrics defined for other realms. The user associated with a realm can see metrics for any defined sub-realms. For more information, see "Realms Definitions".

The exception to this rule is the admin user who is able to see all metrics defined for each realm. The admin user can select a realm from the **Realms** list below the metrics tree, as shown in Figure 4–13.

Figure 4–13 Selecting a Realm



After choosing a realm, the admin user is able to see the metrics tree defined for that realm in the **Library** panel.

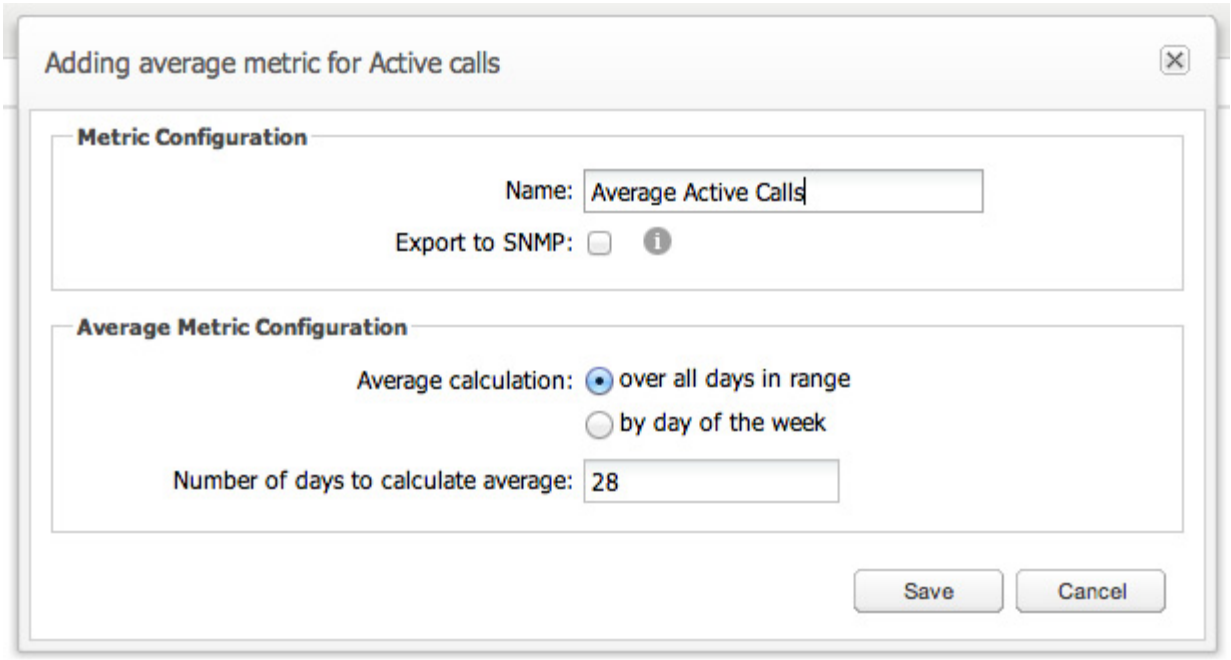
Note: The default Metrics/KPIs for a realm is created first when a user belonging to that realm logs in. For example, a realm bound user that logs in for the first time might be able to see old calls, but the *Active calls* metric only has data from the time the user first logged in and onwards.

Average Metrics/KPIs

An average metric or KPI can be created from any metric or KPI from the **Library** panel. In order to generate the *average metric*, first select the metric or KPI in the **Library** panel and then click on the **Create Avg** button (see Figure 4–2).

Average metrics require some configuration parameters such as a name, specifying if it is exported as SNMP, and the period it is calculated (see Figure 4–14).

Figure 4–14 Configure an Average Metric/KPI



As the traffic during the weekends might be significantly lower than during the working days, Operations Monitor offers the ability to calculate the average only for working days as Figure 4–14 shows. The period of time for which the average has been calculated is given in number of days.

After the average metric/KPI is created, it is added under the metric or KPI to which it belongs.

Adding Metrics to All Trunks

It is possible to specify a device-specific metric to be added to all existing trunks and to all trunks that will be defined in the future. This is done by selecting any trunk, right-clicking the metric in question, and checking **Add to all TRUNK devices**.

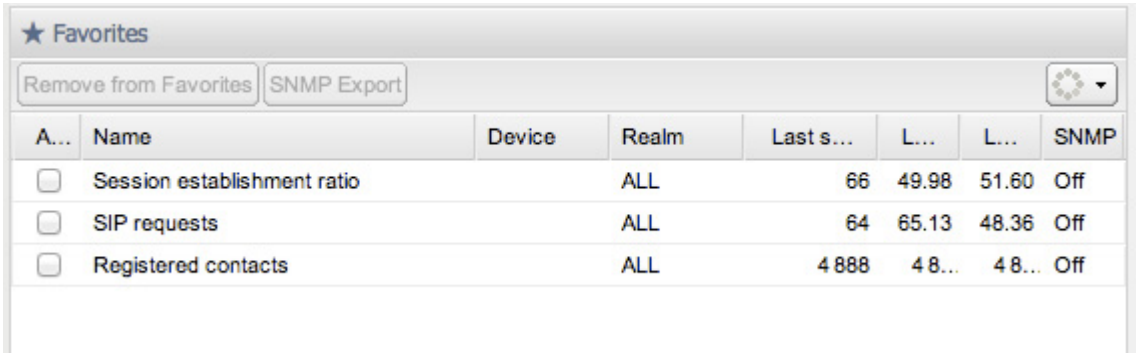
Metrics that have been added to all trunks may still be removed manually from individual trunks.

Deselecting **Add to all TRUNK devices** removes the metric from trunks it has been added to using this functionality, but not from trunks it has been added to manually, whether this happened before or after the functionality was invoked. Editing a metric on a trunk also prevents it from being removed from the trunk in this manner.

Favorite Metrics/KPIs

Any metric or KPI from the *Library* can be dragged to the **favorites** list for an at-a-glance view of a small set of metrics. Each Operations Monitor user has their own personalized **Favorites** list as it is saved by user, as shown in Figure 4–15.

Figure 4–15 Favorite List of Metrics/KPIs



Metrics in the Favorites list are displayed with the following information:

- *Name*, which is the name of the metric.
- *Device*, which is the device it is measured on.
- *Realm*, which is the realm it belongs to.
- *Last second*, which is its value in the last second.
- *Last minute avg*, which is its average on the last minute.
- *Last hour avg*, which is its average on the last hour.
- *SNMP*, which is **On** if the metric is exported via SNMP. Else it is **Off**.

> **Note:** In case of a *platform-wide* metric, the **Device** field is empty.

To remove a metric/KPI from the favorites list, first select the metric/KPI, and then click on the **Remove from Favorites** button. In case you want to enable/disable exporting the selected metric/KPI via SNMP click on the **Enable SNMP Export** or **Disable SNMP Export** button.

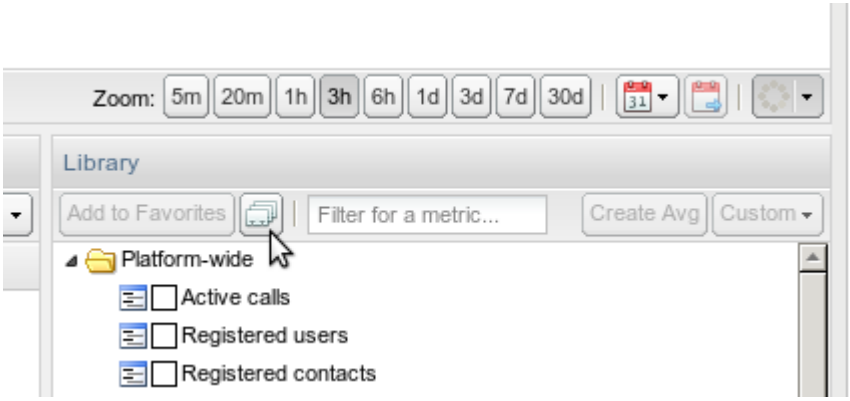
Bulk KPI/Metrics Creation and Removal

Operations Monitor provides the functionality for simplifying the addition and removal of multiple metrics in the **Bulk KPI/Metrics** dialog box. Using this tool, users can add and remove counters for multiple devices and IP Tags in a single operation. It is also useful for providing information about current KPI/counters usage.

Invocation

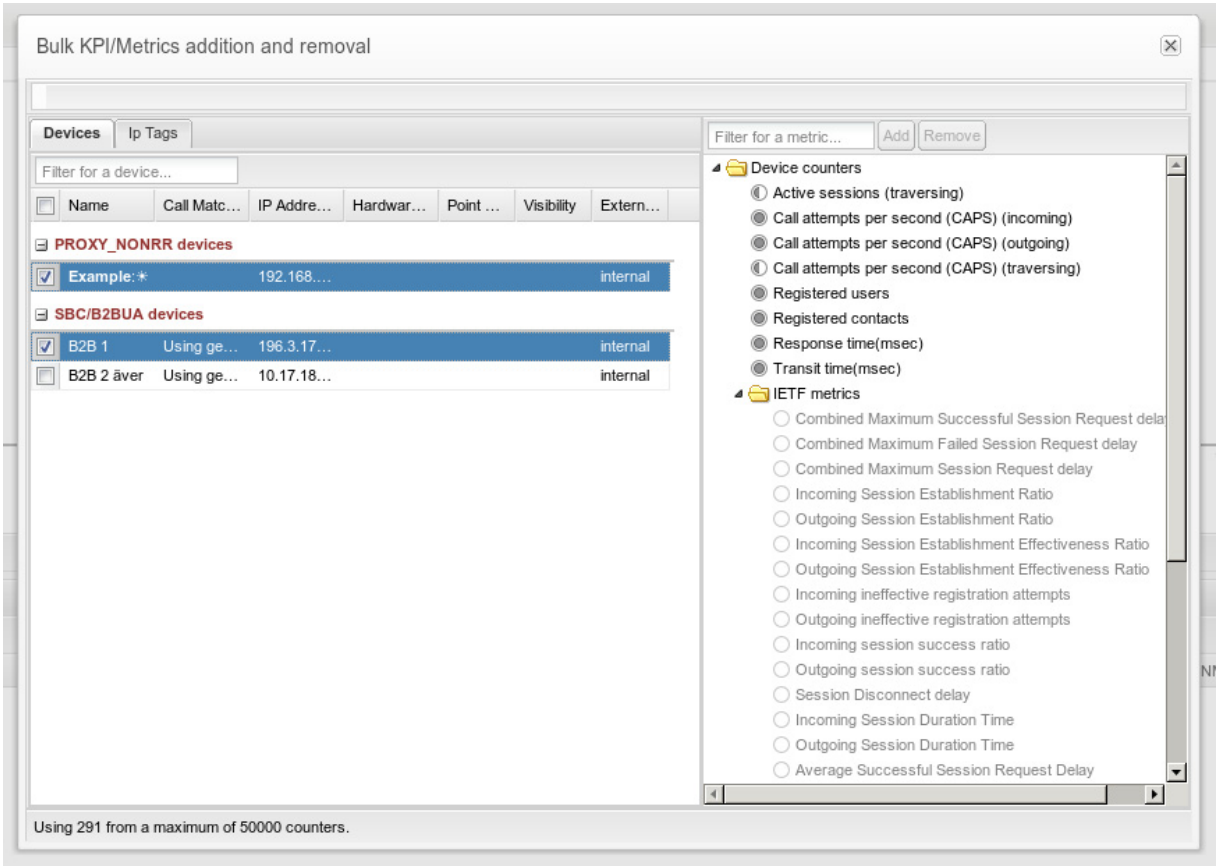
To invoke it, press the corresponding button you will find on the top of the Library toolbar on the **KPI/Metrics** page.

Figure 4–16 Starting Bulk KPI/Metrics



The following dialog will appear:

Figure 4–17 Bulk KPI/Metrics Addition and Removal Invocation



Information Shown

The Bulk KPI/Metrics addition and removal dialog window is composed with the following elements:

- **Devices** and **Ip Tags**

This panel contains two tabs, **Devices** and the **Ip Tags**. Each tab allows users to select different devices/iptags, which are show in their corresponding groups. The same operations can be performed on both tabs.

- **Counters**

  Shows a tree with the counters, as in the single device/IPtag widget but showing a symbol notation and a tooltip explains its meaning. These symbols have the following legend:

  – Light gray: not existing counter.

  – Half dark gray: counter exists for a subset of the selected devices.

  – Full dark gray: counter exists for all the selected devices.

  – Lock: counter cannot be added or removed.

  The top toolbar filter lets users filter by counter name.

- **Progress Bar**

  In the top of the dialog, it will show the progress of the bulk action. The process runs in background mode, so other actions can be performed in parallel.

- **Bottom Toolbar**

  Shows the current number of counters created and the limit enforced for the bulk counters operations. These limits are not enforced for single counter operations.

Usage

Select one or more devices/IPtags on the device and then select a counter definition in the **KPIs/counters** panel. Adding or removing counters for those devices is implemented using the **Add** and **Remove** buttons on top of the counters tree panel.

Depending on the number of existent enabled counters, the following behavior applies:

- Fully disabled counters are only allowed to be enabled.
- Fully enabled counters (when counter is present for all selected devices) are only allowed to be removed.
- Half enabled counters will be offered two options:
  – Add the ones that are not present in some devices.
  – Remove all the existing ones from all the devices.

Limitations

Some device counters are created by Operations Monitor and cannot be added or removed. They are shown with a **lock** icon. The user is not allowed to add or remove them.

Bulk operations cannot be performed in some critical counters. Those counters will not be shown in the bulk **KPIs/counters** dialog box but they appear in the single device Library widget, in the parent **KPI/counters** page. They cannot be removed.

Once the limit for the counters amount is surpassed, users will only be able to perform removal operations.

Calls

The **Calls** page is the central repository for call analysis in Operations Monitor. You can analyze call information by traversing the platform in real-time or historically. This data can be inspected as a high-level overview of all active calls, or a single call and its messages in detail. Operations Monitor can display SIP or ISUP calls together with MEGACO, MGCP, ENUM, and Diameter Cx.

This page contains two panels: the **Calls Chart** displaying the number of calls currently in progress, and **Recent Calls** showing details about recent calls. For more information, see "Active Calls Chart" and "Recent Calls".

Call Legs

A *call leg* is the portion of the call between two platform devices. Operations Monitor gathers messages from multiple points on the network, correlates them into call legs and merges them into *calls*. For example, the call from Figure 4–18 contains four call legs:

1. From the Trunk to the Load Balancer device.
2. From the Load Balancer to the SIP Proxy.
3. From the SIP Proxy back to the Load Balancer.
4. From the Load Balancer to the callee.

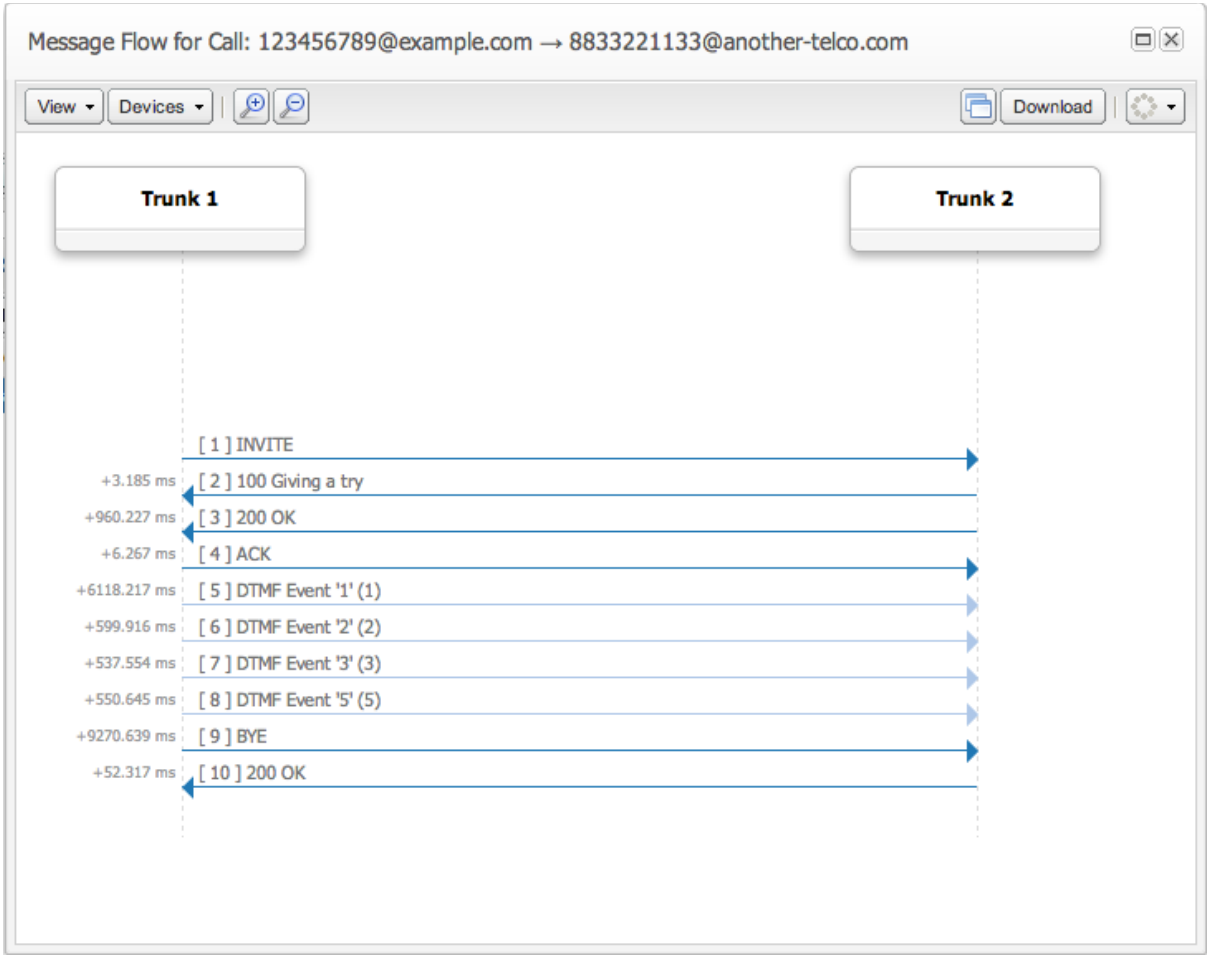
Figure 4–18 Example Call Flow

Merging call legs into *call flows* is important for network troubleshooting and gathering accurate statistics. For call merging to work properly, you need to configure the platform devices in the Platform Devices section. For more information about configuring platform devices, see "Initial Configuration".

DTMF Tones in Call Flow

The call flow shows the DTMF tones (see Figure 4–19). This is subject to certain user permissions.

Figure 4–19 Call Flow with DTMF Tones



Call States

Table 4–1 lists the call states:

Table 4–1 Call States

Call States	Description
Unauthorized	The call was answered with '401' or '407'. The UAC typically sends another INVITE containing the credentials. If these are accepted by the UAS, the state will change to **Proceeding** or **Established**.
Proceeding	The call enters this state immediately after the first INVITE is received, and stays in it until an answer changes the state. The **State details** column may provide extra information (INVITE seen, ACK seen but no final response yet).
Ringing	The call enters this state when the first '180 Ringing' answer is seen.
Established	The call enters this state when the first successful 2xx answer is seen. The **State details** column mentions if the ACK is not yet received (200 OK seen, waiting for ACK).

Table 4–1 (Cont.) Call States

Call States	Description
Finished	An *Established* call enters this state when the first BYE message is seen. The **State details** column mentions if the related 200 OK message is not yet received (BYE seen, waiting for confirmation).
Timed out	An *Established* call enters this state if it lasts longer than the configured *session timeout* limit. This limit can be changed from the **Session-timeout For Calls** system option. For more information, see "Session-Timeout For Calls".
Error	A call enters this state if Operations Monitor is unable to follow the call, due to an unexpected sequence of message.
Failed	A *Proceeding* call enters this state if a non-successful final response is seen or the 'INVITE' transaction times out. In the latter case, the **State details** column shows the 'Timeout during call setup' message.
Not Found	A *Proceeding* call enters this state if the 404 response code is seen.
Moved	A *Proceeding* call enters this state if the 301 or the 302 response code is seen.
Off-line	A *Proceeding* call enters this state if the 480 response code is seen.
Busy	A *Proceeding* call enters this state if the 486 or the 600 response code is seen.
Canceled	A *Proceeding* call enters this state if the 487response code is seen.

If a call has multiple call legs, the call state is computed from the call legs' individual states.

Active Calls Chart

The **Active calls** chart represents the number of established calls at the time of the measure. The minimum, maximum, and average for the sample period are displayed. Zooming and panning are available in this chart. For more information, see "Charts".

By default, the **Auto Refresh** is set to 30 seconds for this graph. For more information, see "Refresh Button".

Example

Figure 4–20 depicts a typical view for one day of the **Active Calls** graph. The number of established calls decreases at night and increases during peak hours.

Figure 4–20 Evolution of Active Calls Over 20 Minutes

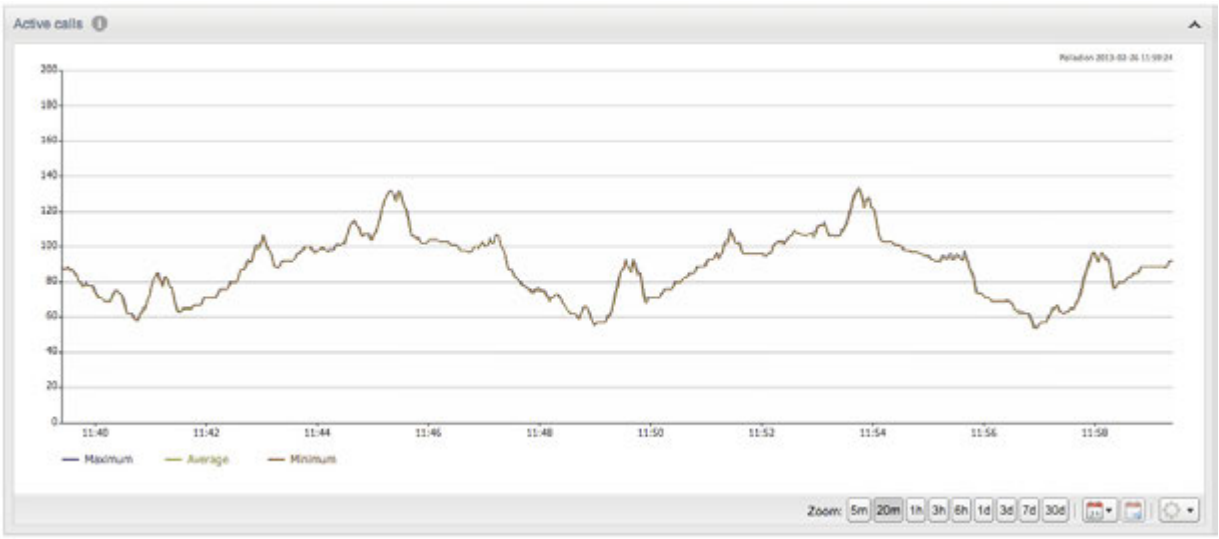


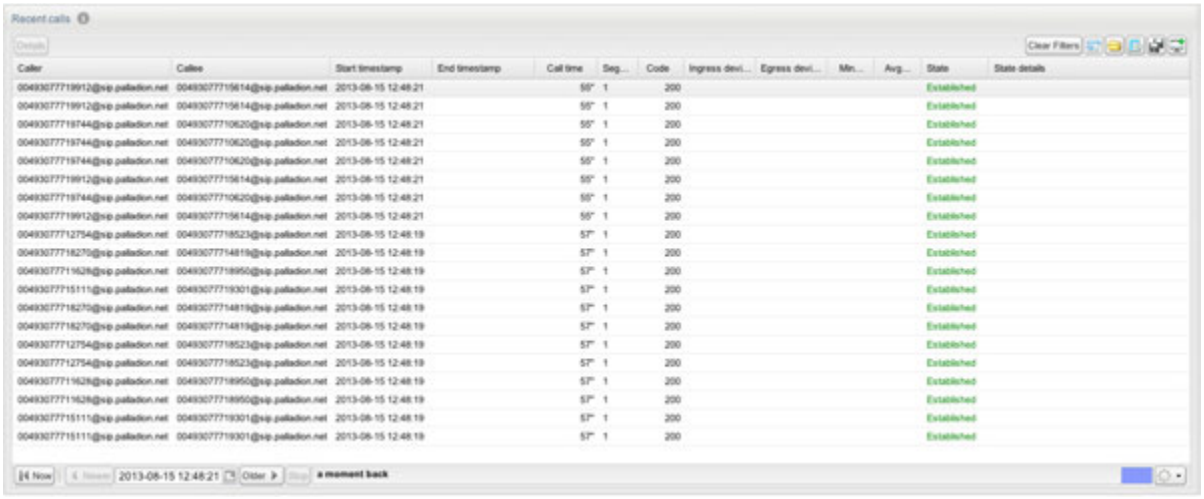
Figure 4–21 shows a view for 3 days.

Figure 4–21 Evolution of 'Active calls' Over 3 days

Recent Calls

The **Recent Calls** table displays recent and historical information for calls made in the last few days. The calls from this table are updated in real-time as their state changes.

Figure 4–22 Recent Calls Table



Recent Calls Table

Table 4–2 describes the columns of the Recent Calls table:

Table 4–2 Recent Calls Table Columns

Column	Description
Avg. MOS	The voice quality estimation for **Finished** calls. If Operations Monitor does not receive RTP traffic, or the RTP module is not loaded, this field is empty. The value is displayed in green, orange, or red depending on the thresholds set in the system options High Threshold for MOS and Low Threshold for MOS. For more information about how Operations Monitor estimates the MOS value of a call, see "Voice Quality".
Avg RTCP delay	The average round-trip delay time reported by RTCP.
Call time	If the call is *Finished*, this field represents the call length, measured from the 200 OK message of the INVITE transaction until the first BYE message. If the call is *Established*, this field represents the time elapsed since the call establishment (the arrival of the 200 OK message of the INVITE transaction) and it is updated on each refresh. If the call is not yet *Established*, this field is empty. This field has a precision of milliseconds but cannot be filtered for unless the call is *Finished*.
Call-Transfer	True if this call has been transferred using the call transfer capabilities in SIP.
Callee	The user to which the call is addressed. This is usually taken from the **To** header field of the first *call leg*.
Callee codecs	The comma delineated list of codecs proposed in the SDP body by the UAS. Usually this appears in the responses from INVITE transactions and the UAS includes a single codec in the answer, and this is the codec used in the call. On each *re-INVITE* from inside the dialog, this field is updated to the last proposed list of codecs. By default, this column is hidden.
Callee Initial codecs	The comma delineated list of codecs proposed in the SDP body by the UAS, usually in the first response from the 'INVITE' transaction. Unlike the **Callee codecs**, this field is not updated on *re-INVITEs*. By default, this column is hidden.

Table 4–2 (Cont.) Recent Calls Table Columns

Column	Description
Callee IP Address	IP address of the called user that connected first. It is possible to filter this column by IP address or IP address mask. By default this column is hidden.
Callee User Agent	The *User-Agent* string advertised in the 200 OK message of the last *call leg*. This string usually contains the brand and the firmware version of the SIP device answering the call. If the User-Agent header is not present, the string is taken from the 'Server' header if present. By default, this column is hidden.
Caller	The user making the call. This is usually taken from the **From** header field of the first *call leg*.
Caller codecs	The comma delineated list of codecs proposed in the SDP body by the UAC, usually in the 'INVITE' message. On each *re-INVITE* from inside the dialog, this field is updated to the last proposed list of codecs. For example, this is useful for detecting T.38 calls. By default, this column is hidden.
Caller Initial codecs	The comma delineated list of codecs proposed in the SDP body by the UAS, usually in the first 'INVITE' message. Unlike the **Caller codecs**, this field is not updated on *re- INVITEs*. This is useful for gathering statistics about the supported codecs. By default, this column is hidden.
Caller IP Address	IP address of the device initiating the call. It is possible to filter this column by IP address or IP address mask. By default this column is hidden.
Caller User Agent	The User-Agent string advertised in the 'INVITE' message from the first *call leg*. This string usually contains the brand and firmware version of the SIP device making the call. By default, this column is hidden.
Code	The *SIP response code* of the last received message from the 'INVITE' transaction. For *Failed* calls, this represents the *SIP error code*.
Diversion	Diversion URI of first Diversion header in call.
Diversion Type	Diversion type of first Diversion header in call. Possible values are: ■ deflection (CD) ■ do-not-disturb (DND) ■ follow-me ■ no-answer (CFNA) ■ time-of-day (CFTOD) ■ unavailable (CFUNV) ■ unconditional (CFUNC) ■ unknown: for **unknown** and all cases that could not be matched to any of the above ■ user-busy (CFB) ■ out-of-service
DPC	Destination Point Codes (DPC) contains the address of the destination for the ISUP call. This is always taken from the first ISUP leg seen by Operations Monitor.
DTMF	Displays 'Yes' if the there is DTMF information available for this call. Unless the user has the correct rights this field will not be available.

Table 4–2 (Cont.) Recent Calls Table Columns

Column	Description
Egress device(s)	This field contains a comma delineated list of the *egress devices* for the call, that is through which the call leaves the platform.
End timestamp	The timestamp of the message that closes the main leg of the call (usually the first BYE message).
Gateway Devices	As the MEGACO Gateway column contains the IP address of the MEGACO Gateway, the Gateway Devices column contains its name.
Ingress device(s)	This field contains a comma delineated list of the *ingress devices* for the call, that is through which the call enters the platform.
Initiator device	If the call was started from a *SIP device* of the platform (for example, a media server), this field contains the name of this device. The call must not be relayed by this device, but actually created by it. Otherwise, the field is empty. By default, this column is hidden.
Max RTCP delay	The maximum round-trip delay time reported by RTCP.
Media	Indicates if RTP recordings were requested for this call.
Media types	Indicates the media types that were negotiated in the call. Multiple media types are separated by a comma (for example: audio, video).
MEGACO Commands	Commands placed by the MEGACO Gateway Controller to the MEGACO Gateway in a transaction. For example, Commands exist to add/modify/subtract Terminations from the Context. See Figure 4–23.
MEGACO Context ID	Defines an identifier for each MEGACO connection. See Figure 4–23.
MEGACO Gateway	IP address of the MEGACO Gateway.
MEGACO Gateway Controller	IP address of the MEGACO Gateway Controller.
MEGACO Termination ID	A MEGACO TerminationID is defined for a PSTN line, a channel in a Trunk or rtp stream. Their format is a string like: line/1 or rtp/1 for RTP streams. See Figure 4–23.
MEGACO Transaction IDs	A MEGACO Transaction is identified by a Transaction ID. See Figure 4–23.
MGCP Call IDs	Hexadecimal strings of maximum of 32 characters that identify uniquely a call. See Figure 4–24.
MGCP Capabilities	Defines the capabilities of the endpoints. See Figure 4–24.
MGCP Connection IDs	The connection identifier is encoded as a hexadecimal string, at most 32 characters in length. See Figure 4–24.
MGCP Gateway IP	IP address of the MGCP Gateway.
Min. MOS	The minimum value of the voice quality estimation for **Finished** calls. If Operations Monitor does not receive RTP traffic, or the RTP module is not loaded, this field is empty. The value is displayed in green, orange, or red depending on the thresholds set in the system options **High Threshold for MOS** and **Low Threshold for MOS**. For more information on how Operations Monitor estimates the MOS value of a call, see "Voice Quality".
OPC	Originating Point Codes (OPC) contains the address of the originator for the ISUP call. This is always taken from the first ISUP leg seen by Operations Monitor.
P-Asserted-ID	The content of the P-Asserted-ID header from the initial INVITE SIP request.
Preferred Callee Number?	The number of the callee determined by the configurable number determination mechanism, if available.
Preferred Caller Number?	The number of the caller determined by the configurable number determination mechanism, if available.

Table 4–2 (Cont.) Recent Calls Table Columns

Column	Description
Prefix Group	Prefix Tags matching this call (0 to n tags).
Q.850 Code	Q.850 cause code for the ISUP call.
Q.850 Details	Q.850 Details for the ISUP call.
Q.850 State	Q.850 State for the ISUP call.
Reason	The content of the Reason header from the BYE, CANCEL, SIP request or from a failure SIP reply.
Remote-Party-ID	The content of the Remote-Party-ID or P-Preferred-Identity header from the initial INVITE SIP request.
RTCP streams	The number of RTCP streams belonging to the call.
Segments	The number of *call legs* in this *call*. By default, this column is hidden.
Setup Delay	This field represents the time elapsed between the initial' INVITE' message and the first valid network response, like '180 Ringing', '183 Session in progress', '480 Temporarily Unavailable', etc. It fulfills Session Request Delay for RFC6076. If the call contains ISUP, *Setup Delay* is computed on the first SIP leg. This field has a precision of milliseconds. By default, this column is hidden.
Setup Delay Type	Shows the type of *Setup Delay* computed. Can be 'Successful Session Request Delay' or 'Failed Session Request Delay'. By default, this column is hidden.
Setup time	If the call is *Established* or *Finished*, this field represents the time elapsed between the initial INVITE message and the call establishment, marked by the 200 OK answer in the INVITE transaction. If the call is not yet *Established*, this field is empty. This field has a precision of milliseconds. By default, this column is hidden.
Start timestamp	The timestamp of the first INVITE or IAM message from the call.
State	A short text representation of the call state. For more information on the possible values for this field, see "Call States".
State details	Extra details about the call state. For example, the state of a call can be *Proceeding*, and this field adds the information: ACK seen, but no final answer. For more information on the possible values for this field, see "Call States".
Terminator device	If the call was ended by a *SIP device* (for example a media server), this field contains the name of the device. The call must not be relayed by this device, but actually terminated by it. Otherwise, the field is empty. By default, this column is hidden.

Figure 4–23 shows examples of MEGACO call details in the **Recent Calls** panel.

Figure 4–23 MEGACO Details in the Calls Panel

MEGACO Terminati...	MEGACO Context ID	MEGACO Commands	MEGACO Transaction IDs
rtp/39	13	"Reply-Add","Reply-...	"1474450","1474451","1474465"...
rtp/246	110	"Reply-Add","Reply-...	"1474398","1474399","1474407",
rtp/65	106	"Reply-Add","Reply-...	"1474393","1474394","1474426"...
rtp/135	92	"Reply-Add","Reply-...	"1473791","1473792","1474285"...
rtp/259	88	"Reply-Add","Reply-...	"1474268","1474269","1474282"...
rtp/258	89	"Reply-Add","Reply-...	"1474161","1474162","1474171"...
rtp/139	31	"Reply-Add","Reply-...	"1474378","1474379","1474386"...
rtp/25	43	"Reply-Add","Reply-...	"1474227","1474228","1474248"...
rtp/211	80	"Reply-Add","Reply-...	"1474354","1474433",
rtp/3	38	"Reply-Add","Reply-...	"1474239","1474240","1474246"...
rtp/200	63	"Reply-Add","Reply-...	"1474348","1474350",
rtp/56	49	"Reply-Add","Reply-...	"1474340","1474341","1474349",
rtp/75	57	"Reply-Add","Reply-...	"1473292","1473293","1473328"...
rtp/123	5	"Reply-Add","Reply-...	"1474065","1474066","1474068"...
rtp/232	82	"Reply-Add","Reply-...	"1474315","1474316","1474322"...
rtp/207	37	"Reply-Add","Reply-...	"1474040","1474041","1474044"...
rtp/144	16	"Reply-Add","Reply-...	"1473905","1474027","1474303"...
rtp/19	23	"Reply-Add","Reply-...	"1474301","1474361",
rtp/112	25	"Reply-Add","Reply-...	"1474290","1474317",
rtp/189	69	"Reply-Add","Reply-...	"1474279","1474280","1474305",

Figure 4–24 shows examples of MGCP call details in the **Recent Calls** panel.

Figure 4–24 MGCP Details in the Calls Panel

MGCP Call IDs	MGCP Capabilities	MGCP Connection I...
"0004f495000000fd0000001f",		"4408351",
"0004e037000001080000001...		"4586620",
"00051d8f000000cc00000009",		"5274443",

Note: To save horizontal space, several columns are hidden by default. To enable the hidden columns, click the drop-down menu of any column header and select the columns menu sub-menu. For more information, see "Tables".

Filtering

You can filter the columns of the **Recent calls** table for specific criteria. When you apply a single filter, the **Filters** toolbar displays the label **SIMPLE**.

Use a simple filter to filter the **Recent calls** table for one specific criteria.

Applying a Simple Filter

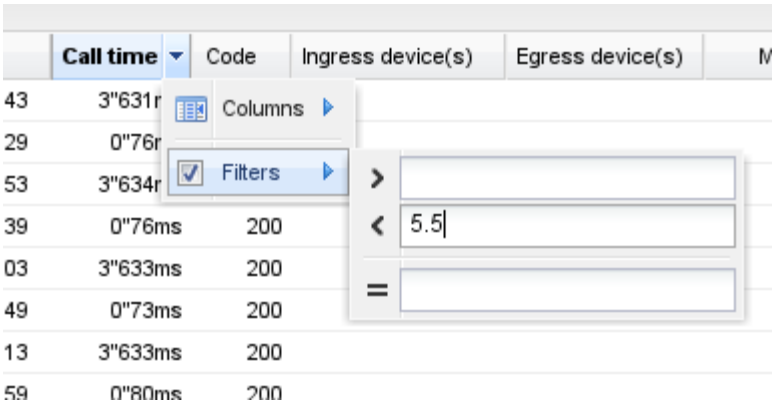
To apply a simple filter:

1. From the navigation list, select **Calls**.
2. On the **Recent calls** table, select the list of options on a desired column.
3. Select the **Filters** option.
4. Enter or select the filter criteria for the filter.

   The table is filtered for the criteria and the toolbar displays the label **SIMPLE**.

Figure 4–25 shows an example of a filter applied to the **Call time** column of the **Recent calls** table. In the example, a filter value has been entered as criteria.

Figure 4–25 Applying a Filter



Changing a Simple Filter to an Advanced Filter

To change a simple filter to an advanced filter:

1. On the **Filters** toolbar, click **Edit Advanced**.

   The **(Unnamed Filter)** dialog box appears.
2. Click **+**.

   A list of filter fields appears.
3. Select the fields to include in the advanced filter.
4. Click **Save**.

   The **Save filters** dialog box appears.
5. Enter a name for the filter.
6. Click **Save**.

   The dialog box now displays the name you assigned the advanced filter.
7. Click **Hide** which closes the dialog box.

Text-Based Filter Fields

Some text-based filter fields can accept queries. In text-based filter fields, you can enter REGEX pattern matching or SQL pattern matching to filter recent calls. REGEX filtering provides slower search times than SQL filtering.

To enter REGEX pattern matching to filter calls, enter the **.*** character combination before the search values.

When regular expressions are filtered, expressions supported are the same as MySQL 5.5.

The following example uses the REGEX filter:

```
.*[0049]
```

To enter SQL pattern matching to filter calls, there are two methods. SQL filtering is faster and more efficient than the REGEX pattern matching. SQL pattern matching is triggered as follows:

- Entering the SQL wildcard character % into the search field. It can be placed in front of or following the filter value.
- Entering ^ at the beginning of a string.

Table 4–3 describes the SQL search filters and provides examples of usage.

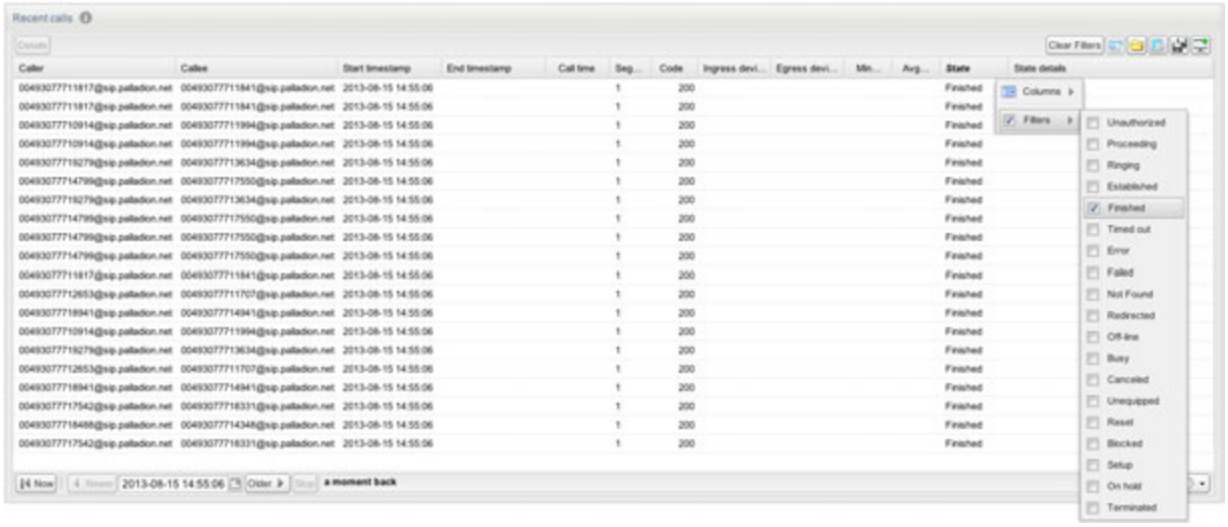
Table 4–3 SQL Filter Conditions

Search Filter	Search Example	Description
(none)	`0049`	Filters for strings that contain 0049.
^	`^0049`	Filters for strings that begin with 0049.

%	`0049%`
	or
	`%0049`
		Filters for strings that begin with 0049.
		
		Filters for strings that end with 0049.

Figure 4–26 shows an example of filtering that selects all finished calls. After the filter is set and the view is refreshed, the total number of finished calls is displayed in the bottom-right corner.

Figure 4–26 Filter All Finished Calls



The filtered columns can also be combined to provide more precise answers. For example, to limit the table to all finished calls initiated by SuperTrunk, set the following filters:

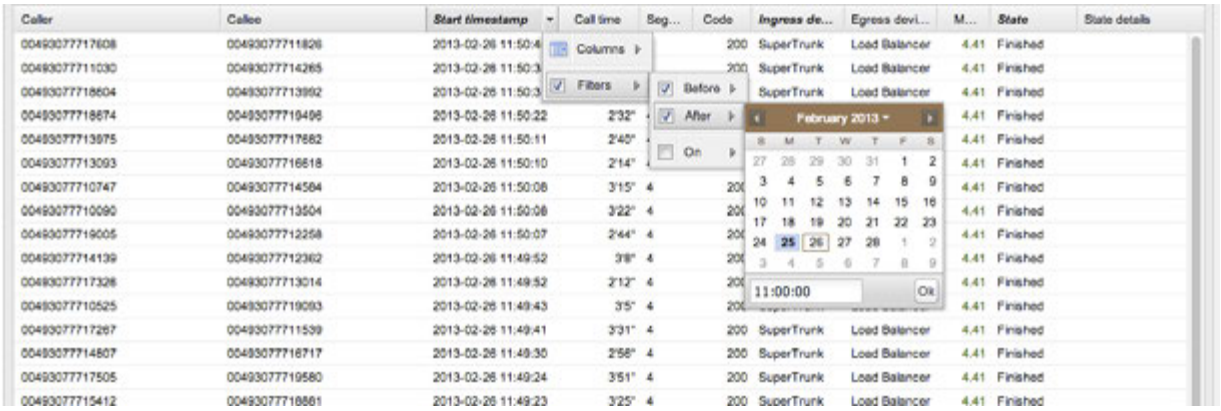
- **State** to **Finished**.
- **Ingress device** to **SuperTrunk**.

- **Start timestamp** to after 11:00.
- **End timestamp** to before 12:00.

See also Figure 4–27.

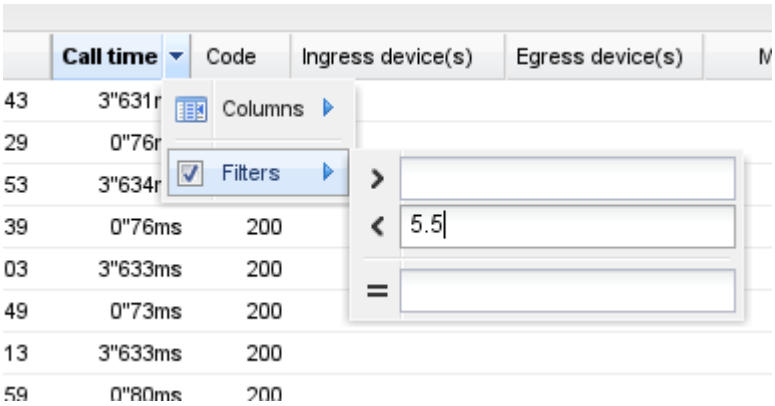
In this table, filtering can be set from the right-click menu. For more information, see "Right-Click Menu".

Figure 4–27 Filter Finished Calls Initiated by SuperTrunk Between 11:00 and 12:00



Filtering for this field is done in seconds.For example, in Figure 4–28, all calls that lasted less than 5.5 seconds are selected.

Figure 4–28 Filtering By Call Length



Note: When filtering for call length, only the finished calls are considered. Thus, it is not possible to filter for all the established calls with a certain call length.

To filter the calls list for calls that contain RTP streams, use **Media types** filter. You can set the **Media types** filter to list the calls that contain audio, video, text, image, application, message, or other media type.

When Media types filter is used, any call that contains any of the media types is listed. For instance, when the filter is set to **audio**, any call where audio is one of the media types that was negotiated in the call is listed, for example, a call containing audio and video.

When multiple media types are set in the filter, for example **image** and **video**, any call where image or video is one of the media types that was negotiated in the call is listed, for example, a call containing image and audio.

Advanced Filtering

You can create advanced filters to filter the **Recent calls** table for specific combinations of results. Advanced filters enable you to select a specific combination of columns from the **Recent calls** table, then enter criteria against which to filter for matching calls.

See Table 4–2 for details on the metrics available in the **Recent calls** table columns.

Some filter types require that you customize the logic that is applied in the filter. You can customize the logic by toggling options or entering specific criteria to be matched in each advanced filter you create. The logic that can be customized depends on the combination of filters selected.

When an advanced filter has been created, the **Filters** toolbar displays the label **ADVANCED**.

> **Note:** Advanced filtering requires higher resource consumption. Complex advanced filters utilize more system resources and may impact overall system performance.

Working with advanced filters involves the following tasks:

- Creating an Advanced Filter
- Adding Criteria to an Advanced Filter
- Adding Scope to an Advanced Filter
- Editing an Advanced Filter
- Applying an Advanced Filter
- Clearing Advanced Filters

Creating an Advanced Filter

To create an advanced filter:

1. From the navigation list, select **Calls**.
2. On the **Filters** toolbar, click **Clear** which removes any applied filters.
3. On the **Filters** toolbar, click **Edit Advanced**.

   The **(Unnamed Filter)** dialog box appears.
4. Click **+**.

   A list of filter fields appears.
5. Select fields to include in the advanced filter.
6. Click **Save**.

   The **Save filters** dialog box appears.
7. In the **Name** field, enter a name for the filter.
8. Click **Save**.

The **(Unnamed Filter)** dialog box is now named the title you assigned the advanced filter.

9. Click **Hide** which closes the dialog box.

Adding Criteria to an Advanced Filter

To add criteria to an advanced filter:

1. From the **Filters** list, select an advanced filter.
2. Click **Edit Advanced**.

   A filter dialog box appears.
3. Click **+**.
4. Select fields to include in the advanced filter.

   Repeat this step to add additional fields.

   Tip: You can drag column headings from the **Recent calls** table into the dialog box.
5. In the filter dialog box, customize the logic of the filter you choose by entering or selecting logic values to filter for.
6. Click **Save**.
7. Do one of the following:
   - Click **Apply Now** which applies the filter to the **Recent calls** table.
   - Close the dialog box and apply the saved filter at a later time.

Adding Scope to an Advanced Filter

You could create an expression with criteria **a**, **b**, and **c**. You can include brackets in the filter criteria to isolate terms that are dependent on each other, such as:

(a = *2* AND b = *4*) OR c = *10*

or

a = *2* AND (b = *4* OR c = *10*)

To add scope to an advanced filter:

1. From the **Filters** list, select an advanced filter.
2. Click **Edit Advanced**.

   The filter dialog box appears.
3. Click **()** in the row you want to add scope to.
4. Click **+** in the same row and insert filter conditions.

   Note: You can move filter criteria within a dialog box by dragging it to a new area.

   Tip: You can drag criteria in to or out of bracketed statements.
5. Click **Save**.

6. Do one of the following:
   - Click **Apply Now** to apply the filter to the **Recent calls** table.
   - Close the dialog box and apply the saved filter at a later time.

Editing an Advanced Filter

To edit and advanced filter:

1. From the **Recent calls** window, select an advanced filter.
2. Click **Edit Advanced**.

   The filter dialog box appears.
3. Edit the filter criteria.
4. Click **Save**.
5. Do one of the following:
   - Click **Apply Now** to apply the filter to the **Recent calls** table.
   - Close the dialog box and apply the saved filter at a later time.

Applying an Advanced Filter

To apply an advanced filter:

1. From the **Recent calls** window, select an advanced filter.
2. Click **Edit Advanced**.

   A filter dialog box appears.
3. Click **Apply Now** and close the dialog box.

Clearing Advanced Filters

To clear advanced filters:

1. From the **Filters** toolbar, select **Clear**.

Toolbar

The **Recent calls** table has, in order, the actions as listed in Table 4–4.

Table 4–4 Recent Calls Table Actions

Actions	Description
Details	Opens the **Call Details** window for the selected call. See Figure 4–33.
Filters:	Select from the list of advanced filters.
Edit Advanced...	Opens the selected filter for editing.
Save	Saves the changes to the advanced filter.
Clear	Resets any selected filters.
Message flow	Shows the *message flow diagram* for the selected calls. You can select multiple calls in the call grid and show a combined message flow diagram. Use the Ctrl key on Windows/Oracle Linux machines and the Cmd key on Macintosh machines. For more information, see "Working with Message Flows".

Table 4–4 (Cont.) Recent Calls Table Actions

Actions	Description
Bulk export	Exports the current calls grid contents (with the current set of filters applied) as a zip archive. The archive contains one **.csv** file which is similar to the one generated with the normal CSV export (see above), but with an additional column named *pcap_filename*. This references one of the pcap files included in the archive, which contains both SIP and RTP data for the call. If *column filters* are set, they are also applied to the exported CSV file. It is recommended to filter the calls list to the set of interest before exporting the CSV file.
CSV export	Exports the calls data content in CSV format, for post-processing purposes. If *column filters* are set, they are also applied to the exported CSV file. The CSV files are truncated to 50 thousand records. It is recommended to filter the calls list to the set of interest before exporting the CSV file.
View saved calls	Shows the list of already saved calls in Operations Monitor. In order to save a call, the **Save** option from the calls details page must be used.
Show in Dashboard	Adds a reduced version of the **Recent calls** table to the Dashboard.
Auto-Refresher	From the **Refresh** menu, the table can be set to auto-update, in order to display the most recent calls and their state changes in real-time. You can set the table to auto-update from the **Refresh** menu in order to display the most recent calls. The state changes in real-time. For more information, see "Refresh Button". The default auto-refresh interval for this table is 5 seconds.

Right-Click Menu

Right-clicking a row in the **Recent calls** table shows the following contextual options:

- **Track caller and Track callee**

  Choosing one of these two options opens the **User Tracking** page, pre-filled with the caller or the callee user of the selected call. For more information, see "User Tracking".

- **Create trace with**

  Choosing this option opens the **Traces** page, pre-filled with the caller or the callee user of the selected call. For more information, see "Traces".

- **Record media of caller** and **Record media of callee**

  Choosing one of these two options opens the **Record media for a new user** window. For more information, see "Media Recording".

- **Call details window**

  Choosing this option opens the **Call Details Window**. For more information, see "Call Details Window".

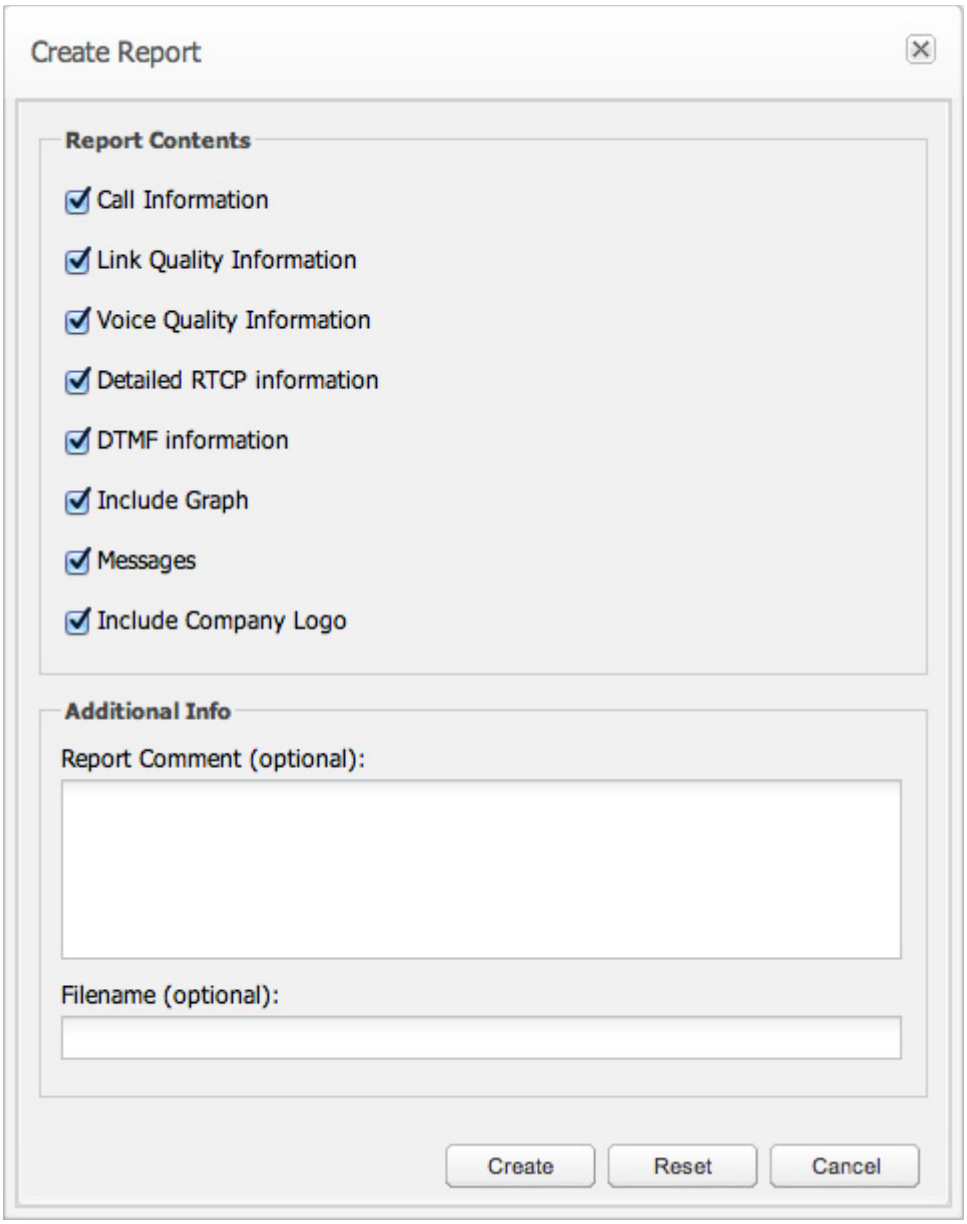
- **Message Flow**

  Shows the *message flow diagram* for the selected call. For more information, see "Working with Message Flows".

- **PDF report**

  Creates a PDF report for the selected call. A dialog appears that allows the user to select the information to include, and to provide a comment to add to the report (see Figure 4–29).

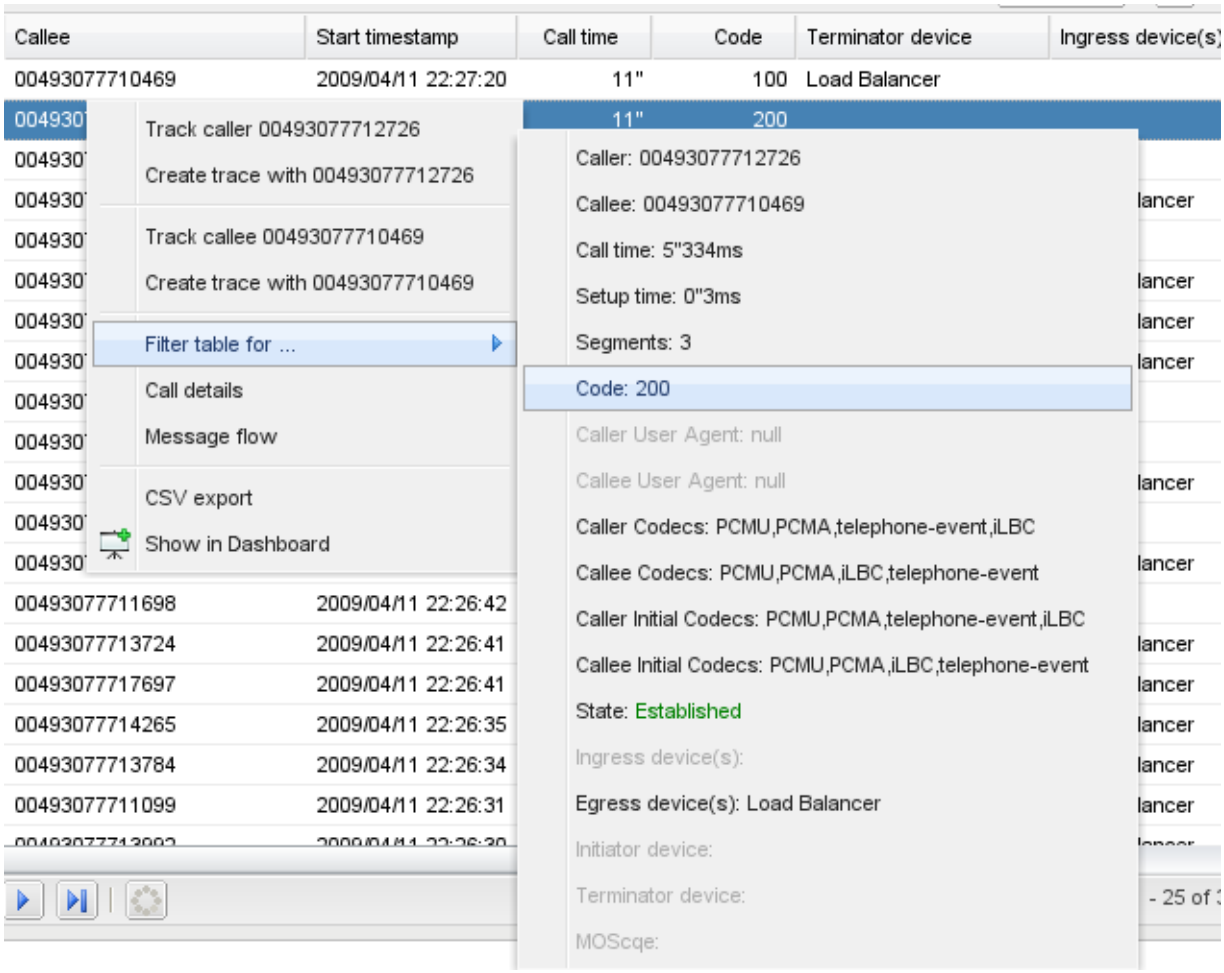
Figure 4–29 Call Report Creation Dialog



- **Filter table for ...**

  Shortcut to *column filters*. This gives a convenient way of finding calls that are similar with the one selected. See Figure 4–30.

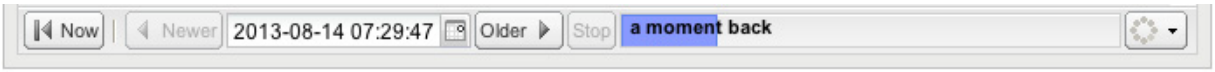
Figure 4–30 Filter Using the Right-Click Menu



Paging

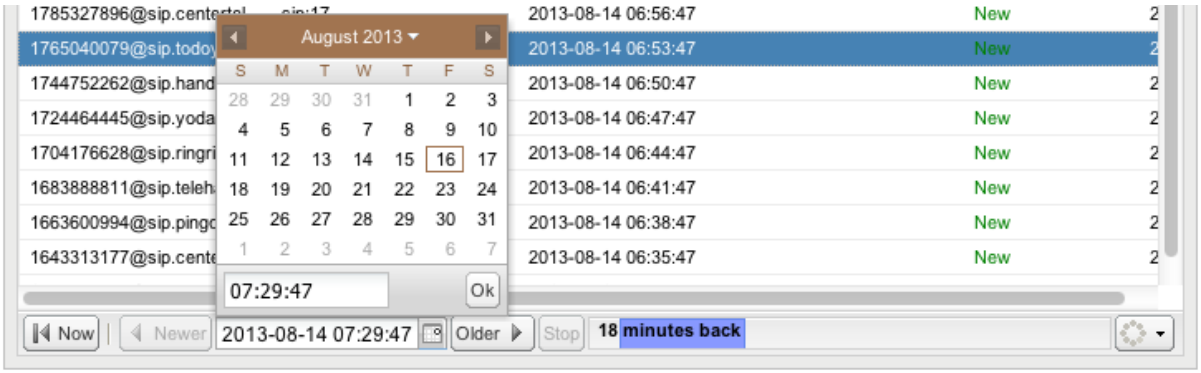
The paging bar allows you to navigate the entries. With the **Newer** and **Older** buttons you can step to the next page of newer respective older entries (see Figure 4–31).

Figure 4–31 The Paging Bar



The **Now** button takes you to the first page, showing the newest entries. The date picker allows you to chose an arbitrary first date and time of the time period you want to view (see Figure 4–32).

Figure 4–32 The Date Picker



The loading bar will show when entries are being fetched. To fill up a page while applying filters, this might require several calls to the server and might take some time. The blue bar in the loading bar shows the range that has been searched.

The refresh button to the right reloads the current entries and shows their updated states.

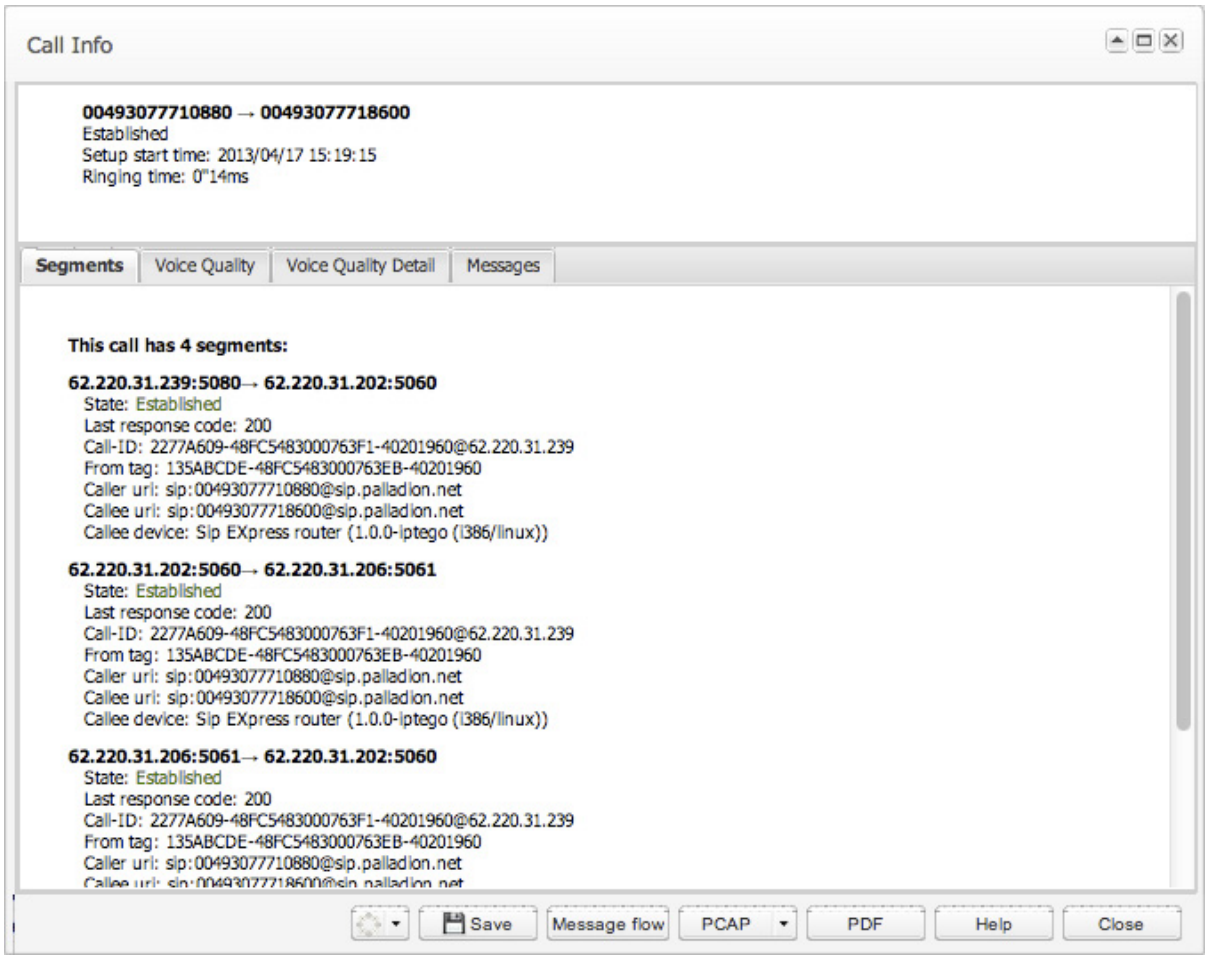
You can also set the refresh button to auto-refresh mode.

Note: The auto-refresh mode in combination with filtering of a grid can lead to longer refresh times than selected.

Call Details Window

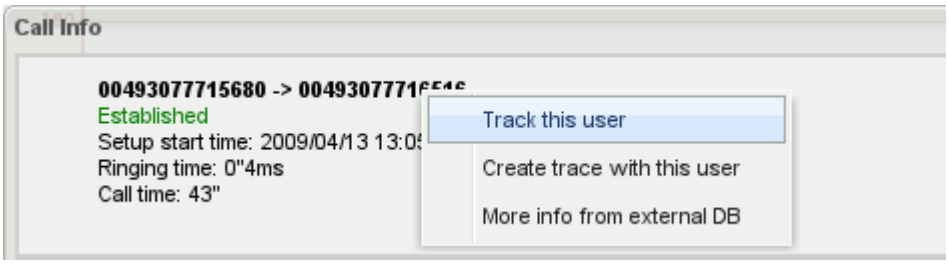
To open the call details panel (see Figure 4–33), double-click on a row from the **Recent Calls** table or click the **Details** button from the toolbar. You can also access the call details panel from the Voice Quality page. For more information, see "Voice Quality".

Figure 4–33 Call Details Window



The top section contains a brief identification of the call: the numbers involved, current state, starting time, ringing, and call duration. Right-clicking on the numbers or user names shows a contextual menu, as shown in Figure 4–34. For more information on these options, see "User Tracking".

Figure 4–34 Right-Click Menu for the Caller and Callee



The bottom section contains four tabs: Segments, Voice Quality, Voice Quality Detail and Messages.

Segments

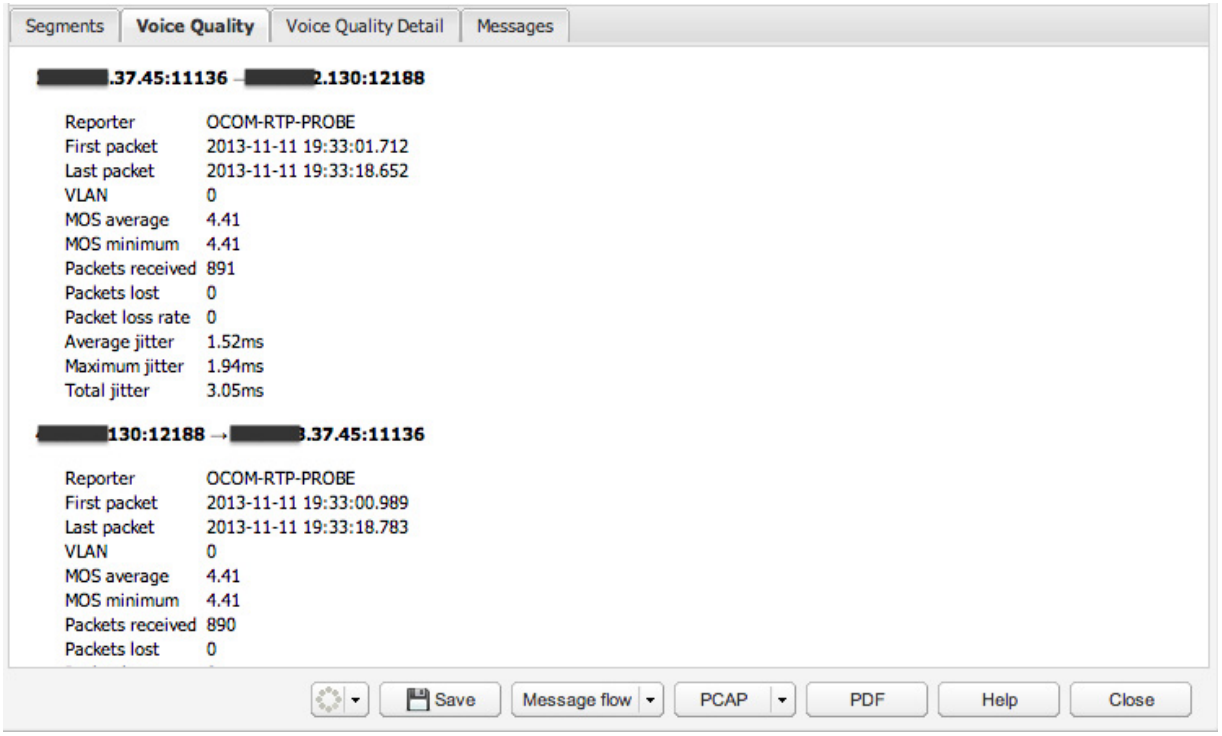
The **Segments** tab (see Figure 4–33) shows details about each call leg within the call, including:

- State, per call leg.
- Call-ID.
- From and To tags.
- Request-URI.
- Caller and callee devices.
- Source and destination devices set with Platform Devices. For more information, see "Platform Devices".

Voice Quality

The **Voice Quality** and **Voice Quality Details** tabs display the voice quality summary of the *finished* calls. It displays source and destination devices when set in Platform Devices. For more information, see "Voice Quality" and "Platform Devices".

Figure 4–35 Call Details Voice Quality Tab



Messages

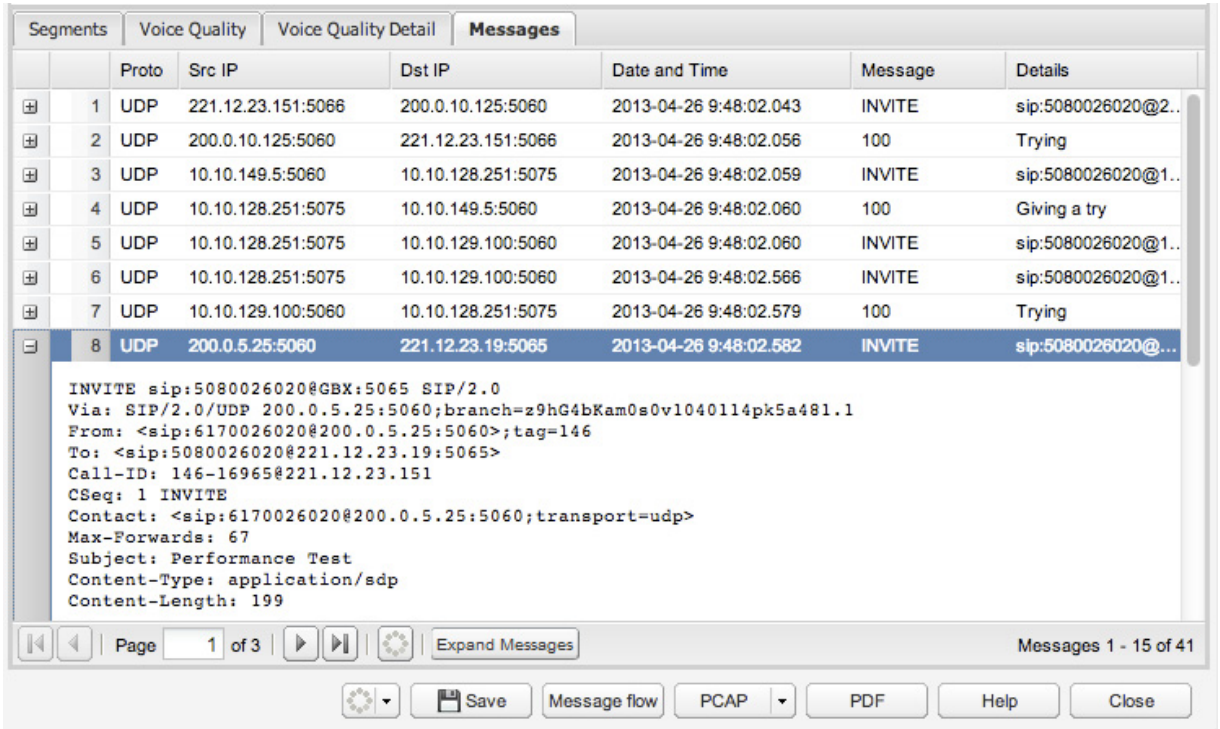
The **Messages** tab shows a table representation of the messages from the call (see Figure 4–36).

Table 4–5 lists the table columns:

Table 4–5 Messages Table

Column	Description
Date and Time	The absolute time when this message was seen by Operations Monitor.
Details	In case of SIP messages, it is equal to Request-URI for SIP requests, or the reason phrase for SIP replies. In case of ISUP messages, it is equal to point codes. In case of ENUM messages, an example might be 8.4.1.1.7.7.9.7.9.7.4.4.e164.arpa. In case of MEGACO messages, an example might be Add or Modify.
Dst IP	The destination IP address and port number.
Dst MAC	The destination hardware address. This column is hidden by default.
Dst PC	The destination PC address. This column is hidden by default.
Message	In case of SIP messages, it is equal to the SIP method for SIP requests, or the response code for SIP replies. In case of ISUP messages, it is equal to one of the ISUP method types: *IAM*, *ACM*, *RLC*, etc. In case of MEGACO messages, it is equal to either *MEGACO Request* or *MEGACO Reply*. In case of ENUM messages, it is equal to either *ENUM Query* or *ENUM Answer*.
Proto	The transport layer protocol. The supported transport protocols are *UDP* and *TCP*.
Src IP	The source IP address and port number.
Src MAC	The source hardware address. This column is *hidden* by default.
Src PC	The source PC address. This column is hidden by default.

Figure 4–36 Call Details Messages Tab



Click the **Expand Messages** button to view the raw messages as seen on the network. The raw messages can be also viewed in context in the **message flow** diagram.

Call Details Toolbar

You can save the call details for debugging later.

Click the **Save** button to save all the call details, including the raw messages and voice quality information.

Click the **PCAP** button to download the raw signaling messages and the media streams of a call into PCAP file format. See "Downloading Call Details to a PCAP File" for more information.

Click the **PDF** button to create a PDF report for the call with details about segments, raw messages and voice quality information.

Click the **Message flow** button to open the **message flow** diagram.

Downloading Call Details to a PCAP File

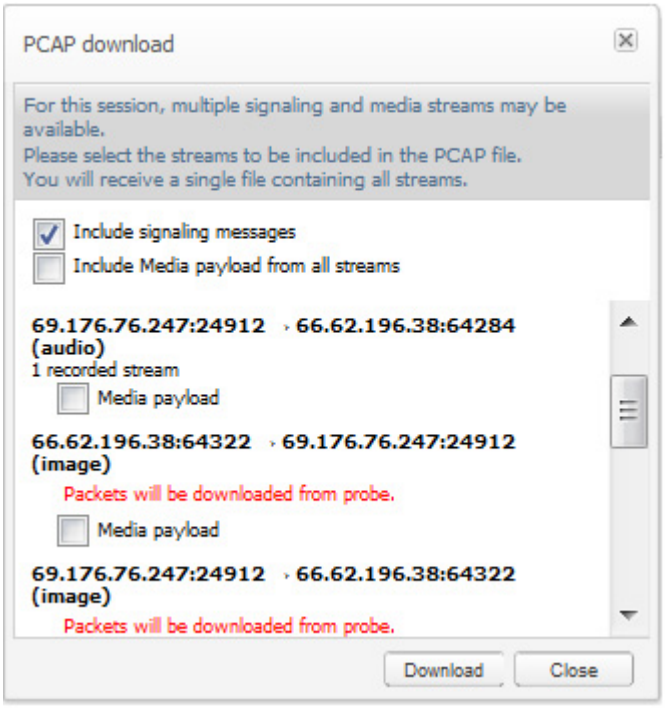
You can save the raw signaling messages and the media streams of a call to a PCAP file. When a call has multiple media streams, a single PCAP file is created containing all the streams.

To save the call details to a PCAP file:

1. In the call details panel, click the **PCAP** button.

   The PCAP download dialog box appears, as shown in Figure 4–37.

Figure 4–37 PCAP download Dialog Box



   If the media stream is not already saved on the Operations Monitor system, it is downloaded from the Probe.

Note: If the media packets are no longer available on the Probe, the PCAP file will not contain the packets of those streams.

2. Select the media streams you want to save. The content available for downloading depends on your user permissions.

Note: Downloading media streams from the Probe generally takes more time than downloading from the Operations Monitor system.

3. Click the **Download** button.

   The Export status dialog box appears, which contains a link to the PCAP file that is generated for the call.

4. Click the PCAP file link.

5. Select the option to save the file, and click **OK**.

Device Visibility in Realms

Device visibility in a realm works as a whitelist for visible devices and a blacklist for hidden devices. By default (if no visibility is set for device), users can see all information for every device.

If the user belongs to a realm which has limited device visibility (only some devices are visible to the realm):

- Information about call segments between visible devices are shown.
- Information about call segments between visible devices and hidden devices are not shown.
- Information about call segments between visible devices and unknown hosts are shown. An unknown host is a host that does not belong to any device. To hide this information from the realm, create a new device for the respective unknown host and set it as hidden for this realm.

This behavior affects the call information on the **Segments** and **Messages** tabs, in the message flow diagram, PCAP, and PDF generation.

Voice Quality

Under **Operations** of the main menu, the **Voice Quality** page allows you to analyze the transmission effects of the network on the voice quality as perceived by the users of the VoIP platform. It provides a graphical overview for both platform-wide calls and device specific calls.

Voice Quality Source

Operations Monitor gathers voice quality information from four sources:

- Operations Monitor RTP Probes
- User Agent Statistics
- SIP Voice Quality Events (application/vq-rtcpxr)
- Other Voice Quality Monitors

Operations Monitor RTP Probes

Probes monitor the traffic in passive mode and record several measures for each RTP stream:

- The number of lost packets.
- The variance of delays between the received packets (*jitter*).
- The distribution of lost packets inside the stream.
- The *codec* used.

Based on these values, Operations Monitor uses the ITU E-Model to estimate the MOS.

The following restrictions apply:

- MOS is calculated for blocks of 10 seconds (chunks), if a chunk contains more than eight seconds of RTP data and contains only packets of a single supported codec.
- No MOS is calculated if a chunk contains multiple payload types other than the telephone- event (DTMF). That is a change in audio codec or comfort noise packets will prevent MOS computation.

Supported codecs for MOS calculation:

- AMR-NB
- AMR-WB
- G.711 A-law (PCMA)
- G.711 u-law (PCMU)
- G.723.1
- G.726
- G.728
- G.729
- iLBC
- OPUS
- Speex

User Agent Statistics

Some SIP devices have the capability to include RTP statistics in a BYE message header, or in the reply to the BYE message. In general, the information provided is not enough to estimate the MOS, with the exception of the *'X-RTP-Stats' version 3* header.

Table 4–6 User Agent Headers Providing RTP Stats

Header Name	Devices	Data Provided
X-RTP-Stats (v2)	AVM	packet loss, jitter, burst lost, codecs.
X-RTP-Stats (v3)	AVM	packet loss, jitter, burst lost, gaps, codecs.
P-RTP-Stats	Cisco, Linksys, Sipura	packet loss, end-to-end latency, burst lost, codecs.
RTP-RxStats	Snom	packet loss.

SIP Voice Quality Events (application/vq-rtcpxr)

Some SIP devices (like: Polycom and Snom) are able to send voice quality information derived from RTCP-XR messages to Operations Monitor's voice quality collector,

configurable under *VQ Collector Configuration*. Operations Monitor gets the values of *jitter*, *packet loss*, *burst loss*, and *MOS* from the voice quality collector to display them under the Call details info page.

Operations Monitor only reports data for end of stream statistics (*CallTerm* marker, refer to RFC 6035).

RTCP reports may be generated by endpoints. Such data may not be present or may not be trustworthy. Due to this we do not consider RTCP quality information for our own voice quality metrics. Nonetheless we display RTCP information to complete the view on the voice quality experienced by the users.

Other Voice Quality Monitors

Oracle Communications SBCs can be configured to forward Voice Quality data to Operations Monitor. Operations Monitor relays this information through its intuitive UI.

Voice Quality Chart

The **Voice Quality** page shows a stacked area chart. Five series are displayed, each based the levels of user satisfaction as specified by ITU recommendation G.107.

- Very satisfied: MOS >= 4.34
- Satisfied: 4.03 <= MOS <= 4.33
- Some users dissatisfied: 3.60 <= MOS <= 4.02
- Many users dissatisfied: 3.10 <= MOS <= 3.59
- Nearly all users dissatisfied: MOS <= 3.09

We analyze the quality of a media stream over a time interval of either one or 15 minutes (determined by the **Voice Quality Chart Scale in Minutes/Minute** system setting) and determine how many minutes and seconds were spent in each of the above MOS ranges. For more information, see "Voice Quality Chart Scale in Minutes/Minute".

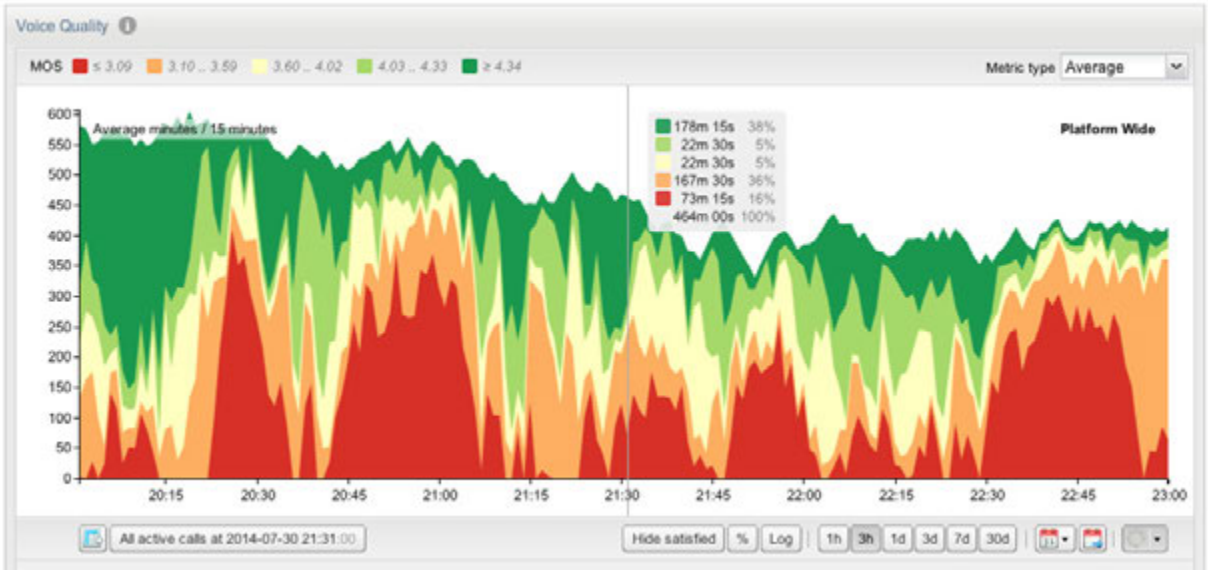
Over a minute a stream may have had 40 seconds with a MOS of >= 4.34, 10s between 4.34 and 4.03, and another 10s between 3.60 and 3.10. If using a 15 minute interval, the values are normalized accordingly to project the average minute and seconds over a 15 minute time period.

In this case, with 40 seconds over one minute of MOS of >= 4.34 is 00:40 minutes, but normalized over a 15 minute interval it is 10:00 minutes.

We sum up all the minutes of MOS >= 4.34 across all active media streams. This determines the height of the area labeled >= *4.34* at T=12:01.

The chart's resolution and range can be controlled with the toolbar directly below it (see Figure 4–38).

Figure 4–38 Voice Quality Overview



The toolbar offers couple of more options:

- Display the active calls during a period of bad voice quality.
- Hide statistics about streams with satisfactory quality (do not show the two best categories).
- Switch to a percentage scale.
- Switch to a logarithmic scale.

Additionally, you can select metric with the drop-down in the upper-right corner of the diagram. The metrics are **Average** (default) and **Maximum**.

The reason for using the **Maximum** metric is that for older data only hour resolution is available. A peak that happened during a couple of minutes will then be flattened out when calculating the average over one hour. By preserving the maximum value over a time period, the peaks can still be detected in old data.

When aggregating maximum values from minute to hour resolution, the sum of minutes of all voice quality levels will most likely be greater (at least equal) than that of the minute resolution. This is because the maximum value for each voice quality level over 60 minutes will be chosen.

Note: Only voice quality measurements calculated by the Operations Monitor probes are shown in this chart. This monitoring has certain limitations:

- Voice quality (MOS) is calculated for blocks of 10 seconds (chunks), if a chunk contains more than eight seconds of RTP data of a single supported codec. For more information, see "Operations Monitor RTP Probes".
- Chunks shorter than one second or with less than 50 packets are normally not shown.
- Streams containing larger gaps (that is due to comfort noise) might be shown as two separate streams.
- Streams that are not matched to calls (no correlation between SDP record received in a SIP message and RTP IP/ports) are not included in Operations Monitor's KPIs.

CSV Export

You can export voice quality data for the displayed period by pressing the CSV export button. The CSV export dialog will open, and when ready provide you with a downloadable CSV file.

Table 4–7 lists the fields that are exported:

Table 4–7 CSV Export Fields

Name	Format	Example
Local timestamps	human readable	2014-07-25 13:42:00
UTC timestamps	ISO 8601	2014-07-25T11:42:00Z
Interval	integer	60
MOS: 0 .. 3.09	integer	300
MOS: 3.10 .. 3.59	integer	329
MOS: 3.60 .. 4.02	integer	28
MOS: 4.03 .. 4.33	integer	255
MOS: 4.34 .. 5	integer	212

The MOS fields contain the number of seconds of each voice quality level, for the time period given in seconds in the **Interval** field. The interval depends on the available resolution for the time period: either minute or hours. If minute resolution values are available, Interval is typically 60, otherwise 3600. The Interval can in rare cases be less than 60 or 3600, if the machine was rebooted or was down when the measurement period started.

Minute values are only available for the last couple of days.

Active Calls During a Period of Bad Quality

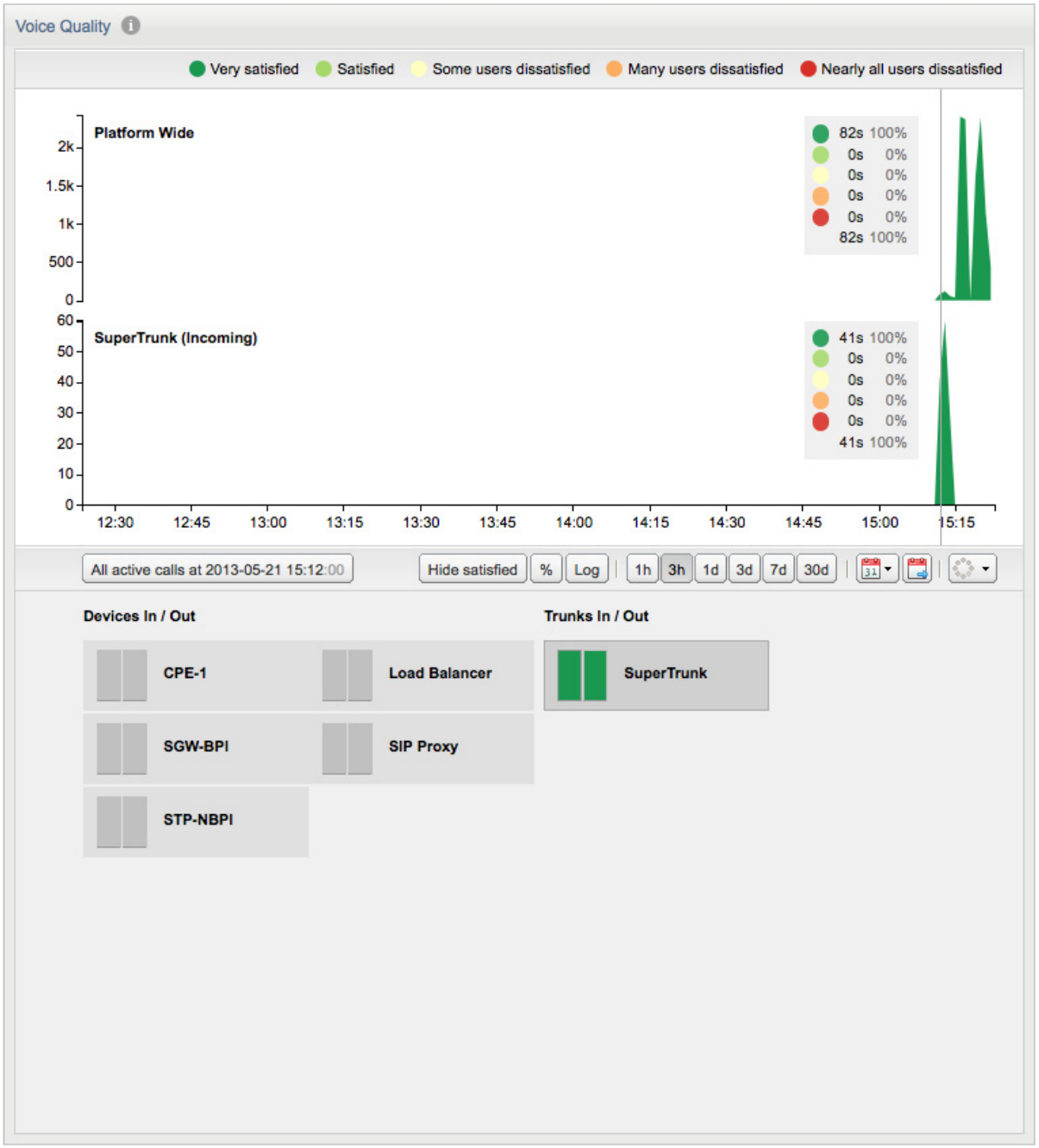
Selecting a time on the main chart will update the **Active calls** button below the chart. This button navigates to the calls page, appropriately filtered to display only calls active at that time.

Device Voice Quality Charts

To help diagnose areas of particularly bad MOS, clicking on the main chart at time T loads the Voice Quality data for all platform devices at that time. The MOS stats for a device are displayed next to it. The devices are sorted by MOS quality, those with bad quality at the selected time are listed first.

Clicking on a device augments the main chart with a second chart displaying the Voice Quality for the currently active time resolution and range (see Figure 4–39).

Figure 4–39 Voice Quality per Device



Media Summary

You can see the media summary information of any call listed in the **Calls** table.

To view the media summary details:

1. Double-click a row in the **Calls Table**.

   The **Call Info** window opens.

2. Click the **Media Summary** tab which opens a textual report of the voice quality data for the call.

Table 4–8 describes the fields in the **Media Summary** tab.

Table 4–8 Media Summary Fields

Field	Description
Average jitter	The average of the measured jitter values of the 10 seconds intervals
Average MOS	The average of the estimated MOS values of the 10 seconds intervals
First packet	The time stamp of the first RTP packet from the stream
Last packet	The time stamp of the last RTP packet from the stream
Maximum jitter	The maximum of the measured jitter values of the 10 seconds intervals
Minimum MOS	The minimum of the estimated MOS values of the 10 seconds intervals
Packet loss rate	The number of received RTP packets divided by the number of expected RTP packets
Packets lost	The total number of RTP packets that were lost for this stream. This can have multiple causes: ■ Packets were dropped by the network ■ Packets were dropped at sniffing time (visible in PSA Media Signaling page) ■ Packets were dropped by the internal jitter buffer used for computing Voice Quality stats
Packets received	The total number of RTP packets that were received in the stream
Reporter	The source of the report: ■ OCOM-RTP-PROBE: Operations Monitor probe ■ *VQ-RTCPXR*: Voice quality reports received in SIP PUBLISH messages ■ *SBC-RTP*: Data based on end-of-call media QoS reports from an SBC ■ *SBC-INTERIM-RTP:* Data based on interim media QoS reports from an SBC ■ *SBC-RTCP*: Data based on an RTCP summary from an SBC ■ *P-RTP-STAT, X-RTP-STAT, RTP-RXSTAT*: Data from one of the supported SIP headers that carry VQ information
Total jitter	The sum of the measured jitter values of the 10 seconds intervals
VLAN	The VLAN tag of the RTP packets

Media Details

The **Media Details** tab displays an estimation of the MOS value for each RTP stream. This data is collected by Operations Monitor RTP probes or may be provided by SBCs in interim media QoS reports. The data is arranged on a grid in 10 second intervals.

You can see the media details grid for any call listed in the **Calls** table.

To view the media details grid:

1. Double-click a row in the **Calls** table.

The **Call Info** window opens.

2. Click the **Media Details** tab which opens a grid report of the media details for the call.
3. Hover over any block on the grid; a tooltip displays data for the interval.

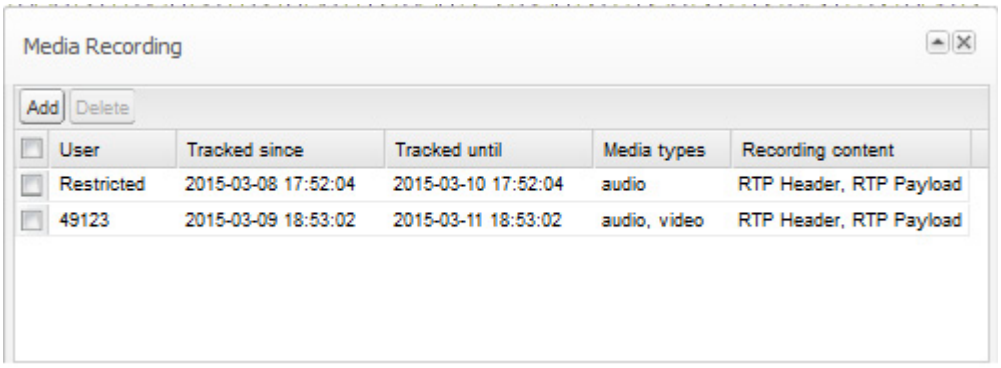
> **Note:** Voice quality report intervals may align differently on the grid, depending on the data source. Intervals reported by Session Monitor probes are aligned to a global grid at 10-second boundaries. Intervals reported by SBCs may not be aligned to this grid because they are provided every 10 seconds after the stream began.

Media Recording

The Media Recording feature allows you to capture RTP (including DTMF) and RTCP streams of calls, in addition to the signaling messages, to track particular user's calls. You can then download the RTP and RTCP streams to a PCAP file for further analysis in protocol analyzer tools. This is helpful for analyzing voice quality issues and for complying with regulations.

Figure 4–40 illustrates the **Media Recording** table, which lists the users that are currently tracked.

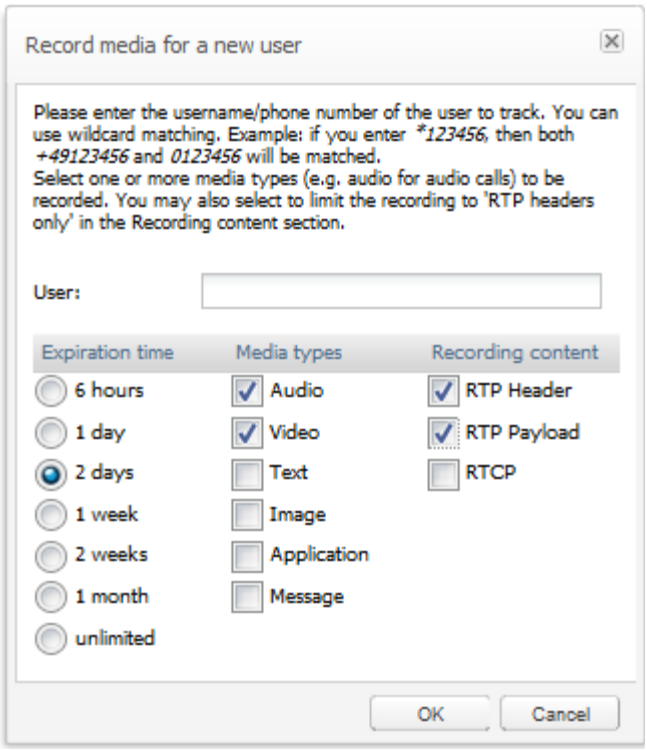
Figure 4–40 Users Tracked in Media Recording



To start capturing RTP and RTCP streams, you set up recording rules. A recording rule specifies the subscriber to track, how long to track the subscriber's calls, the media types to record, and the contents to record.

Figure 4–41 shows the dialog box for specifying a media-recording rule.

Figure 4–41 Media Recording Rule Configuration



Media Recording allows you to record calls with multiple media types, for example, a call containing audio and video or audio and image.

> **Note:** To record audio using Media Recording, you must enable the **RTP Recording** feature on both the Mediation Engine and the Probe. To record all other media types, including video and image, you must enable the **Packet Inspector extension** on both the Mediation Engine and Probe. All media types will display even when **Packet Inspector extension** is not enabled. Packet Inspector is not required for recording audio. See *Session Monitor Installation Guide* for more information about enabling these features.

The content that is available for recording depends on the user permissions. For example, the user may have permissions to download RTP header but not RTP payload.

When the recording content is restricted to RTP Headers only, the system records the following:

- RTP headers for RTP traffic (audio, video, text, T.38 over RTP)
- UDP headers plus the first two bytes (sequence number) for T.38 over UDPTL
- TCP headers only for TCP-based media

Operations Monitor saves the RTP and RTCP packets for all calls made or received by the subscribers that are tracked in Media Recording. For example, if the expiration time is **2 days**, then for two days the system tracks all calls made or received by the subscriber and records the calls that match the rule.

The **Tracked until** column in the **Media Recording** table shows the expiration time. After the expiration time, the recording rule is deleted. This prevents the recording rules list from becoming too long.

All users with the same realm assigned can share a recording job. Users in the realm **ALL** can view all the recordings in the system.

Recording RTP Streams

To start recording the RTP streams for a subscriber:

1. In the **Recent calls** panel of the **Calls** page, click the speaker icon in the menu bar:

   The **Media Recording** table appears. The table shows the subscribers that are currently tracked.

2. Click the **Add** button.

   The Record media for a new user dialog box appears.

3. In the **User** field, enter the phone number or the SIP user name to track.

   Operations Monitor records for all the SIP user names that match the given pattern as a suffix. This is useful when the same subscriber uses various prefixes. For example, if a subscriber uses both *0123456* and *+49123456* phone numbers, enter *0123456*, so that both numbers are matched.

4. In the **Expiration time** section, select the time interval until when the subscriber is tracked.

5. In the **Media types** section, select one or more media types to record.

6. In the **Recording content** section, select one or more content to record.

   The content available for recording depends on the user permissions.

7. Click **OK**.

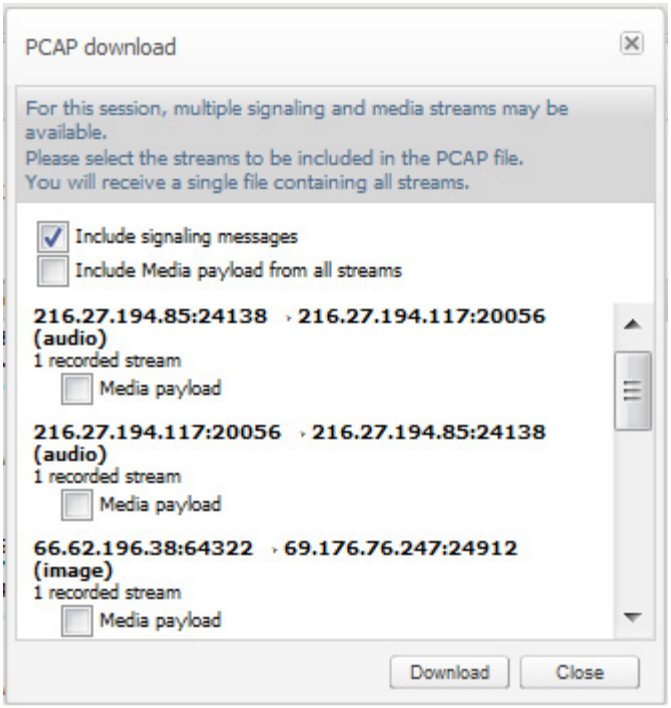
To stop the recording for a particular subscriber, select the user in the **Media Recording** table and click **Delete**.

Downloading Recorded RTP Streams

RTP streams are downloaded in the industry-standard PCAP file format. When a call has multiple media streams, you can select the media streams that you want to download. The content available for downloading depends on the user permissions. A single PCAP file is created containing all the streams.

Figure 4–42 shows a call with multiple signaling and media streams.

Figure 4–42 Download the Signaling and Recorded Media Streams



To download the recorded RTP streams for a call:

1. In the **Recent calls** panel on the **Calls** page, double-click a call that is made or received by the tracked user.

   The Call Info dialog box appears.

2. In the Call Info dialog box, click the **PCAP** button.

   The PCAP download dialog box appears.

3. Select the recorded streams you want to save.

4. Click the **Download** button.

   The Export status dialog box appears, which contains a link to the PCAP file that is generated for the call.

5. Click the PCAP file link.

6. Select the option to save the file, and click **OK**.

Registrations

The **Registrations** page allows you to examine the registration events detected on the monitored platform. Examples of these events are: a SIP device tries to publish itself as an on-line entity willing to engage itself in communication with other devices, changes this status to off-line, or queries a central authority about this status. Registration events are generated using the SIP protocol method 'REGISTER'.

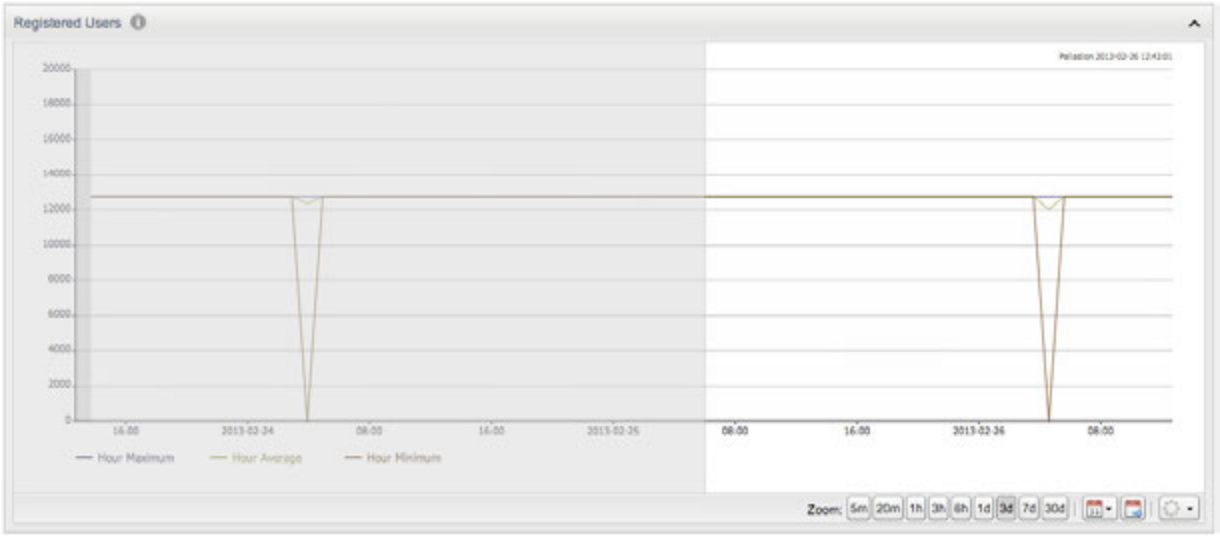
The **Registrations** page contains two panels: the first is the **Registered Users** panel, which provides an overview of the number of registered devices and the fluctuation over time, and the **Registrations Table**, which groups the events for one subscriber (one AOR in SIP terminology) and gives more details.

Registered Users Panel

The top panel contains a chart that traces the total number of registered users over time. The sampling interval is 1 minute for drawing the sample's minimum, maximum, and average values.

By default, a window of three hours is presented. The time frame can be adjusted and the chart allows for panning. For more information on using charts, see "Charts".

Figure 4–43 Registered Users Panel While Dragging Time Window



Note: If Operations Monitor has no registration information over a period of time (due to factors like recent start up, downtime, or lack of traffic), that interval is indicated in the chart by a gray area (as seen on the left side of Figure 4–43).

Registrations Table

The **Registrations** panel contains a table of the registrations events, grouped by SIP user. The list is ordered chronologically, presenting the latest event as the first entry by default. The table can be resorted and/or filtered. For more information, see "Filtering".

Note: If the system setting *Group New Registrations from the Same User* is true, duplicate registration events in state 'New' from the same users are grouped together. However, since the grouping is done on each requested chunk, duplication removal over different chunks does not work.

Table 4–9 lists the column descriptions for the **Registrations** table:

Table 4–9 Registration Table

Column	Description
User	Contains the user part of the AOR; if the system setting **Use users domains** is enabled, the domain part is also included in this field. When specifying a filtering token, all users that contain that token (position independent) will be shown.
Contacts	Filtering is done by matching a given string. All the contacts provided by the registering device are listed here, each as a SIP URI.
Source IP address	The IP address of the network device that initiated the registration event. Filtering is done by matching a given string.
Destination IP address	The IP address of the network server targeted by the registering device. Filtering is done by matching a given string.
Timestamp	The moment in time when the event occurred. Filter by specifying an interval (between *Before* and *After*) or a Date (the *On* date selector).
Device	The device configured in *Settings/Platform Devices* that was targeted by the subscriber's device when registering. When trying to filter by a certain device, a list of configured devices is presented out of which one or more can be selected.
Event	The type of registration event; the table below is detailed on each defined type. For filtering one can choose to filter the presented list to only include some of the possible events.
Code	The SIP code that the registration message exchange finished with. Filter by specifying any valid SIP reply code.

Registration Event Categories

Registration events are categorized as follows:

- **Failed**

  Final response code is bigger than 299.

- **Unauthorized**

  Two consecutive transactions are answered with 401, or when one 401 transaction is not followed by another REGISTER request in the next 5 seconds.

- **New**

  Anew contact binding was created for this user.

- **Gone**

  Another registration request was answered by the registrar, and the binding was not found in the response.

- **Expired**

  A binding is considered expired when the time interval specified in the 'expires' parameter from the registrar's answer is over. No messages are associated with this type of event.

Registrations Table Actions

The **Registrations** table provides a few action buttons.

- Click the **Registration details** button when a row is selected in the table, and a window appears containing *more details* for individual registration events; this window can be also opened by simply double-clicking on the selected row in the table.
- Click the **Message flow** button to open a **SIP message flow diagram** window that depicts the network entities involved and message exchange of the registration event.
- The **CSV export** button allows to have all table entries exported into a CSV file. If the contents of the table have been filtered, the contents of the exported CSV file will only contain the filtered results.

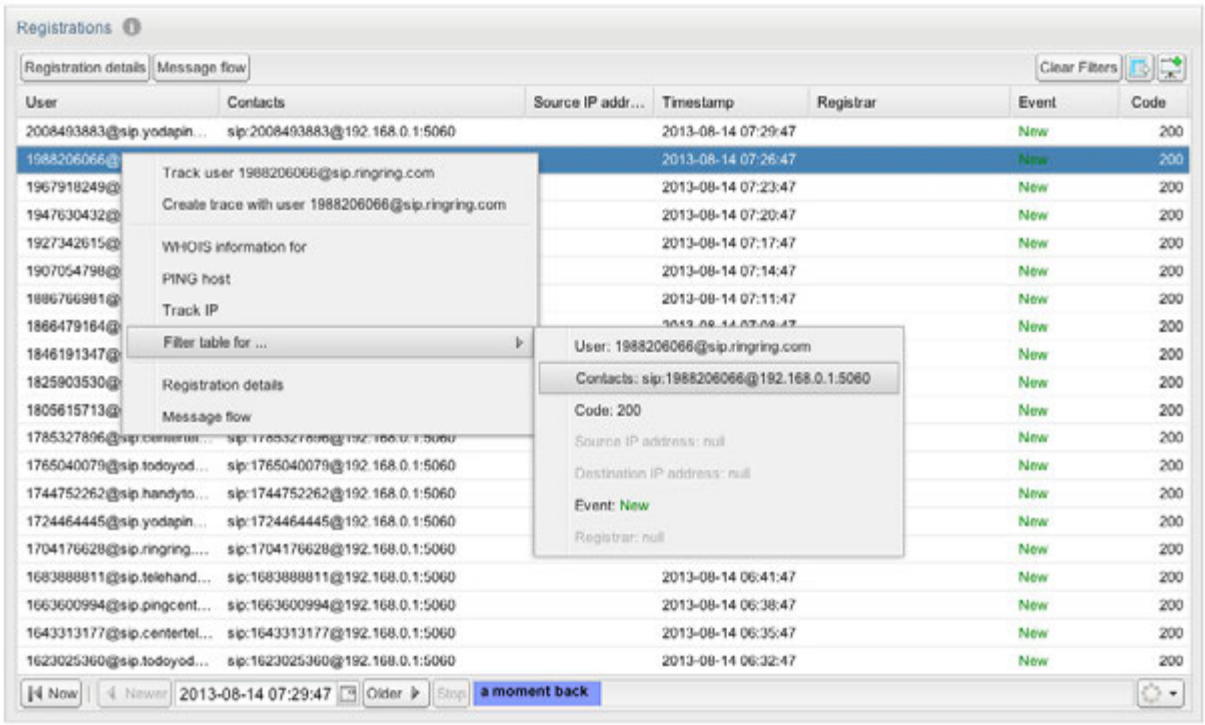
> **Note:** Many of these actions are available by right-clicking (see Table 4–44).

Right-Click Contextual Menu

Right-click menu actions are as follows:

- *Filter table for ...* allows you to filter for:
  - Other registration events of the same user or contact.
  - The same code or event type.
  - All registrations involving the same address or device.
- *WHOIS information for ...* allows you to inquire who is the owner of the IP used as the source of the registration. A window appears that contains the record stored into the WHOIS database for the given address.
- *Create trace with user ...* loads the **Traces** page with the **User** field pre-filled with the value taken from the corresponding registration event.
- *Track user ...* loads the **User Tracking** page with the **User** field pre-filled in the search panel taken from the event.

Figure 4–44 Registrations Table



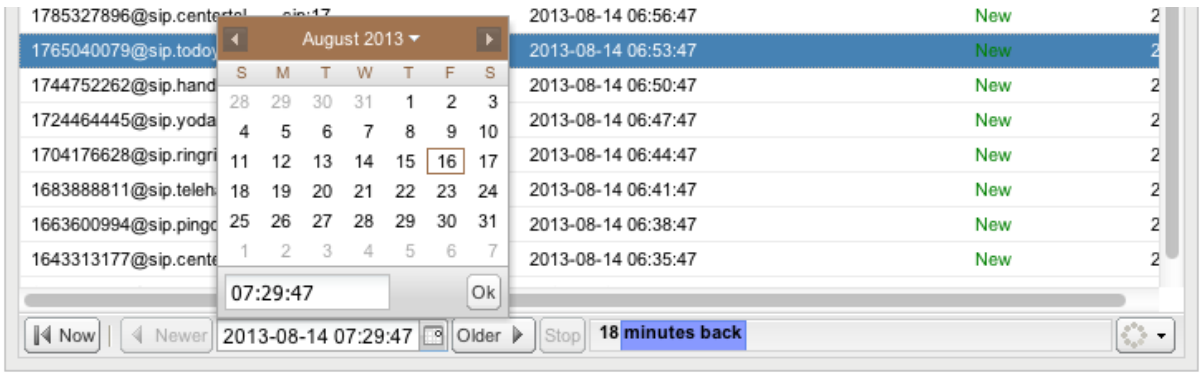
Paging

The paging bar allows you to navigate the entries. With the **Newer** and **Older** buttons you can step to the next page of newer respective older entries (see Figure 4–45).

Figure 4–45 The Paging Bar

The **Now** button takes you to the first page, showing the newest entries. The date picker (see Figure 4–46) allows you to choose an arbitrary first date and time period.

Figure 4–46 The Date Picker



The loading bar will show when entries are being fetched. To fill up a page while applying filters, this might require several calls to the server and might take some time. The blue bar in the loading bar shows the range that has been searched.

The **Refresh** button to the right reloads the current entries and shows their updated states.

You can also set the **Refresh** button to auto-refresh mode. Please note that auto-refresh mode in combination with filtering of a grid can lead to longer refresh times than selected.

Registration Details

The details window for a registration event contains the following information:

- The user of full AOR (subjected to settings).
- The timestamp when the event occurred.
- The network entity of the platform having received the registration request (the ingress device).
- The IP address used to originate the requests.
- The type of registration event.
- A table that contains details about the exchanged messages in this event, listed in chronological order.

Table 4–10 lists the columns in Registration Details window.

Table 4–10 Registration Details Window

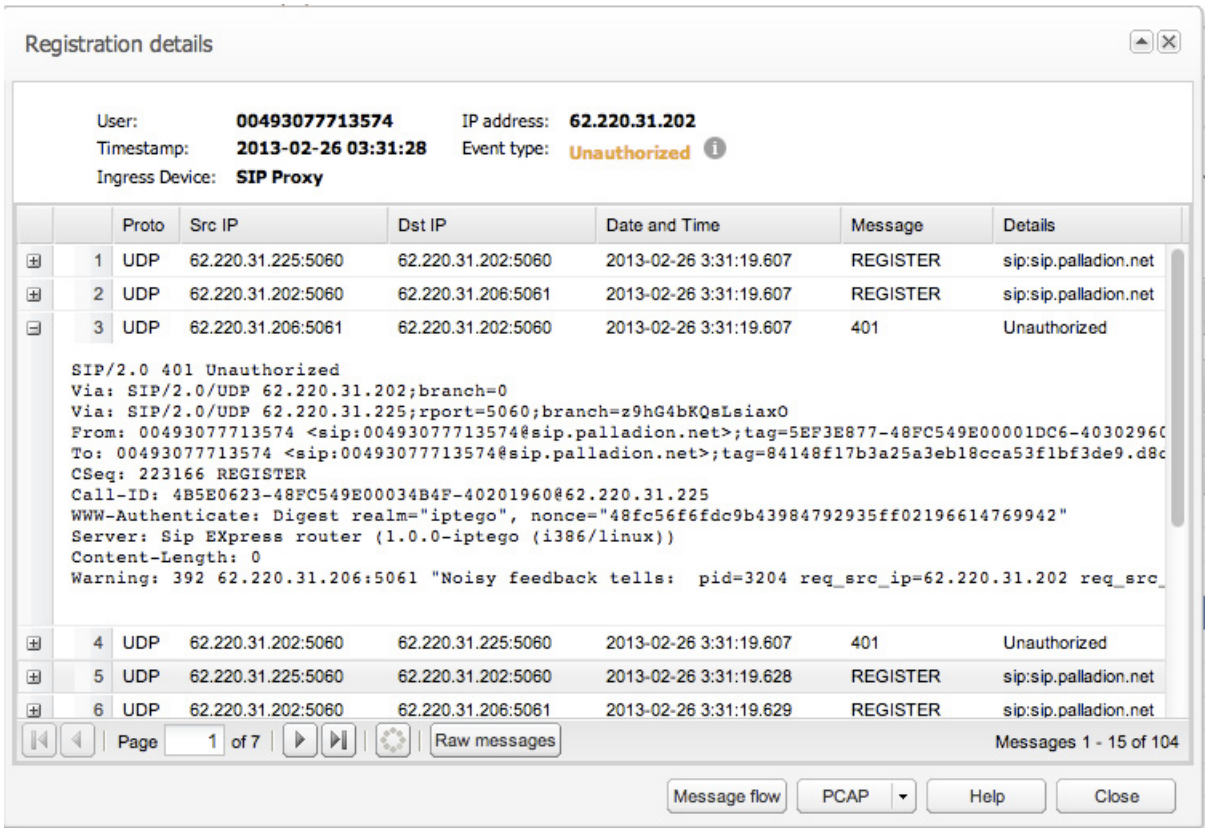
Column	Description
Proto	The network transport protocol used to carry the SIP messages from client to the platform. This field can be either UDP or TCP.
Source	The IP address of the device that generated the message; can be either the client's or one of platform boxes, depending on the direction.
Source Hwaddr	The *Ethernet* address of the device that relayed the network packet just before being mirrored to Operations Monitor.
Destination	The IP address of the device that received the message; just as with *Source*, it can be either the client's or platform's.
Destination Hwaddr	The *Ethernet* address of the device that received the network packet just before being mirrored to Operations Monitor.
Date and Time	The moment in time when the mirrored network packet was received by Operations Monitor.
Method/Code	The type of SIP request or code returned in reply.
RURI/Reason	The SIP Request-URI the request was targeted to (for requests) or the reason phrase (contained in replies).

Registration Details Actions

Click the **Raw messages** button to view the raw SIP message as seen on the network. You can click the button again to display brief messages.

Note: You can view one raw message at a time by double-clicking on any of the table's entries, as shown in Figure 4–47.

Figure 4–47 Registration Details Window with One Expanded Raw Message



Beside the **Help** and **Close** buttons, the details window contains a **Message flow** button and a **PCAP** button. The **Message flow** button creates a *SIP message flow diagram*, and the **PCAP** button allows you to save the shown network packets into a PCAP file.

User Devices

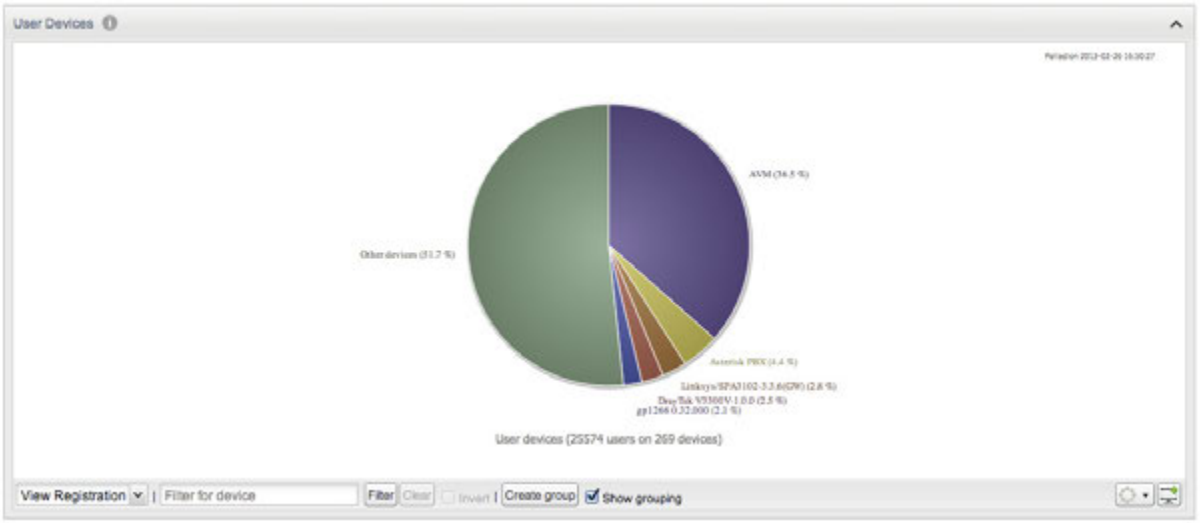
Operations Monitor provides statistics for user agents (user devices) registered on the monitored platform. These statistics are based on the **User-Agent** header of the REGISTER request. Alternatively, Operations Monitor can also evaluate the **User-Agent** headers of INVITE requests and replies in order to provide statistics about the callers and callees of the user device.

The **User Devices** page is located under the **Voice/Video Operation** section of the main menu. The **User Devices** panel displays a pie chart of the found user agents. The **Devices List** table contains detailed information about each user device in the lower area.

User Devices Chart

The **User Devices** chart displays the relative frequency of user agents as shown in Figure 4–48.

Figure 4–48 User Devices Chart



Operations on User Devices

You can filter and group user devices with the following actions in the **User Devices** chart:

- **Select data source**

  You can select whether the statistics displayed on the **User Devices** page is based on registrations, callers, or callees. To choose a display, select one of *View Registrations*, *View Callers*, and *View Callees* from the drop-down menu to the left below the chart. The information stored for callers and callees is based on the list of calls that Operations Monitor keeps in memory.

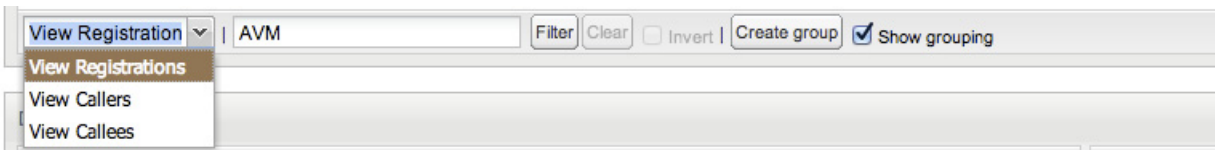
- **Restrict the User Devices page**

  You can specify the statistics on the **User Devices** page to match or not match a certain regular expression. Enter a regular expression into the text field to the left of the filter button below the chart. Click **Filter**. If you want to invert the match, mark the **Invert** check box below the chart.

- **Group devices**

  To create a new group of user devices, enter a regular expression in the **Filter for device** field and click the **Create group** button. The new group rule is applied to the **Device List** panel (below the chart). If the **Show grouping** check box below the chart is marked, the groupings are also applied to the pie chart (see Figure 4–49).

Figure 4–49 Create a Group



Note: When you create filter or group user devices, this is applied to the **User Devices** chart and the **Devices** list.

The **User Devices** chart also contains the **Show in Dashboard** icon button in the bottom-right corner. Click this button to add the panel, with the current filter settings applied, to the dashboard. For more information, see "Dashboard".

Devices List Panel

The **Devices List** panel (see Figure 4–50) contains two tables. The left table lists the devices by group, and the right table displays callers and callees for any device that is selected in the devices table.

Figure 4–50 Devices List Panel

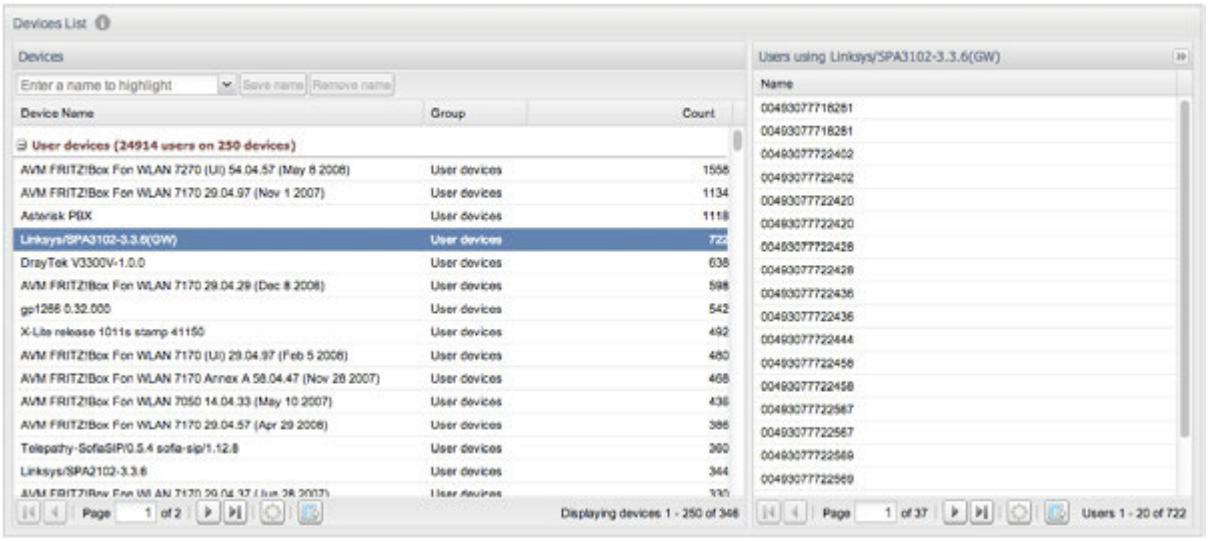


Table 4–11 lists the user devices table columns:

Table 4–11 User Device Table

Column	Description
Device name	Displays the full user agent string for each device.
Group	Indicates the group to which the device belongs.
Count	Shows the number of users per device.

Right-Click Menu

Right clicking on any of the user devices provides the following options (see Figure 4–51):

- **Delete a group**

  Removes a group definition from the system. This only effects the **User Devices** page.

- **Highlight a group**

  Highlights all the devices from a group.

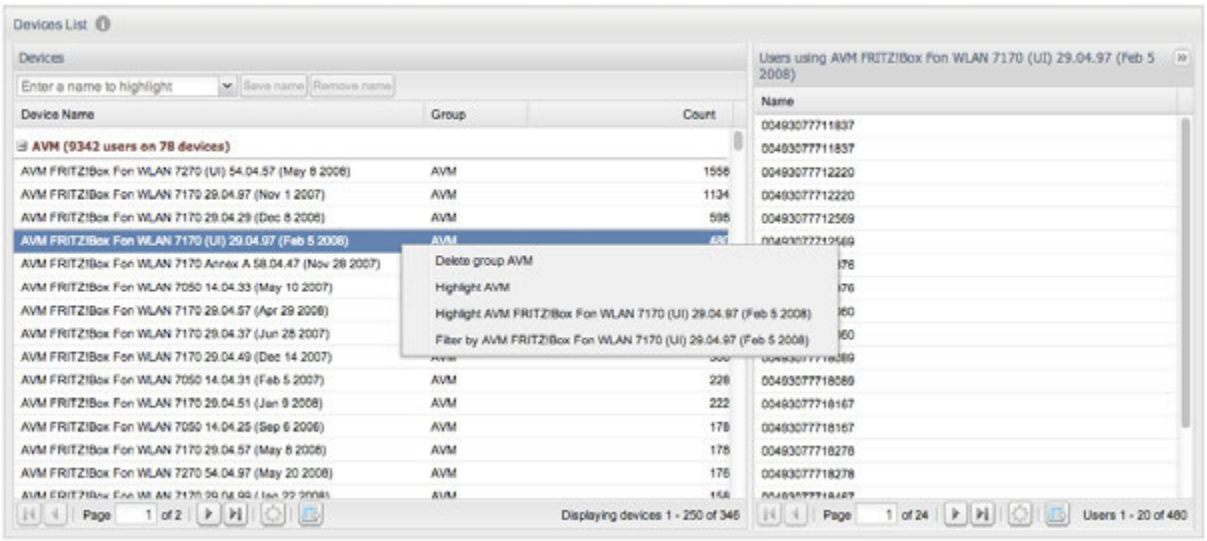
- **Highlight a user device name**

  Highlights all the user devices with a specific name.

- **Filter by a user device name**

Filters the statistics displayed on the page by the selected user device name.

Figure 4–51 Right-Click Menu



Users Using a Specific User Device

The right-side table only contains the **Name** column. An identifier is displayed for each user. If you right-click on a row in the Users table, a context menu appears with the following actions:

- **Track**

  Brings you to the **User Tracking** page with the selected user entered.

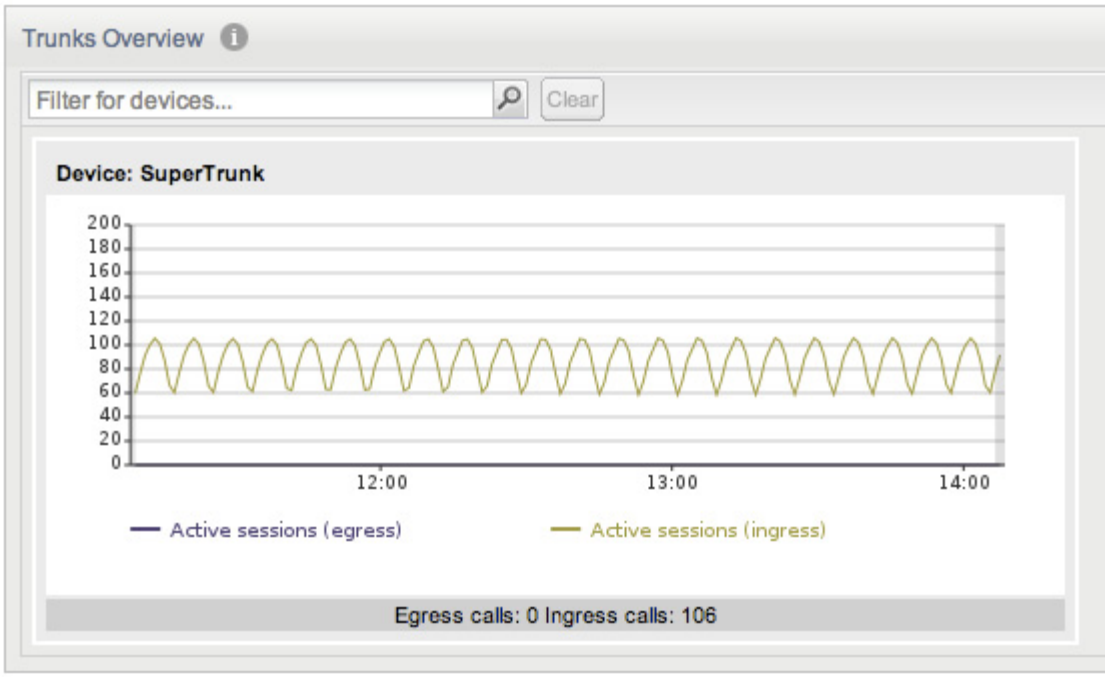
- **Trace**

  Brings you to the **Traces** page, where a new trace is started and filtered for the selected user.

Trunks/Prefixes

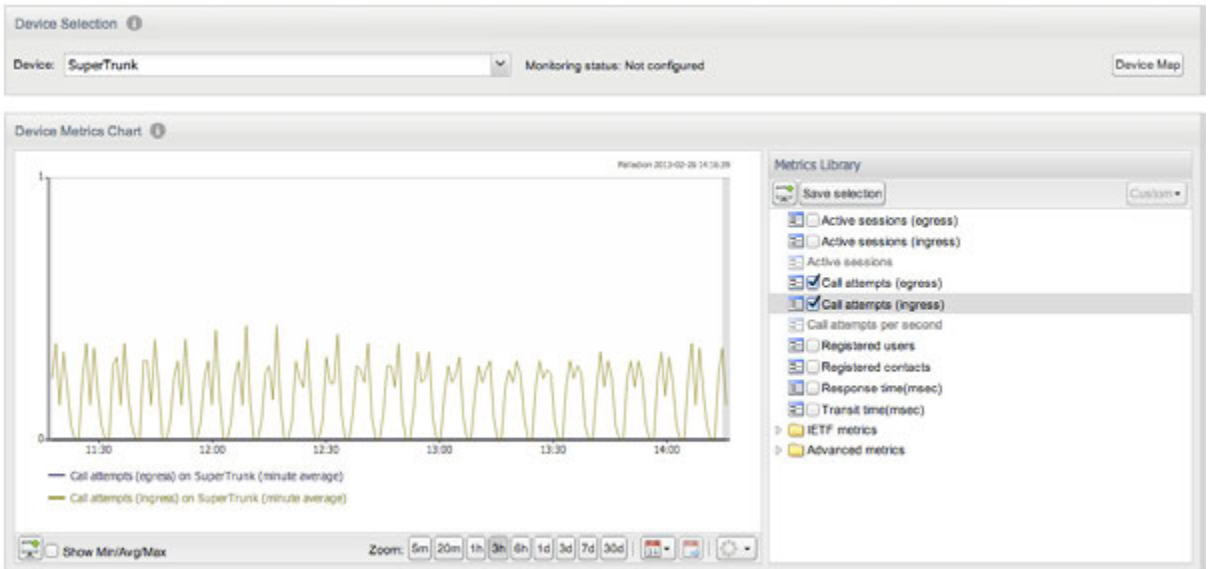
The **Trunks/Prefixes** page presents an overview of the status of all configured trunks and prefixes on the monitored platform. The **Active sessions** chart displays the number of inbound (ingress) and outbound (egress) sessions for each trunk or prefix (see Figure 4–52).

Figure 4–52 Trunks/Prefixes Page Overview



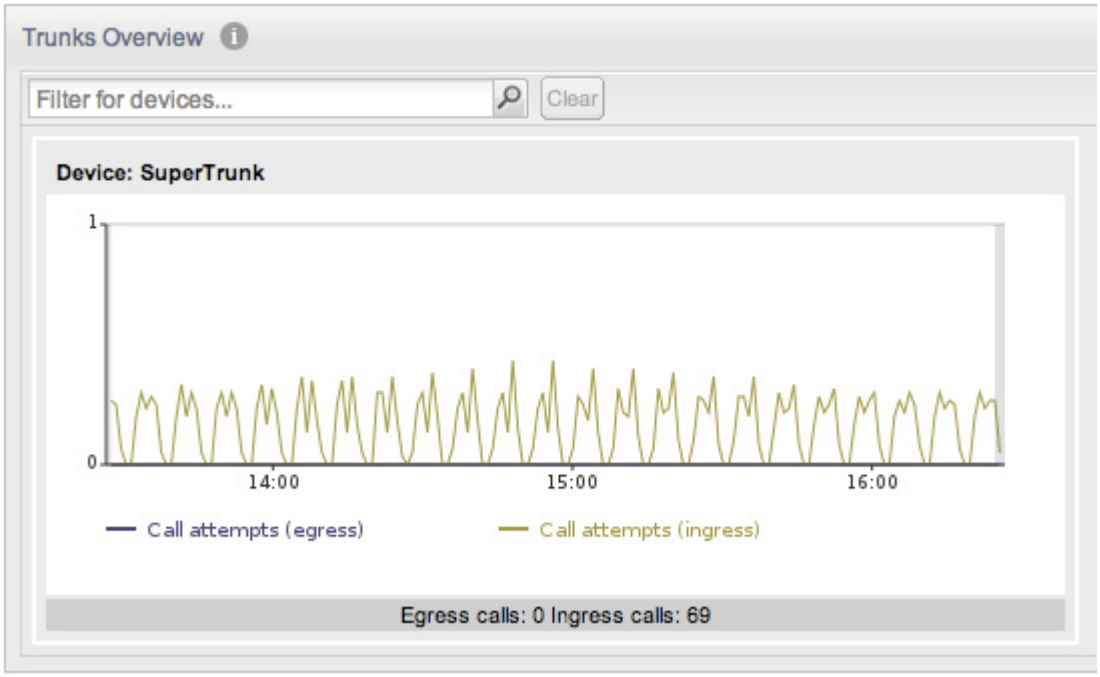
The metric displayed in the chart can be configured in the **Device** page. To change the metric to display for a trunk, select the metric from the *Metric Library* and click **Save selection**, as shown in Figure 4–53.

Figure 4–53 Configuring the Metric to Display for a Trunk



After the change, the **Call attempts** chart is displayed for the *SuperTrunk* (see Figure 4–54).

Figure 4–54 Trunks/Prefix Page Overview

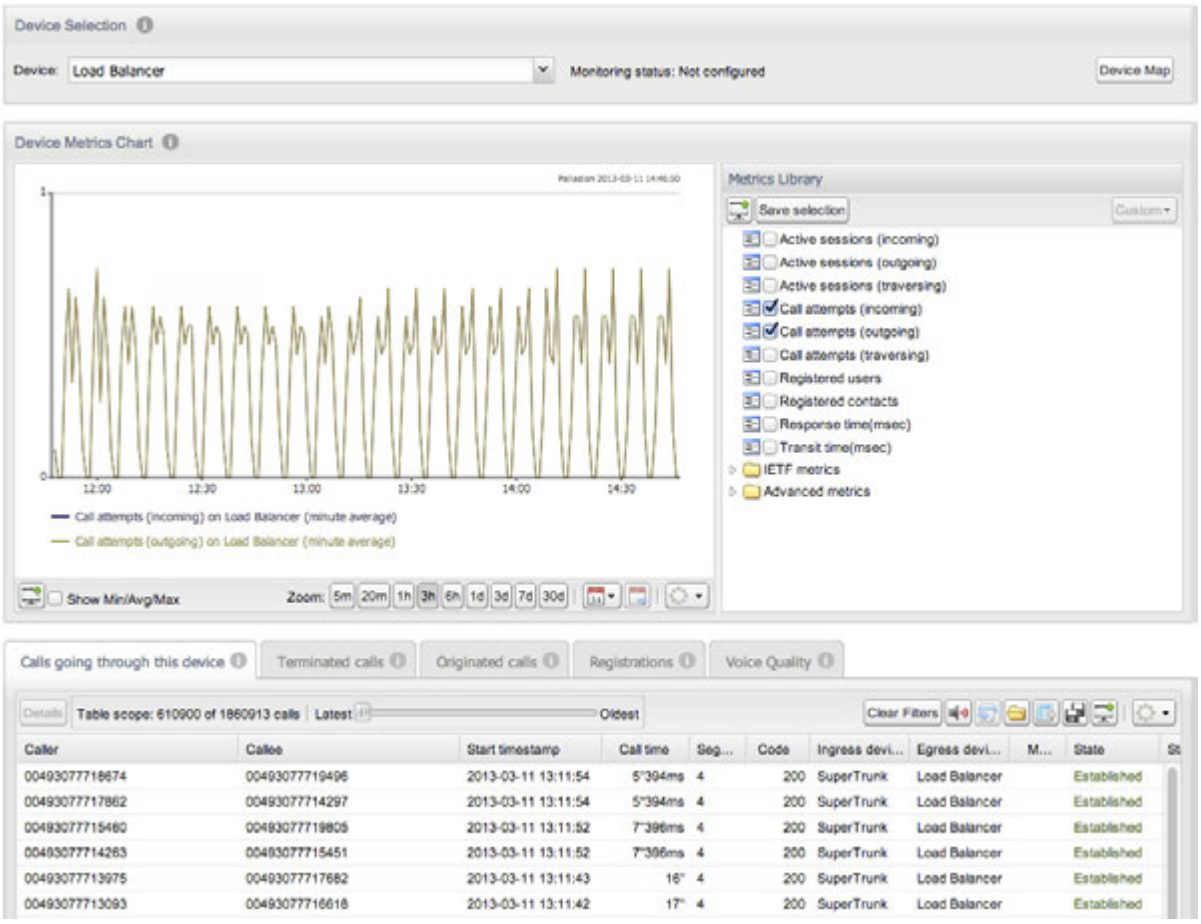


When you click on a trunk (outlined in white as illustrated in Figure 4–54), you are taken to the **Devices** page for this trunk. Clicking on a prefix has no effect. The **Trunks/Prefixes** page also features a **Refresh** button, which can be used to control the update interval for the textual information contained on the page.

Devices

The **Devices** page (see Figure 4–55) shows not only an overview of all configured platform devices, but it also aggregates information about each individual platform device. To use it, the platform devices (or trunks) have to be configured in the platform settings. For more information, see "Platform Devices". This page contains a device selection drop-down menu, a device metrics chart, and a tab panel providing details about calls, registrations, and voice quality for the selected device.

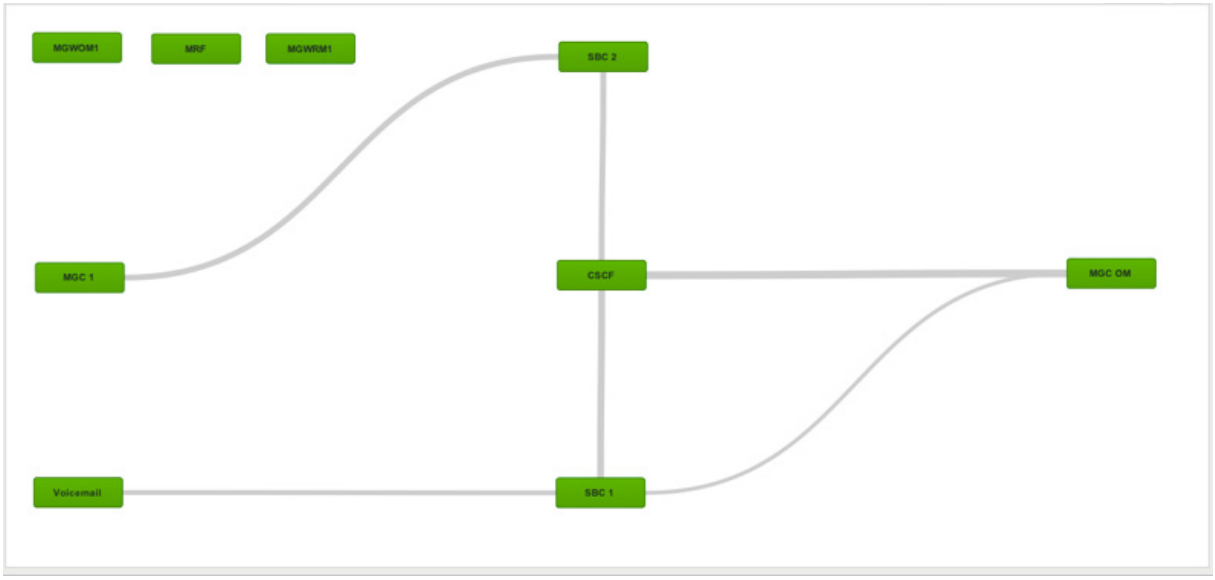
Figure 4–55 Devices Page



Device Map

You can display a map of devices by clicking on **Device Map** button. It shows an overview of the status and activity for all configured devices on the monitored platform (see Figure 4–56). They are displayed as nodes on a topological graph; activity between nodes is shown in the lines connecting them. From this page it is easy to monitor platform activity and status.

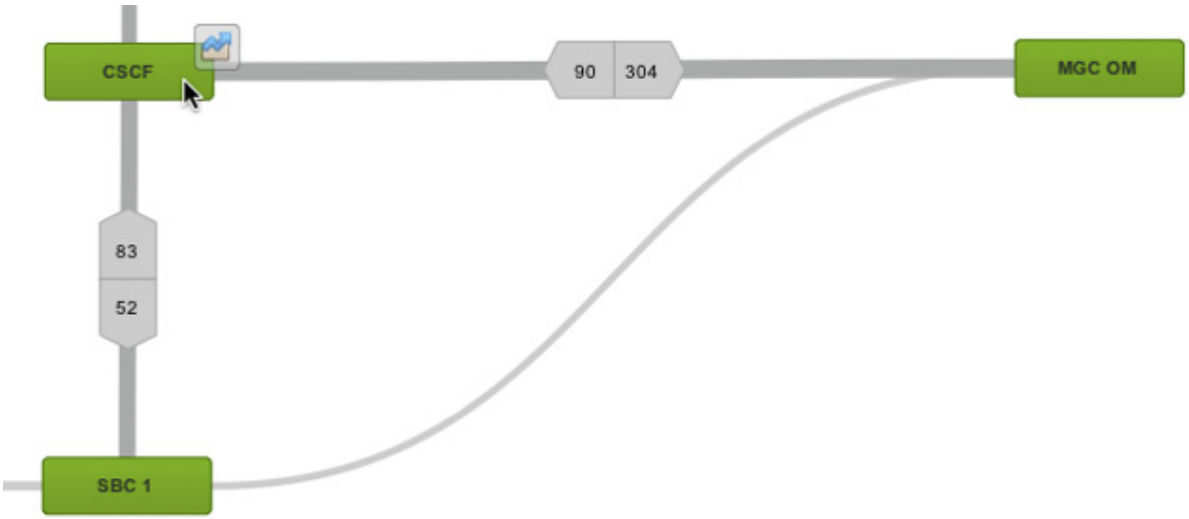
Figure 4–56 Device Map



Once added, you can drag and drop devices around the page to best represent the local network topology. There is a **Reset Layout** button on the toolbar to reset the page.

Hover over the node with the mouse cursor to show the inbound and outbound value of the displayed metric of a node, as shown in Figure 4–57.

Figure 4–57 Hovering Over a Device to Display the Active Metric



Note: By default, Operations Monitor displays a map in real-time of the configured platform devices and the interactions between them. Please note that due to browser and backend performance reasons, the map cannot work for a large number of configured devices. This is due to the high number of counters that need to be updated in real-time. Therefore, for more than 20 platform devices, we recommend you to disable the map through the 'Enabled Device Map' system setting.

Device Map Toolbar

The Devices page has view display options, which are configurable through controls on the toolbar.

Figure 4–58 shows the display options that are available on the Device page toolbar.

Figure 4–58 Device Map Toolbar

The display options are as follows:

- **Metric selection dropdown**

  The metric displayed can be configured in the metric selection drop- down.

- **Always show counters**

  Select the **Always show counters** check box to change the above device hovering behavior so that Operations Monitor always shows metric data.

- **Hide inactive**

  Select the **Hide inactive** check box to hide devices that are not interacting with others. This can often help de-clutter the display. When the page data is refreshed all devices are checked to see whether they have become active.

- **Reset layout**

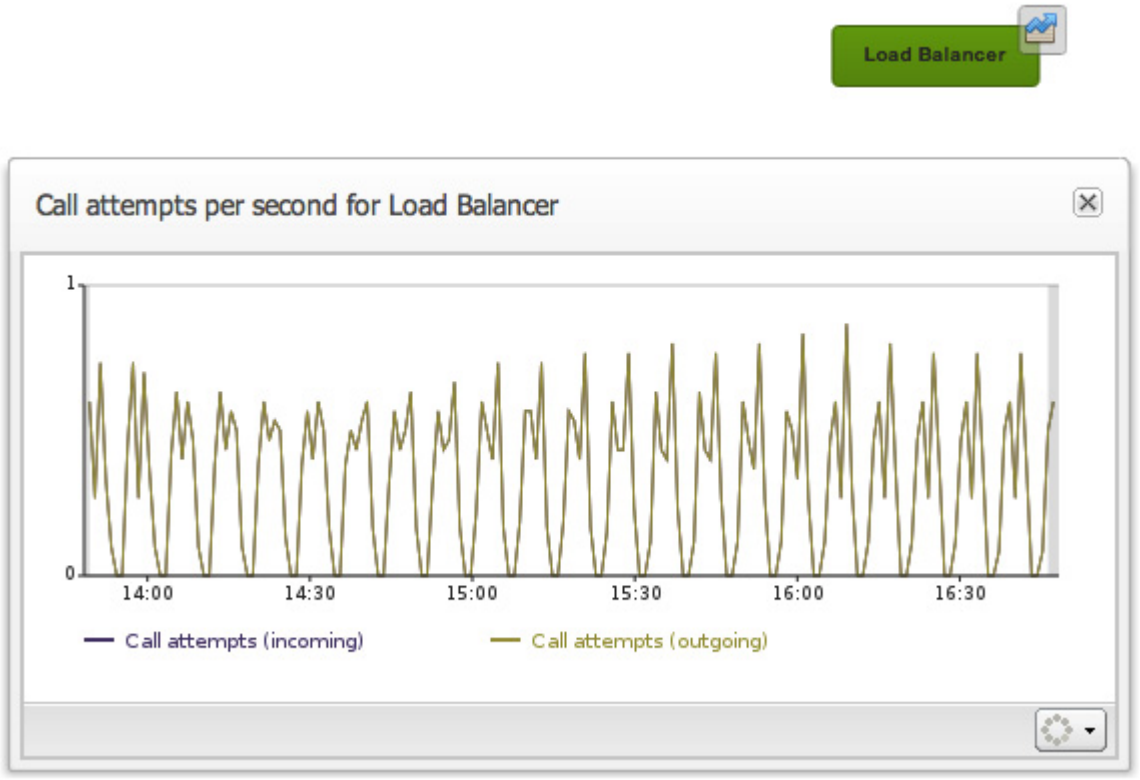
  Applies the default layout if the configuration by the user via dragging and dropping becomes unmanageable.

- **Refresher**

  The **Devices** page features a **Refresh** button to control the update interval for the metric data contained in the page. For more information, see "Refresh Button".

More detailed information about the selected metric (see Figure 4–59) can be displayed per-device by clicking on the **Details** button that appears overlaid on a device while hovering:

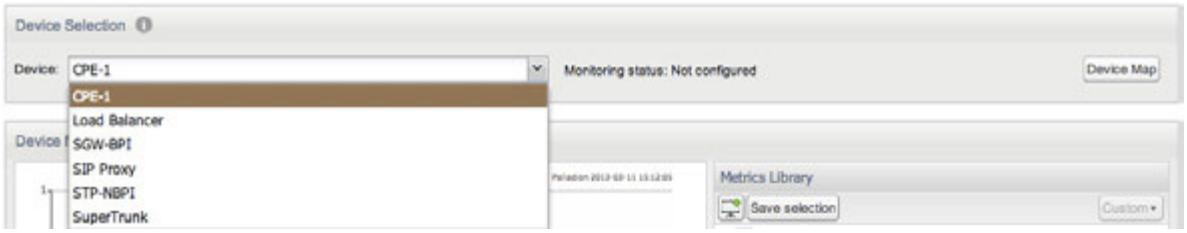
Figure 4–59 Click on a Device to Display Detail Information About the Active Metric



Device Selection Panel

The **Device Selection** panel illustrated in Figure 4–60 determines the device whose details are displayed on the page. Select a device from the drop-down menu.

Figure 4–60 Device Selection Panel



Device Monitoring Status

The **Device Selection** panel displays the monitoring status for the current device (see Figure 4–61). This information is available only if device monitoring is configured for that specific platform device.

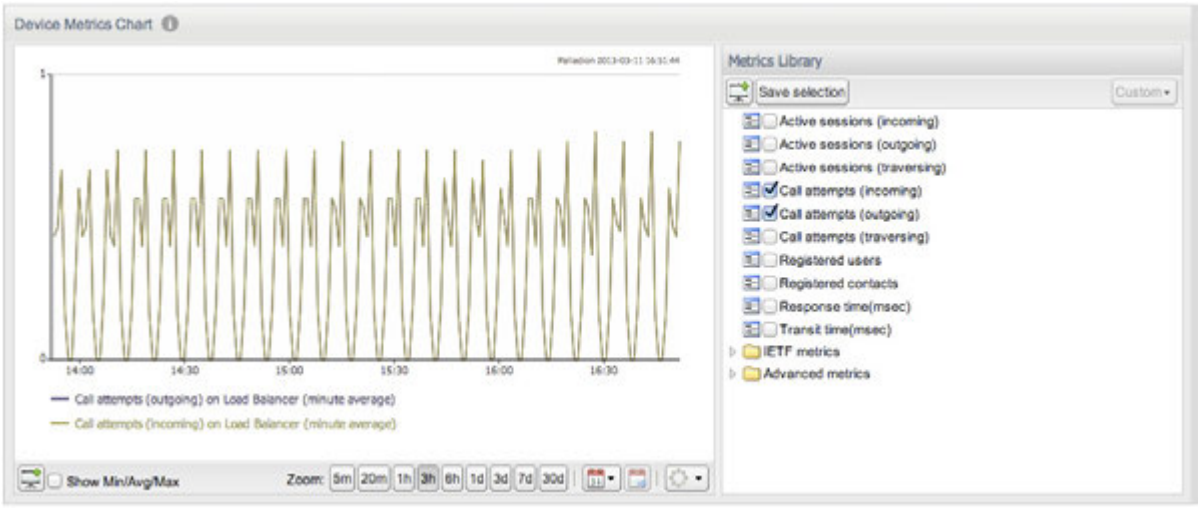
To configure device monitoring, refer to "Platform Devices" and "Device Monitoring".

Figure 4–61 Device Monitoring Status

Device Metrics Chart

The **Device Metrics Chart**, illustrated in Figure 4–62, displays the evolution of the selected device metrics from the *Metrics Library*. For more information about the Metrics Library, see "KPI/Metrics". Device metrics are special in that they measure properties connected to a certain platform device.

Figure 4–62 Device Metrics Chart



Note: You can select up to five metrics by marking the check box of the desired metric.

The **Save Selection** button saves the current set of metrics for the specified device in the **Device Metrics** chart.

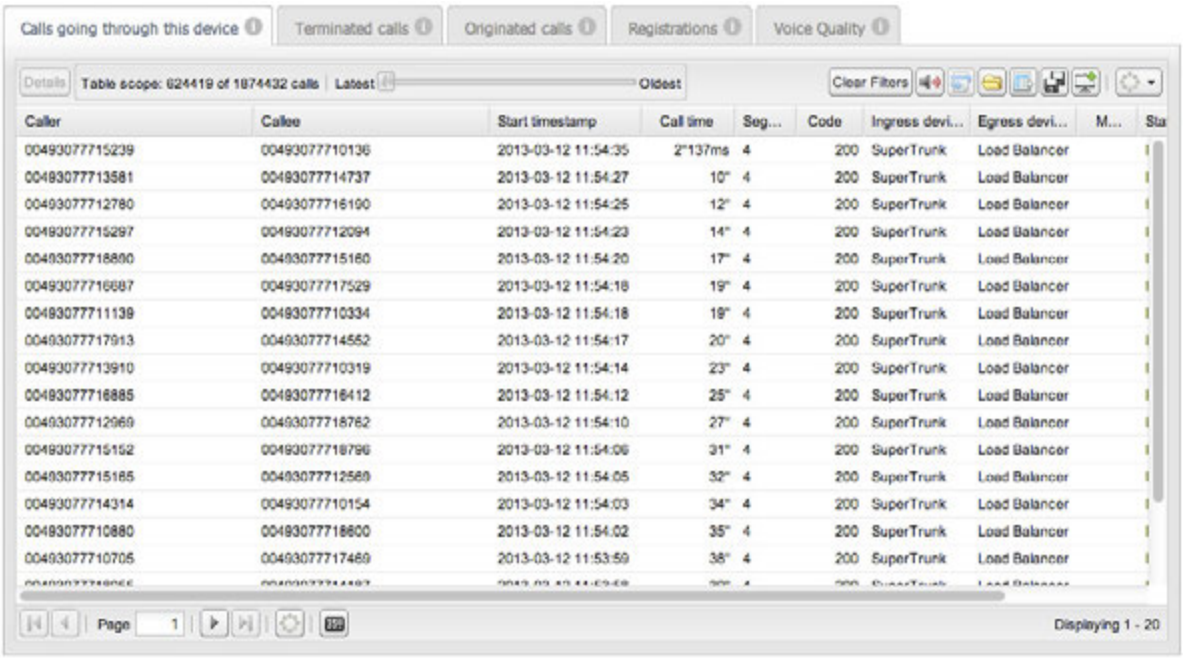
You can add the **Device Metrics Chart** to the **Dashboard** by clicking the **Show in Dashboard** icon button.

At the bottom of the chart, the **Show Min/Avg/Max** check box controls whether minute or hour minima and maxima are displayed for each selected metric. The other controls are described in the section on chart panels. For more information, see "Charts".

Calls Going Through This Device Tab

The **Calls going through this device** tab, as illustrated in Figure 4–63, is similar to the *Recent calls* panel described in the "Calls" section. The table displays only the calls that have an inbound leg to the current device and an outbound leg from the current device.

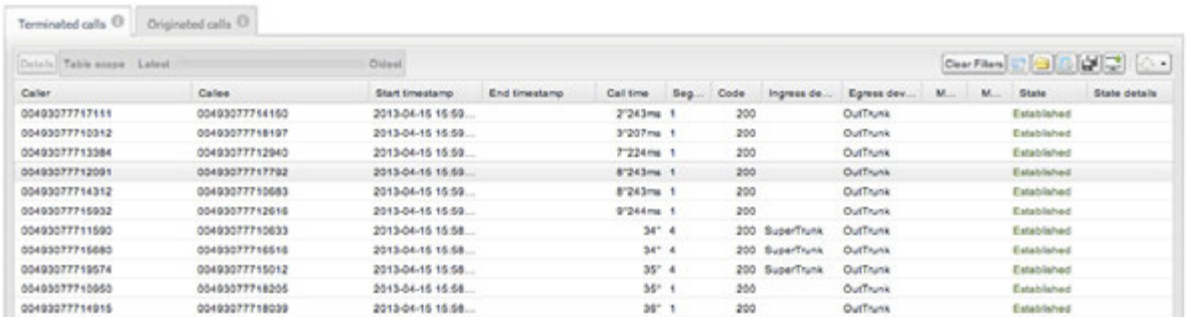
Figure 4–63 Calls Going Through This Device Tab



Terminated Calls Tab

The **Terminated calls** tab, as illustrated in Figure 4–64, is exactly like the **Recent calls** panel described in "Calls". The table displays only the calls which have an inbound leg to the current device. Terminated calls do not have an outbound leg from the current device.

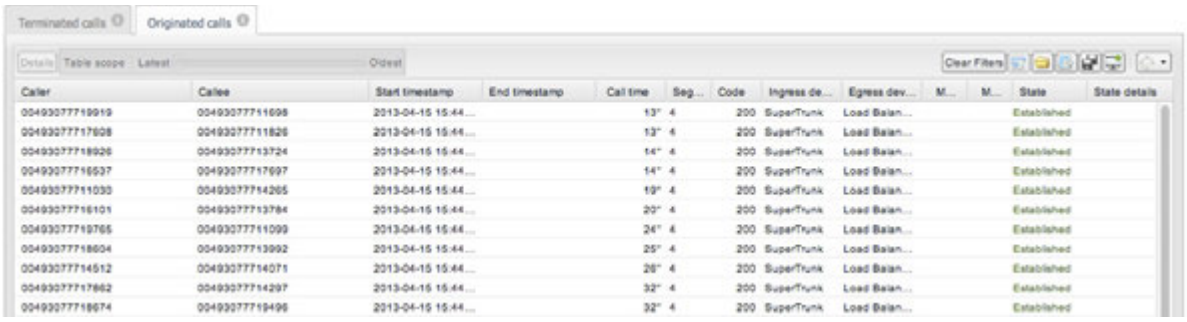
Figure 4–64 Terminated Calls Tab



Originated Calls Tab

The **Originated calls** tab, as illustrated in Figure 4–65, is exactly like the **Recent calls** panel described in "Calls". The table displays only the calls which have an outbound leg from the current device. Originated calls do not have an inbound leg to the current device.

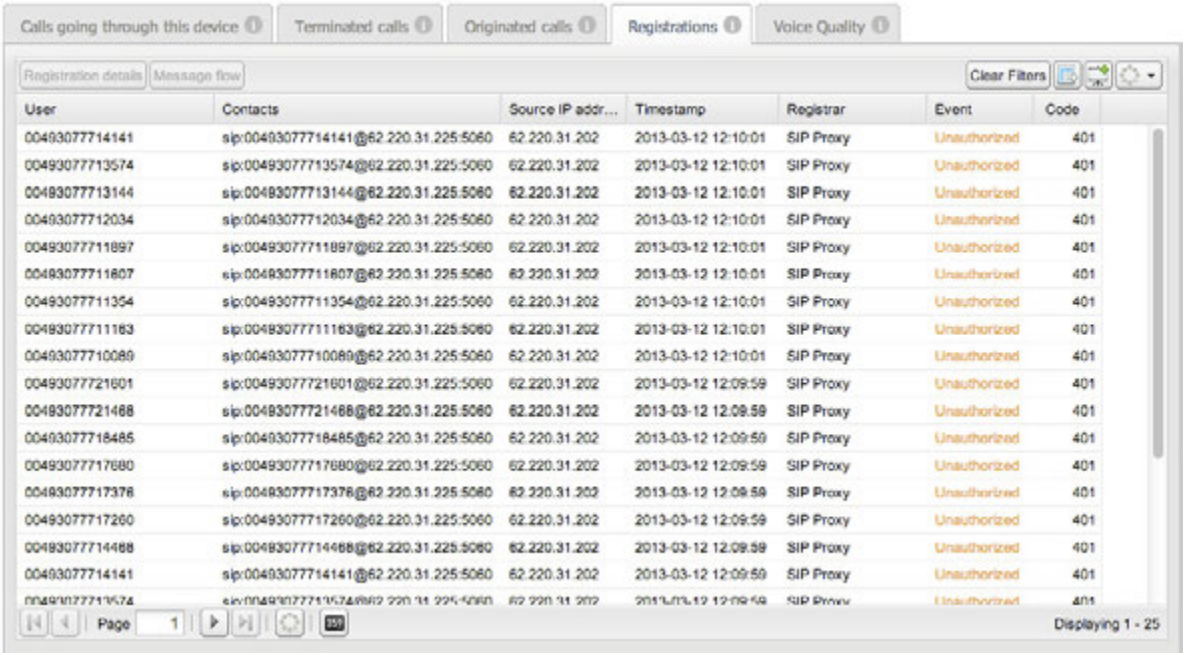
Figure 4–65 Originated Calls Tab



Registrations Tab

The **Registrations** tab, as illustrated in Figure 4–66, is similar to the **Registrations** panel described in "Registrations". However, it only displays registrations which are handled by the current device.

Figure 4–66 Registrations Tab



Note: The **Registrations** tab is not available for gateway devices and trunks.

5

Customers

This chapter describes how to work with the customer experience and troubleshooting features of Oracle Communications Operations Monitor.

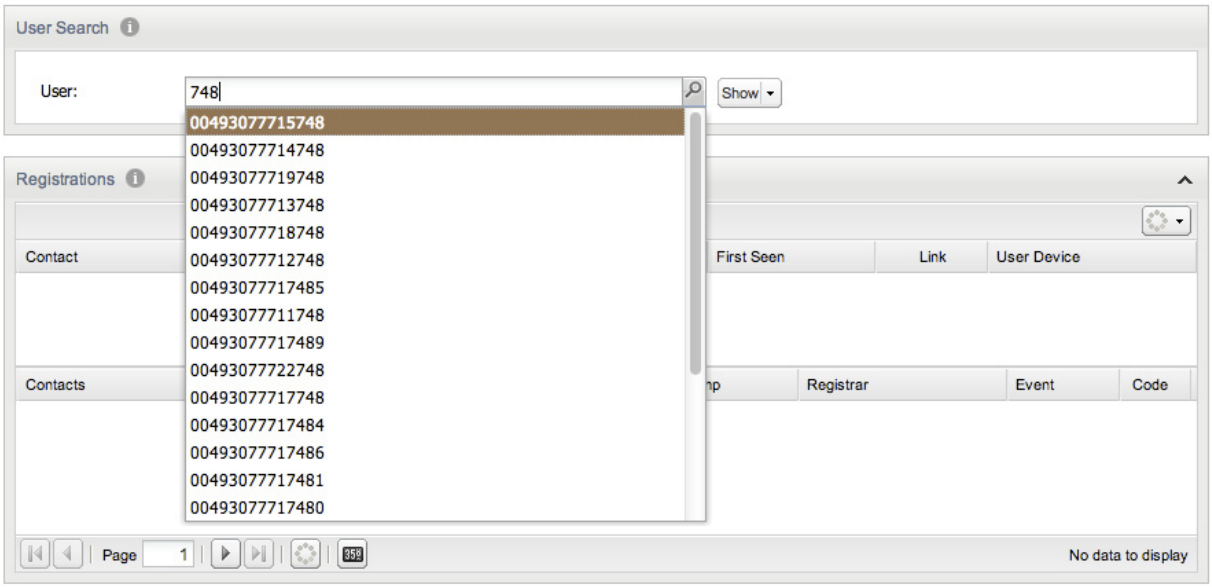
User Tracking

The **User Tracking** page can be accessed under the **Customers** section of the main menu. This page aggregates information for each user of the monitored platform and displays the history of the user's registrations and the calls. A *user* in Operations Monitor is identified by the user name part and optionally by the host name part of a SIP URI.

User Search Panel

The **User Search** panel (see Figure 5–1) allows you to specify which user to track on the **User Tracking** page. It is enabled with *live search* functionality. As soon as you type an identifier in the text field, a background search for users containing the text you entered in the field is performed. A maximum of 30 found users is displayed in the drop down selection.

Figure 5–1 User Search Panel



Registrations Panel

The panel illustrated in Figure 5–2, contains two tables where the first displays the current registered contacts and the second a history of the registrations events for the tracked user.

Figure 5–2 Registrations Panel

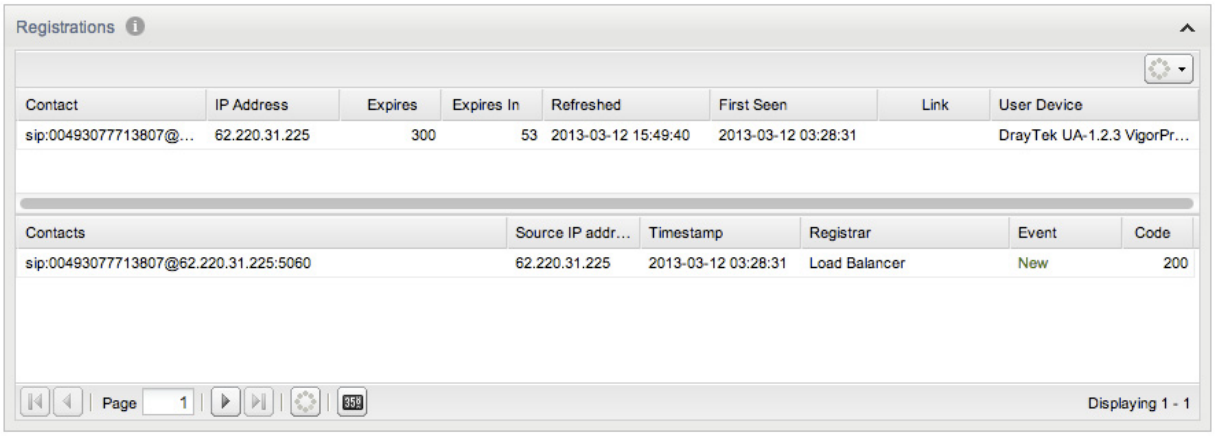


Table 5–1 lists the current registered contacts columns.

Table 5–1 Current Registered Contacts Table

Column Name	Description
Contact	The **Contact** header of the REGISTER request.
IP address	The IP address the REGISTER request originates from.
Expires	The expiration interval set by the server.
Expires In	The remaining time until expiration.
Refreshed	The last time the registrations was refreshed.
First Seen	The first time the tracked user registered successfully.
Link	An indicator for the link quality to the tracked user.
User Device	The **User-Agent** header of the REGISTER request.

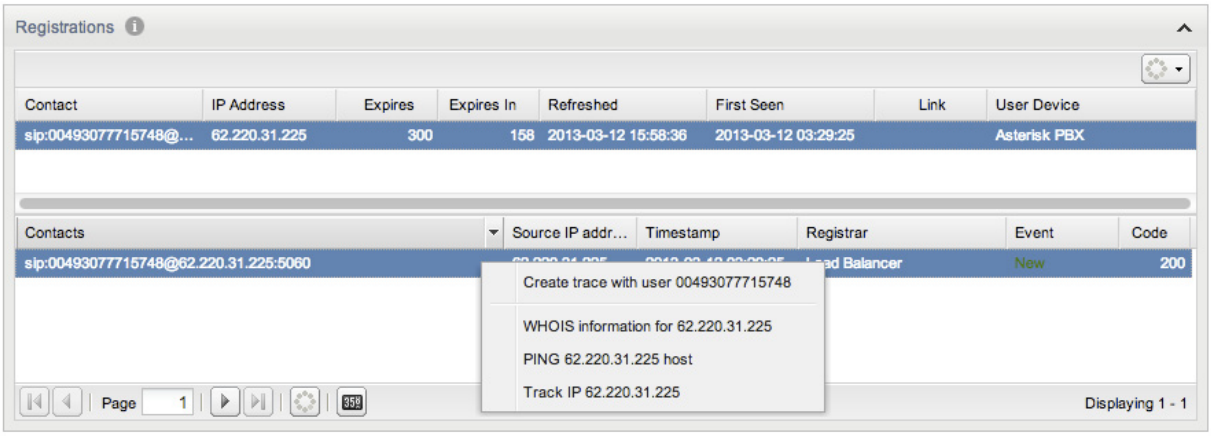
The upper table also contains a hidden column called **Suggested Expires** which can be enabled. It contains the value of the **Expires** header sent with the REGISTER request.

The lower table contains the history of registration events for the tracked user. For the description of this table, refer to "Registrations".

User Actions

Figure 5–3 shows the actions that can be performed on the tracked user.

Figure 5–3 Right-Click On the Registration Event



Right clicking on a register event from the second table of the **Registrations** panel shows the following actions you can perform on the tracked user:

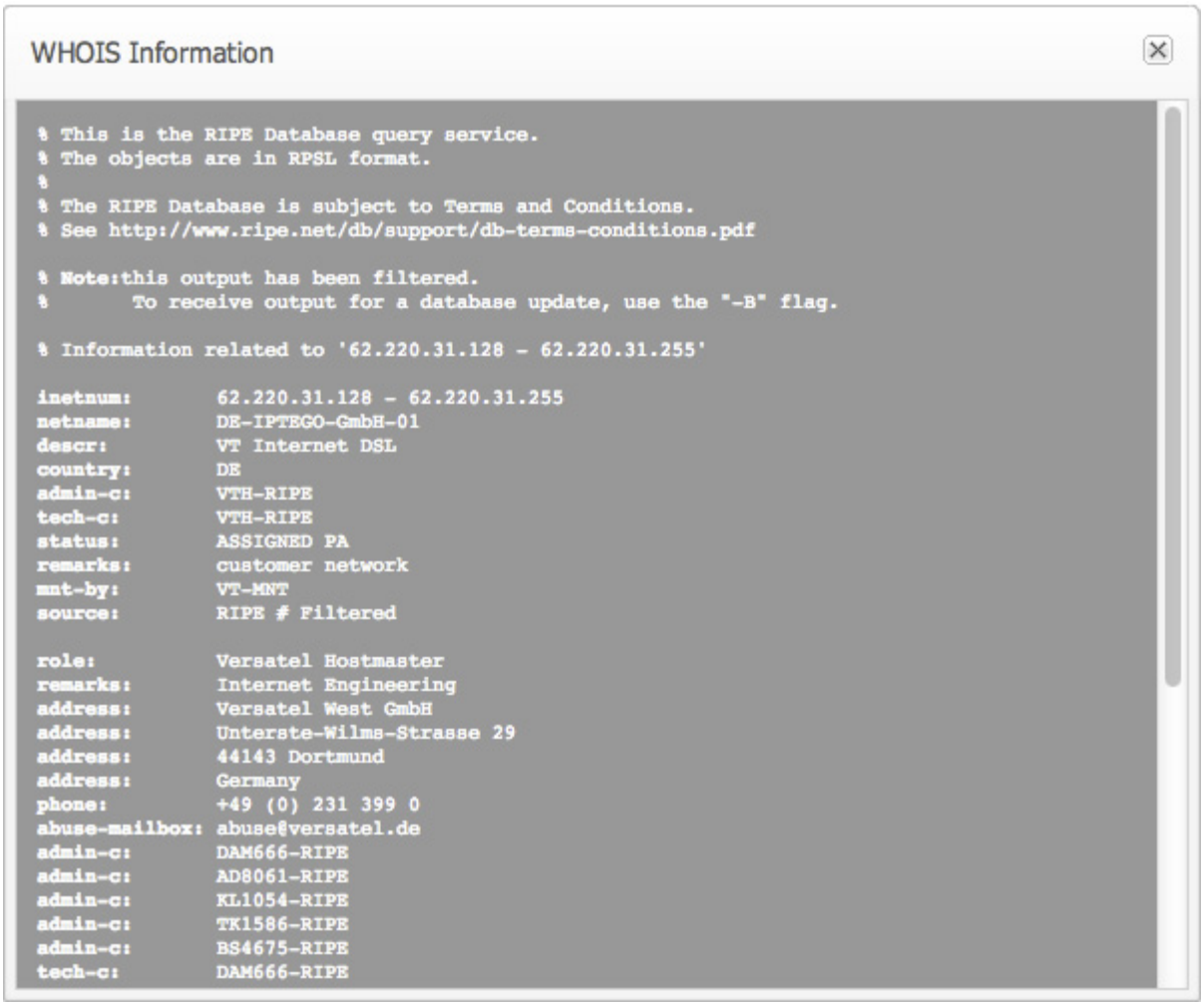
- **Create trace with this user**

  Redirects you to the **Traces** page and starts a trace that is filtered for the tracked user. For more information, see "Traces".

- **WHOIS information**

  Gives the result of a WHOIS query for the IP address of the user (see Figure 5–4).

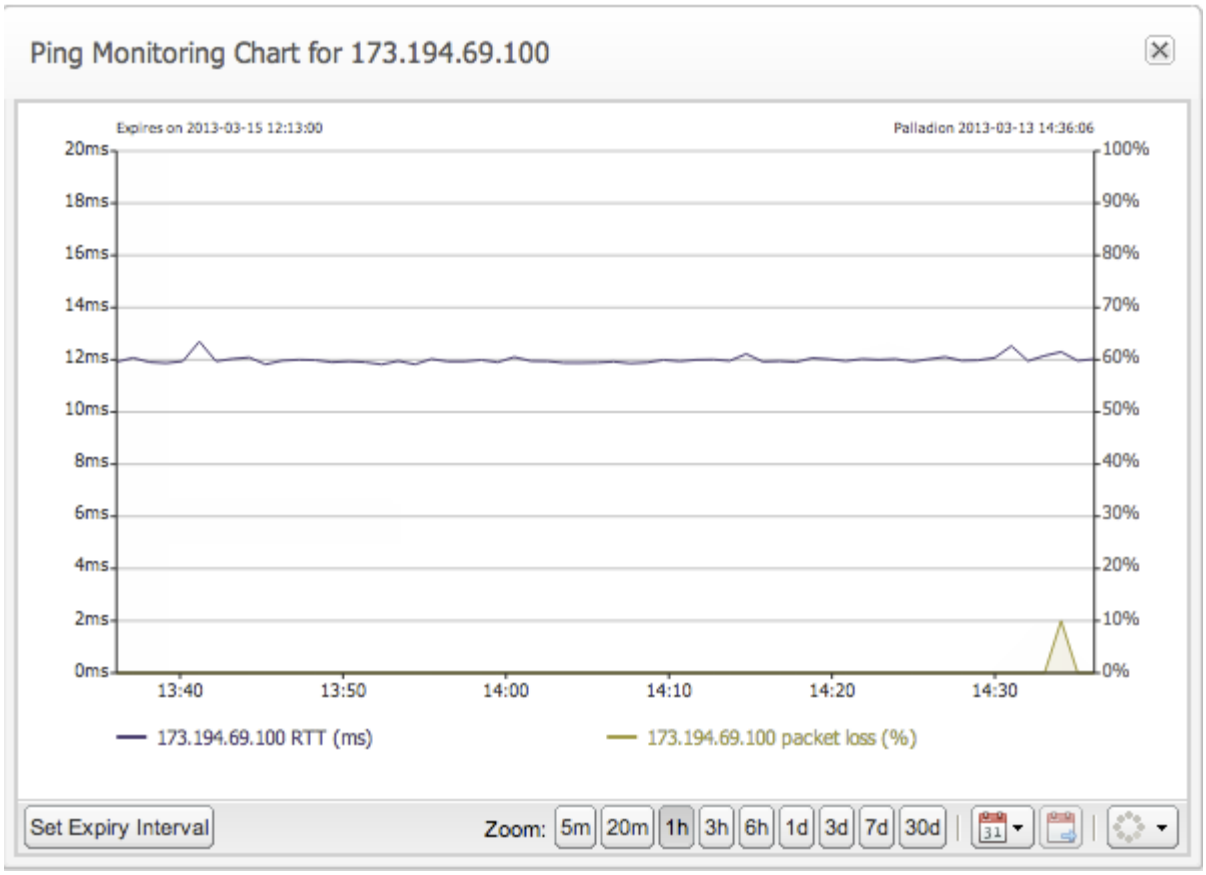
Figure 5–4 WHOIS Results for an IP



- **PING host**

  Checks the quality of the link between the provider and subscriber's host by pinging the host periodically (see Figure 5–5).

Figure 5–5 Link Quality with the host



- **Track IP**

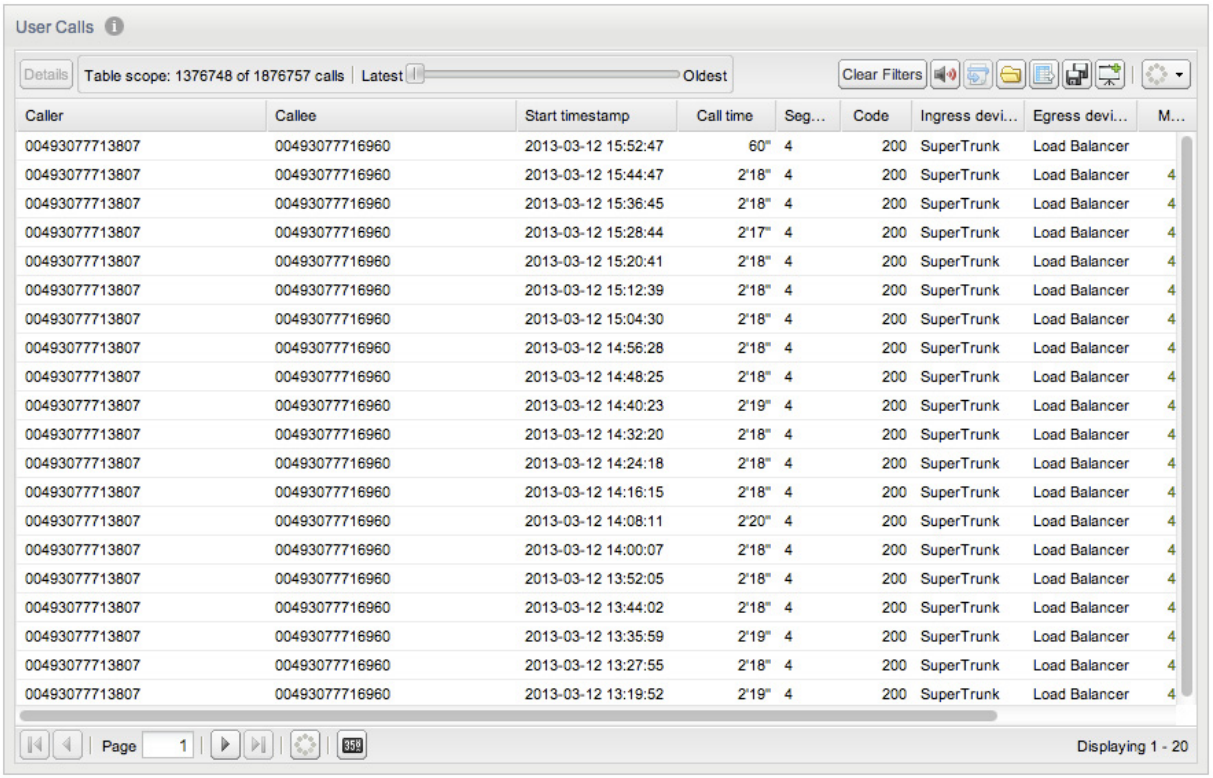
  Redirects you to the **IP tracking** page to get the details about the specific IP. For more information, see "IP Tracking".

Right-click on a registered contact gives you a similar menu as the previous one.

User Calls Panel

The **User Calls** table, as illustrated in Figure 5–6, displays the history of calls that the tracked user participated in as either a caller or a callee.

Figure 5–6 User Calls Panel



You can double click or select a call in the **User Calls** table, and click on **Details** to get to the **Call Info** dialog box. Click on the **Message flow** to display a diagram of the underlying SIP messages of a call. For details on the usage of the **User Calls** table refer to "Calls".

IP Tracking

The **IP Tracking** page aggregates information about a single IP address. It displays all users registered with the given IP address, and provides links to the **User Tracking** page for each user. Either IPv4 or IPv6 addresses may be supplied.

Note: Pinging an IPv6 address is not supported.

IP Search Panel

The **IP Search** panel contains the text input field where you can enter the IP address you want to track. This field accepts network ranges. If a range is entered, Operations Monitor displays the available information for the IP addresses from the range. For example, entering 62.220.31.0/24 displays all the users that are registered from an IP address within that network range.

Figure 5–7 shows the IP Search panel.

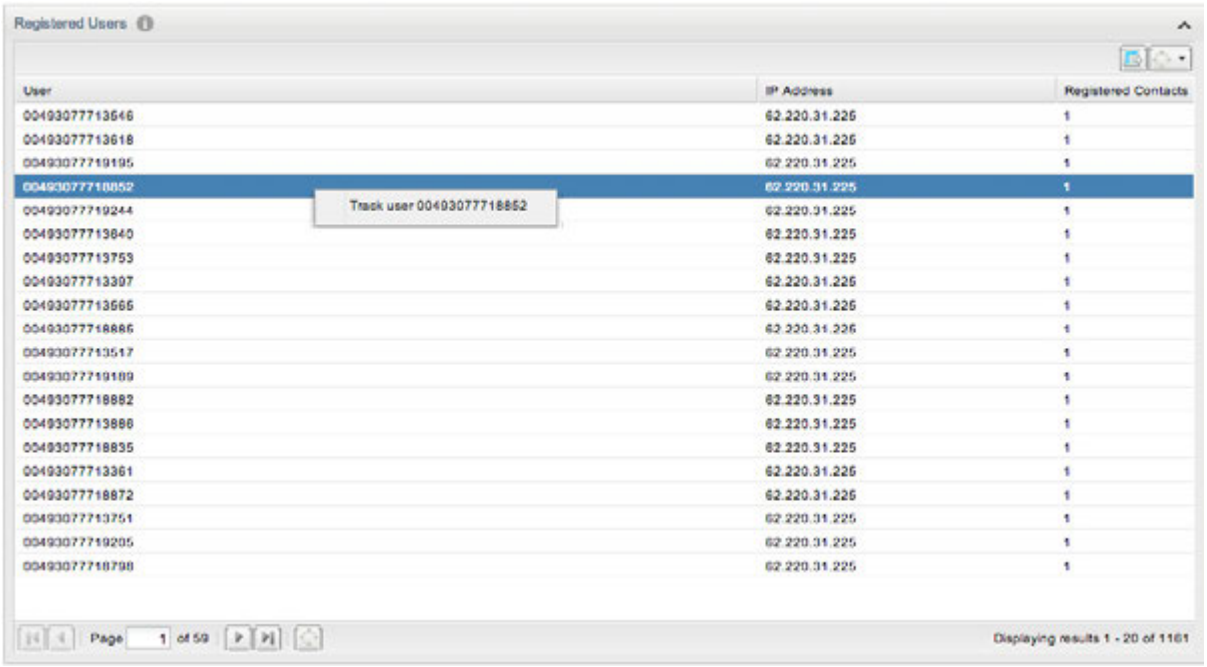
Figure 5–7 IP Search Panel

Registered Users Panel

The **Registered Users** panel, as illustrated in Figure 5–8, displays the users that are currently registered with the searched IP address(es). The results include the user, the IP address, and the number of registered contacts of the user are included in the results.

You can jump to the **User Tracking** page via the right-click menu.

Figure 5–8 Registered User's Panel



Calls Panel

The **Calls** panel displays recent and historical information for calls made in recent days. Calls displayed in this panel were made from or to the searched IP address(es). Calls to the searched address are only listed if they were completed. The calls from this table are updated in real-time as their state changes.

For a description of the columns in the **Calls** panel, refer to Table 4–2:

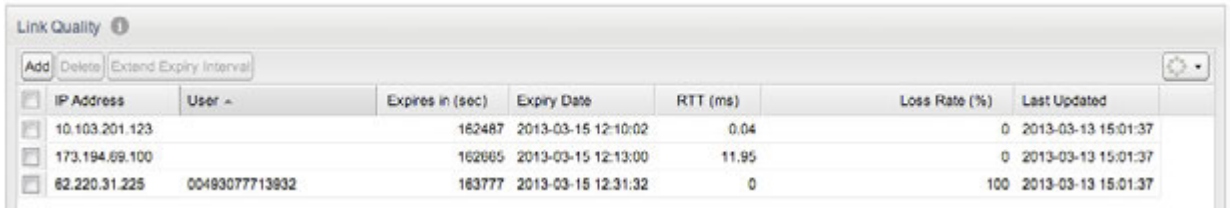
Link Quality

Operations Monitor has the ability to ping the public IP of a subscriber or ping the IP of a proxy. You can access Link Quality under the *Customer Experience & Troubleshooting* section of the main menu. It sends out 10 *ICMP Echo Request* packets every minute with a 500 milliseconds delay between successive packets to an individual target.

Based on the incoming responses and elapsed time for the requests, it reports on the target's reachability, packet loss and round trip delay.

For every pinged address, Operations Monitor measures an average value of round-trip time (RTT) and **loss rate**. The loss rate is the percentage of ICMP Echo Replies lost for a remote target as illustrated in Figure 5–9.

Figure 5–9 Link Quality Table



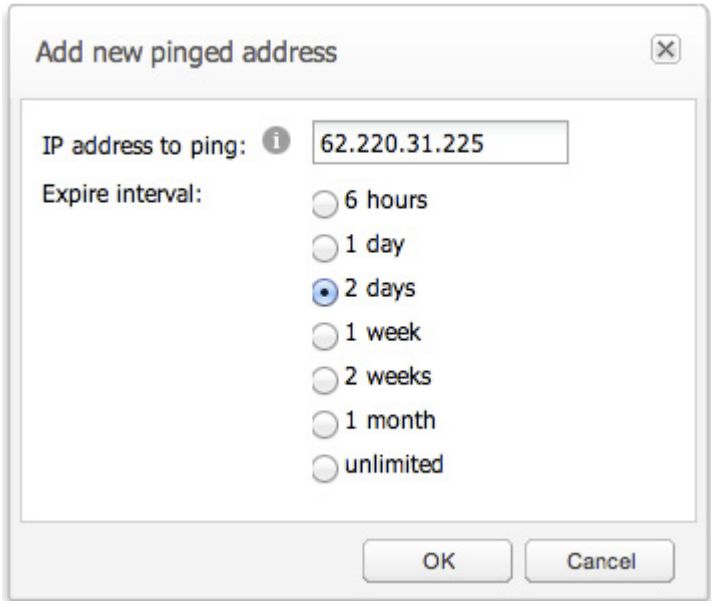
Check Link Quality to a Subscriber

To check the quality of the link between the provider and the subscriber, the provider can start pinging the user's IP from the **User Tracking** page. This is usually helpful to track voice quality problems. For more information, see "User Tracking".

Check Link Quality to a Proxy

To start pinging a new address, you must add the IP address to the **Network Link Quality** page by clicking on the **Add** button. In Figure 5–10 the IP address 62.220.31.225 is added and is configured to be pinged for 2 days. Once you add a pinged address, the interval is extended by pressing the **Extend Expiry Interval** button of the **Link Quality** panel.

Figure 5–10 Add a Target to Ping



To stop pinging a target, you can delete the address by selecting it and pressing the **Delete** button of the **Link Quality** panel.

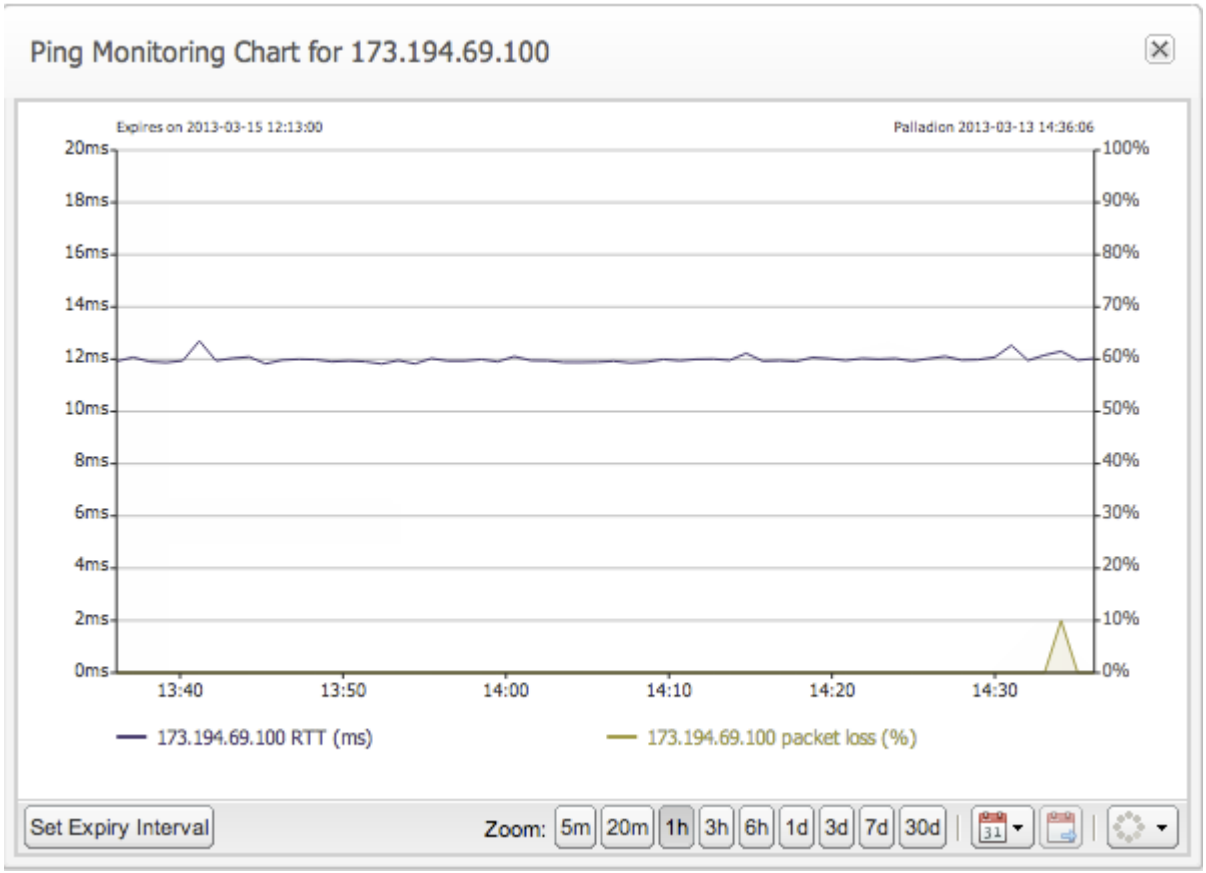
The **Link Quality** panel describes the quality of a link through its columns as listed in Table 5–2:

Table 5–2 Link Quality Panel

Column Name	Description
IP Address	Uses the IPv4 address of the target to ping either the subscriber or a proxy.
User	In case of a subscriber, the **User** field provides the username.
Expiry Date	Define the time when Operations Monitor ceases to ping the target.
Expires (sec)	Define the pinging interval period in seconds.
RTT (ms)	Takes the average value of the *round trip time* in milliseconds.
Loss Rate (%)	Takes the average value of the *loss rate* in percentage.
Last Updated	Describes when the link quality values were last updated.

As an example, Figure 5–9 displays link quality results with three particular targets. The first two targets have a good link quality as their medium loss rate is equal to 0% and the average value of the RTT is 0.04ms and 11.95ms. The last target is unreachable as it doesn't respond at all to ping requests. Thus its average loss rate is equal to 100%.

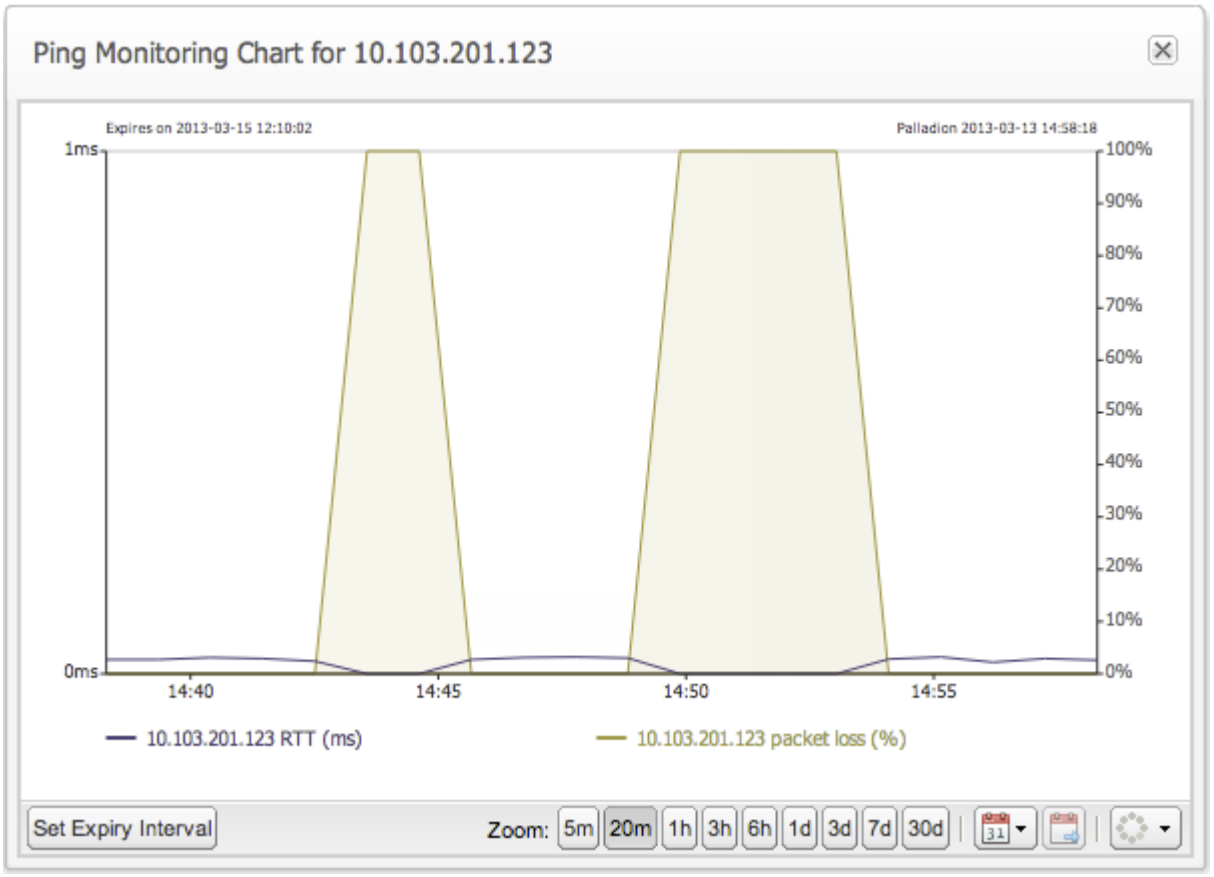
Figure 5–11 Good Link Quality



A time chart of positive link quality presents RTT and Loss Rate values over one hour in Figure 5–11. The dark blue wave defines the RTT values, and the light green wave

defines loss rate. On a scale of 0ms up to 1ms from the left side of the graph, RTT values slightly variate around the medium value of 0.04ms. As the loss rate value stays constant at 0%, it overlaps with the horizontal axis with 2 exceptions where is 100%.

Figure 5–12 Poor Link Quality



A poor link quality evolution over a 20 minutes period is presented in Figure 5–12. The variations of the blue curves for RTT values and the light green lines for Loss Rate values, signifies that the voice quality of the subscriber in some periods of time is quite bad.

6

Control Plane Monitor

The Oracle Communications Control Plane Monitor product offers Diameter transaction monitoring. The following Diameter interfaces are supported:

- S6a
- S6d
- S13
- S13'

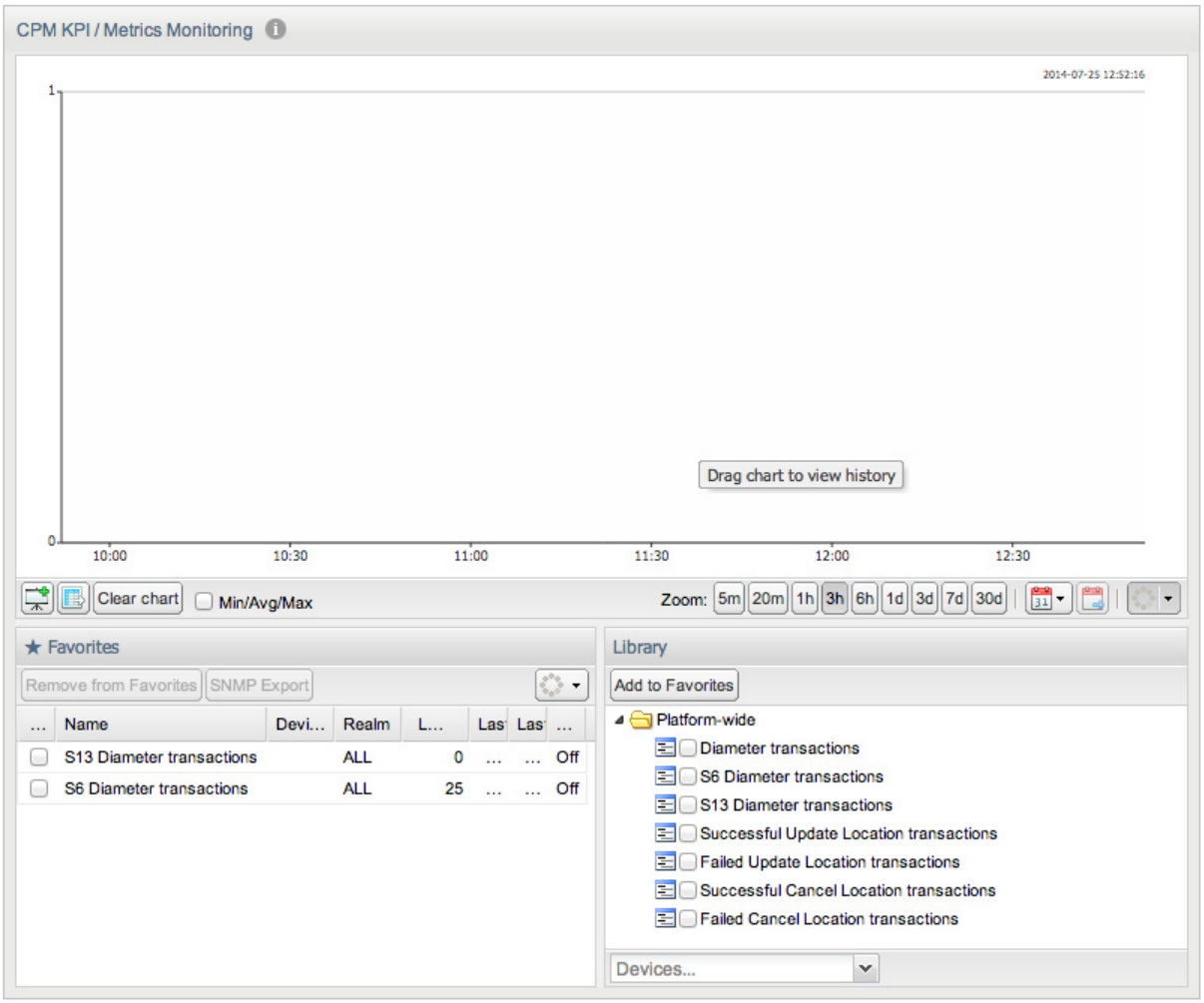
> **Note:** The Oracle Communications Operations Monitor product also supports the IMS Cx interface. The Cx Diameter messages are correlated with SIP calls, so they will show up in the Calls UI but not in the Control Plane Monitor UI.

KPI/Metrics

The Control Plane Monitor **KPI/Metrics** page offers similar functionality as the **KPI/Metrics** page from the Operations Monitor product. The difference is that only Diameter specific KPIs are available, and they are not editable.

Figure 6–1 shows the Control Plane Monitor KPI/Metrics page.

Figure 6–1 Control Plane Monitor KPI/Metrics Page



Currently, the following Platform Wide KPIs are available:

- **Diameter transactions**

  Number of new hop-by-hop Diameter transactions per second that pass the filter configured in the Platform Setup Application, regardless of Application ID.

- **S6 Diameter transactions**

  Number of new hop-by-hop Diameter transactions per second that have the Application ID 16777251 (3GGP S6a/S6d).

- **S13 Diameter transactions**

  Number of new hop-by-hop Diameter transactions per second that have the Application ID 16777252 (3GGPS13/S13').

- **Successful Update Location transactions**

  Number of completed hop-by-hop Diameter transactions per second that have the Diameter command 316 (Update Location) and the result code 2001 (Success).

- **Failed Update Location transactions**

Number of completed hop-by-hop Diameter transactions per second that have the Diameter command code 316 (Update Location) and a result code different from 2001 (Success).

- **Successful Cancel Location transactions**

  Number of completed hop-by-hop Diameter transactions per second that have the Diameter command 317 (Cancel Location) and the result code 2001 (Success).

- **Failed Cancel Location transactions**

  Number of completed hop-by-hop Diameter transactions per second that have the Diameter command code 317 (Cancel Location) and a result code different from 2001 (Success).

The following KPIs are available on a per-device basis:

- **Incoming S6 Diameter transactions**

  Number of new Diameter transactions per second that are received by the given device and have the Application ID 16777251 (3GGP S6a/S6d).

- **Outgoing S6 Diameter transactions**

  Number of new Diameter transactions per second that are created by the given device and have the Application ID 16777251 (3GGP S6a/S6d).

- **Successful incoming Update Location transactions**

  Number of completed Diameter transactions that are received by the given device and have the Diameter command 316 (Up- date Location) and the result code 2001 (Success).

- **Failed incoming Update Location transactions**

  Number of completed Diameter transactions that are received by the given device and have the Diameter command 316 (Update Location) and a result code different from 2001 (Success).

- **Successful incoming Cancel Location transactions**

  Number of completed Diameter transactions that are received by the given device and have the Diameter command 317 (Cancel Location) and the result code 2001 (Success).

- **Failed incoming Cancel Location transactions**

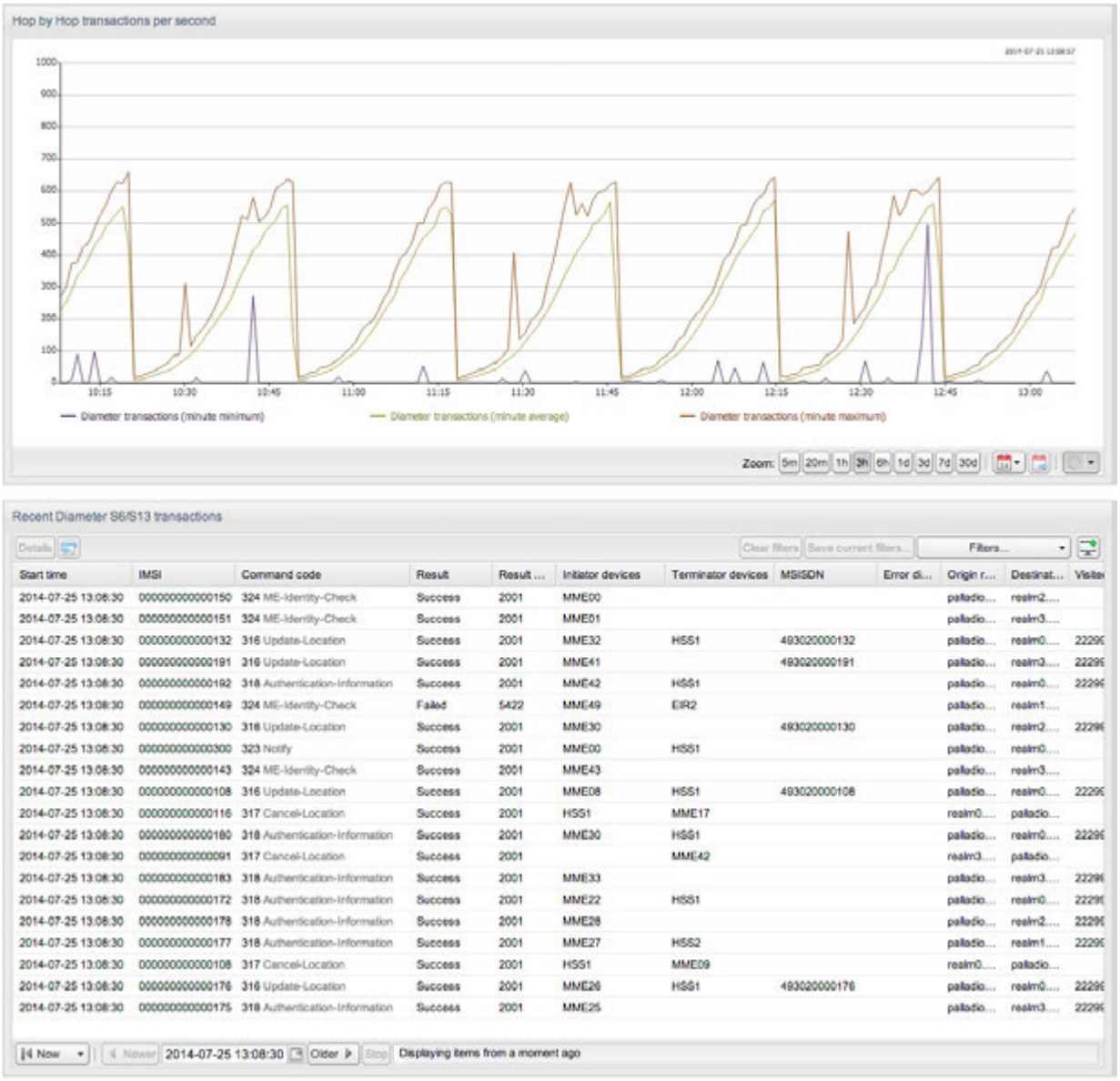
  Number of completed Diameter transactions that are received by the given device and have the Diameter command 317 (Cancel Location) and a result code different from 2001 (Success).

Transactions

The **Transaction** page is the central repository for transaction analysis in Control Plane Monitor. You can analyze the transaction traversing the platform in real-time or historically. Control Plane Monitor automatically correlates the transactions into End-to-End transactions.

This page contains two panels: the top one shows the Diameter transactions KPI while the bottom one contains the recent S6/S13 transactions. As shown in Figure 6–2.

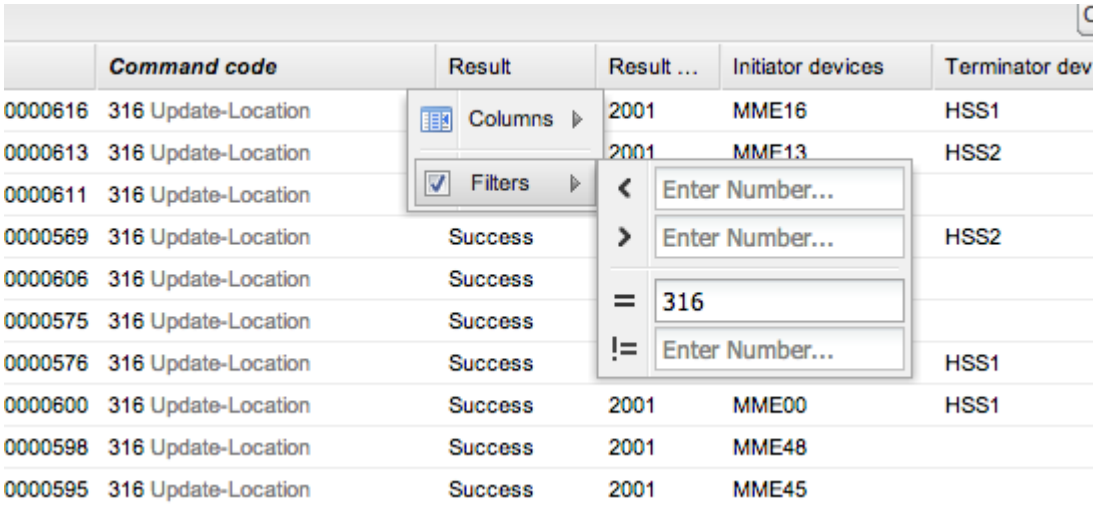
Figure 6–2 Control Plane Monitor Transactions Page



Filtering Columns

The **Transactions** table supports Filtering for all columns. Figure 6–3 shows an example of filtering that selects all transactions with a given command code. For more information, see "Filtering."

Figure 6–3 Control Plane Monitor Filtering



Transaction Details

To open the **Transaction details** window, double-click on a row from the **Recent S6/S13 transactions** table or select the transaction and click the **Details** button.

The upper portion for the window contains brief identification information of the transaction: the IMSI, the Diameter realms, the MSISDN and so on.

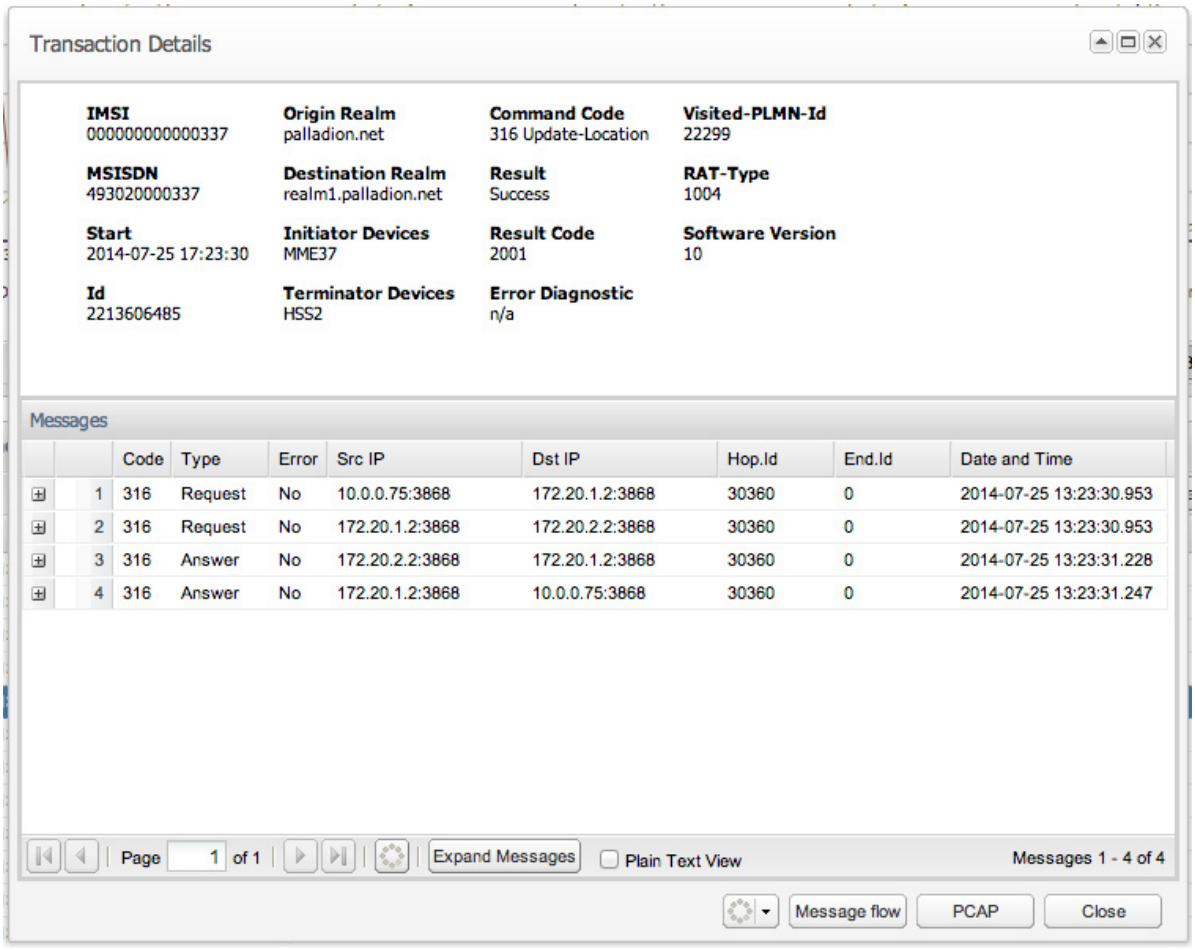
The bottom portion of the window contains the list of messages composing the transaction. You can drill down in the details of each message by expanding the tree.

Click the **Message flow** button to open a *message flow diagram* with the messages involved in this transaction.

To save the raw messages of the transaction into a PCAP file, click the **PCAP** button.

Figure 6–4 shows the Transaction Details window.

Figure 6–4 Transaction Details



Message Flow

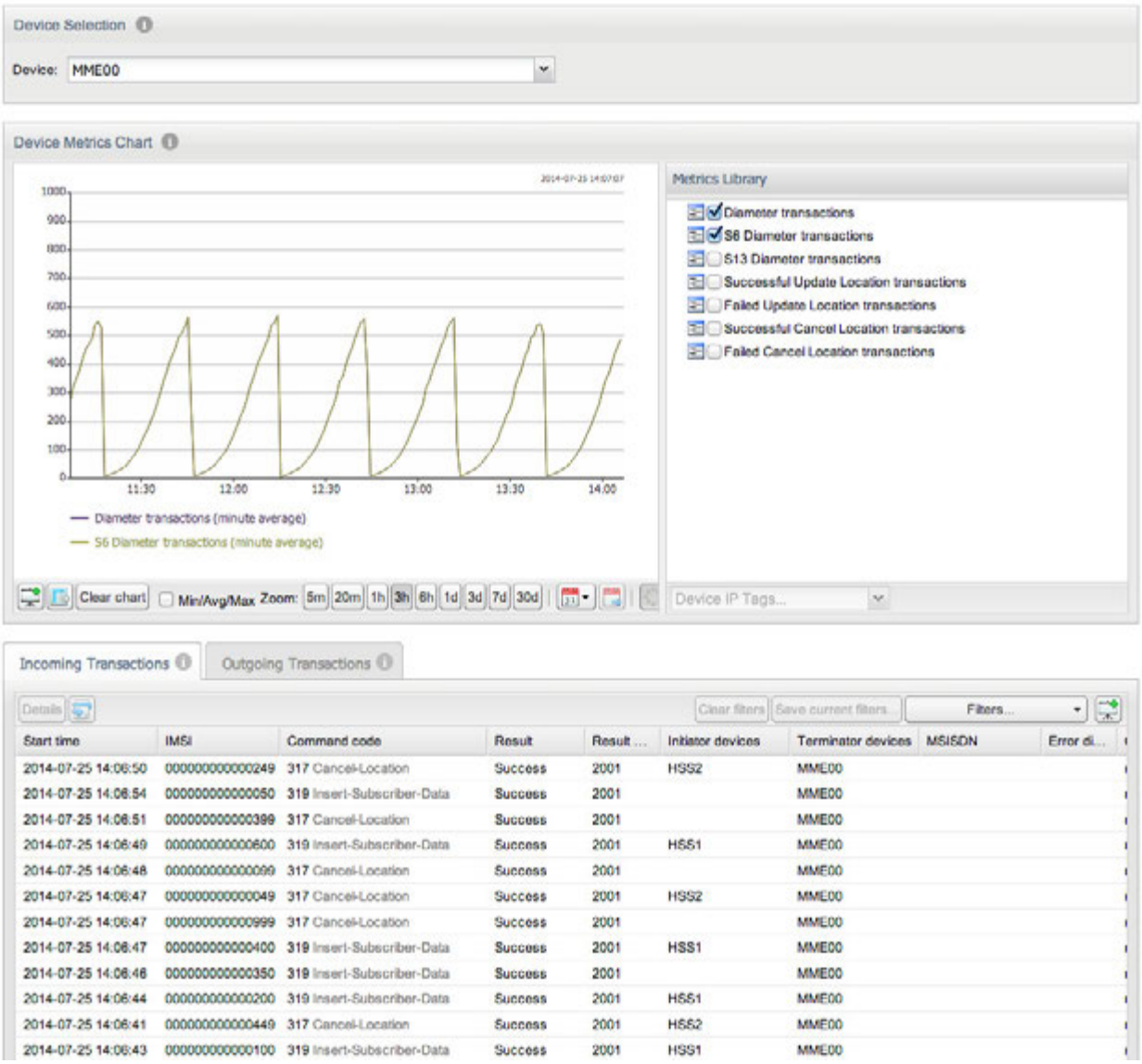
You can generate a *message flow diagram* by two means:

- Select one or more transactions from the **Recent S6/S13 Transaction** list and click the **Draw Diagram** button from the top toolbar. To select more transactions, hold the Shift key while selecting.
- Open the Transaction details window for a given transaction and click the **Message Flow** window.

Devices

The **Devices** page (see Figure 6–5) offers a device centric view of the Diameter transaction and KPIs.

Figure 6–5 Control Plane Monitor Devices Page



The **Device Selection** panel determines the device whose details are displayed on the page. Only configured diameter devices are available in this drop down. For example, HSS, Diameter agent, and Diameter proxy. For more information on the diameter devices, see Table 7–3.

The **Device Metrics Chart** displays the evolution of the selected device metrics from the *Metrics Library*.

You can add the **Device Metrics Chart** to the **Dashboard** by clicking the **Show in Dashboard** icon button.

At the bottom of the chart, the **Show Min/Avg/Max** check box controls whether minute or hour minima and maxima are displayed for each selected metric. For more information on the chart panel controls, see "Charts".

Incoming Transactions

The **Incoming Transactions** tab is similar with the **Recent S6/S13 Transactions** panel, which is described in "Transactions". The table displays only the transactions which have an inbound leg to the current device.

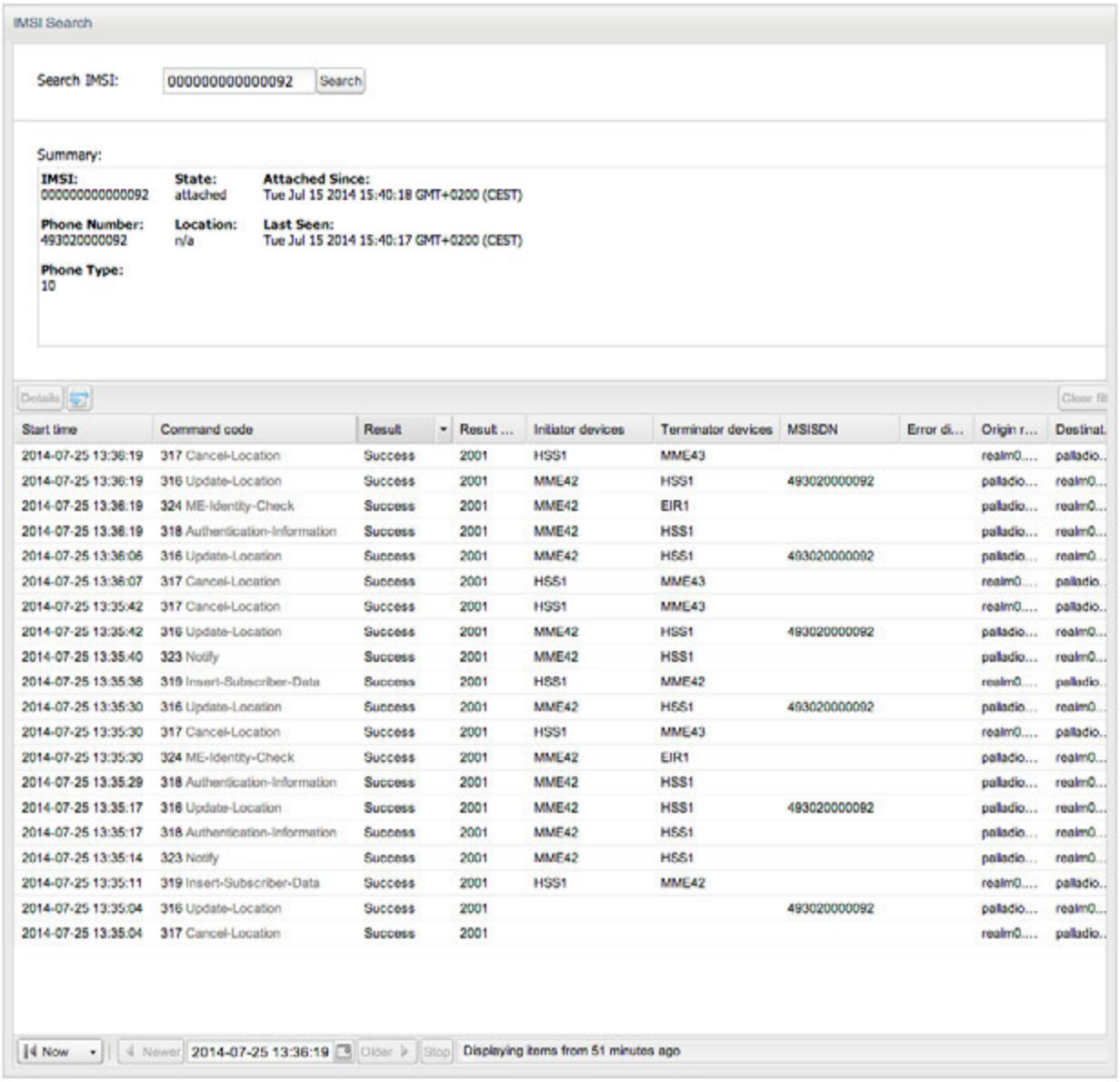
Outgoing Transactions

The **Outgoing Transactions** tab is similar with the **Recent S6/S13 Transactions** panel, which is described in "Transactions". The table displays only the transactions which have an outbound leg from the current device.

IMSI Search

The **IMSI Search** page (see Figure 6–6) displays information about a particular IMSI. To choose the IMSI you can either enter it in the search box from the top of the page or use the right-click menu from the **Transactions** list.

Figure 6–6 Control Plane Monitor IMSI Search Page



The **Summary** section provides high-level information about the selected IMSI number, including the current attachment state and the phone number.

The transactions list from the bottom is similar with the one from the **Transactions** page, but it is pre-filtered by the selected IMSI.

7

Settings

This chapter describes how to work with the *Settings* features of Oracle Communications Operations Monitor.

General Settings

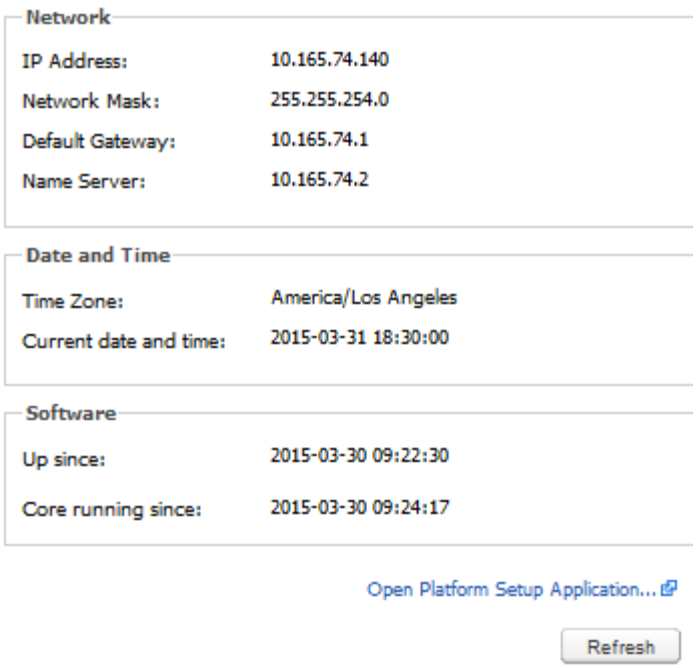
The **General Settings** section of the **Operations Monitor Settings** dialog box allows you to view the status of the Operations Monitor server, configure date and time, update the Operations Monitor software or license, view properties of the currently installed license and execute some system-wide actions.

Status

The **Status** page allows you to view some important information about the Operations Monitor server at a glance. To access the **Status** page, from the **Settings** menu, select **General Settings**, and then select **Status**. The **Status** page has three text boxes, *Network*, *Date and Time*, and *Software*, as shown in Figure 7–1

The **Network** text box displays information about the configuration of the primary network interface (IP Address, Network Mask, Default Gateway, and Name server). The **Date and Time** text box displays information about the configured Time Zone and the Current date and time settings on the Operations Monitor server. Finally, the **Software** text box displays the time since the Operations Monitor software has been up and the time since core has been running.

Figure 7–1 Status Page



Actions

The **Actions** page allows you to block alerts and reset the browser UI state. To access the **Actions** page, from the **Settings** menu, select **General Settings**, and then select **Actions**. To execute one of these actions, click the corresponding button on the **Actions** page (see Figure 7–2). A confirmation dialog box appears. Click **OK** to continue.

Figure 7–2 Actions Page

Table 7–1 lists the Action Page fields.

Table 7–1 Action Page Fields

Field	Description
Block alerts	Blocks the sending of alerts configured on the alerts page.
Reset UI state	For the user's convenience, Operations Monitor stores state information about the tables in HTTP cookies. The state information contains the hidden columns and the filtering criteria. This way the filters are kept even if the web page is refreshed or the browser is restarted. You can drop this UI state by using this action. It will delete all HTTP cookies used for state persistence.

External IP/hostname

The External IP/hostname is used as base for deep links in alerting emails. Typically, this is the hostname you see in the address field of your browser.

This page allows you to view and update the External IP/hostname. To access the page, from the **Settings** menu, select **General Settings**, and then select **External IP/hostname**. The **External IP/hostname** page contains one field in which you can enter new text. Enter text in the field and click **Update** to save the External IP/hostname.

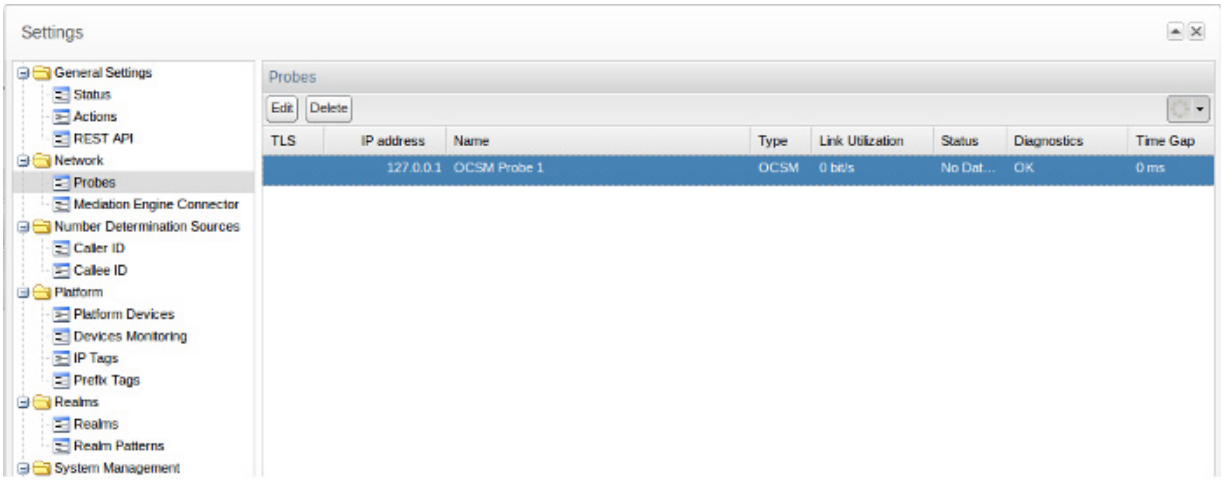
Network

The **Probes** page helps you define the remote agents that are pushing the data to the Mediation Engine of Operations Monitor. To access the **Probes** page, from the **Settings** menu, select **Network**, and then select **Probes**.

Probes

If your Operations Monitor installation has been configured to use probes on remote machines for traffic sniffing, you may view or edit those probes via this panel. By default, Operations Monitor will treat the local machine as its one and only probe (see Figure 7–3).

Figure 7–3 Network Probes Panel



SBC probes are not configured using Operations Monitor, but it is possible to edit their names and IP addresses using the **Probes** panel. Users can remove probes using the **Remove** button in the probes list dialog box, but if a probe still exists, it will eventually reappear in the list of probes. Use the column **Status** to verify if your settings are correct and Operations Monitor can connect to the probes.

There is a process that searches for new probes at a given time interval. If they are correctly configured they will appear in the **Probes configuration** panel. When a probe is removed from the network, it should also be removed from the list.

Table 7–2 lists the columns in the **Probes** panel and the information that is shown for each probe:

Table 7–2 Probe Panel Columns

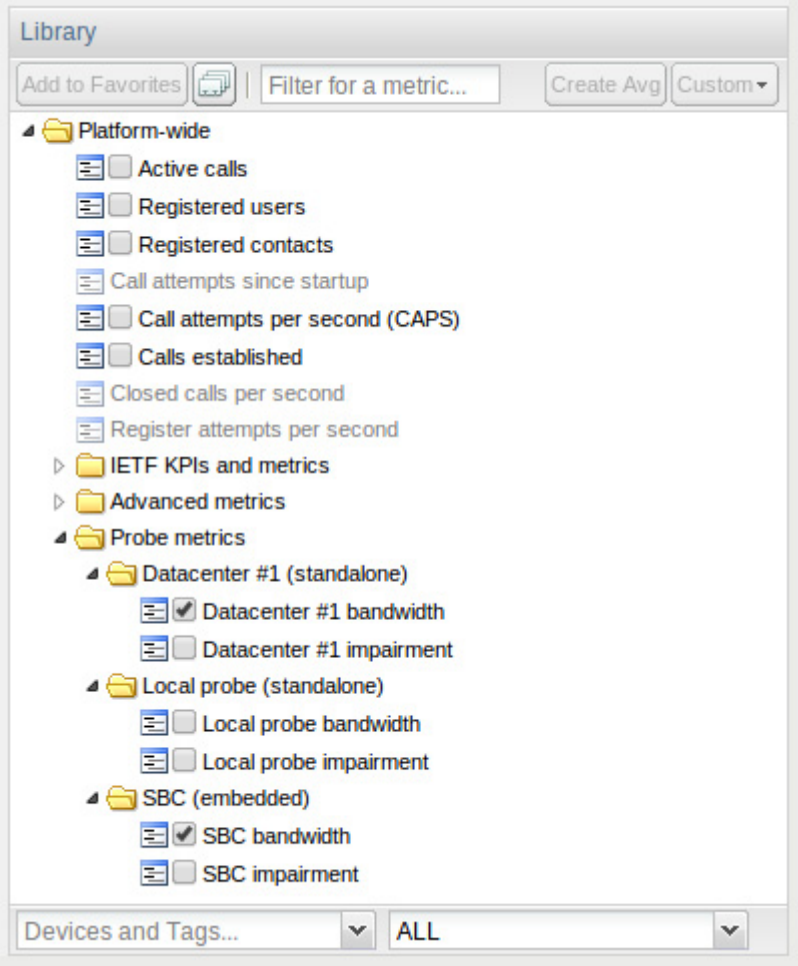
Column	Description
TLS	A lock appears if the connection with this machine is encrypted.
IP address	The IP address of the probe that will be used.
Name	A name to identify the probe. Can be edited by the user.
Type	Two types of probes are shown: ■ *OCSM*. Oracle Communications Session Monitor (Session Monitor) probes are either standalone probes or embedded probes in a Session Monitor Mediation Engine. ■ *SBC*. These kind of probes are configured on the SBC side by adding the IP of the Mediation Engine to the configuration. SBC probes will thus automatically connect to the Mediation Engine. An internal process is triggered every 60 seconds for embedded probes auto-discovery. This process runs in the background and the refresh button will not trigger the auto-discovery process. There is a "System Setting" for modifying the refresh interval. It is not recommended to alter the default value so as not to overload the system. If there is an embedded probe that has the same IP address as the Session Monitor probe, its configuration in Operations Monitor will be overwritten.
Link Utilization	This column shows the traffic volume in bits per second if the probe is reachable.
Status	The value of this cell depends of the data received. If no received data is shown, that may indicate a potential problem.
Diagnostics	Shows information about possible errors that can appear in probes configuration, such as IP conflicts and counters mis-configuration. In the latter case, a button will let the Administrators manually create the missing counters.
Time Gap	Absolute time difference between the Operations Monitor instance and the probe in milliseconds. If the difference is bigger than the established maximum value, the value will appear in red color and an alert symbol will appear in the dashboard, noticing that there is a significative time difference. The setting name for the time difference probe value alert is *vsp.probes.time_alert*.

There are two KPIs that are associated with each added probe:

- *Bandwidth*, which is the amount of traffic, both forwarded by the probe in bits/second. Includes signal and also voice if it was recorded and forwarded to the Mediation Engine (ME).
- *Impairment*, which is a Boolean value. Has the value of 1 for each minute that a problem has been detected with the probe. For more information on the problem the system log should be consulted.

Probe metrics are shown in the **KPI/Metrics** page, inside its own folder, as shown in Figure 7–4.

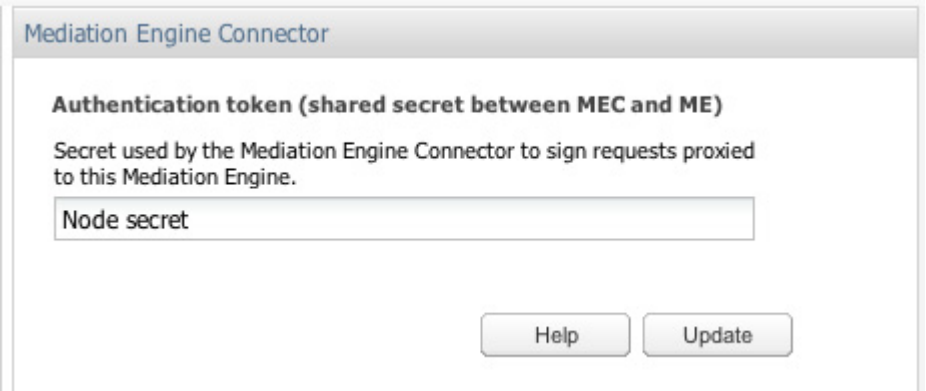
Figure 7–4 Library View of the Probe metrics Folder



Mediation Engine Connector

If your Operations Monitor installation has been configured to use a Mediation Engine Connector, the authentication token, which this Mediation Engine uses to authenticate itself to the Mediation Engine Connector, can be configured on this page as shown in Figure 7–5.

Figure 7–5 Authentication Token Configuration



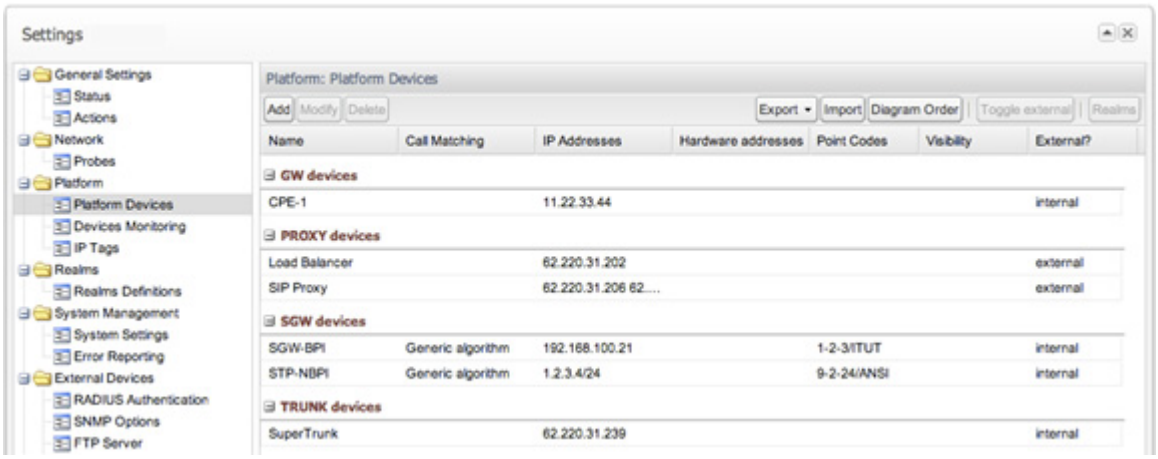
Platform

The **Platform** section of the **Settings** window contains configuration options for defining the monitored SIP network. Operations Monitor requires this information for accurately mirroring SIP messages.

Platform Devices

The **Platform Devices** section is used to define the topology of the monitored network. This section defines the SIP devices and how they behave in order for Operations Monitor to properly merge calls and provide accurate overall and per-device statistics (see Figure 7–6).

Figure 7–6 Platform Devices Settings



To add, modify, or delete a device:

- To add a new device, click the **Add** button from the upper toolbar. A wizard guides you through the device options.
- To edit a device, double click its table row, or select the device and click the **Edit** button. A wizard dialog box opens, pre-filled with the current values for the selected device.
- To delete a device, select it and click the **Delete** button. A confirmation window opens. Click **OK**.

Device Types

Operations Monitor supports the device types shown in Figure 7–7 and described in Table 7–3:

Figure 7–7 Device Type Settings

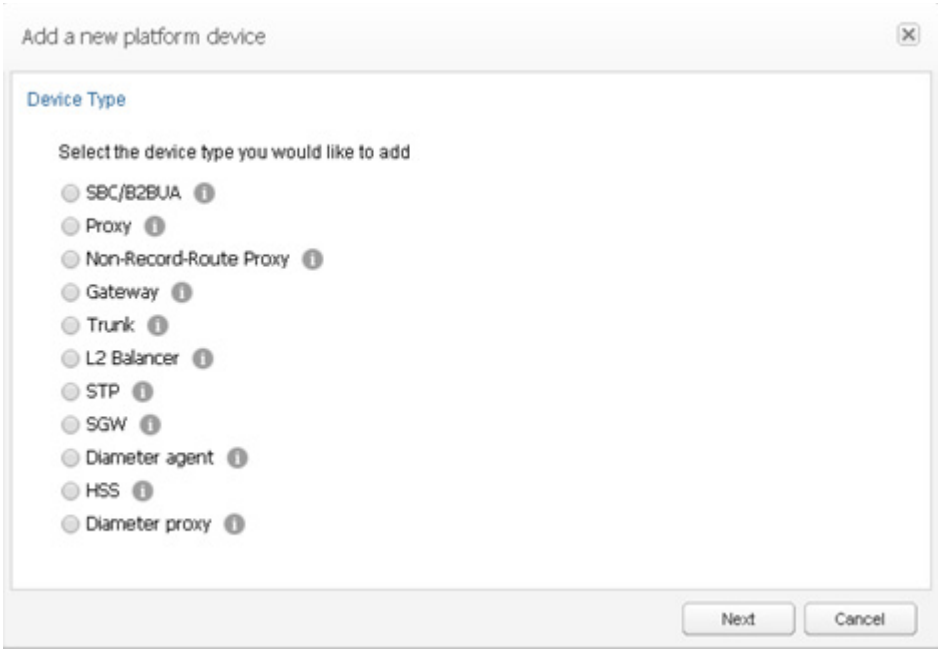


Table 7–3 Device Types

Device Type	Description
SBC/B2BUA	Select this option for devices that terminate an incoming leg and originate one or more independent outgoing legs from the same call (creating at least two SIP sessions). Examples include, session border controllers (SBCs), application servers (AS), and private branch exchanges (PBX). Because a B2BUA works by accepting the incoming call leg as a UAS, and creating a new leg as a UAC, it is difficult to determine whether two call legs are part of the same call. Therefore, on selecting this option, the wizard displays more configuration options for merging the call legs. For more information, see "SBC/B2BUA Call Merging".
Proxy	Select this option for devices that route calls without creating a new SIP dialog. The Call-ID in the outgoing call leg is usually left unchanged and SIP soft-switches and SIP layer load balancers are usually implemented as proxies.
Non-Record-Route Proxy	Select this option for devices that route INVITE messages but are skipped by the next messages in the dialog. These devices usually keep the Call-ID and the tag in the **From SIP** header field, as well as the URIs in the **From** and **To** headers unchanged.
Gateway	A SIP gateway is a device that connects the SIP platform to other types of networks (for example, PSTN, H.323). Operations Monitor only monitors the calls terminated and created by the gateway on the SIP side. Call merging is not created for gateway devices. If you need multi-protocol call merging, use an SGW (Signaling Gateway) device.
Trunk	This is not a physical device, but a peering with other carriers in which the peering IP addresses are known. Operations Monitor can display the calls going to and from this peer, as well as statistics about them. In addition to static IP addresses, the trunks can be defined using the OTG and DTG URI parameters.
L2 Balancer	A SIP load balancer acts at Layer 2 (of the OSI stack), keeping the IP addresses untouched and only modifying the hardware addresses. Operations Monitor supports these setups by identifying the devices by their hardware addresses in addition to the IP addresses.

Table 7–3 (Cont.) Device Types

Device Type	Description
STP	A Signaling Transfer Point (STP) monitors SS7/Sigtran routers and supports call merging for these protocols. For more information, see "SBC/B2BUA Call Merging".
SGW	A Signaling Gateway (SGW) transfers between different types of protocols. These multi-protocol signaling routers convert between ISUP and SIP and allow call merging between different protocols. For more information, see "SBC/B2BUA Call Merging". **Note:** A simple (or SIP) GATEWAY is always seen as a terminating device.
Diameter agent	Select this option if you are monitoring Diameter devices in Control Plane Monitor. For example, Mobile Management Entity (MME).
HSS	Select this option if you are monitoring Diameter messages exchanged with a Home Subscriber Server (HSS) in Control Plane Monitor.
Diameter proxy	Select this option if you are using an intermediate device to monitor a proxy that relays Diameter transactions in Control Plane Monitor.

If the device you want to create is not an SBC or B2BUA, go to "Device Identification".

SBC/B2BUA Call Merging

Call merging across B2BUA devices is a crucial point for SIP troubleshooting. Operations Monitor can use a large set of criteria for matching the calls, has several easy-to-use presets, and can also combine the criteria in a very flexible way by using custom algorithm expressions.

For STP devices, a similar choice of options applies. For SGW devices, the only options are the default or custom matching algorithm.

The wizard panel offers the high-level options shown in Figure 7–8 and described in Table 7–4:

Figure 7–8 Choosing the Call Matching Type for a B2BUA Device

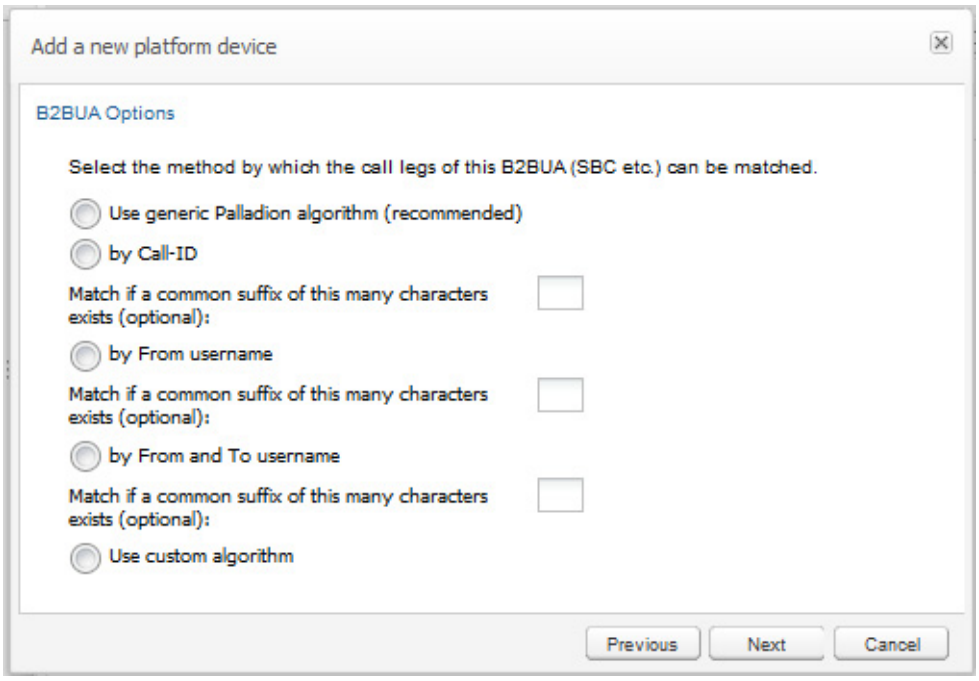
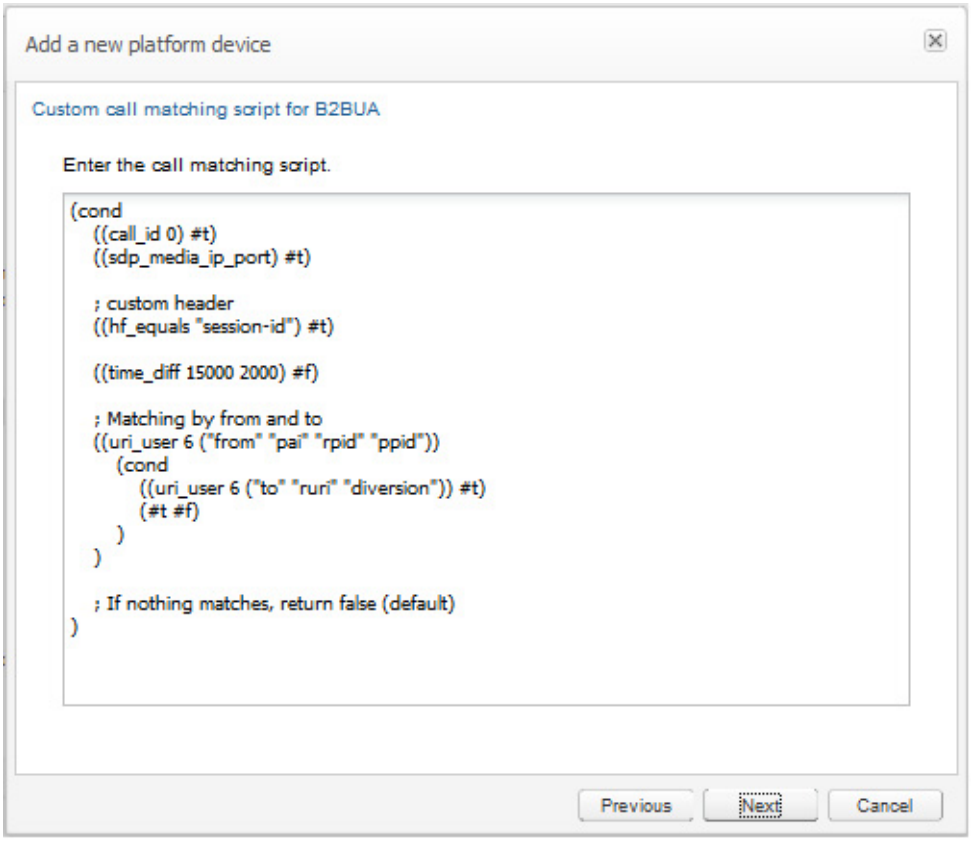


Table 7–4 B2BUA Device Options

Option	Description
Use generic OCOM/EOM algorithm	The algorithm was developed to accurately detect which call legs belong to the same call for the most common SBC/B2BUA configurations. The algorithm takes into account the time difference between the calls, the *session-id* from the SDP, the **From** and **To** users, the **P-Asserted-Identity** header, the **Diversion** header, etc. Oracle recommends using this option and to adjust if necessary. For more information about the generic algorithm, see "Call Merging Algorithms".
by Call-ID	To be used if the SBC/B2BUA is configured to copy the **Call-ID** header from the incoming *call leg* to the outgoing *call leg*. If the *suffix* parameter is given, then only the last *suffix* characters are compared. This is useful, for example, if the SBC/B2BUA only prepends a string to the *Call-ID* to make it unique.
by From username	To be used if the SBC/B2BUA is configured to keep the same **From** user between *call legs*, and this does not lead to false positives in most cases. If the *suffix* parameter is given, only the last *suffix* characters are compared. This is useful if the SBC/B2BUA only changes the prefix of the numbers (for example, from +49 to 0049).
by From and To username	To be used if the SBC/B2BUA is configured to keep the same **From** user and **To** user between *call legs*. If the *suffix* parameter is given, only the last *suffix* characters are compared. This is useful if the SBC/B2BUA only changes the prefix of the numbers (for example, from +49 to 0049).
Use custom algorithm	This allows flexible combining available criteria for matching, ordered tests, and nested tests, as well as matching on arbitrary SIP headers. For more information on how to write a custom algorithm, see "Call Merging Algorithms". If this is selected, another wizard panel will be shown for entering the algorithm (see Figure 7–9). The Operations Monitor generic algorithm is pre-filled as a starting point.

Figure 7–9 Entering a Custom Match Algorithm



Device Identification

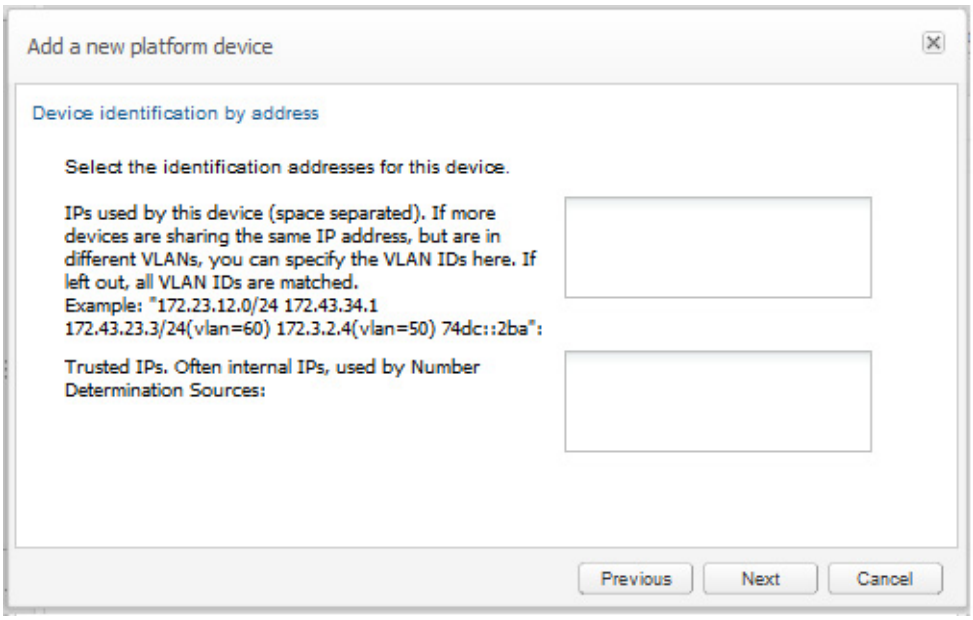
The devices are usually identified by only their IP addresses. Exceptions are the *trunk* devices, which can be also identified by the *OTG* and *DTG* parameters, and the set-ups in which the hardware addresses are significant. Either IPv4 or IPv6 addresses may be supplied. IP ranges with bitmasks smaller than 24 (for IPv4) or 120 (for IPv6) might impact performance significantly.

In the **Device identification by address** wizard panel, specify the identification addresses for the device you create. Enter the IPs used by the device in a list separated by spaces. If multiple devices share an IP address, but are in different VLANs, you can also specify the VLAN IDs.

You can specify VLAN IDs in multiple ways. You can enter:

- **IP(VLAN=*x*)** with *x*>=0, which matches packets with VLAN ID *x*.
- **IP** without a specified VLAN. The IP matches any packets, with or without a VLAN.
- **IP(VLAN=0)** or **IP(VLAN=off)** which only matches packets with no VLAN. This restricts the IP to only non-VLAN packets.

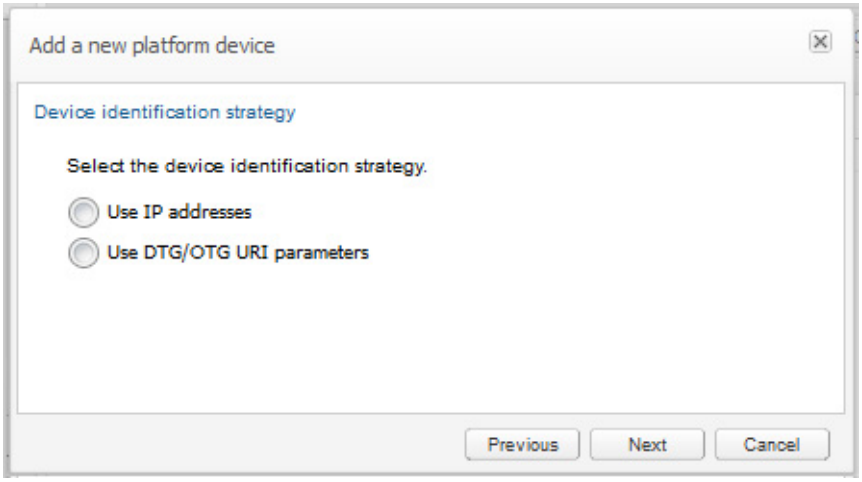
Figure 7–10 Device Identification Settings



Identifying Trunk Devices

For *trunk* devices, there is also the option of identifying by using the *OTG* and *DTG* parameters, offered by an extra wizard panel (see Figure 7–11).

Figure 7–11 Trunk Identification Settings



The parameters *OTG* and *DTG* are the originating and destination trunk identifiers found in the **From** header field of the *URI*, respectively, *Request-URI*.

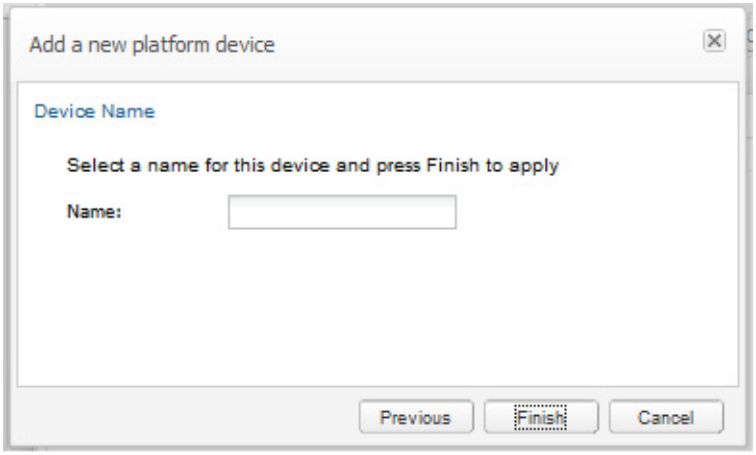
Trusted IP Ranges

For the Number Determination algorithm, only these IPs (or point codes) on each device are considered when identifying a call. Set this field to the source IPs for messages that you wish to take number information from.

Providing a Name

The last step in the creation wizard is to set a name for the device. The name may contain spaces, but it is recommended to keep the name short in order to fit in the table columns and SIP diagrams.

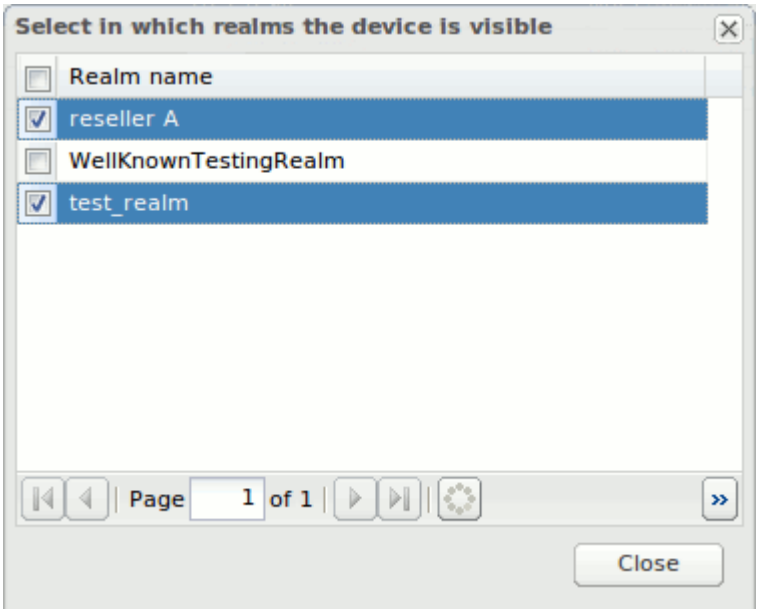
Figure 7–12 Device Name Settings



Visibility Configuration

You can configure a device or trunk to be visible or hidden for each realm. Click the **Realms** button in the toolbar to list the current defined realms as illustrated in Figure 7–13. If one or more realms are checked, the device/trunk will be visible only to the users belonging to those realms.

Figure 7–13 Device Visibility Settings



Configuring Devices for the Mediation Engine Connector

A Mediation Engine Connector allows a user to monitor multiple Mediation Engines that are located in separate geographic locations.

If a call is sent through a device monitored by two Mediation Engines, the Mediation Engine Connector sees the call flow through both Mediation Engines. In order for the Mediation Engine Connector to correlate the call, you must designate one Mediation Engine as internal, and the second Mediation Engine as external. The Mediation Engine Connector consolidates the data from both Mediation Engines and presents the single call flow.

The **Toggle external** button allows you to select a device and determine if the device is visible to the associated "internal" Mediation Engine, or visible to other "external" Mediation Engines in the Mediation Engine Connector platform.

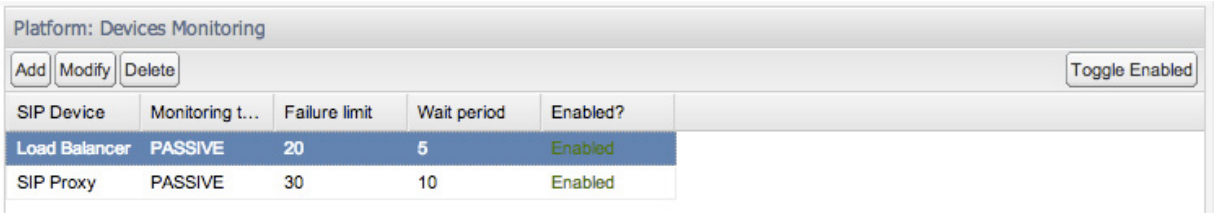
For more information on correlation, see "Understanding Call Correlation" in "Configuring Mediation Engines" in *Mediation Engine Connector User's Guide*.

For more information regarding the Mediation Engine Connector, consult your Oracle representative.

Device Monitoring

The **Device Monitoring** page is used for starting monitoring a device defined in "Platform Devices". Monitoring is realized by counting the number of requests that remain unanswered.

Figure 7–14 Device Monitoring Settings



Operations Monitor uses the following algorithm for monitoring SIP devices:

- A counter is kept with the number of *incoming* SIP requests to the monitored device, which have received no answer. These requests are *not* sent by Operations Monitor, but are regular requests from the SIP network.
- Whenever the monitored device sends a message (regardless of what type of message), the counter is set to zero, because the device must be *up*.
- If the counter exceeds a configurable threshold, a timer is started to fire after a configurable time interval.
- If the monitored device sends a message before the timer fires, the timer is canceled and the counter is set to zero again.
- If the timer fires, the monitored device is marked as *down*.

Enabling Device Monitoring

In the **Device Monitoring** table (see Figure 7–14), each line represents a device being monitored. To enable or disable the monitoring of any device, select the device in the table and click the **Toggle Enabled** button in the toolbar. The **Enabled?** column from the table shows the current status.

Adding and Editing Monitored Devices

Before you add a device to be monitored, you must configure the device in the **Platform Devices** settings section.

To add a new monitoring configuration, click the **Add** button from the upper toolbar. A wizard dialog guides you through the monitoring options.

To edit an existing monitoring configuration line, select the device and click the **Modify** button in the toolbar. A wizard dialog opens pre-filled with the current values for this monitoring configuration. If you want to permanently delete a configuration type for a device, select it from the table and click **Delete**.

Note: Device monitoring does not affect the performance or stability of the monitored device. It is immune to connectivity problems between the monitoring and the monitored device, reducing false positives.

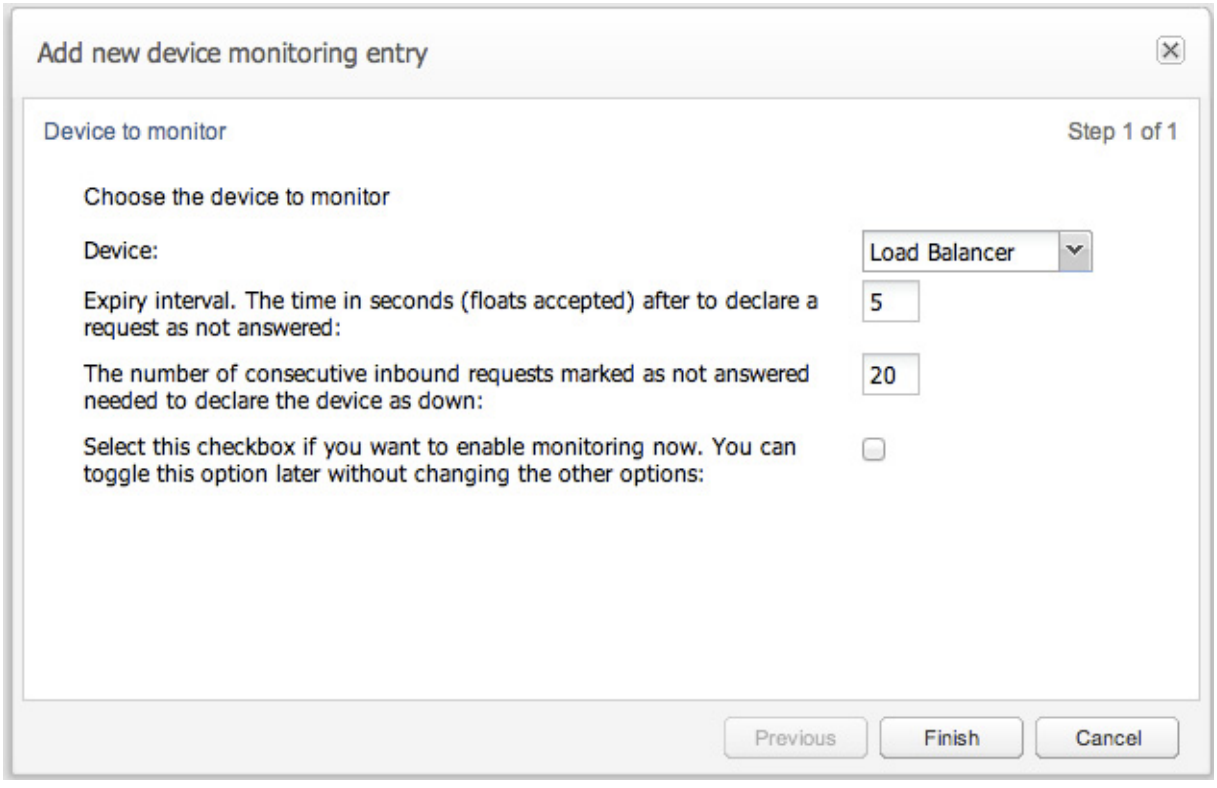
Device Monitoring Parameters

To configure a device monitoring, a couple of parameters are requested (see Figure 7–15):

- Device name to be monitored.
- Expiry interval of the timer.
- Number of unanswered requests needed to start the timer.

Additionally, you can enable device monitoring immediately, or leave it disabled until toggled from the list of monitored devices.

Figure 7–15 Device Monitoring Parameters



Realms Definitions

You configure *Realms Definitions* under the **Realms** section of the **Settings** menu. Each *realm* can have one or more *realm patterns* associated with it.

Realms can be used for limiting the visibility of Operations Monitor's web interface users. They are useful in scenarios such as:

- You have a set of resellers of the SIP service, and you want to give each of them the monitoring abilities of Operations Monitor, but at the same time restrict their view only to the SIP users served by them.
- You have a set of support workers, and you want to give them the fast troubleshooting abilities of Operations Monitor, but in the same time restrict their view to a certain set of users, for privacy reasons.

A *realm* can be defined as a set of *telephone numbers* or as a set of *domains*. The set of rules defining a *realm* are called *realm patterns*. Additionally, a custom header in SIP can contain a realm identifier.

> **Note:** *Realms* do not need to be disjoint. For example, you can have a single telephone number in multiple realms, and you can have one realm containing the union of other two realms.

If a user has his or her visibility restricted to one *realm*:

- The user only sees the registered subscribers that are part of the *realm*. The *User tracking* feature is restricted as well.
- The user only sees calls in which the caller or the callee are part of the *realm*.
- *Statistics* are adjusted to reflect only the **Realm** section, not the whole platform. For example, the **Active Calls** chart counts only the active calls in which one of the participants is part of the *realm*.

> **Note:** Voice quality statistics are an exception. Realm separation does not apply to voice quality statistics. Users assigned to realms see platform wide voice quality values.

- If the user creates a *trace*, it will only contain the messages that are from or to subscribers of the *realm*.
- *Alerts* are only sent to the user if the cause of the alert is part of his *realm*.

In other words, *realms* are segments of the subscriber base, and if an Operations Monitor user's view is restricted to a *realm*, Operations Monitor will only show information about that segment.

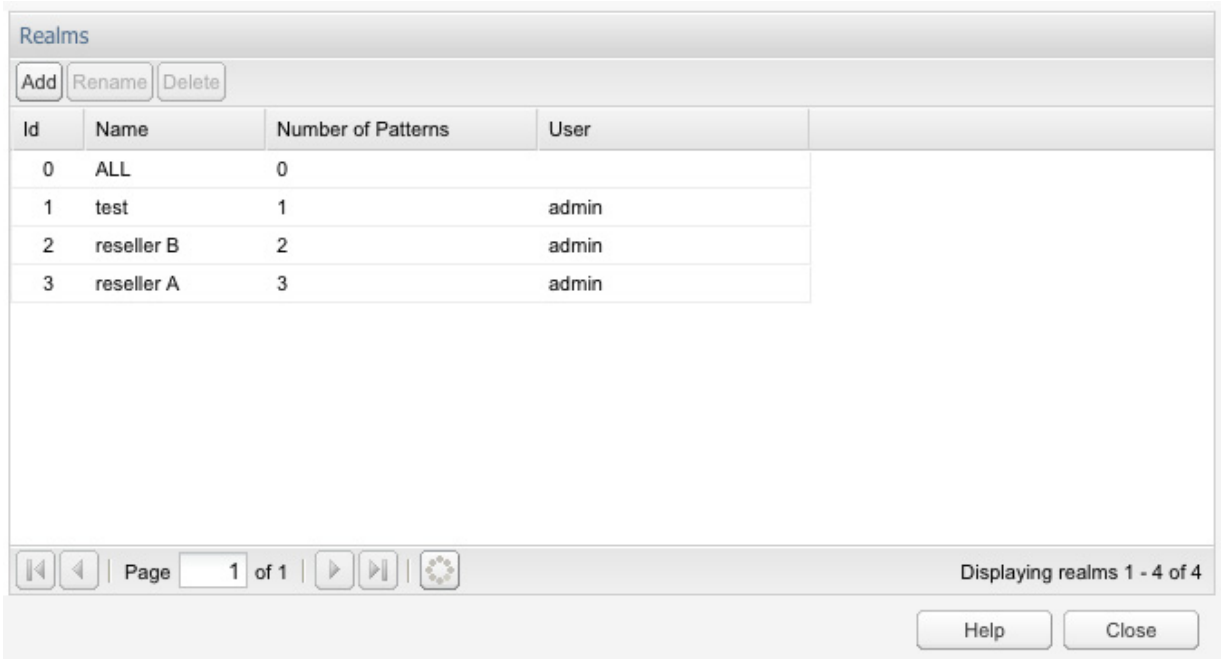
> **Note:** Realm names *cannot be deleted*, only the realm patterns.

Realms are not the only way to limit the view of the web interface users. You can also restrict features by using *user rights*. For more information, see "User Management".

Realms Table

Figure 7–16 shows the panel used for adding, editing and deleting *realms*. Realms can also be created from the Realm Patterns Table edit window (see Figure 7–17). Deleting a realm will also delete all its associated patterns.

Figure 7–16 Realms Definitions



Realm Patterns Table

Figure 7–18 shows the panel used for defining *realm patterns*. Selecting a realm pattern and then clicking the toolbar Edit button, or double-clicking an entry opens an editing window. For an example, see Figure 7–17.

Table 7–5 describes the fields found in the **Add new realm pattern** table:

Table 7–5 Add a New Realm Pattern Fields

Field	Description
Rid	The ID of the realm.
Name	The name of the *realm*. The edit select menu of the field offers the previous used *realms*, but also allows entering a new name. By entering a new name and pressing the **Create** button, a realm with that name is created immediately.
First number	The first phone number from a *number range realm*. If this is specified, then the *Last number* must also be specified. The realm will contain all numbers from the range.
Last number	The last phone number from a *number range realm*. If this is specified, then the *First number* must also be specified. The realm will contain all numbers from the range.
Domain	The domain from a *domain restricted realm*. The realm will contain all the users from the given domain. This can be combined with the *First number* and *Last number* options in which case all restrictions apply.

Table 7–5 (Cont.) Add a New Realm Pattern Fields

Field	Description
Custom Pattern	Give a custom header pattern for matching of calls to this pattern. If a call's invite contains this header pattern then it is counted in the realm. For more information, see "Realms Header Specification".
Creator	The user who created this realm.
Comment	(Optional) Comment line, only for convenience.

Important: In order for the *domain restricted realms* to work, Operations Monitor needs to be configured to consider the domains in the user identifiers. Set the **Use User Domains** to **true**. For more information, see "Use User Domains".

Note: Configuring a large number of realm patterns will affect the performance of the server.

Figure 7–17 The Add or Edit Realm Pattern Window

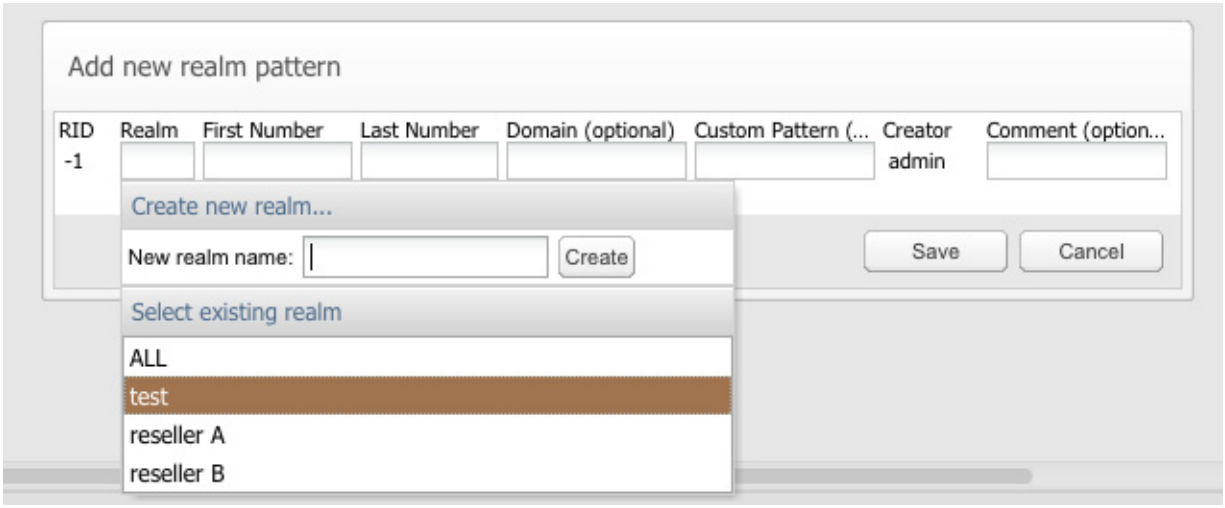
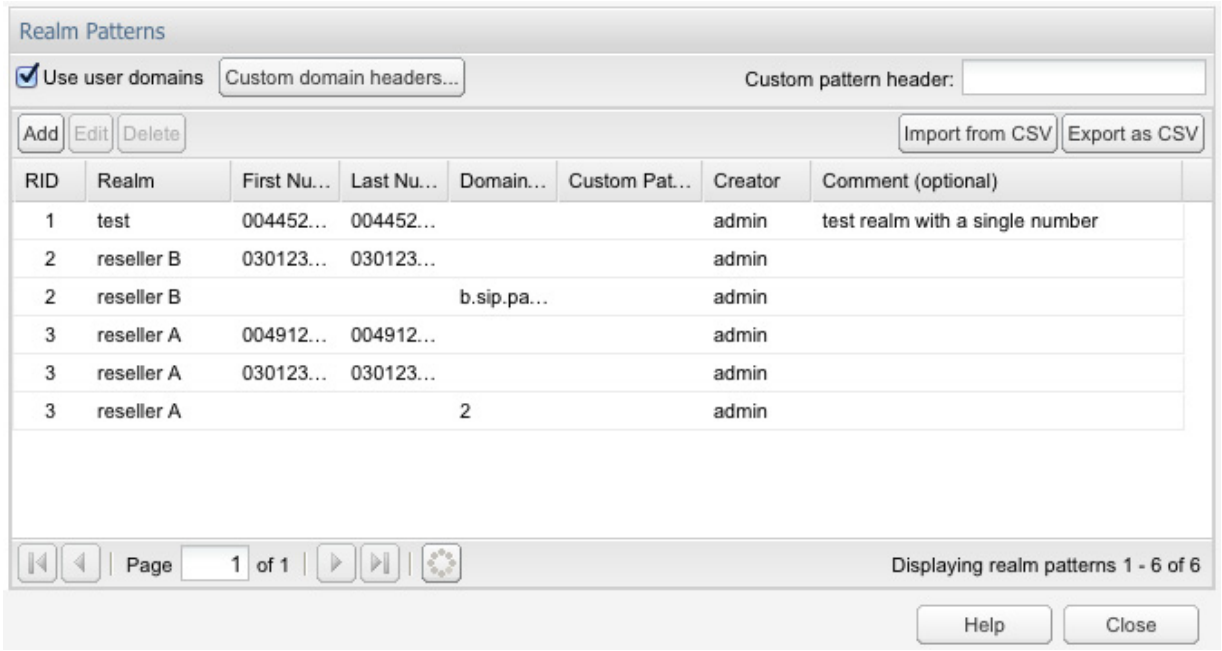


Figure 7–18 Realm Patterns Definitions



For example, the configuration from Figure 7–18 defines the realm *reseller A* to contain:

- The phone number range 00491234000 to 00491234999, regardless of the domain name.
- The phone number range 0301234000 to 0301234555, regardless of the domain name.
- All users that have **a.sip.palladion.net** as their domain name.

The *realms* configuration can be exported to and imported from *CSV* files. This simplifies the *realms* maintenance in case of many *realms patterns*. For example, the configuration in Figure 7–18 would be exported to:

```
name,min,max,domain,comments
reseller A,00491234000,00491234999,,
reseller A,0301234000,0301234555,,
reseller B,0301234556,0301234999,,
reseller B,,,b.sip.palladion.net,
reseller A,,,a.sip.palladion.net,
test,00445213000,00445213500,,Test realm with a single number.
```

The realms patterns can also be provisioned automatically by uploading a similar CSV file via FTP.

Realms Header Specification

There are two System Settings that modify the way Operations Monitor assigns *realms* to calls. By default Operations Monitor checks the **To:** and **From:** header for each SIP invite message associated with a call and associates this call with all matching realms ("or").

Specify additional header fields in the System Setting named **Headers in which to look for realm URIs**. **Diversion** or **P-Asserted-Identity** could be useful values, depending on the configuration. Operations Monitor generally checks all the

occurrences of each of these fields in each SIP Invite message. Exceptions are possible for header fields whose repetition is forbidden by the SIP standard.

Further customization is possible by matching a custom pattern against an arbitrary header field. Use the **Custom header for realm definition** System setting to specify which header to search for; use the **Custom Header Pattern (optional)** column on the **Realms Definitions** page to specify a pattern for each realm. The pattern matching works similar to the number matching.

Be aware that each additional header and the custom header matching are affecting Operations Monitor's performance.

Automatic Realm Pattern Import

You can create an external list of realm patterns and import the list into Operations Monitor. You can compose a list as a CSV file, upload the file by using FTP, and configure Operations Monitor to have the CSV file processed at a defined interval to import the realm patterns.

This is an alternate method of adding realms to Operations Monitor. Importing realm patterns from an external CSV file may be preferable if you have a large number of entries to add or if you make frequent changes to the realm definitions See Realms Definitions for more information on adding realms using the Operations Monitor interface.

Caution: Importing new realm patterns overwrites existing realm patterns. The import feature is not additive; rather, all numbers for all realms should always be present in the CSV file.

About Formatting the CSV File

Each line in the CSV file contains the realm name and the number pattern separated by a comma.

For example:

RealmName, *NumberPattern*

Assign each realm name and number pattern a unique value and order the list alphabetically. Only enter single number entries (even if there are number ranges belonging to the realm). Do not include duplicate entries. The import function of Operations Monitor compacts the numbers to the least possible amount of patterns by grouping them into the ranges.

Following is an example of a list of realm names and number patterns:

```
03940394, 444333222111
40404044, 404040404040
realmone, 133334444555
realmone, 133334444666
second, 12224444333
```

Sorting Realms

If you are unsure of the uniqueness or ordering of realm names and number patterns, you can use the UNIX **sort** function on the file you have created.

From the command line, enter:

```
/var/vsi/ftp/realms# cat list.csv | sort -u > ordered_list.csv
```

If you use the **sort** function in Operations Monitor, use the command to remove the original list after the sorting is complete. Since all entries must be unique, allowing the original list to remain following the sort would introduce duplicate entries. The original list would also exist in an unsorted state.

Use the following command to remove the original list:

```
/var/vsi/ftp/realms# rm list.csv
```

Importing Realms at a Daily Frequency

You can set a specific time at which the new realm patterns are automatically imported each day.

To enable importing realms at a daily frequency, at the command line:

1. Create a folder within the FTP directory:

   ```
   mkdir /var/vsi/ftp/realms
   ```

2. Open the **etc/iptego/vsp.conf** file and add the following text:

   ```
   [realms]
   import-enabled=1
   import-task-hour=hour
   import-task-minute=minute
   ```

   The *hour* and *minute* variables indicate the hour of the day and the minute of the hour that the feature processes the CSV file. Valid values for the *hour* field are 0 through 23. Valid values for the *minute* field are 0 through 59.

   The following example schedules an import to take place every day at 03:10 a.m. server time:

   ```
   import-enabled=1
   import-task-hour=3
   import-task-minute=10
   ```

3. Save and close the file.

4. Restart the respective process to make above changes active:

   ```
   systemctl restart pld-vsp.service
   ```

Editing the CSV File

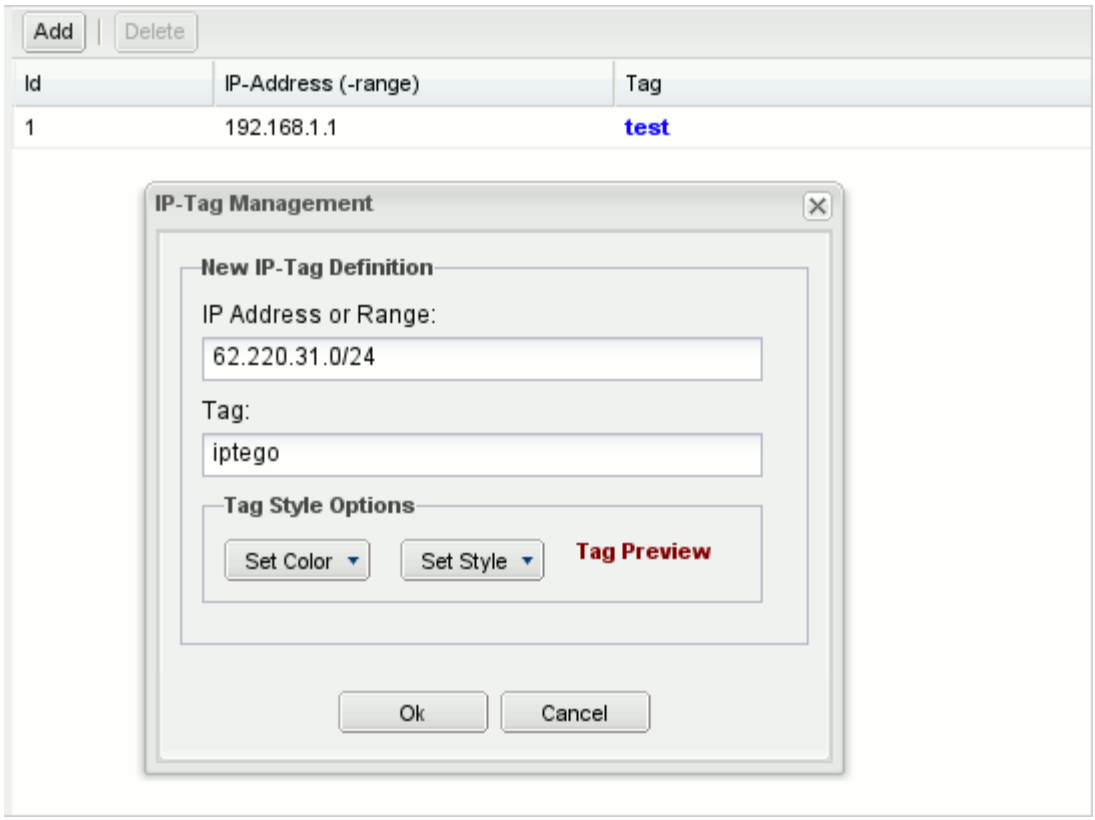
Edit the CSV file externally, then recommit it by using FTP to the **realms** folder. Make sure the CSV file is ordered and with unique entries. On the configured import time, the change will take effect.

There can be multiple CSV files present in Operations Monitor. Ensure that the entries are unique across all the CSV files.

IP Tags

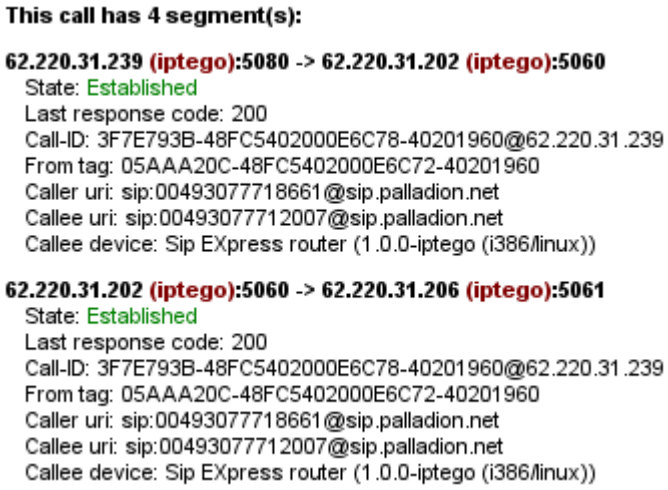
Figure 7–19 shows the New IP Tag Management window, which is used to add new IP addresses.

Figure 7–19 Adding a New IP Tag



IP tags are a way of assigning friendly names to *IP* addresses or *IP subnets* (either IPv4 or IPv6). The IP tag's name appears in brackets in the **Call Details** window, see Figure 7–20 for an example. The color and font style of the tags can be customized. IP tags are visible and editable by all users.

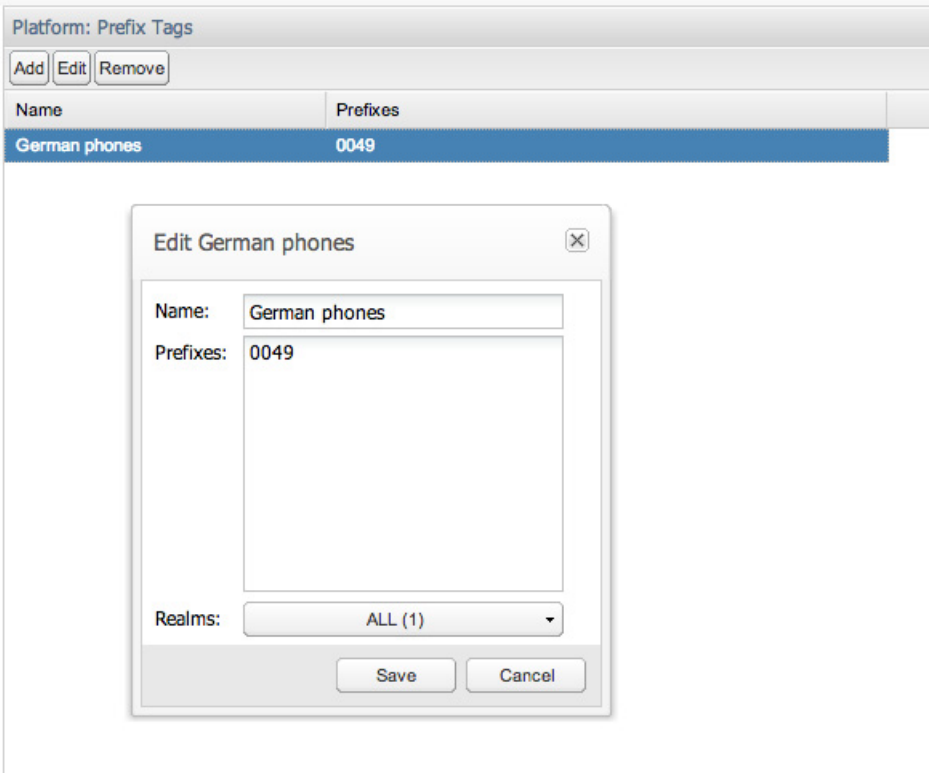
Figure 7–20 IP Tags Shown for Known Addresses



Prefix Tags

Figure 7–21 shows the **Platform: Prefix Tags** window, which is used to add new Prefix tags.

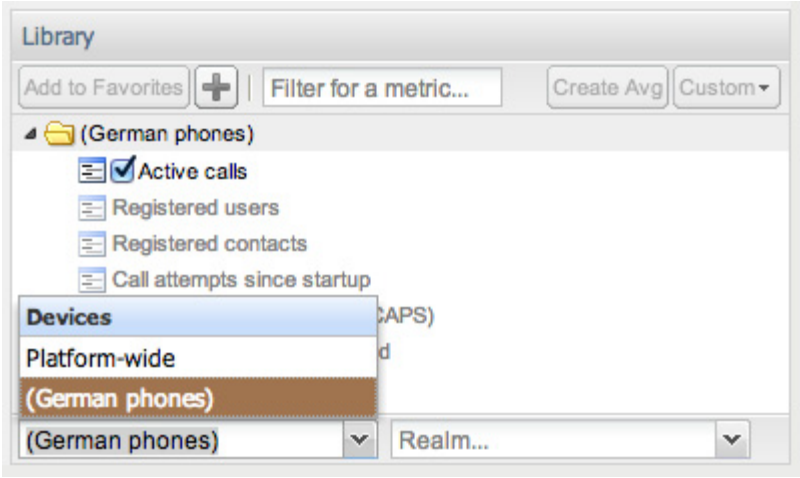
Figure 7–21 Adding a New Prefix Tag



Prefix tags allow assigning a friendly name to phone *prefixes*. Once the prefix tag is created, the user can create a KPI or metric on it, see Figure 7–22 for an example. You can, like with devices, configure a prefix tag to be visible or hidden for each realm, with the difference that prefixes in the *ALL* realm won't be visible for the user in another realm. For more information, see "KPI/Metrics".

When a prefix tag KPI is created in a realm, the value of the KPI is sourced from the calls matching both the prefix and the realm pattern. If you create a prefix tag in one realm and a KPI in a different realm, the KPI will not be visible to users who are restricted to one of the two realms.

Figure 7–22 KPI Based on a Prefix Tag

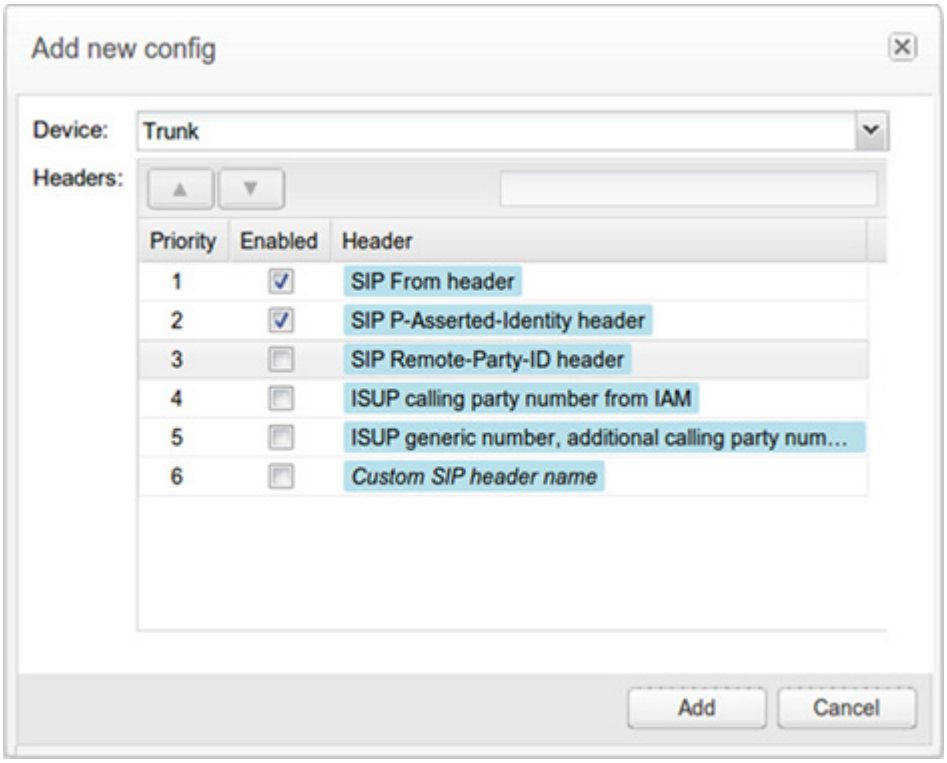


Number Determination Sources

This section of the configuration enables the definition of prioritized lists of trusted devices, from which phone number information may be taken for the caller and callee, along with the headers to be used (for example, From/To).

Note: Only data leaving a device via the Trusted IP ranges is considered for number determination purposes.

Figure 7–23 Enabling and Adding Prioritized Lists



System Management

Operations Monitor's system is configurable and maintainable through the **System Management** section of the **Settings** link in the upper right corner. The **System Management** section includes the following:

- **System Settings**: Use the System Settings page to configure Operations Monitor system parameters by adjusting the default values to meet your business needs. For example, you can adjust the session-timeout value that defines how long Operations Monitor keeps a call active if no SIP messages are received for the call.
- **Oracle SBC Config Upload**: Use the Oracle SBC Config Upload page to import the device configurations from an Oracle SBC configuration file into Operations Monitor or update existing Oracle SBC configurations.
- **Language Settings**: Use the Language Settings page to configure the default display language for all Operations Monitor users.

System Settings

System settings allows you to configure Operations Monitor's internal parameters and several advanced settings.

Important: Some of these settings may affect the performance and stability of Operations Monitor. Please consult with Oracle or your system integrator if you are unsure about a configuration.

Figure 7–24 is an example of possible system settings. A short description about every system option is displayed when you hover with the mouse over the name of the option.

Figure 7–24 System Settings

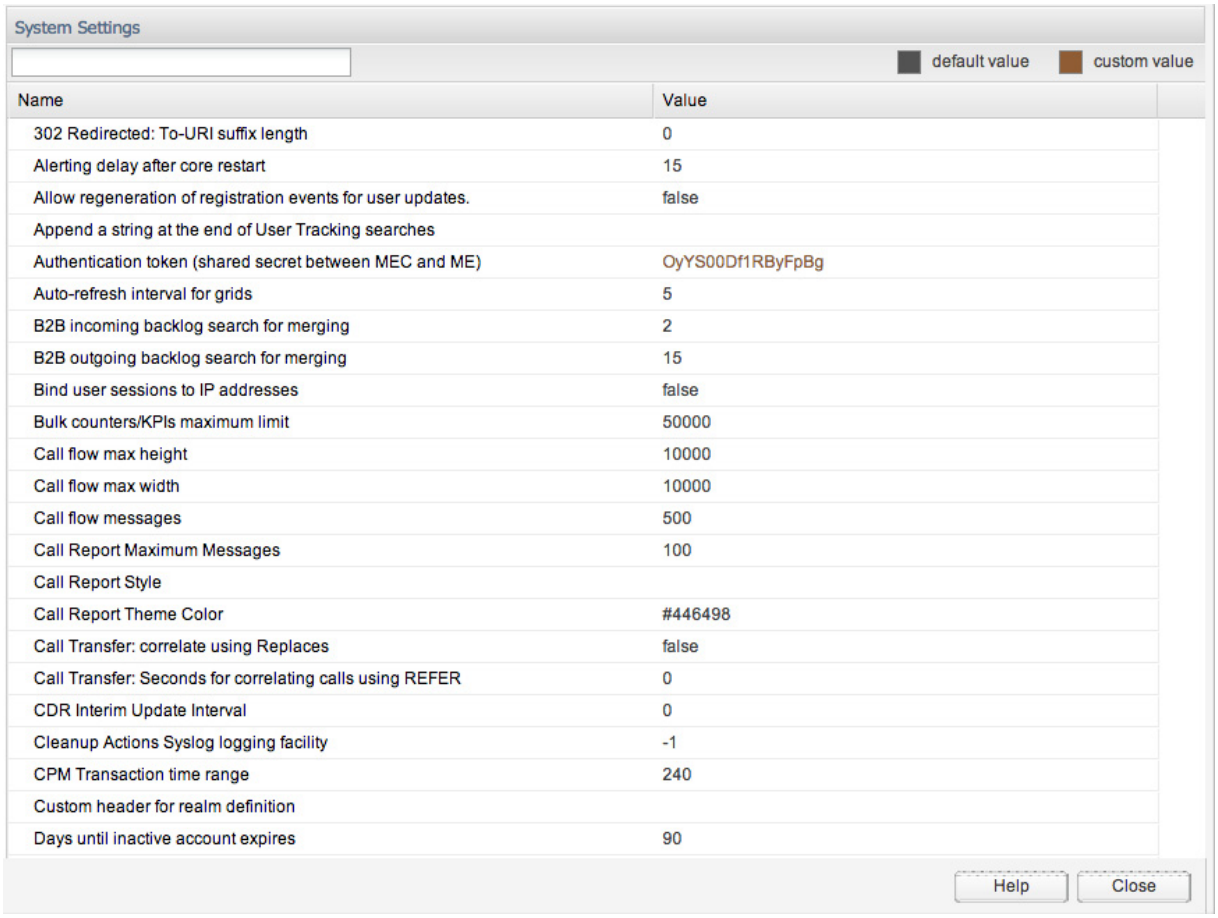


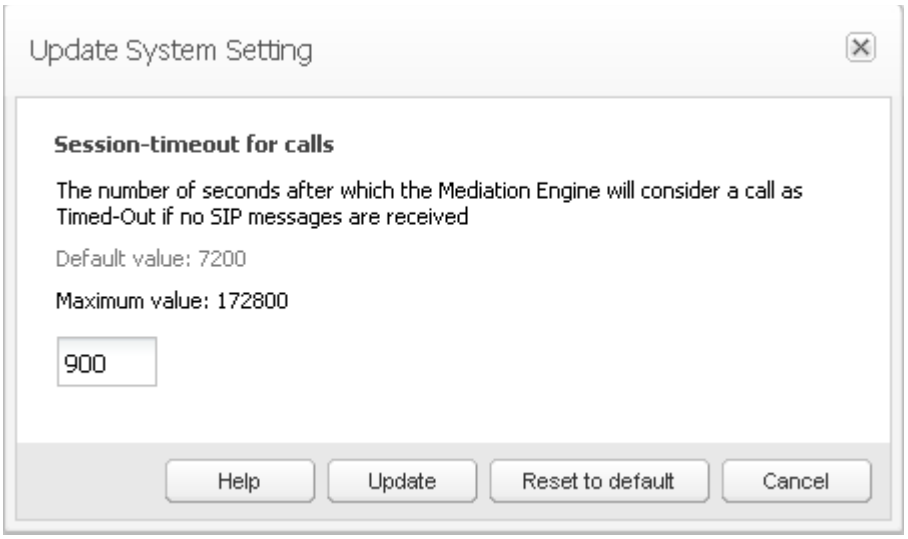
Table 7–6 describes the system options:

Table 7–6 System Options

Option	Description
Name	The name of the system option.
Status	Can be *user defined* if the option was changed by the administrator, or *default* if the option has the factory default value.
Type	The type of the system option. There are three types defined: ▪ string ▪ number ▪ boolean
Value	The current value of the setting.

To change the value for a setting, double-click the table row. A window similar to the one in Figure 7–25 is shown. The **Update System Option** window contains a short description of the system option, the default value, a maximum value in case of a number type option, and a value box or a check box in case of a Boolean type option. The **Update** button is for changing the system option to the new given value, and **Reset to default** resets the current value to the default one.

Figure 7–25 Update System Option



For a list of all system settings, see "System Settings Summary".

B2B Incoming Backlog Search for Merging

When doing call merging across B2B/SBC or Proxy device types, Operations Monitor is searching for the matching call in the recent calls going to or from that device. This options sets the default number of seconds to search the incoming call leg in the past when an outgoing leg is received.

> **Note:** If the call merging algorithm uses the `time_diff` function, then the values defined by it take precedence over this system setting.

Because the outgoing leg is normally seen by Operations Monitor after the incoming leg, we recommend setting a small value for this option.

B2B Outgoing Backlog Search for Merging

When doing call merging across B2B/SBC or Proxy device types, Operations Monitor is searching for the matching call in the recent calls going to or from that device. This options sets the default number of seconds to search the outgoing call leg in the past when an incoming leg is received.

> **Note:** If the call merging algorithm uses the `time_diff` function, then the values defined by it take precedence over this system setting.

Call Flow Messages

The maximum number of messages displayed in a **Message Flow** window. This number should be limited as a large amount of messages can significantly slow down the web browser. When the number is exceeded, a warning icon is displayed. By default, it is set to 100. The maximum value is 5000 messages.

Call Flow Maximum Height/Width

The maximum number of pixels (height or width) for rendered *Message Flow*. Limit the number of pixels if the diagrams take too long to render.

Call Flow Parallel Loading

If enabled, the *Message Flow* will render progressively, instead of all at once.

Calls Partition Size

As of release 2.10.0 the calls table in Operations Monitor is partitioned into several partitions. Operations Monitor only writes to one partition at a time. When this partition reaches its size limit, a new partition is created and subsequent calls are written to this partition. This setting gives the partition size limit in number of calls.

Call Report Maximum Number of Messages

The maximum number of messages to include in a *Message Flow* that is part of a call report (PDF).

Enabled Device Map

Operations Monitor can display a real-time map of the configured platform devices and couple of measurements of the traffic going through these devices. By default, the option is enabled, but please note that this map cannot work properly for a large number of configured device counters. It is therefore recommended to disable this setting when configuring more than 20 devices or realms.

Future Maximum Search

Sets a range of time in which new incoming calls can be included in a search you have initiated. Since new incoming calls can occur after a search has been initiated, this setting allows a range of time in which these incoming calls will be included in search results. The default setting is 10 seconds.

High Threshold for MOS

The upper threshold for good voice quality MOS value. Estimated MOS values higher than this threshold will be displayed in green in the Calls table. Lower values will be displayed in orange or red.

Group New Registrations from the Same User

If enabled, new registrations from the same user will be grouped together. However, the grouping is done per requested chunk, sometimes leading to duplicates still being displayed in the grid if the grid needs to do multiple requests to fill up a page. This is mostly notable when navigating back and forth through recent registration events, sometimes displaying different results for the same page.

Ignore Internal Registrations

A REGISTER SIP message exchanged between two devices from the platform is considered an internal registration. If disabled, registrations from inside the platform will be recorded. Otherwise, only the registrations between the end points and the first device from the platform are recorded. The default value is *disabled*. The value of this option must be the same as the "Registrations Gone Events," both enabled or both disabled.

Low Threshold for MOS

The lower threshold for good voice quality MOS value. Estimated MOS values lower than this threshold are displayed in red in the Calls table. Higher values are displayed in orange or green.

Match Registration Events by Comparing a Suffix of the Username

Operations Monitor matches the related registration attempts by the SIP username. If this value is 0, then the whole username is used. If non-zero, only that many of the last digits/characters are compared.

Set this to a non-zero if the Registration device is changing the SIP username/phone number when forwarding the message.

See also "Search for Matching Registration Segments".

Maximum Concurrent Sessions

The maximum number of parallel sessions a user can be logged into. If the limit is reached, the user cannot start additional sessions. In case the user has the permission to always log in, the user will still be able to log in, but for every new session that exceeds the limit the oldest of the user's existing sessions will be destroyed. For a description of permissions, see "User Management".

Maximum Stored Script Runs

The maximum number of stored app runs. Before an app is executed, scheduled or run manually, we check to make sure we are not violating this limit. The oldest stored app runs are removed as needed. This prevents frequently running or large result producing apps from depleting resources.

Maximum Script Output Size

The maximum size a script output can show. It is defined in bytes and limits the size that can be shown during a script run.

Maximum Simultaneous Script Runs

The maximum number of scripts that be running simultaneously.

Newest Search Cache

Checks how often the newest date in the search cache is refreshed. The default setting is 1 second.

Note: Changing this system setting can result in a loss of incoming calls.

Oldest Search Cache

Checks how often the oldest date in the search cache is refreshed. The default setting is 60 seconds.

Recurrent Alerts Threshold

Operations Monitor compares the time difference between a new alert and any previous unread alert that is identical (same definition and message) to avoid the creation of similar alerts within that period. You can specify the number of days for that period.

Registrations Gone Events

If enabled, when a REGISTER transaction occurs and not all previous known contacts of the user are seen in the successful answer, generates '**Gone**' *registration events* for the missing contacts. In other words, if the subscriber registers a new SIP contact without

explicitly de-registering the old one, a registration 'Gone' event is generated for the old contact. By default, this option is *disabled*.

> **Note:** In some networks in which the REGISTER messages are routed by intermediate devices that change the **Contact** header, this option can cause false **Gone** events. In this case, Oracle recommends you disable this option.

Search for Matching Registration Segments

When viewing the diagram of a *Registration* event, Operations Monitor searches for other related registration events. This option controls the time span during which Operations Monitor searches. If set to 0, the searching of related registration events is disabled.

This is useful for getting the *flow* of the registration request inside the platform. If you have devices that forward registration requests or that register themselves on behalf of the customer (as some SBC devices do), set this option to a non zero value. Also, make sure to disable the **Ignore Internal Registrations** option.

Session-Timeout For Calls

Session-timeout defines the number of seconds that Operations Monitor keeps a call active if no SIP messages are received for it. When the session timer expires, Operations Monitor marks the call with the state *Timed out*.

If the calls are not periodically refreshed in the monitored SIP network at SIP level, you might consider increasing this value. The default value is 2 hours (7200 seconds).

Timeout for Mediation Engine Querying

This setting gives a timeout for requests made by this Mediation Engine to its peer Mediation Engines. Ignore, if you use this Mediation Engine in standalone mode.

User Devices Ignore Internal Registrations

Similar as Ignore Internal Registrations, but for the *User Devices* statistics. There are cases when it is desirable to take the internal registration into account but not to count them for the User Devices statistics. An example of this is when all the internal registrations use the same User-Agent string. If the **Ignore Internal Registrations** option is enabled, this option is ignored.

Users Loose Searching

If enabled, when searching users by using the live search feature, the last digits of the number act as wildcards to suggest similar numbers from the same range. By default it is *disabled*.

Use User Domains

> **Important:** Changing this system setting requires a core restart, which will cause a temporary drop in all metrics.

If *enabled*, Operations Monitor uses the domain part of the SIP URI to differentiate the subscribers, in addition to the user name part/phone number. If the option is enabled, Operations Monitor also displays the domain part of the AOR of the subscribers in the

web interface. See Figure 7–26 and Figure 7–27 for comparative examples. By default, this option is *disabled*.

Figure 7–26 Calls Panel with Use Users Domains Disabled

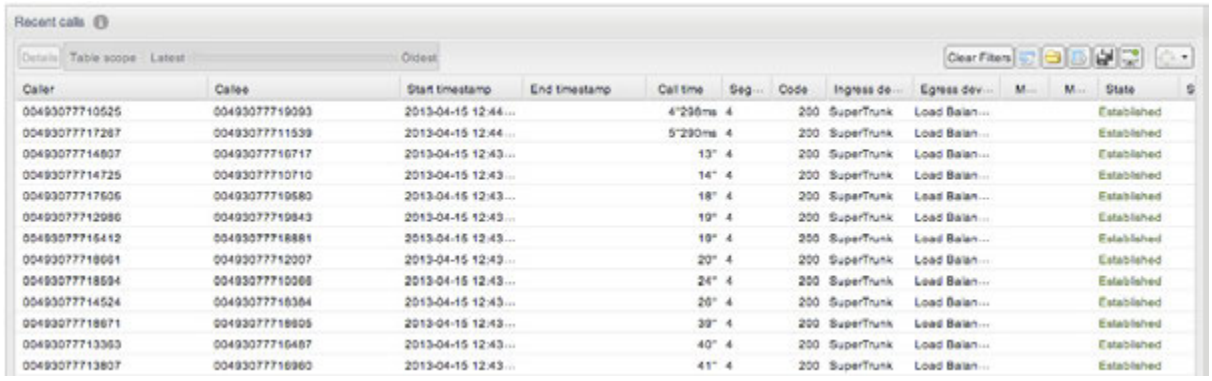
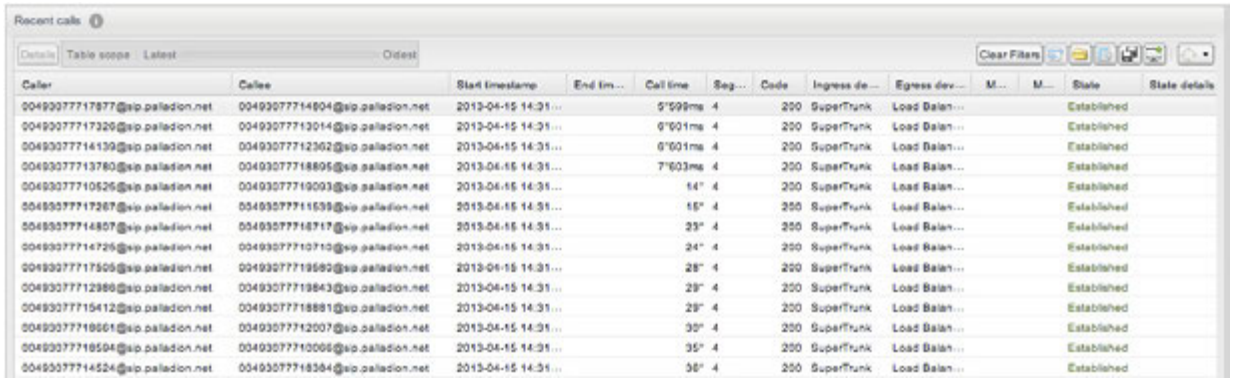


Figure 7–27 Calls Panel with Use Users Domains Enabled



Note: Realms definitions delimited by domain names only work if this setting is enabled.

Voice Quality Chart Scale in Minutes/Minute

Determines the scale of the **Voice Quality** chart. If true, the scale will be minutes/minute instead of minutes/15 minutes. Default is false (minutes/15 minutes).

Matching a Message to a Device

When a new message arrives, Operations Monitor tries to match it to a source device and to a destination device. The match is made based on the IP address, but there are some additional checks:

- If the message has VLAN tag N: in order for the message to be matched to a device, the device must have either VLAN=N or no VLAN configured.
- If source has specific MAC address(es) configured: in order for the message to be matched, the source MAC address of the message must match one of the MAC addresses configured for the device.

- If destination has specific MAC address(es) configured: in order for the message to be matched, the destination MAC address of the message must match one of the MAC addresses configured for the device.

Matching a Message to an Existing Call Leg

When a new message arrives, Operations Monitor tries to match it to an existing call leg. In order for it to match, the following conditions have to be fulfilled:

- In case both source IP and destination IP of the new message are configured as a device, the match to the existing leg is successful only if BOTH source and destination devices matched by the new message match the ones of the existing leg. For more information, see "Matching a Message to a Device".
- In case the source IP or destination IP of a new message is not configured as a device, the following settings are also checked:
  - Transport VLAN aware

    If this setting is 'true', Operations Monitor checks that the VLAN tag is the same (or that there is no VLAN tag) before matching this packet to an existing call leg. If this setting is 'false', the VLAN tag is ignored.

    Note: If the device is defined with **VLAN=0** or **VLAN=off**, then the check only matches packets with no VLAN. See "Device Identification" for more information on VLAN tag matching in device definitions.

  - Transport MAC aware

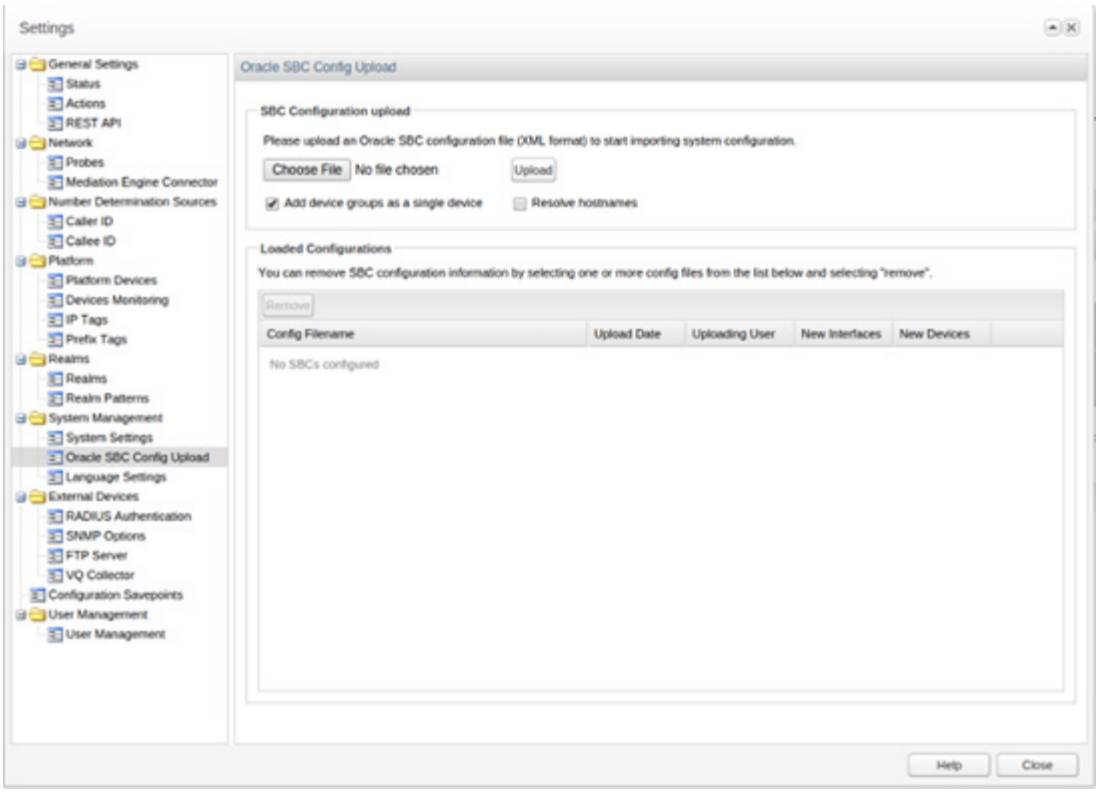
    If this setting is true, Operations Monitor checks that both source and destination MAC addresses are the same before matching this packet to an existing call leg. If one of the endpoints was already matched via device lookups, the MAC address check for that endpoint is skipped. If this setting is 'false', the MAC address is ignored.

Oracle SBC Config Upload

You can create new SIP trunks and devices based on an SBC configuration with the **Oracle SBC Config** function. The **Oracle SBC Config** function can also update existing SBC configurations. Every data point (for example, a **trunk**) that is added from an SBC configuration is marked in the configuration. This allows that all trunks created via an SBC configuration can be identified and later removed.

Figure 7–28 shows the **Oracle SBC Config Upload** page.

Figure 7–28 Oracle SBC Config Upload Page



The SBC Configuration upload section contains the following check boxes:

- **Add device groups as single device**

  If this check box is selected before an upload, the session agents in the configuration file that are part of a session agent group are added to Operations Monitor as a single device with multiple IPs.

- **Resolve hostnames**

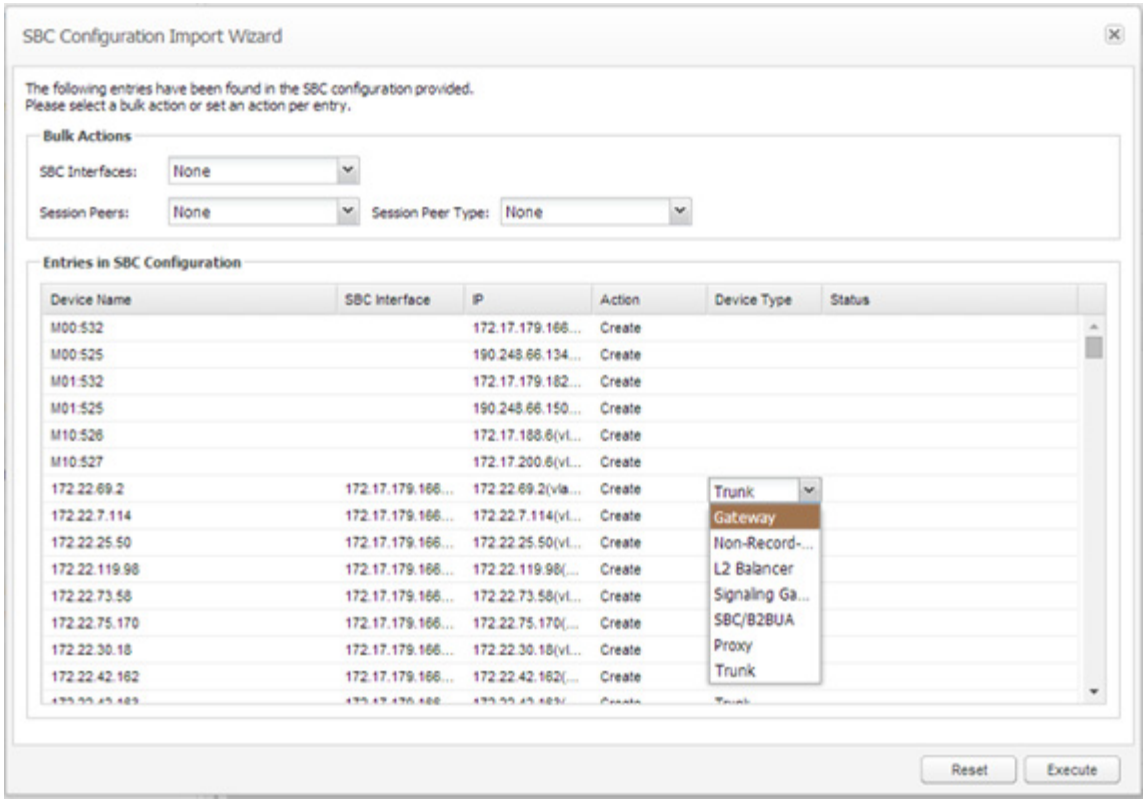
  By default, devices that do not have a numeric IP address specified in the configuration file, are skipped. If the **Resolve hostnames** check box is selected before an upload, Operations Monitor will try to resolve the hostnames for those devices.

  Note: This will increase the processing time for the change.

Import Window

The import window shows all the changes that could be committed based on the given SBC configuration.

Figure 7–29 SBC Configuration Import Wizard Page



The SBC Configuration Import Wizard dialog box contains the following:

- **Bulk Actions** section

  Bulk actions set the per data point action to be applied. It does not automatically execute the action but sets the action for the given data points accordingly.

  The Bulk Actions section contains the following fields:

  – SBC Interfaces

  – Session Peers

  – Session Peer Types

- **Entries in SBC Configuration** table

  This table lists all the entries that were found in the configuration.

  Existing entries are marked with the **Create** check box disabled so that duplicates cannot be created. Valid options are **Create** and **Ignore**. All entries are identified by their IP or IP pair (in case of trunks) address.

  The user can double-click a device to rename it.

- **Execute/Reset** buttons

  The **Execute** and **Reset** buttons at the bottom of the page are used to execute all actions on the page or reset to the default values.

Language Settings

The **Language Settings** page configures the default language for users, as shown in Figure 7–30.

Note: Each user can have an individual language setting that overrides the default setting.

Figure 7–30 Language Settings

External Devices

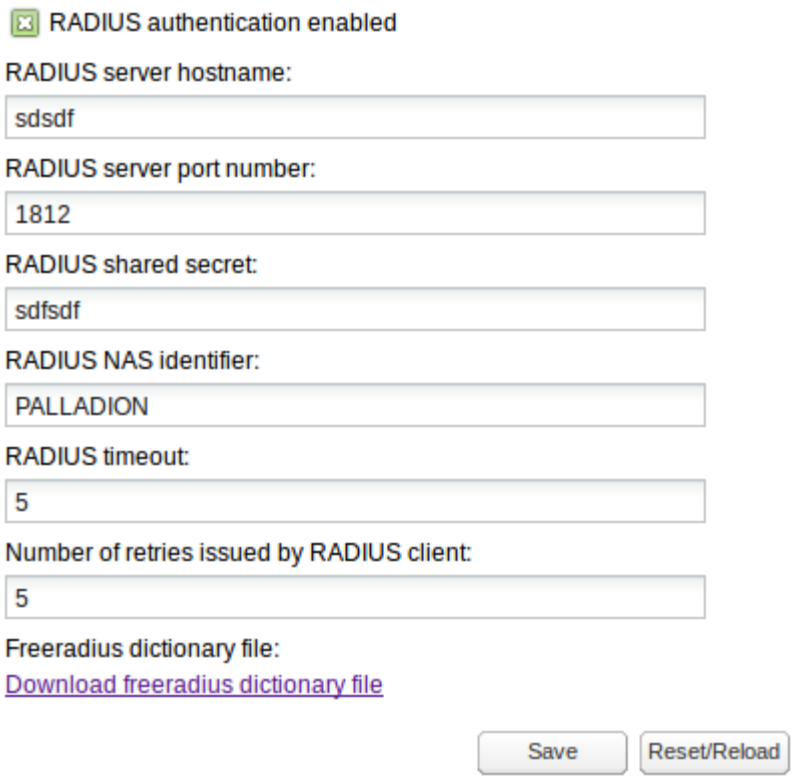
You can configure communication with external devices in the **External Devices** section, which includes the following sub-sections:

- RADIUS Authentication
- SNMP Options
- FTP Server
- Voice Quality Collector

RADIUS Authentication

RADIUS authentication allows you to configure whether or not Operations Monitor will perform RADIUS authentication against a RADIUS authentication server each time you log in. See Figure 7–31 for an illustration:

Figure 7–31 Radius Authentication Settings



To enable RADIUS authentication on an Operations Monitor installation, click the **RADIUS authentication enabled** check box. When enabled:

- The **RADIUS server hostname** and **RADIUS server port number** fields control the address of the RADIUS server against which Operations Monitor performs authentication.
- The **RADIUS shared secret** field must contain the secret that is shared by Operations Monitor and the RADIUS server used for authentication.
- The **RADIUS NAS identifier** is a string identifier and is used by the RADIUS server to determine whether or not the request is coming from Operations Monitor.
- The **RADIUS timeout** is number of seconds to wait response from RADIUS server.
- The **Number of retries issued by RADIUS client** is the number of attempts to connect to server, expressed as positive integer value. Number of retries multiplied by timeout value should not exceed 30 (seconds), otherwise both fields (timeout and number of retries) will be marked as invalid.

For more information about RADIUS authentication in Operations Monitor, see "Requirements for RADIUS Authentication".

SNMP Options

The **SNMP Options** subsection allows you to configure:

- Whether Operations Monitor sends SNMP traps.
- Whether Operations Monitor can be queried using SNMP.

Regarding the queries Operations Monitor can answer, they are in turn split in two:

- All the counters of Operations Monitor are exported as gauges.
- All operating parameters of the system of Operations Monitor.

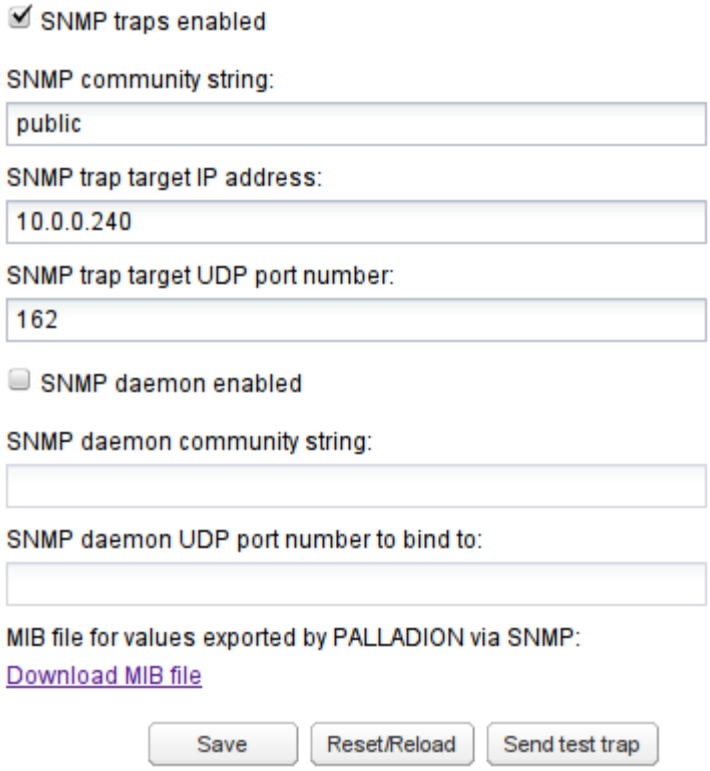
SNMP Traps

To enable SNMP traps, click the **SNMP traps enabled** check box as shown in Figure 7–32. When enabled, SNMP traps can be sent on several *Alerts*. You must configure the SNMP community name, IP address, and port where the SNMP traps will be sent.

To test the settings, a test trap can be sent by clicking **Send test trap**. To send SNMP traps to more than one target, you enter more than one SNMP trap target IP address.

If you specify more than one community string, Operations Monitor will use the first community string and the first IP address, and so on. You cannot specify more than one port.

Figure 7–32 SNMP Traps Settings



SNMP traps are interpreted using the *MIB* file, which is downloaded from the **SNMP Options configuration** page. See Figure 7–32 for an illustration.

The traps include descriptive text which interprets the trap as well as the subsystem that caused the trap to be generated. Included in this description are the current value assigned to the system (value of the counter, IP address, actual user or device name), and the severity of the trap.

Some instances that generate traps are:

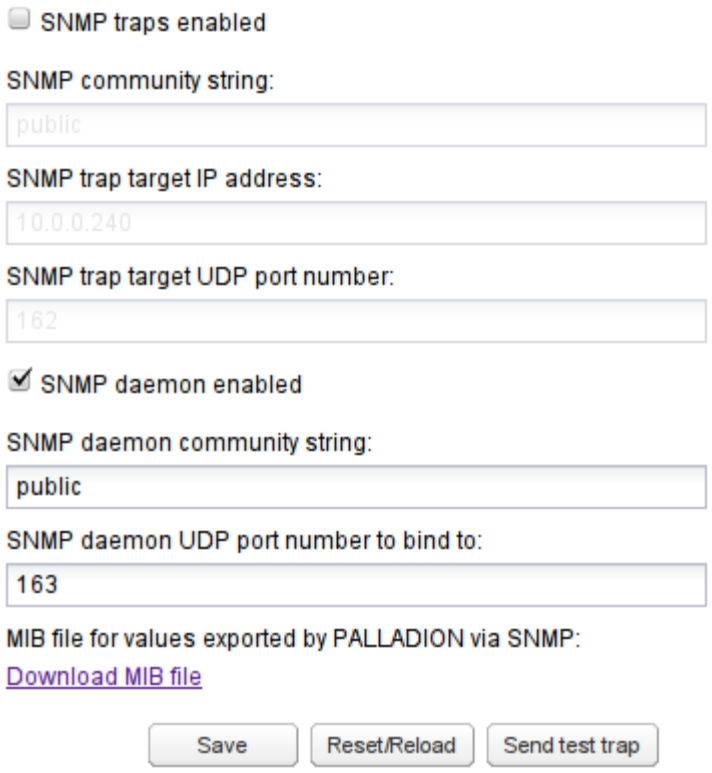
- Counters exceeding thresholds.

- IP addresses or user names meeting certain conditions.
- Failing network devices.

SNMP Daemon

To enable SNMP Daemon, click the **SNMP daemon enabled** check box. With the SNMP daemon enabled, you can query Operations Monitor using SNMP. See Figure 7–33. For SNMP Daemon to work properly, you must also specify the community name and UDP port, which listens for SNMP requests coming in. Operations Monitor supports SNMP Version 2 over UDP.

Figure 7–33 SNMP Daemon Settings



Exported Counters

The counters are available in the MIB tree under the OID: **.1.3.6.1.4.1.29631.2.2.0.256**. Enter the following command to list all counter information using Net-SNMP's snmpwalk:

```
$ snmpwalk -Os -c public 10.1.0.78:162 -v 2c \
   .1.3.6.1.4.1.29631.2.2.0.256
```

The next node is the counter ID. Under it, the counter object has the following fields as listed in Table 7–7:

Table 7–7 Counter Object Fields

Field	Description
.0.0	The counter ID.
.1.0	The name of the counter.

Table 7–7 (Cont.) Counter Object Fields

Field	Description
.2.0	The device name (if it is a device counter).
.3.0 .4.0 .5.0	The counter parameters.
.6.0	The current second value.
.7.0 .8.0 .9.0	The last minute average/min/max.
.10.0 .11.0 .12.0	The last 5 minutes avg/min/max.
.13.0 .14.0 .15.0	The last hour avg/min/max.

For example, here is how to get the active calls counter. Note, that the ID of the counter (422 as used in this example), will vary from one box to another:

```
$ snmpwalk -Os -c public 10.1.0.78:162 -v 2c  \
     .1.3.6.1.4.1.29631.2.2.0.256.442

enterprises.29631.2.2.0.256.442.1.0 = STRING: "Active calls"
enterprises.29631.2.2.0.256.442.3.0 = INTEGER: 0
enterprises.29631.2.2.0.256.442.4.0 = INTEGER: 0
enterprises.29631.2.2.0.256.442.5.0 = INTEGER: 0
enterprises.29631.2.2.0.256.442.6.0 = Counter64: 136
enterprises.29631.2.2.0.256.442.7.0 = Counter64: 13751
enterprises.29631.2.2.0.256.442.8.0 = Counter64: 133
enterprises.29631.2.2.0.256.442.9.0 = Counter64: 141
enterprises.29631.2.2.0.256.442.10.0 = Counter64: 10319
enterprises.29631.2.2.0.256.442.11.0 = Counter64: 66
enterprises.29631.2.2.0.256.442.12.0 = Counter64: 141
enterprises.29631.2.2.0.256.442.13.0 = Counter64: 10068
enterprises.29631.2.2.0.256.442.14.0 = Counter64: 63
enterprises.29631.2.2.0.256.442.15.0 = Counter64: 141
```

Note: When using snmpwalk the following should be considered:

- Because of large values, the Counter64 type is used, but this is not available in SNMPv1. You must use SNMPv2 or later.
- Even with Counter64 type in use, the counters act like gauges in SNMP talk. For example, over time the value of the counter is presented, and not the sum.
- The averages are computed per-second. For example, for the last 5 minutes, all the values measured every second are added and then divided by 300. Since we cannot transport floats over SNMP, the avg * 100 is returned. In *cacti <http://www.cacti.net/>*, you would divide back with 100. In this example, the last 5-minute average, or .10.0, means: On average, there was 103.19 active calls in the last 5 minutes (measured every second).

We recommend this when creating the graphs. The min and max values might also be informative.

System Monitoring

Operations Monitor runs Net-SNMP which provides a large set of monitoring objects for the Operations Monitor system itself. For example, you can get the UC Davis *http://www.net-snmp.org/docs/mibs/ucdavis.html* values with:

```
$ snmpwalk -Os -c test1 10.1.0.72:161 -v 2c .1.3.6.1.4.1.2021
```

As these are standard Unix MIBs, most SNMP managers are able to interpret them.

FTP Server

You can create CDRs, which are based on the traffic displayed and the platform topology configured. For more information, see "Public CDR Generation".

The CDRs generated CDRs, can be retrieved and deleted using FTP or FTPS. To enable access via FTP or FTPS, click the **FTP access enabled** check box. You must also configure a username and password.

Voice Quality Collector

Operations Monitor can gather the voice quality information sent by IP phones in SIP PUBLISH messages and display it on the **Voice Quality** page.

As defined in RFC 6035, SIP User Agents may send voice quality reports derived from RTCP Extended Reports (RTCP-XR) and Operations Monitor acts as the collector of SIP vq-rtcpxr events using the PUBLISH method.

The vq-rtcpxr event collector is configurable as shown in Figure 7–34.

Figure 7–34 Configure VQ Collector

To configure the voice quality collector:

1. Select the **VW collector enabled** check box to enable the collection of voice quality metric reports.
2. In the **VQ Collector IP address** field, enter the IP address that is used by the vq-rtcpxr event collector to listen for PUBLISH messages.
3. In the **VQ Collector port number** field, enter the port number.
4. Click **Save**.

> **Note:** The same collector must be configured at the IP phone in order to let it send the voice quality metric reports to Operations Monitor.

Configuration Savepoints

Operations Monitor allows you to create configuration snapshots. Savepoints include most of the configuration values:

- Settings dialog sections:
  - Platform
  - System Management
  - External Devices
- Operations Monitor user accounts
- Counters currently configured on pages:
  - Monitoring
  - Devices

> **Note:** Custom KPIs and Alerts are not saved in savepoints.

The management functionality for configuration savepoints is only available to the administration user. Savepoints can be created and used to restore the above mentioned settings. They can be downloaded as a file and uploaded again.

> **Note:** Savepoints are only valid for the exact version of the Operations Monitor software by which they have been created.

When you upgrade or downgrade its system software, Operations Monitor automatically creates a configuration savepoint before the actual installation. Please check PSA manual for more details about upgrading and downgrading the system. However, you won't see this savepoint unless you downgrade back to the version the savepoint was created under.

Figure 7–35 Configuration Savepoints

The **Configuration Savepoints** section (see Figure 7–35) lists all savepoints with name and creation date.

Table 7–8 lists the actions, which are provided by the buttons in the table's toolbar:

Table 7–8 Configuration Savepoint Fields

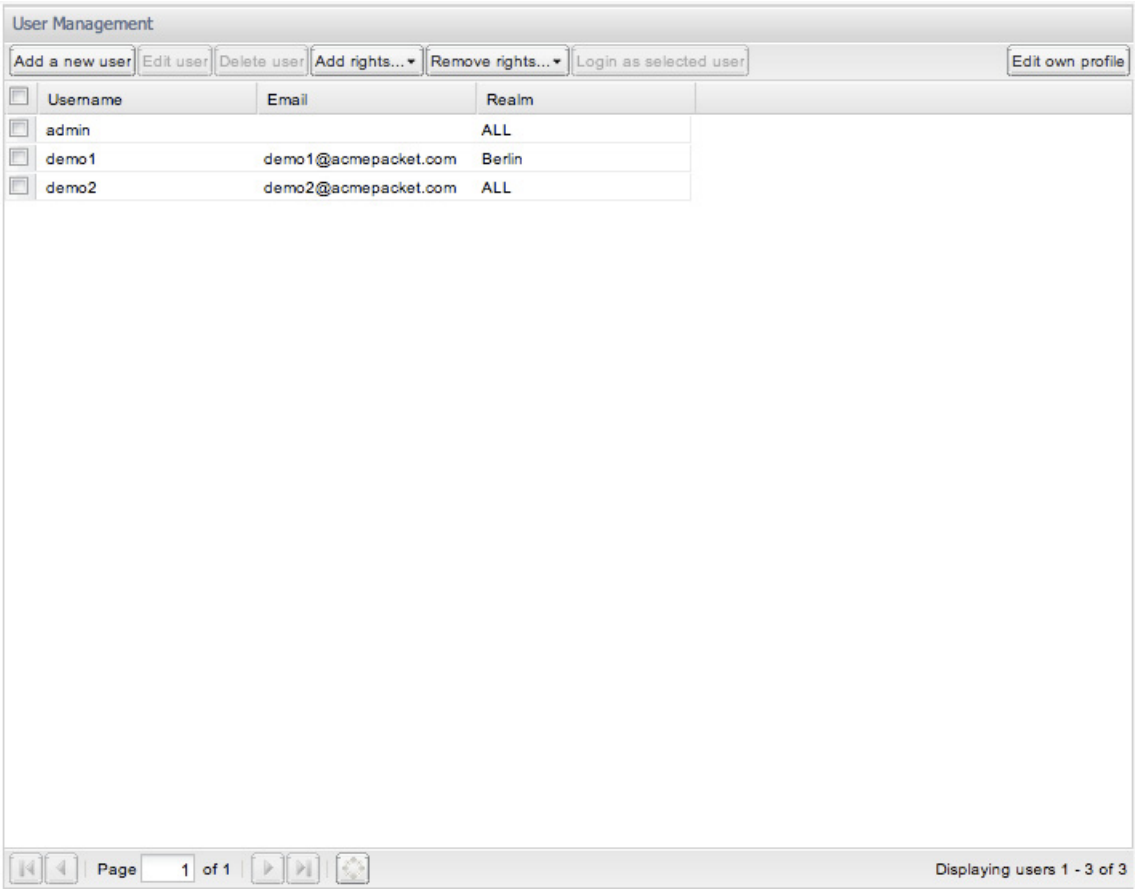
Field	Description
Create Savepoint	Creates a new savepoint from the current Operations Monitor configuration.
Restore Savepoint	Restores the Operations Monitor configuration contained in the selected savepoint. Note, that on restoring a savepoint, Operations Monitor needs to restart some internal services, which can result in a downtime in monitoring. Restoring a savepoint in an Operations Monitor deployment will result in downtime. The duration of downtime depends on the size and the configuration of the deployment.
Download Savepoint	Downloads the selected savepoint as a file.
Upload Savepoint	Uploads an Operations Monitor savepoint contained in a file.
Delete Savepoint	Removes the selected savepoint from the list and the system.

Important: Do not manually alter savepoint files unless you know what you are doing, as this may lead to severe inconsistencies in the Operations Monitor setup.

User Management

This section of the configuration allows to administer the access control over the web interface. The Operations Monitor web resources can be accessed concurrently from multiple network locations. This is done using a traditional user access mechanism.

Figure 7–36 The User Management Panel



A *user* is a data set whose most important elements are: a log-in name, a password, a set of access rights, the set of relations to other users, and the realm the user has access to; other less significant elements are name, email address, logo image.

The users are organized into a hierarchy, having *admin* as root user (with the default password 'oracle'). Any user can create a set of additional *sub-users* (provided the permission to add extra users is granted). A user has total control over the sub-users that he created, he can change any of the elements that constitute a sub-user.

The access to the web interface is fragmented into individual, non-overlapping domains. In order to access such a domain, a user needs to have the permission (or *right*) to do so. These rights, that control which parts of the web interface a user can see, are granted by the parent user that created it. Of course, the user *admin* has unrestricted access.

The **User Management** panel features seven buttons, as shown in Figure 7–36.

The left most three of them allow a user to manage his sub-users: Add a new, edit an already added one, or remove a no longer needed user. To modify or delete a sub-user, its parent-user needs to select a row out of the table and press the corresponding button. Both the **Add a new user** and the **Edit user** buttons invoke a wizard (in a new window) asking for the needed detail in order to set-up a new or change an existing user. The **Add rights...**and the **Remove rights...** buttons make it possible to add selected rights to selected or all users created by the current user, respectively to remove selected rights from selected or all users created by the current user. The rights to add or remove can be selected in a wizard, that is in accordance with the last two pages of the wizard. The **Login as selected user** button allows a parent user to login as

a sub-user without having to enter a password. The **Edit own profile** button to the right of the page allows the current user to edit one's own profile information.

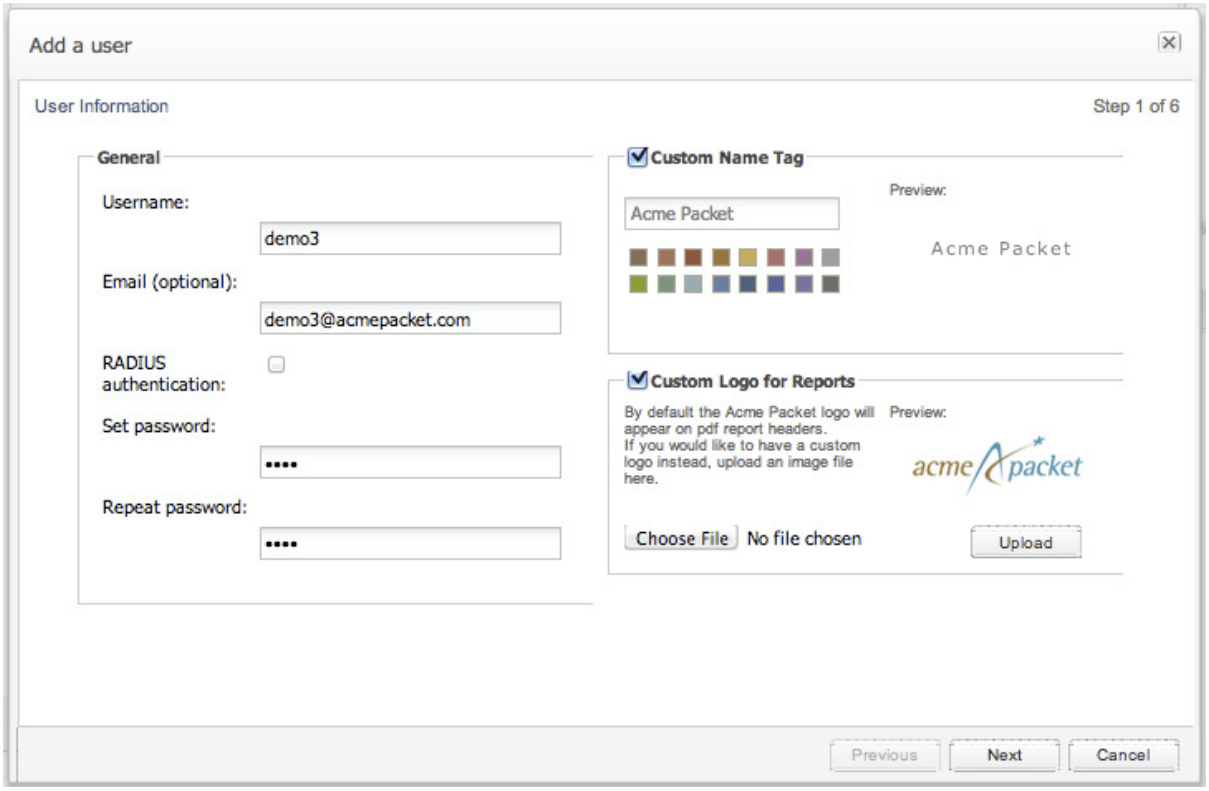
Add a New User

We'll next walk through the steps needed to add a new user. The same steps need to be performed for editing a user, with the difference that a field for the name of the new user is present when adding. When editing one's own profile, only the first step of this process is necessary.

User Information

The first step, named *User Information*, requires providing the log-in name (the username), an optional email address, and a password (asked two times to prevent accidental mistyping) as can be seen in Figure 7–37. Instead of entering a password, it is also possible to opt for *RADIUS* authentication, if RADIUS authentication has been set up for this Operations Monitor instance. To learn more about Operations Monitor's RADIUS authentication feature, see "Requirements for RADIUS Authentication".

Figure 7–37 Setting User Information for a New User



A logo image is located in the upper left corner of the interface. By default, this is the Oracle logo, but it can be customized for each individual user; this is useful when the web interface is accessed by multiple clients of the same provider, sharing the same Operations Monitor setup. The **Custom Logo for Reports** section provides an upload field where the user can insert the picture to use as a logo.

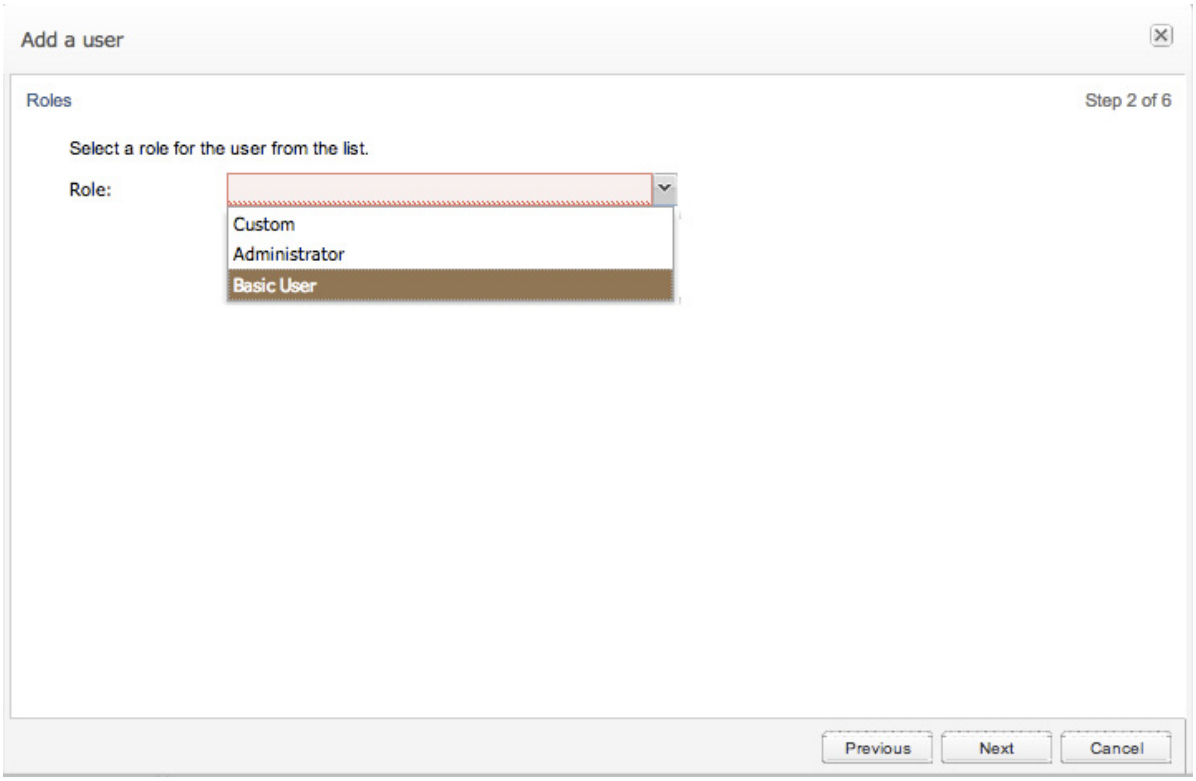
Note: The following logo formats are supported: TIF, TIFF, EPS, PDF, JPG, JPEG, GIF, and PNG.

After specifying the path, one must upload the file by pressing on the **Upload image** button.

Selecting a Role

After choosing a username and password, the user is given the option of selecting a role (See Figure 7–38) providing roles have been defined in addition to the custom role. If no roles are available, the dialogue will continue directly to the realms selection if they exist, otherwise onto the individual permissions selection.

Figure 7–38 Setting a Role for a New User



User Permissions

In case *User Roles* is enabled (see Figure 7–38), a role is set for the user that gives the user a predefined set of permissions that are tied to the role. For more information on role management, see "Role Management".

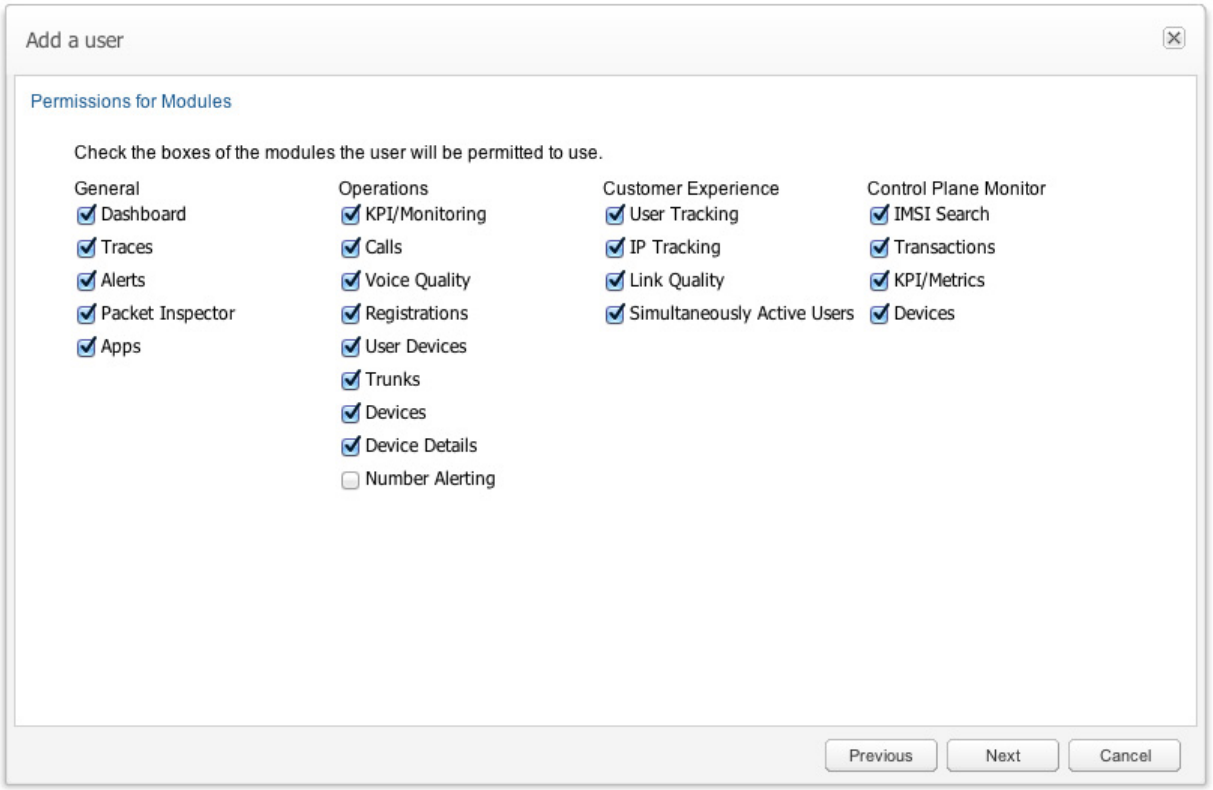
In case *User Roles* is disabled, the permissions need to be set individually for each module (see Figure 7–39) and for each setting.

The **Permissions for Modules** page, is where the parent user can control what the new user will be able to access. The set of permissions are grouped in four, reflecting the grouping of functionality on the actual web interface:

- General
- Operations
- Customer Experience
- Control Plane Monitor

Each check box in these groups corresponds to a menu item in the main menu of Operations Monitor. If it is checked, the new user will have the corresponding menu item in his main menu.

Figure 7–39 Setting Permissions for Modules for a New User



The **Permissions for Settings** page can control what settings the created user will be allowed to modify. The access right to some advanced functionality of Operations Monitor can also be controlled here.

The check boxes in the *Settings* group to the left of the section roughly correspond to sections in the settings dialog.

Table 7–9 describes the user setting options.

Table 7–9 User Settings Options

Field	Description
Changing of platform settings	If checked, the created user is allowed to access the **Platform Settings** section in the Operations Monitor settings dialog.
Changing of prefix tags	If checked, the user is allowed to change the number prefix tags in the Platform Settings. However, the user will still be able to view them even without this permission.
Create users	If checked, the created user is allowed to create and manage new users himself using the **User Management** section of the Operations Monitor settings dialog.
Grant create users	If checked, the created user can grant the 'Create users' permission described above.

Table 7–9 (Cont.) User Settings Options

Field	Description
View/Edit all users	If checked, the user is able to see and modify users created by other users.
Create roles	If checked, the user is able to view and modify the roles management section in the user management folder. For more information, see "Role Management".
Changing of external devices	If checked, the created user is allowed to access the **External Devices** section in the Operations Monitor settings dialog.
Changing the system settings	If checked, the created user is allowed to access the **System Settings** section in the Operations Monitor settings dialog.
Modify realms	If checked, the created user is able to view and modify the realms management section of the settings. If the user is assigned a realm, on a previous screen, he will only be able to modify realms that intersect or are contained within the user's own realm.

Table 7–10 describes the user interface setting options.

Table 7–10 User Interface Settings Options

Field	Description
View SIP messages	If checked, the created user can see only the SIP message headers in the messages tab of the **Call Details** window and in the **Registration Details** window. For more information, see "Calls" and "Registrations".
View SIP message bodies	If checked, the created user can view SIP message bodies in the messages tab of the **Call Details** window and in the **Registration Details** window. For more information, see "Calls" and "Registrations". Message bodies cannot be captured in packet capture downloads unless they are set as viewable to the user.
Create PDF reports	If checked, the new user can create PDF call reports on the Calls page. For more information, see "Calls".
View registration history	If checked, the new user is allowed to see the registration history of a platform user in the **User Tracking** page. For more information, see "User Tracking".
Incremental User Search	If checked, the new user is allowed to use the incremental user search on the **User Tracking** page. For more information, see "User Tracking".
Read the OCSM manual	If checked, the new user is allowed to see the online version of the Operations Monitor User's Guide.
Change the dashboard page	If checked, the new user is allowed to add and remove items from the **Dashboard** page.
May always log in	If checked, the new user is always allowed to log in, even if the concurrent session limit for one user or the global concurrent session limit is surpassed. In case the global concurrent session limit is surpassed, another user will be logged out in order to allow this user to log in.
Save calls	If checked, the new user is allowed to save calls for further debugging.

Table 7–10 (Cont.) User Interface Settings Options

Field	Description
View saved calls from other users	If checked, the new user is allowed to view the saved calls created by other users.
View all realms	If checked, the new user is allowed to view all the realms not only the ones created by itself.
RTP Header Recording	If checked, the new user is allowed to record the header part of RTP packet in a call.
RTP Payload Recording	If checked, the new user is allowed to record the entire RTP packets of a call. This includes audio, video, and image data, voice quality data, and DTMF tones.
RTCP Recording	If checked, the new user is allowed to record RTCP voice quality data.
View DTMF Tones	If checked, the new user is allowed to view DTMF tones in the call flow diagram and to see the presence of DTMF tones in a call in the calls grid.

User Realm

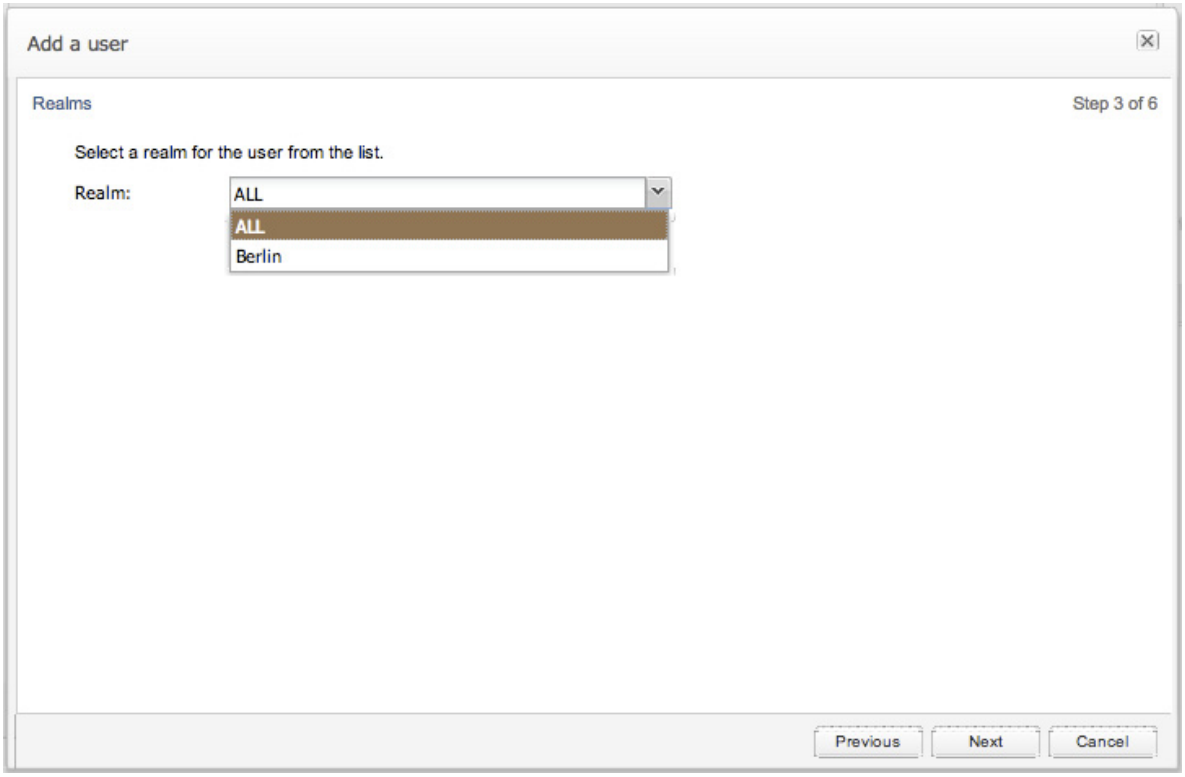
The second step sets the realms that the future user will be able to access.

A user assigned to a realm can only see devices that are also assigned to that realm. Users assigned to the realm **ALL** are unrestricted and can see all devices, regardless of the realm the device is assigned to. For more information about the realm concept in Operations Monitor, see "Realms Definitions."

On the **KPI/Metrics** page, users also only see the devices listed in their assigned realm. For more information, see "KPI/Metrics."

A user can access either one or **ALL** the realms (see Figure 7–40).

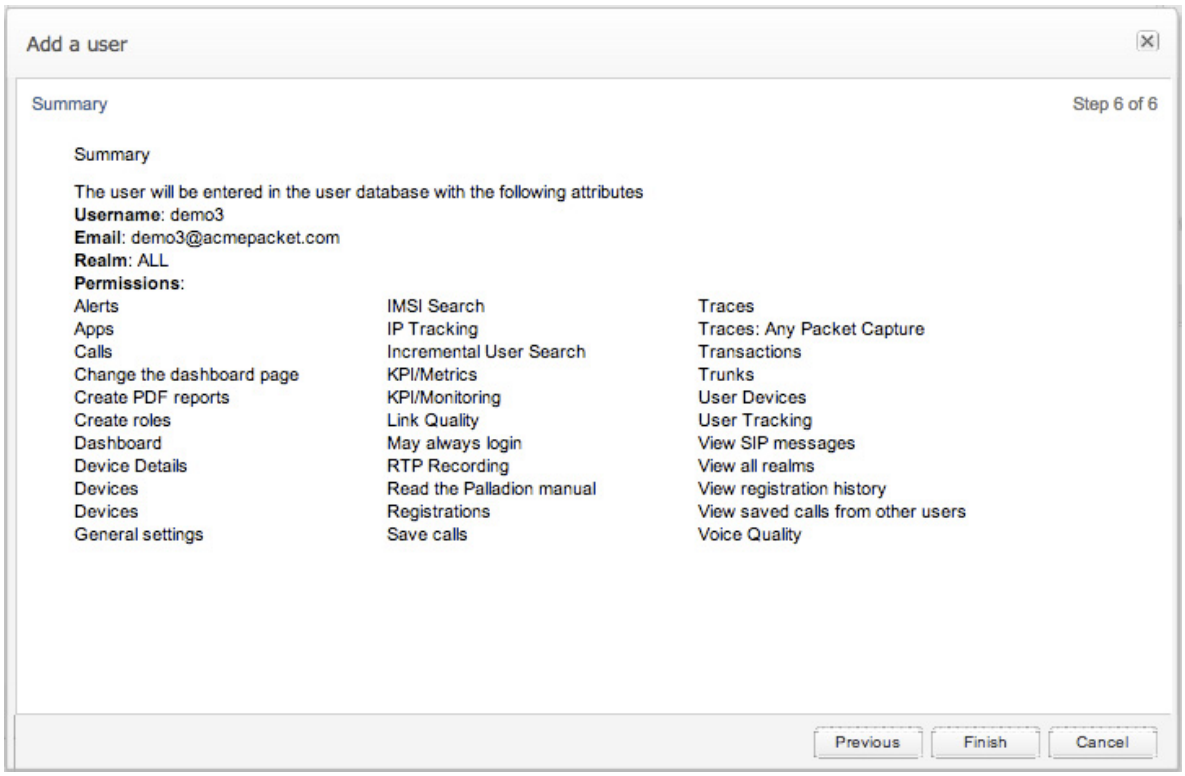
Figure 7–40 Setting Realm for a New User



Overview Information

The summary as shown in Figure 7–41 gives a brief overview of the new user. Included is the username, email address (if entered), role, realm and permissions bitmask.

Figure 7–41 Summary of the New User



Requirements for RADIUS Authentication

When a user logs in, Operations Monitor supports RADIUS authentication if it has been enabled in the **RADIUS Authentication** section of the Settings dialog box. RADIUS authentication is performed if it has been enabled for the user.

For RADIUS authentication to work, the RADIUS server has to respond to the Operations Monitor access request sent with the set of required user attributes. Operations Monitor sends its access request with a configurable **NAS_Identifier** attribute, which can be used to identify requests from Operations Monitor on the server side, and with the encrypted password as the **Password** attribute. Upon successful authentication, Operations Monitor expects the following user attributes:

Table 7–11 RADIUS Authentication User Attributes

Attribute	Description
PLD-User-First-Name	Not applicable.
PLD-User-Last-Name	Not applicable.
PLD-User-Email	(Optional) The e-mail address of the user.
PLD-User-Realm	The integer realm ID of the user's realm.
PLD-User-Creator	The user ID of the parent user for the user.
PLD-User-Permissions	A bitmask describing the permissions of the user.
PLD-User-Logo	(Optional) A field containing the logo image for the user.
PLD-User-Options	A JSON-encoded object describing per user options.

The numeric IDs for these user attributes are defined in a RADIUS dictionary file, which is delivered together with Operations Monitor.

> **Note:** The **PLD-User-First-Name** and **PLD-User-Last-Name** fields in the radius definition are not used

The **PLD-User-Permissions** attribute deserves special explanation. It contains a bitmask describing the permissions that a user has. Each of the permissions introduced above has a position in the bitmask. To generate a specific permissions bitmask you can use the wizard for adding users (choose **RADIUS authentication** on the first page of the wizard, then select the appropriate permissions, and the permissions bitmask will be displayed on the last page of the wizard).

Password Settings

In this folder of the Operations Monitor settings, you can define the password settings that are followed by users.

Select the **Force users to change their password regularly** check box for access to password change settings. Modify the period (in days) to force a password change:

- For users with access to sensitive data, the default setting is 90 days.
- For users without access to sensitive data, the default setting is 180 days.

> **Note:** The entries in these fields are applied when the user next changes the password.

Select the **Enforce stringent password rules** check box to increase the level of security required in user passwords.

Click **Save** when you are finished.

Role Management

In this folder of the Operations Monitor settings, the user roles are administered to define default sets of permissions for users. It is not required to assign a role to a user, however it is recommended as it simplifies the user creation process. An example of this panel is seen in Figure 7–42.

For instance an administrator role would generally contain all permissions, a power user may contain most screens and a subset of system permissions, and a basic user a subset of screens. During the user creation process if a role other 'custom' is selected, then later on the individual permission check boxes will be skipped.

> **Note:** Regarding role management, the terms *permissions* and *rights* are used interchangeably.

Figure 7–42 Roles

Note: By default, the roles feature is disabled. In order to make use of the roles, the **User Roles** system setting must be enabled.

Adding a Role

Adding a role is performed after selecting the **Add a new role** button. As shown in Figure 7–43, Figure 7–44, and Figure 7–45. This is equivalent to the permissions screens of adding a new user, however after setting the permissions there is a prompt for the role name, see Figure 7–45. The role name is mandatory.

Figure 7–43 Setting Permissions for Modules for a New Role

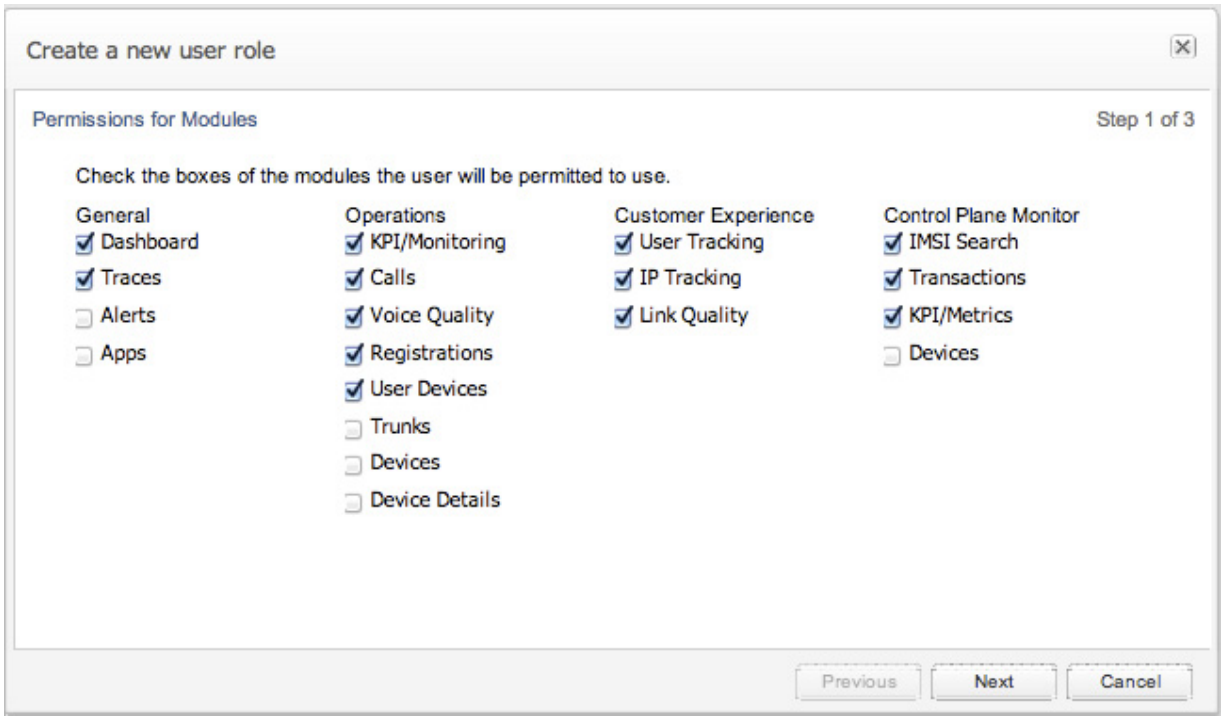


Figure 7–44 Setting Permissions for Settings for a New Role

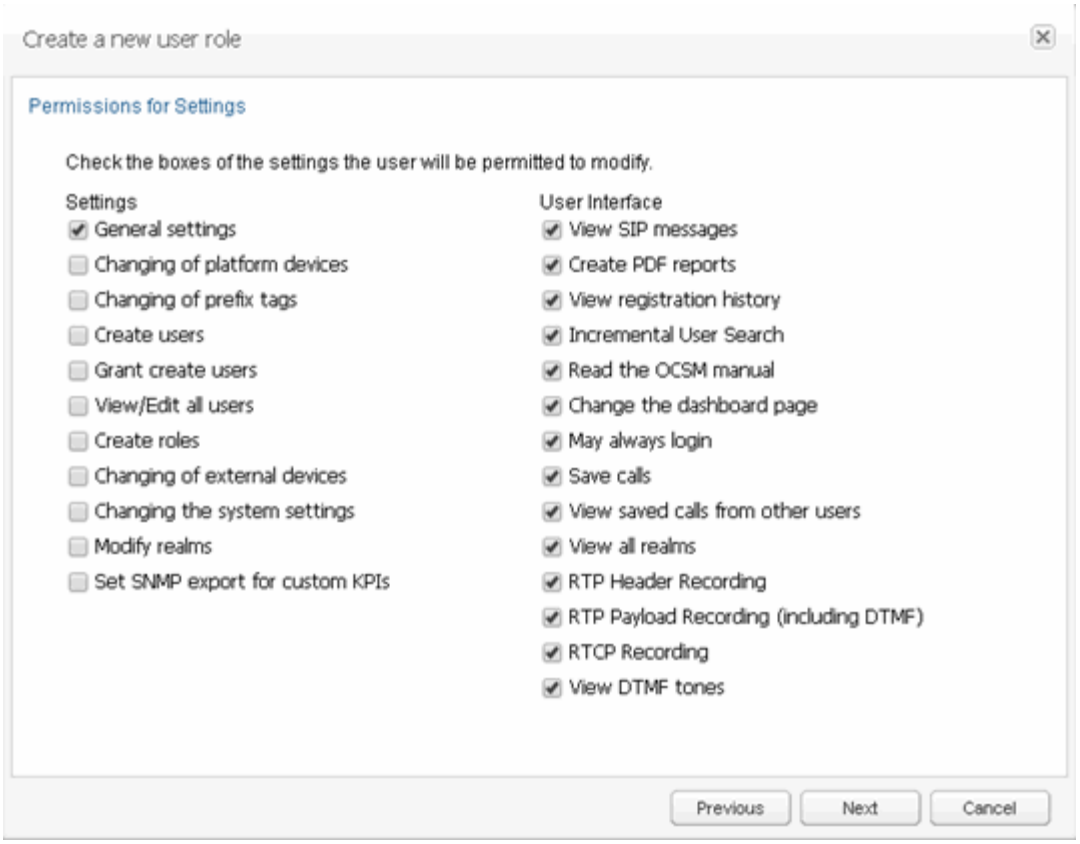
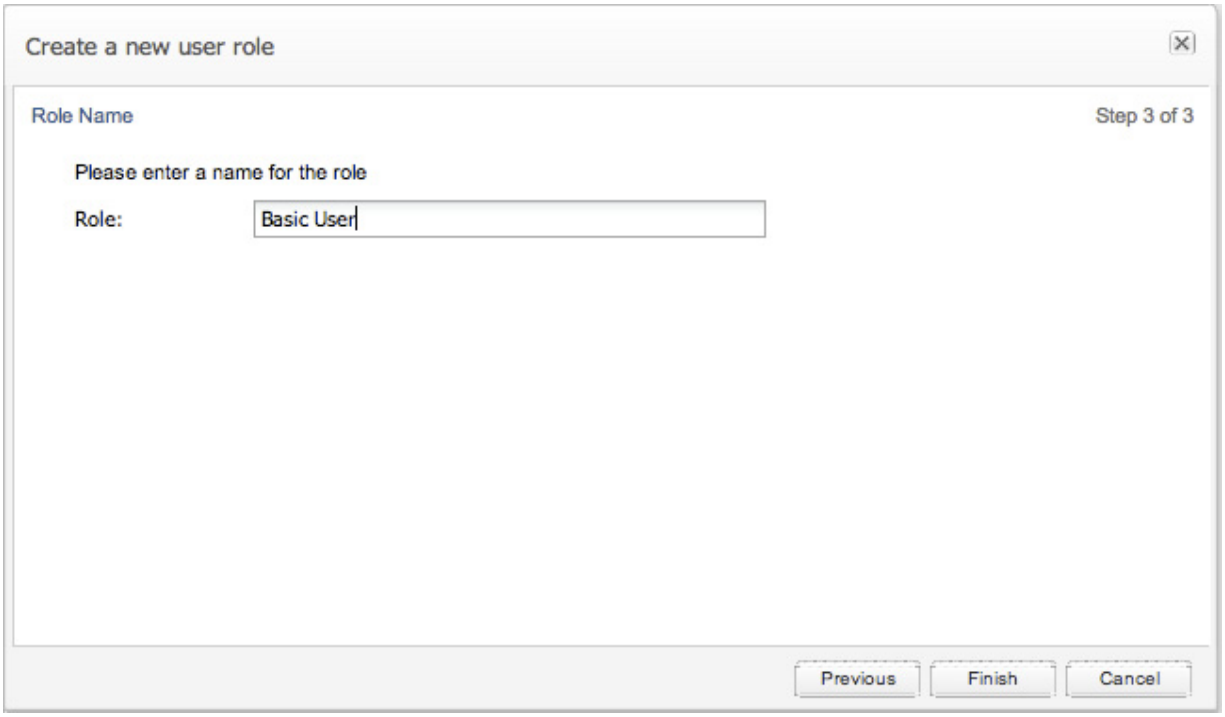


Figure 7–45 Setting the Name of the Role



Editing a Role

After selecting only one role, it may be edited by selecting the **Edit** option in the **role management** panel. When multiple roles are selected this option is not available.

Deleting a Role

Deleting a role may also only be performed after selecting a single role. There is a confirmation prompt that must be confirmed before the role is deleted.

Adding Rights to a Role

One can either add rights to a selected role or generally to all roles. This can be selected as demonstrated in Figure 7–46 and Figure 7–47. The following dialogue is equivalent to the normal role dialogue, only without the option to change a role name. The selected check boxes designate which rights will be added to the selected role(s).

Figure 7–46 Adding Rights to All Roles

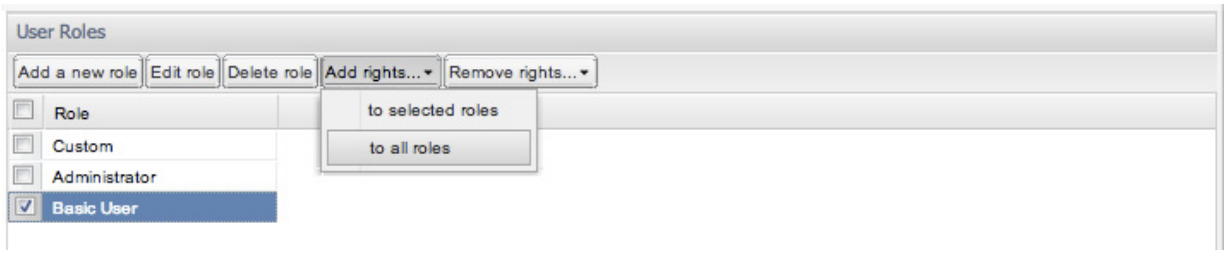
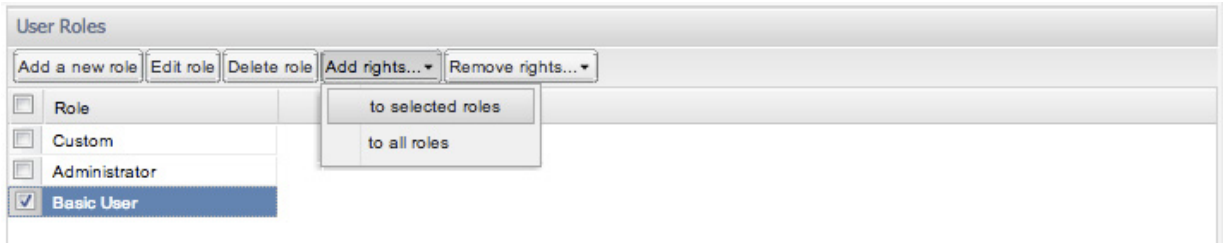


Figure 7–47 Adding Rights to Selected Roles



Removing Rights from a Role

Likewise, one can remove rights from individual roles or generally across all roles. This is selected as shown in the images Figure 7–48 and Figure 7–49. Removing rights is the mirror process to adding rights, with the selected check boxes will be removed from the selected role(s).

Figure 7–48 Removing Rights from All Roles

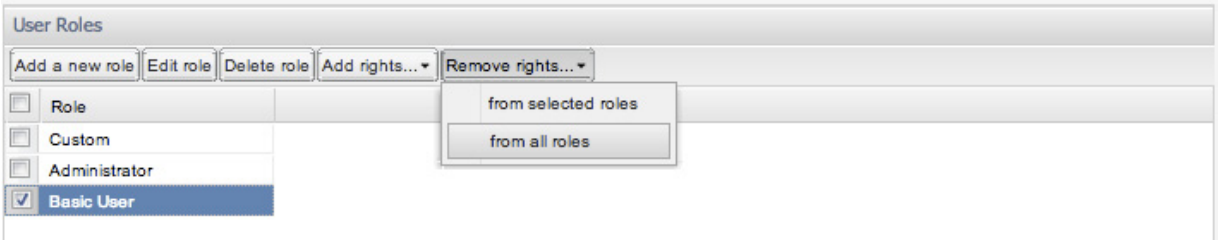
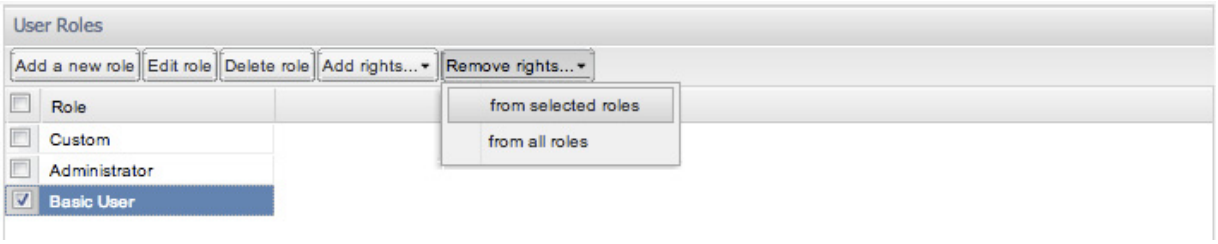


Figure 7–49 Removing Rights from Selected Roles



8

Call Merging Algorithms

Oracle Communications Operations Monitor has the ability to find the *call legs* of the same call. This is an essential feature for troubleshooting and for providing deep device visibility and accurate statistics. While in the case of some SIP devices (like Proxies and B2BUAs that keep the **Call-ID** header) finding the matching call legs is easy, many other devices change most of the fields from the incoming leg for purposes like topology hiding, decoupling, and interoperability. This makes the call merging problem hard and impossible to solve in all cases with a single algorithm.

Operations Monitor gives the network operator the power to adjust the call merging algorithm by providing a comprehensive list of *tests* and a flexible way to combine them in **call merging** algorithms. This section describes the available tests and how to write the algorithms. The resulting scripts can be entered in the **Platform Devices** section for *SBC/B2BUA* devices by choosing the **Use custom algorithm** option.

The call merge algorithms always return a single Boolean value: **true** if the *call legs* are part of the same *call*, and **false** otherwise. They provide ordered tests, tests with parameters, and nested decisions trees. To keep the real-time nature of Operations Monitor, the tests are implemented natively in the Operations Monitor core, and the algorithms are compiled to an internal representation which can be executed with maximum performance.

This chapter also contains links to KM notes that describe how to enable various types of call merging and call correlation. For more information, see "Merging and Correlating Calls".

Note: When call correlation is based on the correlation algorithm of a device, an incoming leg can be correlated with multiple outgoing legs. An outgoing leg, however, can be correlated to only one incoming leg.

For global call correlation based on the SIP Call ID, the restriction for outgoing legs does not apply. Call ID based correlation can be enabled with the **Merge globally by Call ID** setting in **System Settings** under **System Management** on the Settings page. For more information on this setting, see "Merge globally by Call-ID".

Merging and Correlating Calls

This section contains links to KM notes that describe how to enable various types of call merging and call correlation. For more information, see the KM note located at: https://support.oracle.com/epmos/faces/DocumentDisplay?id=1638050.1

For information on call merging and call correlation using the Replaces parameter, see the KM note at:
https://support.oracle.com/epmos/faces/DocumentDisplay?id=2012218.1

Syntax

Syntax wise, they are formed from a single LISP expression. Apart from the tests, there is only the special function `cond`, and two terminals:

- `#t` for true.
- `#f` for false.

The LISP `cond` function accepts an arbitrary number of arguments. Each argument is a list of two elements: a **condition** and a **result**.

```
(cond
    (condition1 result1)
    (condition2 result2)
    ...
    (conditionN resultN)
)
```

The conditions are evaluated in order, and for the first being **true**, the result is returned:

if `condition1` is **true** return `result1`, **else if** `condition2` is **true** return `result2`, **else if** ...; **else if** `conditionN` is **true** return `resultN`.

If none of the conditions is **true**, then **false** is returned for the whole expression, as a convenience.

Usually in *call merging algorithms*, the conditions are tests and the results are terminals:

```
(cond
    (test1 #t)
    (test2 #t)
    (test3 #f)
    (test4 #t)
)
```

But the result does not have to be a terminal. For example, this is a possible **AND** statement:

```
(cond
    (test1 test2)
)
```

The result part can also be another `cond` expression for providing nested tests:

```
(cond
    (test1 (cond
                (test2 #t)
                (test3 #t)
           )
    )
    (test4 #t)
)
```

The above example translates in the following pseudo-code:

```
if test1:
    if test2:
```

```
        return True
    elif test3:
        return True
    else:
        return False
elif test4:
    return True
else:
    return False
```

The tests can accept an arbitrary number of parameters which have to be provided right after the test name, for example:

```
(cond
    (test1 param1 param2)
)
```

translates to:

```
if test1(param1, param2):
    return True
else:
    return False
```

The parameters can be strings, integers, or list of strings:

```
(test "string" 12 ("list" "of" "parameters"))
```

Line comments can be inserted by using the semicolon symbol:

```
; this is a comment
```

Tests Reference

This section lists the tests available for custom call merging algorithms.

call_id ([*suffix*])

Returns `#t` if the **Call-ID** headers of the two *call legs* are equal.

Parameters:

suffix - If this parameter is provided and it is non-zero, then only the last `suffix` characters are compared from the two **Call-ID** headers, and if they are equal, it returns `#t`.

Example:

```
(call_id 12)
```

returns `#t` if the last 12 characters from the **Call-ID** headers are equal.

hf_equals (*header*)

Returns `#t` if the values of the header whose name is given as a parameters are equal in the two call legs. The header name is case-insensitive.

For example, the following snippet returns `#t`, if the INVITE messages from both call legs contain a header named **Session-ID** and their values are equal:

```
(hf_equals "Session-ID")
```

hf_any_equals (*headers*)

Like `hf_equals`, but compares any header value in headers list with any header value from the second call; for a more complete example what this means see also `uri_user`.

hf_any_equals_all (*headers*)

This function is like `hf_any_equals`, but differs in the way that it treats the case where a header field contains a list of values, either separated by commas or in several header field rows with the same field name. In that case `hf_any_equals` will use only the first header field row with a given field name and use the entire row as one value, whereas `hf_any_equals_all` will try to match each individual element of the list.

hf_equals_prefix (*length*, *header*)

Limits the comparison performed by `hf_equals` to `length` characters from the beginning of the header value.

hf_param_equals (*header*, *param*)

Works like `hf_equals` but compares only the value of the named param.

sdp_media_ip_port ()

Returns `#t` if the *SDP* bodies from the initial INVITE messages of the two call legs contain the same media *IP* address and *UDP* port number in the first media description.

This is usually a good indicator, if the *B2BUA* device does *not* relay media.

sdp_session_id ()

Returns `#t` if the *SDP* bodies from the initial INVITE messages of the two call legs contain the same non-zero session ID.

time_diff (*interval1*, *interval2*)

Returns `#t` if the time difference between the two legs is *larger* than the accepted intervals.

Parameters:

- **interval1** - The minimum time difference between the initial INVITE message of *incoming* call leg and the initial INVITE of the *outgoing* call leg.
- **interval2** - The minimum time difference between the initial INVITE message of *outgoing* call leg and the initial INVITE of the *incoming* call leg.

For example, the following code returns `#t` if the outgoing leg comes after the incoming leg with more than 15 seconds, **or** if the incoming leg comes after the outgoing leg with more than 2 seconds:

```
(time_diff 15000 2000)
```

Important: The maximum value that you can use for the **time_diff** function is the same as the Operations Monitor system settings. Where,

- The maximum value for **B2B incoming backlog search for merging** is 5 seconds.
- The maximum value for **B2B outgoing backlog search for merging** is 1800 seconds.

If you do not include the **time_diff** function in your algorithm, your Operations Monitor **B2B incoming backlog search for merging** and **B2B outgoing backlog search for merging** system settings are used.

Note: The **time_diff** function is treated differently than the others within the call correlation algorithm. It is considered for each correlation regardless of its position inside the algorithm.

uri_user (*suffix*, *list_of_headers*)

Returns `#t` if the user part of any of the *URIs* is specified in the *list_of_headers* matches.

Parameters:

- **suffix** - If non-zero, only the last *suffix* characters from each URI is compared with the rest. URI length must be at least this size.
- **list_of_headers** - A list containing the headers from which to extract the URI usernames to compare. A set of shortcuts are defined:
  - **ruri** - Request-URI.
  - **from** - From header URI.
  - **to** - To header URI.
  - **diversion** - Diversion header URI.
  - **pai** - P-Asserted-Identity URI.
  - **ppid** - P-Preferred-Identity URI.
  - **rpid** - Remote-Party-ID URI.

The URIs are compared in all possible permutations. For example, this snippet:

```
(uri_user 6 ("from" "pai"))
```

compares:

- The **From** user from leg 1, with the **From** user from leg 2.
- The **From** user from leg 1, with the **P-Asserted-Identity** user from leg 2.
- The **P-Asserted-Identity** user from leg 1, with the **From** user from leg 2.
- The **P-Asserted-Identity** user from leg 1, with the **P-Asserted-Identity** user from leg 2.

Similarly, the following snippet makes at most 9 comparisons:

```
(uri_user 6 ("to" "ruri" "diversion"))
```

In addition to the specified shortcut header names, URIs can be also specified by the name of the header containing them.

For example, the following expression returns `#t` if the last 7 characters from the username of the first **Contact** header from both calls are equal:

```
(uri_user 7 ("Contact"))
```

The header name is case-insensitive.

uri_user_max (*suffix*, *list_of_headers*)

The same as `uri_user` function, but it tries to match the call if a string is shorter than the *suffix*. For example, phone numbers **5551234** and **1234** will match, even if the suffix is **6**.

uri_user_all (*suffix*, *list_of_headers*)

This function behaves in the same way as `uri_user` , except that if one of the relevant headers of one the call legs contains more than one URI, then all of these URIs are used for comparison, whereas `uri_user` would only use the first. Therefore if in the example:

```
(uri_user 6 ("from" "pai"))
```

one leg contains one **P-Asserted-Identity** and the other three, then up to 8 comparisons will be performed.

uri_user_all_max (*suffix*, *list_of_headers*)

As `uri_user_max` , but treats lists in header fields in the same way as `uri_user_all`.

cxpn_uri (*suffix*, *list_of_headers*)

Matches an in- with an out-going call comparing all possible header permutations like `uri_user`. It also supports ISUP *cdpn* and *cgpn* and allows for cross matching of SIP and ISUP calls.

Parameters:

- **suffix** - If non-zero, only the last *suffix* characters from each URI is compared with the rest. URI length must be at least this size.
- **list_of_headers** - In addition to the values supported by `uri_user`, cxpn_uri also supports:
  - **cdpn** - Matching on the callee.
  - **cgpn** - Matching on the caller.

cxpn_uri_max (*suffix*, *list_of_headers*)

The same as `cxpn_uri` function, but it tries to match the call if a string is shorter than the *suffix*. For example, phone numbers **5551234** and **1234** will match, even if suffix is **6**.

cxpn_uri_all (*suffix*,*list_of_headers*)

Variants of `cxpn_uri` that use all URIs if a header of the SIP leg contains several, just as `uri_user_all`.

cxpn_uri_all_max (*suffix*,*list_of_headers*)

Same as `cxpn_uri_max`, but treats lists in header fields in the same way as `uri_user_all`.

cdpn (*suffix*)

Like `uri_user`, but matches only on the callee.

cgpn (*suffix*)

Like `uri_user`, but matches only on the caller.

Test Usage Limitations

Depending on the kind of device that is used for matching, different sets of test functions are available:

- **B2BUA**

  Supports SIP matching which comprises these tests: `call_id` , `uri_user`, `time_diff`, the `hf_equals` family of tests and `sdp_session_id` as well as `sdp_media_ip_port`.

- **STP**

  Supports ISUP call matching with `cdpn`, `cgpn` and `time_diff` as tests.

- **SGW**

  Supports all the tests available for the other two device types and the `cxpn_uri` test.

Examples

The following examples describe how to create an algorithm using the test references.

Match by the Caller and the Callee

The following algorithm matches two call legs if:

- The last 6 characters of caller are equal, as found in either the **From** header or in the *P-Asserted-Identity*,
- **And** the last 6 characters of the callee are equal, as found in either the **To** header or in the *Request-URI*.

```
(cond
    ((uri_user 6 ("from" "pai"))
        (uri_user 6 ("to" "ruri"))
    )
)
```

The following example modifies the one above to return **true**, if either the caller or the callee matches:

```
(cond
    ((uri_user 6 ("from" "pai")) #t)
    ((uri_user 6 ("to" "ruri")) #t)
)
```

Match by Generic Algorithm

The following algorithm is an example of a *generic* algorithm, which should give good results for a wide range of SIP devices.

```
(cond
    ((call_id 0) #t)
    ((sdp_media_ip_port) #t)

    ; custom header
    ((hf_equals "session-id") #t)

    ((time_diff 15000 2000) #f)

    ; Matching by from and to
    ((uri_user 6 ("from" "pai" "rpid" "ppid"))
        (cond
            ((uri_user 6 ("to" "ruri" "diversion")) #t)
            (#t #f)
        )
    )

    ; nothing matched, return false
    (#t #f)
)
```

The above algorithm:

- Checks first the **Call-ID**. If the two call legs have the same **Call-ID** header, then it returns **true**.
- Checks the **SDP** bodies. If the two call legs have the same media IP and port number, or if the session ID are equal, then it returns **true**.
- Checks if the custom **Session-ID** header is present in both call legs, and if their values are equal, then it returns **true**.
- Checks if the start time stamps of the two call legs are too far one from each other, and if so, returns **false**.
- Checks if both the caller and the callee match, by comparing the URIs from several headers. If a match is found for both of them, it returns **true**.

If none of the above is **true**, it returns **false**.

9

Implementing Apps

Apps provide an easy way to extend Oracle Communications Operations Monitor with analytical capabilities that are not present in the user interface. Apps can query information from Operations Monitor, process this information, and store results in an output table.

Structure of an App

A valid Operations Monitor extension app consists of a Python 2.7 module and of an app specification file, that declares the parameters of the app and the schema of the app's result table. The Python module file and the app specification file must be packed into a ZIP archive, containing no other files and no directories. Such a ZIP archive can be uploaded to Operations Monitor as described in "Apps".

Further requirements for the app module and the app specification file apply. The app module must contain a **run** function with the following signature:

run (*facade*, *params*)

Parameters:

- **facade**

  An instance of **libpalladion.scripting.Facade** which provides access to the Apps API.

- **params**

  A dict instance which represents the parameters supplied to the app.

This function is called by Operations Monitor when the app is executed from the web interface or via remote procedure calls.

The app specification file must be an XML file which complies with the Operations Monitor app-specification DTD cited below:

```
<!DOCTYPE script [
<!ELEMENT script (param-spec?,result-schema?)>
<!ATTLIST script xmlns CDATA #FIXED "http://iptego.de/palladion/script-spec">
<!ATTLIST script name CDATA #REQUIRED>
<!ATTLIST script description CDATA #IMPLIED>
<!ATTLIST script result-type (custom|calls) #REQUIRED>

<!ELEMENT param-spec (param+)>

<!ELEMENT param EMPTY>
<!ATTLIST param name CDATA #REQUIRED>
<!ATTLIST param label CDATA #REQUIRED>
```

```
<!ATTLIST param type (string|datetime|numeric) #REQUIRED>
<!ATTLIST param required CDATA #IMPLIED>
<!ATTLIST param default CDATA #IMPLIED>

<!ELEMENT result-schema (column+,primary-key?,unique-key*)>

<!ELEMENT column EMPTY>
<!ATTLIST column name CDATA #REQUIRED>
<!ATTLIST column type CDATA #REQUIRED>
<!ATTLIST column null (true|false) #IMPLIED>
<!ATTLIST column auto-increment (true|false) #IMPLIED>
<!ATTLIST column default CDATA #IMPLIED>

<!ELEMENT primary-key EMPTY>
<!ATTLIST primary-key columns NMTOKENS #REQUIRED>

<!ELEMENT unique-key EMPTY>
<!ATTLIST unique-key name CDATA #REQUIRED>
<!ATTLIST unique-key columns NMTOKENS #REQUIRED>
]>
```

The top-level **script** element is used to configure some important properties of an app. Its **name** attribute specifies a name for the app which is displayed in the **Available Apps** table. The **description** attribute can contain a longer description for the app. Most importantly, the **result-type** attribute specifies what kind of result table the app will have. It can be either **custom**, in which case you must specify the result table in the **result-schema** element, or **calls**, in which case the result table is fixed as a calls table.

The **param-spec** element is a container for parameter specifications declared by the **param** element. Parameter specifications are used by the web interface to generate the right kind of parameter entry dialog when an app is started. The **name** attribute of the **param** element specifies the name of the parameter, that is passed to the app. The **label** attribute specifies a label that is displayed in the parameter entry dialog. The **type** attribute specifies how the parameter entry is rendered in the parameter entry dialog. The **required** attribute decides whether the parameter must be entered or can be left blank. The **default** attribute specifies a default value, that is filled into the parameter entry.

The **result-schema** element is used to specify a schema for the result table of the app. This is required when the **result-type** attribute of the main **script** element is set to **custom**. The contents of the **result-schema** element are translated into an SQL CREATE TABLE statement. You can specify columns in the result table, as well as a primary key and multiple unique keys. The **column** element specifies a column. The **name** attribute gives the name of the column. The **type** attribute specifies the type of the column. It can contain any of the data type specifications valid in MySQL. For more information, see the *Create Table Syntax* in the *MySQL 5.0 Reference Manual*:

http://dev.mysql.com/doc/refman/5.0/en/create-table.html

The **null** attribute decides whether the column is nullable or not. The **auto-increment** attribute specifies whether the column can be filled via auto increment on insert. The **default** attribute gives a default value for the column. The **primary-key** element declares a primary key for the result table. Its **columns** attribute is a list of columns that make up the primary key. The **unique-key** element declares a unique key in the result table. Its **name** attribute gives the name for the unique key. Its **columns** attribute is a list of columns that make up the unique key.

Apps API

class libpalladion.scripting.Call (*facade*)

This class represents a call in Operations Monitor. For general information on how Operations Monitor handles calls, see"Calls".

Table 9–1 describes the fields for the Call class.

Table 9–1 Call Class Fields

Field	Description
callid: STRING	ID of the call.
nlegs: INTEGER	The number of legs this call consists of.
state_msg: STRING	A string describing the call state. One of 'Unauthorized', 'Proceeding', 'Ringing', 'Established', 'Finished', 'Timed out', 'Error', 'Failed', 'Closed by CleanBye', 'Not found', 'Moved', 'Off-line', 'Busy', 'Canceled'.
state_details: STRING	An additional description of the call state.
setup_start_ts: DATETIME	A timestamp specifying when the call was initiated.
setup_time: INTEGER	The time, in milliseconds, needed for the call to reach the 'Established' state. Only available if state_msg equals 'Established'.
call_time: INTEGER	The duration of the call in milliseconds, that is the time the call was in the 'Established' state. Only available if state_msg equals 'Finished'.
code: INTEGER	The final response code of the INVITE transaction associated with this call.
src_user: STRING	The user creating the call. This is usually taken from the **From** header field of the first *call leg*.
src_ua: STRING	The user agent string for the caller.
src_codecs: STRING	The comma separated list of codecs proposed in the SDP body by the UAC, usually in the INVITE message. On each re-INVITE from inside the dialog, this field is updated to the last proposed list of codecs. This is useful for detecting T.38 calls, for example
src_initial_codecs: STRING	The comma separated list of codecs proposed in the SDP body by the UAC, usually in the first INVITE message. This field is not updated at *re-INVITES*.
dst_user: STRING	The user to which the call is addressed. This is usually taken from the **To** header field of the first call leg.
dst_ua: STRING	The user agent string for the callee.
dst_codecs: STRING	The comma separated list of codecs proposed in the SDP body by the UAS. On each *re-INVITE* from inside the dialog, this field is updated to the last proposed list of codecs.
dst_initial_codecs: STRING	The comma separated list of codecs proposed in the SDP body by the UAS, usually in the first INVITE message. This field is not updated at *re-INVITES*
term_devs: STRING	A comma separated list of device IDs for devices that this call has an inbound leg to.
init_devs: STRING	A comma separated list of device IDs for devices that this call has an outbound leg from.
ingress_devs: STRING	A comma separated list of devices that are ingress devices for this call.

Table 9–1 (Cont.) Call Class Fields

Field	Description
egress_devs: STRING	A comma separated list of devices that are ingress devices for this call.
MOSlqe_avg: FLOAT	Average MOSlqe value for all relevant streams in the call.
MOSlqe_min: FLOAT	Minimum MOSlqe of all stream chunks associated with the call.
getLegs ()	Query legs that belong to this call. Note that this method works only for calls that are in the permanent history (that is, finished for more than 15-20 seconds). **Return type:** QueryIterator over Leg instances.
getMessages (*filter*=[])	Query all SIP messages, that belong to this call. Note that this method works only for calls that are in the permanent history (that is, finished for more than 15-20 seconds). **Return type:** QueryIterator over Message instances.
getOneWayAudio (*filter*=[], *keys={}*, *orderBy*=[])	Query all voice quality information, that belong to this call. **Return type:** integer Values: ■ 0: no audio. ■ 1: there is a media leg with audio only in one direction. ■ 2: all media legs with two way audio.
getRawMessages (*filter*=[])	Query all raw signaling messages, that belong to this call. Note that this method works only for calls that are in the permanent history (that is, finished for more than 15-20 seconds). **Parameters** ■ **filter** - A filter specification to be applied to the queried RawMessage instances. **Return type:** QueryIterator over RawMessage instances.
getVoiceQuality (*filter*=[], *keys={}*, *orderBy*=[])	Query all voice quality information, that belong to this call. **Parameters** ■ **filter** - A filter specification to be applied to the queried VoiceQuality instances. ■ **keys** - Keys that the queried VoiceQuality instances must match. ■ **orderBy** - Ordering specification. **Return type:** QueryIterator over VoiceQuality instances.
getVoiceQualityChunks (*filter*=[], *keys={}*, *orderBy*=[])	Query all voice quality information, that belong to this call. **Parameters** ■ **filter** - A filter specification to be applied to the queried Chunk instances. ■ **keys** - Keys that the queried Chunk instances must match. ■ **orderBy** - Ordering specification. **Return type:** QueryIterator over Chunk instances.

class libpalladion.scripting.Leg (*facade*)

The Leg class represents a leg found by Operations Monitor. A leg is the portion of a call that happens between two given devices. Table 9–2 describes the fields for the Leg class.

Table 9–2 Leg Class Fields

Field	Description
id: INTEGER	Corresponds to the **Call.id** attribute of the associated call.
state: INTEGER	A numeric representation of the call state.
state_msg: STRING	A string describing the call state. One of 'Unauthorized', 'Proceeding', 'Ringing', 'Established', 'Finished', 'Timed out', 'Error', 'Failed','Closed by CleanBye', 'Not found', 'Moved', 'Off-line', 'Busy', 'Canceled'.
state_details: STRING	An additional description of the call state.
setup_start_ts: INTEGER	Timestamp in seconds of the first message in this leg.
setup_time: INTEGER	The time in milliseconds that the leg took to reach the 'Established' state. Only available, if state_msg equals 'Established'.
call_time: INTEGER	The time in milliseconds that the leg was in the 'Established' state. Only available, if state_msg equals 'Finished'.
code: INTEGER	The final response code in this leg.
src_user: STRING	The identifier of the caller.
src_uri: STRING	The URI of the caller.
src_ip: STRING	The IP address of the caller.
src_mac: STRING	The hardware address of the caller.
src_port: INTEGER	The port the caller is listening on.
src_ua: STRING	The user agent string of the caller.
dst_user: STRING	The identifier of the callee.
dst_uri: STRING	The URI of the callee.
dst_ip: STRING	The IP address of the callee.
dst_mac: STRING	The hardware address of the callee.
dst_port: INTEGER	The port the callee is listening on.
dst_ua: STRING	The user agent string of the callee.
ruri: STRING	The request URI.
callid: STRING	The value of the Call-ID header.
from_tag: STRING	The value of the tag parameter of the From header.
to_tag: STRING	The value of the tag parameter of the To header.
getMessages (*filter*=[])	Query all SIP messages, that belong to this leg. **Parameters** ■ **filter** - A filter specification to be applied to the queried Message instances. **Return type:** QueryIterator over Message instances.
getRawMessages (*filter*=[])	Query all raw signaling messages, that belong to this leg. **Parameters** ■ **filter** - A filter specification to be applied to the queried Message instances. **Return type:** QueryIterator over RawMessage instances.

class libpalladion.scripting.RegistrationEvent (*facade*)

This class represents registration events found by Operations Monitor. For a general description of registration events in Operations Monitor, see "Registrations".

Table 9–3 describes the fields for the Registration Event class.

Table 9–3 Registration Event Class Fields

Field	Description
id: INTEGER	A unique ID for this registration event.
ts: DATETIME	The timestamp of this registration event.
realms: INTEGER	The realms bit mask for this registration event.
type: INTEGER	The type of this registration event.
type_msg: STRING	An additional message associated with the type of registration event.
user: STRING	The identifier of the user that is associated with this registration event.
ip: STRING	The source IP address of this registration event.
dest_ip: STRING	The destination IP address of this registration event.
contacts: STRING	The content of the contact header of the REGISTER request.
code: INTEGER	The response code for this registration event.
dev_id: INTEGER	The unique identifier of the device that handled the registration event.
getMessages (*filter*=[])	Query all SIP messages, that belong to this registration event. **Parameters** ▪ **filter** - A filter specification to be applied to the queried Message instances. **Return type:** QueryIterator over Message instances.
getRawMessages (*filter*=[])	Query all raw SIP messages, that belong to this registration event. **Parameters** ▪ **filter** - A filter specification to be applied to the queried RawMessage instances. **Return type:** QueryIterator over RawMessage instances.

class libpalladion.scripting.Message(*facade*)

This class encapsulates a single SIP message. Table 9–4 describes the fields for the Message class.

Table 9–4 Message Class Fields

Field	Description
ts: DATETIME	The timestamp, when the message was received by Operations Monitor.
request: BOOLEAN	True if the message is a request, False otherwise.
proto: STRING	Protocol used.
src_ip: STRING	Source IP address.
src_mac: STRING	Source MAC address.
dst_ip: STRING	Destination IP address.
dst_mac: STRING	Destination MAC address.

Table 9–4 (Cont.) Message Class Fields

Field	Description
method: STRING	The request method of this message. Only present, if this is a request message.
ruri: STRING	The request URI of this message. Only present, if this is a request message.
code: INTEGER	The response code of this message. Only present, if this is a response message.
reason: STRING	The response reason of this message. Only present, if this is a response message.
headers: LIST	A list of (fieldname, field body) tuples, representing the headers of this message.
body: STRING	The message body.
getBody ()	Returns the body of the message. **Return type:** str.
getHeaders ()	Returns a list of (fieldname, field body) tuples for all defined header fields. Field name and field body are both instances of *str*. **Return type:** list of tuples.
getHeadersByName (*name*)	Returns a list of header field bodies for header fields where the field name equals the given *name*. A field body is an instance of *str*. **Parameters** ▪ **name** - Name to match header fields against. **Return type:** list of str.
isRequest ()	Returns, whether this message is a request. **Return type:** bool.

class libpalladion.scripting.RawMessage(*facade*)

This class encapsulates a raw signaling message. Table 9–5 describes the fields for the Raw Message class.

Table 9–5 Raw Message Class Fields

Field	Description
ts: DATETIME	The timestamp, when the message was received by Operations Monitor.
h: DICT	A dictionary representing the PCAP header.
frame: STRING	The raw data.

class libpalladion.scripting.VoiceQuality(*facade*)

This class represents voice quality measurements for a call found by Operations Monitor, or reported by one of the User Agents from the call. A call can have more Voice Quality instances associated, one for each RTP stream.

It cannot be invoked directly. It is used for the return objects of the method **getVoiceQuality** for **Call** objects.

For more information on how Operations Monitor gathers voice quality data, see "Voice Quality".

Table 9–6 describes the fields for the Voice Quality class.

Table 9–6 Voice Quality Class Fields

Field	Description
start_ts: DATETIME	The timestamp of the first measured RTP packet.
probe: STRING	IP address of the probe that received the data.
vlan: INTEGER	VLAN of the probe.
direction: STRING	The direction of the RTP stream that was measured. One of 'src2dst' or 'dst2src'.
dst_ip: STRING	The destination IP of the call.
dst_port: INTEGER	The destination port of the call.
src_ip: STRING	The source ip of the call.
src_port: INTEGER	The source port of the call.
start_ts: DATETIME	The timestamp of the call.
packets_received: INTEGER	The number of RTP packets received.
packets_lost_rate: INTEGER	Packet loss rate for the RTP stream.
forward: BOOLEAN	True if forward.
source: STRING	Source Device Name.
moscqe_avg: FLOAT	MOS average for voice quality.
jitter_total: FLOAT	Sum of the jitter value for each packet, in milliseconds.
jitter_max: FLOAT	Average jitter value for the RTP packets from the call.
latency: INTEGER	Latency as reported by the client's User Agent, when available.
r_factor: FLOAT	R-factor value for voice quality.
rtcp: OBJECT	If present, this optional object contains the following fields: ■ start_ts: DATETIME The timestamp of the first measured RTP packet. ■ end_ts: DATETIME ■ The timestamp of the last measured RTP packet. ■ jitter_max: FLOAT Maximum jitter value for a single packet, in milliseconds. ■ jitter_avg: FLOAT Average jitter value for the RTP packets from the call. ■ packets_lost: INTEGER Packets lost for the RTP stream. ■ source: STRING Source Device Name.

class libpalladion.scripting.Chunk(*facade*)

This class represents a single chunk of voice quality data.

It cannot be invoked directly. It is used for the return objects of the method **getVoiceQualityChunks** for **Call** objects.

For more information on how Operations Monitor gathers voice quality data, see "Voice Quality".

Table 9–7 describes the fields for the Chunk class.

Table 9–7 Chunk Class Fields

Field	Description
start_ts: DATETIME	The timestamp of the first measured RTP packet.
end_ts: DATETIME	The timestamp of the last measured RTP packet.
received: INTEGER	The number of RTP packets received.
expected: INTEGER	The number of RTP packets expected.
codec: STRING	The codec used for this chunk.
pkt_delay_variation_us: DOUBLE	Variation on packet delay.
no_moscqe_reason: STRING	If there is no MOS data, its cause.

class libpalladion.scripting.Device(*facade*)

This class represents a platform device configured in Operations Monitor. For a general introduction on platform devices in Operations Monitor, see "Platform Devices".

Table 9–8 describes the fields for the Device class.

Table 9–8 Device Class Fields

Field	Description
id: INTEGER	The unique numeric ID of the device.
type: STRING	The type of the device. One of 'SBC', 'PROXY','L2LB', 'GW', or 'TRUNK'.
name: STRING	The user visible name of the device.
description: STRING	A description for the device.
match_type: STRING	If the device is of type 'SBC', this value indicates how call matching is performed.
suffix: INTEGER	Some call matching types require a suffix parameter.
match_script: STRING	This attribute contains a script that specifies the call matching algorithm.
ipranges: STRING	The IP address ranges which are occupied by this device.
hw_addrs: STRING	The hardware addresses of this device.
up_since: DATETIME	A timestamp specifying the time since when the device is known to be running.
dtg: STRING	If the device is of type TRUNK and device identification via DTG/OTG URI parameters was configured, this attribute contains the value of the DTG request URI parameter to look for.
otg: STRING	If the device is of type TRUNK and device identification via DTG/OTG URI parameters was configured, this attribute contains the value of the OTG From URI parameter to look for.

Table 9–8 (Cont.) Device Class Fields

Field	Description
getCreatedCalls (*filter*=[], *keys*={}, *orderBy*=[])	Query calls that are created by this device, that is, calls that have an outbound leg from this device and no inbound leg to this device. See Facade.query() for the meaning of the parameters. **Parameters** ■ **filter** - A filter specification to be applied to the queried Call instances. ■ **keys** - Keys that the queried Call instances must match. ■ **orderBy** - An ordering specification. **Return type:** QueryIterator over Call instances.
getRegistrationEvents (*filter*=[], *keys*={}, *orderBy*=[])	Query registration events that are handled by this device. See Facade.query() for the meaning of the parameters. **Parameters** ■ **filter** - A filter specification to be applied to the queried RegistrationEvent instances. ■ **keys** - Keys that the queried RegistrationEvent instances must match. ■ **orderBy** - An ordering specification. **Return type:** QueryIterator over RegistrationEvent instances.
getRelayedCalls (*filter*=[], *keys*={}, *orderBy*=[])	Query calls that are relayed by this device, that is calls that have an inbound leg to this device and an outbound leg from this device. See Facade.query() for the meaning of the parameters. **Parameters** ■ **filter** - A filter specification to be applied to the queried Call instances. ■ **keys** - Keys that the queried Call instances must match. ■ **orderBy** - An ordering specification. **Return type:** QueryIterator over Call instances.
getTerminatedCalls (*filter*=[], *keys*={}, *orderBy*=[])	Query calls that are terminated by this device, that is, calls that have an inbound leg to this device and no outbound leg from this device. See Facade.query() for the meaning of the parameters. **Parameters** ■ **filter** - A filter specification to be applied to the queried Call instances. ■ **keys** - Keys that the queried Call instances must match. ■ **orderBy** - An ordering specification. **Return type:** QueryIterator over Call instances.

class libpalladion.scripting.Counter(*facade*)

A Counter is an object in Operations Monitor that takes periodic measurements of a numeric property of the monitored platform. For a general introduction on counters, see "KPI/Metrics".

Table 9–9 describes the fields for the Counter class.

Table 9–9 Counter Class Fields

Field	Description
id: INTEGER	A unique identifier for the counter.
name: STRING	The name of the counter.
maintype: INTEGER	The main type of the counter.
subtype: INTEGER	The sub type of the counter.
device: STRING	The device this counter belongs to. May be None.
p1: STRING	The first parameter for the counter.
p2: STRING	The second parameter for the counter.
p3: STRING	The third parameter for the counter.
sec: INTEGER	The last second measurement.
m_avg: FLOAT	The last minute average of measurements.
h_avg: FLOAT	The last hour average of measurements.
getHourValues (*span*, *offset=0*)	Query hour average values from the counter history. The returned query iterator yields a dict instance for each hour in the supplied time span. The yielded dict instances contain the keys *avg*, *min*, *max* and *sum*, with the obvious meanings. The time span is given by the parameters *span* and *offset*, where *offset* specifies the end point of the time span relative to the current time. **Parameters** ■ **span** - The length of the queried time span in seconds. ■ **offset** - The offset of the queried time span from the current time, specified in seconds. **Return type:** QueryIterator over dict instances.
getMinuteValues (*span*, *offset=0*)	Query minute average values from the counter history. The returned query iterator yields a dict instance for each minute in the supplied time span. The yielded dict instances contain the keys *avg*, *min*, *max*, and *sum*, with the obvious meanings. The time span is given by the parameters *span* and *offset*, where *offset* specifies the end point of the time span relative to the current time. **Parameters** ■ **span** - The length of the queried time span in seconds. ■ **offset** - The offset of the queried time span from the current time, specified in seconds. **Return type:** QueryIterator over dict instances.

class libpalladion.scripting.Alert(*facade*)

This class represents alerts raised by Operations Monitor. Table 9–10 describes the fields for the Alert class.

Table 9–10 Alert Class Fields

Field	Description
ts: DATETIME	The time when this alert was raised.
type: STRING	The type of the alert raised.
subtype: STRING	The sub type of the alert raised.
new: BOOLEAN	Whether this alert is new or read.

Table 9–10 (Cont.) Alert Class Fields

Field	Description
message: STRING	The message issued when raising the alert.
details: STRING	Some details about the raised alert.
prio: INTEGER	The priority of the alert.

class libpalladion.scripting.RegisteredUser(*facade*)

This class represents a registered user. Table 9–11 describes the fields for the Registered User class.

Table 9–11 Registered User Class Fields

Field	Description
identifier: STRING	Username of the registered user.

class libpalladion.scripting.QueryIterator(*facade*,*query_handle*, *cls=None*)

Instances of this class can be used to iterate over query result sets. You cannot instantiate a QueryIterator directly, but various query methods in this API return QueryIterator instances.

Instances conform to the Python iterator protocol and can be used in 'for item in iterable:' constructs. You can also use the next() method directly to retrieve one result at a time. QueryIterator instances also provide a close() method, which frees resources tied by the query. Once close() has been called, iteration stops. close() is called automatically when all references to a QueryIterator instance are gone.

Table 9–12 describes the fields for the Query Iterator class.

Table 9–12 Query Iterator Class Fields

Field	Description
close()	Free the resources tied up by the query.
next()	Return the next item in the query result set. **Return type:** Depends on the type of query.

Using the Apps API Examples

The following app searches the Operations Monitor database for calls which have a given Call-ID header and have been set up between two given points in time:

```
from libpalladion.scripting.Model import Call, Leg

def run(facade, args):

    # Get input parameters
    start_ts = args.get("start_ts")
    end_ts = args.get("end_ts")
    callid = args.get("callid")

    callid = callid.strip()

    for call in facade.getCalls(
          filter=[Call.setup_start_ts >= start_ts, Call.setup_start_ts <= end_ts]
        ):
        # search for the specific callid in all the call legs
```

```
        # the callid of each call leg might be different
        for leg in call.getLegs():
             if leg.callid == callid:
                 facade.addResult(
                    {"id": call.id},
                    on_duplicate_key='ignore'
                 )
```

This app has to be packaged with an app specification file, which should specify the type of the result table (*calls* in this case), and the names and types of the parameters *callid*, *start_ts*, and *end_ts*. The following specification file achieves this:

```
<?xml version="1.0" encoding="utf-8"?>
<script xmlns="http://iptego.de/palladion/script-spec"
   name="Call-Id Search"
   description="This script will find all the calls with a given Call-ID value"
   result-type="calls">
   <param-spec>
     <param name="start_ts" label="Started after" type="datetime" required="yes"/>
     <param name="end_ts" label="Started before" type="datetime" required="yes"/>
     <param name="callid" label="Call-Id" type="string" required="yes"/>
   </param-spec>
</script>
```

Note, that the parameters specified as *datetime* are passed as the standard Python datetime type. If the *result-type* of an app is specified as calls, it has a result table with one integer column, ID. When displaying the results table of such an app in the user interface, a join with the calls table is performed. In the example above, the call to *facade.addResult* has to be surrounded by *try ... except* because there can be multiple **libpalladion.scripting.Leg** objects, with the same ID attribute. This is because a call can have multiple legs. Adding the same ID twice to the calls result table results in a duplicate key error.

In the next example we look at an app that requires a custom result table. We will see how this custom result table can be specified in the app specification file. The app calculates a statistics of which user agents use which codecs how often. Again the source data for the app is restricted to calls that were set up between two given points in time.

```
from libpalladion.scripting.Model import Call

def run(facade, args):
    start_ts = args.get("start_ts")
    end_ts = args.get("end_ts")

    stats = {}
    filter = [Call.setup_start_ts > start_ts, Call.setup_start_ts <= end_ts]
    for call in facade.getCalls(filter=filter):
        ua = call.src_ua
        if ua is not None:
            codecs = call.src_codecs
            if (ua, codecs) not in stats:
                stats[(ua, codecs)] = 0
            stats[(ua, codecs)] += 1

    for ((ua, codecs), count) in stats.items():
         facade.addResult({'ua': ua, 'codecs': codecs, 'count': count})
```

The **facade.addResult** call on the last line makes the assumption that the result table has the columns **ua**, **codecs**, and **count**. So this schema has to be specified in the script specification file.

```
<?xml version="1.0" encoding="utf-8"?>
<script xmlns="http://iptego.de/palladion/script-spec"
    name="Statistics of which user agent uses which codecs"
    description="This app calculates a statistic of which user agents use which
codecs"
    result-type="custom">
    <param-spec>
        <param name="start_ts" label="Start time" type="datetime" required="yes"/>
        <param name="end_ts" label="End time" type="datetime" required="yes"/>
    </param-spec>
    <result-schema>
        <column name="ua" type="VARCHAR(255)" null="false"/>
        <column name="codecs" type="VARCHAR(255)" null="false"/>
        <column name="count" type="INT(11)" null="false" default="1"/>
        <primary-key columns="ua codecs"/>
    </result-schema>
</script>
```

Caution: When creating your own applications, or using third-party applications, test your scripts in a test environment to ensure they are safe before uploading them to your production environment.

Applications approved by Oracle are safe to use in your environments. However, non-approved applications could cause security and performance issues. Oracle is not responsible for any loss, costs, or damages incurred from using your own applications, or third-party applications.

Remote App Procedure Calls

Remote app procedure calls provide programmatic access to the app functionality of Operations Monitor. It is possible to invoke apps installed on Operations Monitor and get their results via a simple HTTP GET interface. For more information, see "Apps" and "Implementing Apps".

Apps are possibly long running processes. Therefore invoking an app and retrieving the results of an app run have been split into two separate steps.

Invoking an App

Invoking an app is done by an HTTP GET request to the URL **/scripts/run/***app_name* on the Operations Monitor host, where *app_name* is to be replaced by the identifier of an app. Note, that you can lookup the identifier of an app in the **Id** column of that app's entry in the Available Apps table. The parameters of the app have to be encoded as URL parameters into the GET request URL. For more information on the Apps table, see "Apps".

If invoking the app succeeds, an app run is created, and the GET request returns the numeric run ID for the app run in the response body. This run ID can later be used to retrieve the results of the app run.

There are two possible failure cases when invoking an app. First, the given app name could be invalid. In this case Operations Monitor responds with an HTTP '404 Not

found' error code. Secondly, the parameters given in the request could be invalid when a required parameter is missing or a given parameter cannot be parsed. In this case Operations Monitor responds with an 'HTTP 400 Bad request' error code.

A note regarding parameter encoding is required here: Datetime values have to be encoded in a special way to make Operations Monitor recognize them. Operations Monitor expects datetime values to be in the format YYYY-mm-dd HH:MM:SS, where the 4-digit string YYYY gives the year, mm gives the month, dd gives the day, HH gives the hour, MM gives the minute, and SS gives the second.

Retrieving the Results of an App Run

Retrieving the results of an app run is done by an HTTP GET request to the URL **/scripts/get/***run_id* on the Operations Monitor host, where *run_id* is a numeric run ID. Run IDs are returned when invoking an app. They can also be looked up in the **Id** column of the **App runs** table. For more information, see "Apps".

There are three possible outcomes of this request. First, if the run ID given is invalid an HTTP '404 Not found' error code is returned. Secondly, if the run ID is valid but the app run is not yet finished, an HTTP '202 Accepted' status code is returned. In this case the body of the response contains the string **202 Accepted**. The third case is when the run ID is valid and the app run is finished already. In this case an HTTP '200 OK' status code is returned, and the body of the response contains the app run result table in CSV format.

10

REST API

A simple REST API facilitates access to much of the information generated by Oracle Communications Operations Monitor.

REST is a way of implementing Remote Procedure Call (RPC) systems on top of HTTP. REST APIs are stateless, since they don't use mechanisms of application state keeping that are traditionally employed together with HTTP, such as cookies or sessions. Everything necessary for keeping the application state must be encoded into the HTTP requests. REST APIs are structured around addressable resources, that means all information, that is provided or manipulated by the API, is addressable with HTTP URLs. Conceptually, resources are like the files and directories of a file system, organized into a tree of parents and children.

From the generally used HTTP methods GET, POST, PUT, and DELETE, Operations Monitor's REST API only uses the GET method, as the interface is intended to only query information from Operations Monitor.

Note: The remote application needs to authenticate itself with the administrator password over the standard HTTP digest mechanism.

Note: REST API searches are not guaranteed to be real-time without advanced hardware sizing.

Example call:

```
$ curl -L --digest -u admin:<passwd>
https://10.1.0.81/r/users/00493077715680/calls/recent

  {"data": [
    {"state_msg": "Finished",
     "code": 200,
     "dst_codecs": "PCMU,PCMA,iLBC,telephone-event",
     "src_codecs": "PCMU,PCMA,telephone-event,iLBC",
     "pid": 1250018007,
     "url": "\/r\/users\/00493077715680\/calls\/1250018007:16739",
     [...]
    }
    [...]
 ]}
```

Interface Description

The resources accessible via the REST interface are encoded in JSON. You can access the following resources on each Operations Monitor system (with variables denoted in angle brackets):

/r/calls

A structure containing information about the available calls. It contains the fields listed in Table 10–1:

Table 10–1 /r/calls Fields

Field	Description
end_call_id	The call ID of the last call.
end_ts	A timestamp giving the second in which the last call was started.
name	Is always **calls**.
start_call_id	The call ID of the first call (not the same ID used in conjunction with a partition ID).
start_ts	A timestamp giving the second in which the first call was started.
total	The number of calls in the partition.

Note: In the Operations Monitor version before 3.1 this method returned a list of structures with partitioning information. Starting with version 3.1, partitions are no longer used.

/r/calls/recent

The list of recent calls similar to the one that is displayed in the recent calls table in the web interface. This resource is a list resource and has the common format of list resources and allows filtering and paging. For more information, see "The Format of List Resources".

The list items are structures consisting of the fields listed in Table 10–2:

Table 10–2 /r/calls/recent Fields

Field	Description
call_time	The time the call was in state ESTABLISHED.
code	The code of the last response for the first INVITE message.
dst_codecs	The codecs proposed by the callee.
dst_initial_codecs	The codecs initially proposed by the callee.
dst_ip	The IP address of the called user that connected first. This may be an empty string.
dst_ua	The user agent of the callee.
dst_user	The identifier of the callee.
egress_devs	The egress device(s).
id	Together with **pid** makes up the unique identifier of a call.
in_devs	The in device(s).

Table 10–2 (Cont.) /r/calls/recent Fields

Field	Description
ingress_devs	The ingress device(s).
init_devs	The initiator device.
MOSlqe_avg	The average MOS estimation for this call.
MOSlqe_min	The minimum MOS estimation for this call.
nlegs	The number of legs in the call.
out_devs	The out device(s).
pid	Together with **id** makes up the unique identifier of a call.
realms	A bitmask of the realms this call belongs to.
setup_start_ts	The time the first INVITE message was issued.
setup_time	The time before the call reached the state ESTABLISHED.
src_codecs	The audio codecs proposed by the caller.
src_initial_codecs	The audio codecs initially proposed by the caller.
src_ip	The IP address of the device initiating the call. This includes the IP address string.
src_ua	The user agent of the caller.
src_user	The caller identifier.
state_details	Details about the current state of the call.
state_msg	The current state of the call.
term_devs	The terminator device.
url	The url of the full call rest resource.

/r/calls/<*start_ts*>/<*end_ts*>

A list of calls started between <*start_ts*> and <*end_ts*>. This list is structured exactly like the list of recent calls, but the calls are from a definable timespan. The timestamps have to be written in ISO 8601 format and then be url encoded.

The following query gets the calls started between, 22 Mar 2010 16:30 and 22 Mar 2010 17:00:

```
$ curl -L -digest -user admin:<passwd> https://10.1.0.81/r/calls/2010-03-
22+16:30:00/2010-03-22+17:00:00
```

A list of calls might be filtered by following parameters:

- MOSlqe_avg
- MOSlqe_min
- init_devs
- term_devs
- traversing_devs
- ingress_devs
- egress_devs
- gateway_devs

For Example:

```
$ curl -L -digest -user admin:<passwd> https://10.1.0.89/r/calls/2014-01-
01+00:00:00/2014-02-01+00:00:00?term_devs__eq=52
```

/r/calls/<*start_ts*>/<*end_ts*>/<*pid*>:<*id*>

The call with the given <*pid*> and <*id*>. The returned resource is a structure with the fields listed in Table 10–3:

Table 10–3 /r/calls/<start_ts>/<end_ts>/<pid>:<id> Fields

Field	Description
call_id	An identifier that makes up a unique identifier of a call. Can be used instead of <*pid*> and <*id*>.
call_time	The time the call was in state ESTABLISHED.
code	The code of the last response for the first INVITE message.
dst_user	The identifier of the callee.
id	Together with **pid** makes up the unique identifier of a call.
legs	A list of legs belonging to this call.
messages_url	The url of the messages resource for this call.
nlegs	The number of legs in the call.
pid	Together with **id** makes up the unique identifier of a call.
realms	A bitmask of the realms this call belongs to.
setup_start_ts	The time the first INVITE message was issued.
setup_time	The time before the call reached state ESTABLISHED.
src_user	The caller identifier.
state_details	Details about the current state of the call.
state_msg	The current state of the call.
vq_url	The url of the voice quality resource for this call.

Note: <*call_id*> can be used to identify a call instead of the <*pid*>:<*id*> combination.

/r/calls/<*start_ts*>/<*end_ts*>/<*pid*>:<*id*>/messages

A representation of the SIP messages of a given call. This resource is a list resource and has the common format of list resources and allows paging but not filtering. For more information, see "The Format of List Resources".

The list items are structures with the fields listed in Table 10–4:

Table 10–4 /r/calls/<start_ts>/<end_ts>/<pid>:<id>/messages Fields

Field	Description
code	If the message is a response message - the response code.
data	The whole message as a string.
dst_ip	The destination IP address.

Table 10–4 (Cont.) /r/calls/<start_ts>/<end_ts>/<pid>:<id>/messages Fields

Field	Description
dst_mac	The destination hardware address.
dst_port	The destination port.
method	If the message is a request message - the request method. For example, INVITE or ACK.
msgid	The unique ID of this message.
proto	UDP or TCP.
reason	If the message is a response message - the response reason.
ruri	If the message is a request message - the request uri of this message.
src_ip	The source IP address.
src_mac	The source hardware address.
src_port	The source port.
ts	The time this message was received by Operations Monitor.

/r/calls/<*start_ts*>/<*end_ts*>/<*pid*>:<*id*>/vq

Voice quality information for each RTP stream that belongs to the given call. This resource is a list resource and has the common format of list resources. It does not allow for filtering or paging. For more information, see "The Format of List Resources".

The list items are structures with the fields listed in Table 10–5:

Table 10–5 /r/calls/<start_ts>/<end_ts>/<pid>:<id>/vq Fields

Field	Description
avg_jitter	Average jitter value for the packets from the RTP stream.
burst_cnt	The number of sequences of lost RTP packets (gaps).
cid	Corresponds to the ID of the call the voice quality information belongs to.
decoder	The codec used by the RTP stream.
dir	The direction of the RTP stream either 'src2dst' or 'dst2src'.
expected	Expected number of packets in the RTP stream.
first_ts	The timestamp of the first measured RTP packet.
Ieeff	Effective Equipment Impairment Factor.
last_ts	The timestamp of the last measured RTP packet.
lid	No meaningful value.
max_burst	The maximum number of RTP packets lost in sequence (in a single gap).
max_jitter	The maximum jitter value for the packets from the RTP stream.
MOS	MOS estimation for voice quality.
pid	The pid of the call this voice quality information belongs to.
pl_rate	Packet loss rate for the RTP stream.
R	R-value score for voice quality.

Table 10–5 (Cont.) /r/calls/<start_ts>/<end_ts>/<pid>:<id>/vq Fields

Field	Description
received	Received number of packets in the RTP stream.
source	The source that generated the voice quality information. For example, Operations Monitor.
total_jitter	Total jitter value for the packets from the RTP stream.
ts	The timestamp.

/r/registrations

The list of recent registration events. This resource is a list resource and has the common format of list resources and allows for filtering and paging of the list items. For more information, see "The Format of List Resources".

The list items are structures with the fields listed in Table 10–6:

Table 10–6 /r/registrations Fields

Field	Description
code	The response code of the last response to the REGISTER request.
contacts	The content of the contact header of the REGISTER request.
dest_ip	The destination IP address of the registration event.
dev_id	The numeric ID of the platform device that handled the registration event.
device	The name of the device that handled the registration event.
id	The unique ID of the registration event.
ip	The source IP address of the registration event.
realms	A bitmask of the realms this event belongs to.
ts	The time stamp of the registration event.
type	The type of registration event.
type_msg	Human readable form of the type of registration event. One of New, Failed, Unauthorized, Expired, or Gone.
url	The url of the full registration event resource.
user	The identifier of the registered user.

/r/registrations/<*id*>

The registration event with the given <*id*>. This resource is a structure with the fields listed in Table 10–7:

Table 10–7 /r/registrations/<id> Fields

Field	Description
code	The response code of the last response to the REGISTER request.
contacts	The content of the contact header of the REGISTER request.
dest_ip	The destination IP address of the registration event.
dev_id	The numeric ID of the platform device that handled the registration event.
device	The name of the device that handled the registration event.

Table 10–7 (Cont.) /r/registrations/<id> Fields

Field	Description
id	The unique ID of the registration event.
ip	The source IP address of the registration event.
messages_url	The url of the messages resource for this registration event.
realms	A bitmask of the realms this event belongs to.
type	The type of registration event.
type_msg	Human readable form of the type of registration event. One of New, Failed, Unauthorized, Expired, or Gone.
user	The identifier of the registered user.

/r/registrations/<*id*>/messages

The SIP messages belonging to a given registration event. This resource is a list resource and has the common format of list resources and allows paging but not filtering. The list items are exactly like the list items of the **/r/calls/**<*pid*>**:**<*id*>**/messages** resource. For more information, see "The Format of List Resources".

/r/devices

The list of configured platform devices. This resource is a list resource and has the common format of list resources. For more information, see "The Format of List Resources".

The list items are structures with the fields listed in Table 10–8:

Table 10–8 /r/devices Fields

Field	Description
id	The device ID.
inbound	The number of inbound calls to this device.
name	The device name.
outbound	The number of outbound calls from this device.
url	The resource url for this device.
users	The number of registered users at this device.

/r/devices/<*devid*>

A representation of the given platform device. This resource is a structure with the fields listed in Table 10–9:

Table 10–9 /r/devices/<devid> Fields

Field	Description
created_calls_url	The url to the created_calls resource for this device.
id	The device ID.
inbound	The number of inbound calls to this device.
name	The device name.
outbound	The number of outbound calls from this device.

Table 10–9 (Cont.) /r/devices/<devid> Fields

Field	Description
registrations_url	The url to the registrations resource for this device.
relayed_calls_url	The url to the relayed_calls resource for this device.
terminated_calls_url	The url to the terminated_calls resource for this device.
users	The number of registered users at this device.

/r/devices/<*devid*>/created_calls

A list of calls created by the given platform device. The format of this resource is exactly like the format of the **/calls** resource.

/r/devices/<*devid*>/relayed_calls

A list of calls which are relayed by the given platform device. The format of this resource is exactly like the format of the **/calls** resource.

/r/devices/<*devid*>/terminated_calls

A list of calls which are terminated by the given platform device. The format of this resource is exactly like the format of the **/calls** resource.

/r/devices/<*devid*>/registrations

A list of registration events handled by the given platform device. The format of this resource is exactly like the format of the **/registrations** resource.

/r/users/<*userid*>

Information belonging to a user registered on the monitored platform, similar to the information displayed on the user tracking page. This resource is a structure with the fields listed in Table 10–10:

Table 10–10 /r/users/<userid> Fields

Field	Description
calls_url	The url to the calls resource for this user.
current_registrations	A list containing current registrations of the user.
registrations_url	The url to the registrations resource for this user.
user	The user ID.

The **current_registrations** field is a list whose list items are structures with the fields listed in Table 10–11:

Table 10–11 current_registrations Fields

Field	Description
dev_id	The numeric ID of the platform device that handled the current registrations event.
expires	The negotiated expiry time of the registration.
expires_in	The time remaining until registration expiry.
first_seen_ts	First time the user registered.
last_refreshed_ts	The last time the user refreshed the registration.

Table 10–11 (Cont.) current_registrations Fields

Field	Description
last_seen_ts	The last time the user registered.
last_suggested_expires	The value of the expires header of the REGISTER request.
link_quality	An indication of the quality of the network link between platform and user.
srcip	The source IP address of the current registrations event.
uri	The uniform resource identifier (URI) for the current registrations resource.
usrdev	The name of the device that handled the current registrations event.

/r/users/<*userid*>/calls

Calls belonging to the given platform user. The format of this resource is exactly like the format of the **/r/calls** resource.

/r/users/<*userid*>/registrations

Registration events belonging to the given platform user. The format of this resource is exactly like the format of the **/r/registrations** resource.

/r/counters

The list of counters for the admin user and the **ALL** realm. This resource is a list resource and has the common format of list resources. For more information, see "The Format of List Resources".

The list items are structures with the fields listed in Table 10–12:

Table 10–12 /r/counters Fields

Field	Description
datatype	The datatype of the counter.
h_avg	The average of measurements over one hour.
h_max	The maximum of measurements over one hour.
h_max	The maximum of measurements over one hour.
h_sum	The sum of measurements over one hour.
id	The numeric counter ID.
m_avg	The average of measurements over one minute.
m_max	The maximum of measurements over one minute.
m_min	The minimum of measurements over one minute.
m_sum	The sum of measurements over one minute.
maintype	The numeric counter main type.
name	The counter name.
p1	Counter parameter one.
p2	Counter parameter two.
p3	Counter parameter three.
subtype	The numeric counter sub type.

Table 10–12 (Cont.) /r/counters Fields

Field	Description
url	The url to the full counter resource.
value	The current second measurement.

/r/counters/<*id*>

A representation of the given counter. This resource is a structure with the fields listed in Table 10–13:

Table 10–13 /r/counters/<id> Fields

Field	Description
autoreset	If the counter is reset every second.
datatype	The counter datatype.
days	For average counters, the number of days to average over.
hours_url	The url to the hours resource for this counter.
id	The numeric counter ID.
maintype	The numeric counter main type.
minutes_url	The url to the minutes resource for this counter.
name	The name of the counter.
owner	The owner of the counter: either "system" or "user".
p1	Counter parameter one.
p2	Counter parameter two.
p3	Counter parameter three.
realm	The realm that this counter belongs to.
seconds_url	The url to the seconds resource for this counter.
src_cnt	For average counters, the source counter for the average.
subtype	The numeric counter sub type.
user_id	The user ID of the user who owns the counter.
value	The current second measurement.
weekdays	For average counters, whether the average is calculated for each weekday separately.

/r/counters/<*id*>/seconds

The list of recent second measurements for the given counter. This resource is a list resource and has the common format of list resources and allows paging. The list items are integer values. The list contains up to 3600 values, one value for each of the last 3600 seconds, in chronological order from oldest to youngest. For more information, see "The Format of List Resources".

/r/counters/<*id*>/minutes

The list of minute values for the given counter. This resource is a list resource and has the common format of list resource and allows paging. The list items are float values. The list contains all the minute averages of the counter measurements stored in the database of Operations Monitor in chronological order from oldest to youngest. For

more information, see "The Format of List Resources".

/r/counters/<*id*>/hours

The list of hour values for the given counter. This resource is a list resource and has the common format of list resources and allows paging. The list items are float values. The list contains all the hour averages of the counter measurements stored in the Operations Monitor database in chronological order from oldest to youngest. For more information, see "The Format of List Resources".

The Format of List Resources

List resources like the **/r/calls/recent** resource are structures with the fields listed in Table 10–14:

Table 10–14 List Resources Fields

Field	Description
data	The actual list. The format of the list item depends on the resource.
limit	The number of returned items.
start	The index of the first returned item.
total	The total number of items that exist.

All list resources allow paging. Paging is enabled by default and can be controlled by giving the **start** and **limit** GET parameters when requesting the resource. The **start** parameter gives the index of the first item to be returned, the **limit** parameter gives the number of items to be returned.

Note, that *limit* cannot exceed the value given in the **vsp.rest.max_limit** system setting and is adjusted if it does exceed this value. For the counter seconds, minutes and hours resource you can also give *start_ts* instead of *start*, meaning that the request will only return items with an associated timestamp that is greater or equal than the given timestamp.

Some list resources also allow filtering. Filtering is controlled by additional GET request parameters. A filter controlling parameter must have the form <*fieldname*>_<*verb*>, where <*fieldname*> is the name of a field of the list items of the resource and <*verb*> determines the logic of the filter. The known verbs and their meaning are listed in Table 10–15:

Table 10–15 List Resources Verbs

Verb	Description
contains	This filter applies if the set valued field contains the set given. The parameter must be given as a list separated by '+'.
deq	This filter applies if the date contained in the field and the given date are equal. The date must be given as "mm/dd/yyyy", where mm is the month, dd the day of the month and yyyy the year.
dgeq	This filter applies if the date contained in the field is younger or equal to the given date. The date must be given as "mm/dd/yyyy", where mm is the month, dd the day of the month and yyyy the year.

Table 10–15 (Cont.) List Resources Verbs

Verb	Description
dgt	This filter applies if the date contained in the field is younger than the date given in the parameter. The date must be given as "mm/dd/yyyy", where mm is the month, dd the day of the month and yyyy the year.
dleq	This filter applies if the date contained in the field is older or equal to the given date. The date must be given as "mm/dd/yyyy", where mm is the month, dd the day of the month and yyyy the year.
dlt	This filter applies if the date contained in the field is older. The date must be given as "mm/dd/yyyy", where mm is the month, dd the day of the month and yyyy the year.
dteq	This filter applies if the date and time contained in the field is equal to the date and time given. The date and time must be given as "yyyy/mm/dd HH:MM:SS", where yyyy is the year, mm the month, dd the day of the month, HH the hour, MM the minute and SS the second.
dtgeq	This filter applies if the date and time contained in the field is younger or equal to the date and time given. The date and time must be given as "yyyy/mm/dd HH:MM:SS", where yyyy is the year, mm the month, dd the day of the month, HH the hour, MM the minute and SS the second.
dtgt	This filter applies if the date and time contained in the field is younger than the date and time given in the parameter. The date and time must be given as "yyyy/mm/dd HH:MM:SS", where yyyy is the year, mm the month, dd the day of the month, HH the hour, MM the minute and SS the second.
dtleq	This filter applies if the date and time contained in the field is older or equal to the date and time given. The date and time must be given as "yyyy/mm/dd HH:MM:SS", where yyyy is the year, mm the month, dd the day of the month, HH the hour, MM the minute and SS the second.
dtlt	This filter applies if the date and time contained in the field is older than the given date and time. The date and time must be given as "yyyy/mm/dd HH:MM:SS", where yyyy is the year, mm the month, dd the day of the month, HH the hour, MM the minute and SS the second.
eq	This filter applies if the numeric value contained in the field is equal to the number given.
geq	This filter applies if the numeric value contained in the field is greater than or equal to the number given.
gt	This filter applies if the numeric value contained in the field is greater than the numeric value given in the parameter.
leq	This filter applies if the numeric value contained in the field is smaller than or equal to the number given.
lt	This filter applies if the numeric value contained in the field is smaller than the number given.
oneof	This filter applies if the value contained in the field is equal to one of the values given. The values must be given as a list separated by '+'.

Table 10–15 (Cont.) List Resources Verbs

Verb	Description
substr	This filter applies if the value of the parameter is contained in the field.

In case the GET request parameters cannot be parsed, an HTTP 400 Bad Request status code is returned.

11

New REST API

With the new REST API almost all Oracle Communications Operations Monitor features are now easily accessible through HTTPS REST calls. Nearly every feature you already use through the web interface of Operations Monitor can now be automated and used with your HTTPS capable toolchain of choice.

Plus with the new API Key, authentication is now more simple and secure than before. Authentication is now as simple as passing your personal API Key along with your request.

API Key

The new API Key is used to authenticate the user when using the new REST API. The API Key is connected to your user account — so when using the REST API with the API Key you have the same access rights as with your user account. Before using the new REST API you first have to enable and generate your personal API Key. The API Key is comprised of the username and a hexcode delimited by a semicolon.

For security reasons the API Key is disabled by default.

To enable your API Key:

1. From the **Settings** menu, select **General Settings**, and then select **REST API**.
2. Select the **Enable REST API** option, which generates a new API Key.

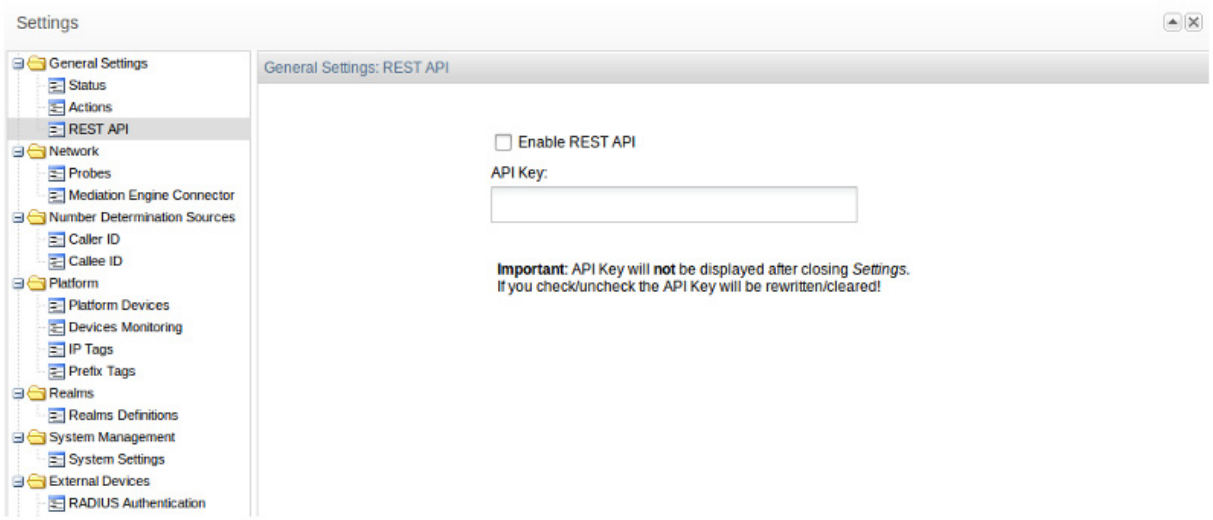
If the **Enable REST API** option is deselected, the API Key will automatically be deleted, which disables the REST API. For more information how to use the API Key and the new REST API, see "Getting Started".

Important: It is strongly recommended that you store your API Key in a safe and secret place. For example, your script's configuration files.

Important: After closing the **Settings** dialog box the API Key will no longer be displayed. The API Key is displayed only once when it is first created.

Figure 11–1 shows the General Settings REST API page.

Figure 11–1 How to Enable the New REST API



Getting Started

To get started with using the new REST API please make sure you have your API Key enabled as described in "API Key".

To test if everything is setup correctly please open a terminal window and have your API Key ready. We are going to use the curl command to call the REST API. Please note that this is an example. What we are doing is essentially just placing a HTTPS request with a custom header **-X-Api-Key** (the custom header contains the API Key). (We use the **-k** option to make sure that we don't run into any certificate complications when using the API).

Example call:

```
$ curl -k -X POST -H "X-API-Key: musterman;samplehexcode"
https://mypalladion.com/me/saveApplianceConfiguration
```

Please replace **musterman;samplehexcode** with your personal API Key. If the API Key is correct and all parameters are provided as per reference, then the HTTP response will be 200 OK with the JSON response containing a **success** field that will have the value **true**.

Usecase Backup

With the new REST API it is now possible to create and download savepoints of configurations automatically. To achieve this we are going to use the **saveApplianceConfiguration** and **downloadApplianceConfiguration** methods.

First we will call the **saveApplianceConfiguration** method, which will create a new savepoint on the server and return the name of the savepoint upon success.

In the second step we will then call the **downloadApplianceConfiguration** method with the filename we just received as a parameter to download the configuration file to our local computer.

The two scripts below automate the task of downloading a configuration savepoint using the two API functions we just discussed. In both scripts it is assumed that the IP address of the instance we want to backup is ME_IP and our API Key is **API_KEY**.

Python 2.7:

```
import requests

API_KEY = 'username;somehexcode'
URL = 'https://'+ ME_IP  +'/me/'
HEADERS = {'X-Api-Key': API_KEY}

resp = requests.get(URL+'saveApplianceConfiguration', headers=HEADERS)
remote_cfg = None

if resp.status_code == 200:
   data = resp.json()
   remote_cfg = data['name']
else:
    print('Unexpected status code: ' + str(resp.status_code))
    exit()

resp = requests.get(URL+'downloadApplianceConfiguration',
                    headers=HEADERS,
                    params={"name": remote_cfg})

if resp.status_code == 200:
    with open(remote_cfg, 'w') as f:
       f.write(resp.text)
else:
    print('Unexpected status code: ' + str(resp.status_code))
```

Even less code is required when Bash scripting is used. The Bash scripting example uses the **curl** command to call the REST API.

```
#!/usr/bin/env bash

API_KEY="X-Api-Key: username;somehexcode"
URL="https://ME_IP/me/"
SAVE_URL="${URL}saveApplianceConfiguration"
DOWNLOAD_URL="${URL}downloadApplianceConfiguration"

remote_cfg=$(curl -k -s -X GET -H"${API_KEY}" $SAVE_URL | grep -Po
'Palladion\-configuration.*\.cfg')

curl -k -s -H "${API_KEY}" -d"name=${remote_cfg}" -o ${remote_cfg} $DOWNLOAD_URL
```

Usecase High Availability

High availability can be achieved by setting up two Mediation Engines — one master and one slave — with a load balancer in front of the two.

In a high-availability setup, it is important to keep the configuration of the master and slave instances synchronized. With the new REST API, you can utilize a simple bash script to download the configuration from the master (as seen in the previous example) and upload and restore it on the slave.

For uploading a configuration file to an instance, use the **uploadApplianceConfiguration** API call, and for restoring the configuration after it has been uploaded, use the **restoreApplianceConfiguration** API call.

The following example shows how you can use bash script to automate your setup:

```
#!/usr/bin/env bash
```

```
API_KEY="X-Api-Key: my_api_key "
URL="https://my_ip/me/"
UPLOAD_URL="${URL}uploadApplianceConfiguration"
RESTORE_URL="${URL}restoreApplianceConfiguration"
local_cfg="Palladion-configuration-10042014-1.cfg"

curl -k -s -X POST -H"${API_KEY}" -F configuration=@"${local_cfg}" ${UPLOAD_URL}
curl -k -s -X POST -H"${API_KEY}" -d"configuration=${local_cfg}" ${RESTORE_URL}
```

Examples

List the last 20 calls:

curl -k -X POST -H "X-API-Key: *API_Key*" **-d "limit=20"**
"https://*IP*:*Port*/me/getPagedCalls"**

where,

- *API_Key*, is your personal key code. For more information on how to generate this key, see "API Key".
- *IP*, is the IP address of Oracle Communications Session Monitor Mediation Engine.
- *Port*, is the port number of the SSL web interface of Mediation Engine. The *:Port* section can be omitted if the default HTTPS port (443) is used.

Get the next 20 calls (after executing the previous example):

curl -k -X POST -H "X-API-Key: *API_Key*" **-d "limit=20&older_than=***next_value*"
"https://*IP*:*Port*/me/getPagedCalls"**

Where *next_value* is the returned content of the **next_value** field from the previous example.

List all failed calls starting from last week, from a specific caller:

```
curl -k -X POST
-H "X-API-Key:API_Key"
-d 'filters=[{"field":"src_user","data":{"type":"string","value":"src_user"}},
{"field":"setup_start_ts","data":{"type":"datetime","comparison":"gt",
"value":"start_date"}},
{"field":"state_msg","data":{"type":"list","value":["Failed"]}}]'
"https://IP:Port/me/getPagedCalls"
```

Where,

- *src_user*, is the ID of the user whose failed calls your retrieving.
- *start_date*, specifies the starting date to retrieve failed calls. The format is *YYYY-MM-DD hh:mm:ss*, for example 2015-04-13 00:00:00.

Reference

This section lists the most common API calls. However if you want to access a feature of Operations Monitor through the new REST API that is not listed here, see "Reverse Engineering of API Calls".

Note: All parameter names and options (given in round brackets, separated by vertical lines) are case sensitive.

Important: All parameters are mandatory with all REST API calls unless marked as **optional**. This is not enforced by the server. For many parameters fall back values are defined. You are strongly advised not to rely on these fall back values as any software updates could break your code in the case that fall back values have been changed.

Fundamental Concepts

Please note that there are some basic principles to the new REST API that you should know in order to have a better understanding of the API:

- All REST API calls are made using HTTPS. Any requests using plain HTTP will result in a HTTP/1.1 301 Moved Permanently.
- All parameters of calls are transmitted as HTTP header fields.
- Responses are JSON Objects.
- If the API call succeeded the response will be a **200 Ok** HTTP message and the **success** field of the response will have the value **true**.
- If a call doesn't succeed, the response will be a **200 Ok** HTTP message too. The **success** field of the response however will be **false** and there will be a field **errorMsg** containing a brief description of what went wrong.

Note: In some cases instead of a **200 Ok** HTTP message, a **404** type message will be returned. In this rare event instead of a JSON object a HTML string, containing an error page, will be returned. So please be sure to also check for **404**, and **301** HTTP responses (see above) when checking for errors!

- Often calls have a method for limiting the result set. This is done just as in the database world where you can specify from which element you want to start and how many elements you want to have returned at most.
- The filtering concept that is used in **getPagedCalls** is also available for any other data that is being presented as a sortable grid with optional filters in Operations Monitor. For more information on how to design filters and reverse engineer the API for any other data accessible through Operations Monitor, see "Examples" and "Reverse Engineering of API Calls".

In order to get a better understanding of all these concepts please feel free to play around with the little snippets presented in the "Examples" section.

Configuration Savepoints

Table 11–1 describes the **/me/saveApplianceConfiguration** savepoint object.

Table 11–1 /me/saveApplianceConfiguration

Description	Method	Parameters	Returns
Creates a savepoint on the server and returns the name of the savepoint.	POST	None.	*name* Name of the save point.

Table 11–2 describes the **/me/uploadApplianceConfiguration** savepoint configuration.

Table 11–2 /me/uploadApplianceConfiguration

Description	Method	Parameters	Returns
Uploads a savepoint to the Server and returns the name of the savepoint.	POST	*configuration* Local file name. *name* "configuration".	*name* Name of the save point on the server.

Table 11–3 describes the **/me/downloadApplianceConfiguration** savepoint object.

Table 11–3 /me/downloadApplianceConfiguration

Description	Method	Parameters	Returns
Downloads a previously saved or uploaded configuration savepoint.	POST	*name* Name of the savepoint to be downloaded.	The requested file. File name is included in the content-disposition Response Header field.

Table 11–4 describes the **/me/restoreApplianceConfiguration** savepoint object.

Table 11–4 /me/restoreApplianceConfiguration

Description	Method	Parameters	Returns
Restores a previously saved to uploaded savepoint.	POST	*configuration* Name of the savepoint to be restored.	*warnings* Warnings that occurred during the restore process, separated by newline characters.

Table 11–5 describes the **/me/deleteApplianceConfiguration** savepoint object.

Table 11–5 /me/deleteApplianceConfiguration

Description	Method	Parameters	Returns
Deletes a savepoint.	POST	*name* Name of the savepoint to be deleted.	None.

Table 11–6 describes the **/me/listApplianceConfigurations** savepoint object.

Table 11–6 /me/listApplianceConfigurations

Description	Method	Parameters	Returns
Gets list of save points.	POST	*start* Starting offset for the results to be returned. *limit* Limits the number of results to be returned. *sort* Column by which to sort the results (“date” | “name”). *dir* Whether to sort results ascending or descending (“ASC” | “DESC”).	*configurations* Array with informations about the individual savepoints. *totalCount* Number of savepoints in total (not the number of returned savepoints).

Calls

Table 11–7 describes the **/me/getPagedCalls** call object.

Table 11–7 /me/getPagedCalls

Description	Method	Parameters	Returns
Gets a list of calls around a given point in time or around a given call.	POST	*filters* See the filter design template and the tables below to design your own filters for the result set. If you do not want to filter the result set, please pass an empty list instead ([]). *older_than* Pass an ID of an element of the result set here to retrieve elements older than the given element. *older_than_ts* Same as the above, only here you pass the timestamp of an object, not its ID. *newer_than*, *newer_than_ts* Same as the above, just into the opposite direction. *limit* Limits the amount of results to be returned.	*calls* Array with calls objects. See the below table for details. *first_page* Whether or not the returned result set is the first page of results available (true | false). *last_page* Whether or not the returned result set is the last page of results available (true | false). *next_value* When you want to retrieve the next page of results using the same parameters, use *next_value* for the *older_than* field in the request header.

Filter Design Template

```
{"field":"<field>","data":{"type":"<type>","comparison":"<comparison>",
"value":"<value>"}}
```

Where:

- *<field>* is the name of the field that the filter applies to.

- <*type*> is the type of the filter. Depending on the type of the field there are different filters available. See Table 11–10 for details.
- <*comparison*> is the comparator to be used for filtering. See Table 11–10 for details.
- <*value*> is the value to compare against in the filter. See Table 11–10 for details.

Creating a Filter for a Specific Column

To create a filter for a specific column:

1. Locate the <*field*> value for the desired column (see the **Field (<*field*>)** column in Table 11–8).
2. Locate the type of the column (see the **Type** column in Table 11–8).
3. Refer to Table 11–10 to see which filters are available and how to use them.
4. Refer to "Examples" and "Reverse Engineering of API Calls" for in depth examples on how to use **getPagedCalls** and how to watch network traffic to get a better understanding of the API.

Table 11–8 lists the columns and fields in the calls objects that can be filtered.

Note: Table 11–9 lists fields, which are also part of the calls objects that cannot be filtered.

Table 11–8 Fields and Types (in alphabetical order)

Field (<*field*>)	Type	Column in the Calls Table
call_time	Numeric	Call time
code	Numeric	Code
dpc	String	DPC
dst_codecs	String	Callee Codecs
dst_initial_codecs	String	Callee Initial Codecs
dst_ip	IP address	Callee IP address. For the data field of the filter, set type to **ipaddr** and value to the IP address string (e.g. 192.168.1.123 or 10.0.0.0/8).
dst_ua	String	Callee User Agent
dst_user	String	Callee
dst_user_pref_tag	Boolean	Preferred Callee Number? For the data field of the filter: ■ To filter to **True**, set type to **string** and comparison to **notnull**. ■ To filter to **False**, set type to **string** and comparison to **null**.
dtmf	Boolean	DTMF
egress_devs	List	Egress device(s)
end_ts	Datetime	End timestamp
gateway_devs	List	Gateway Devices
in_devs	List	In devices
ingress_devs	List	Ingress device(s)

Table 11–8 (Cont.) Fields and Types (in alphabetical order)

Field (<*field*>)	Type	Column in the Calls Table
init_devs	List	Initiator device
megaco_cmds	String	MEGACO Commands
megaco_context_id	String	MEGACO Context ID
megaco_mg_ip	String	MEGACO Gateway
megaco_mgc_ip	String	MEGACO Gateway Controller
megaco_termination_id	String	MEGACO Termination ID
megaco_txids	String	MEGACO Transaction IDs
mgcp_call_ids	String	MGCP Call IDs
mgcp_capabilities	String	MGCP Capabilities
mgcp_connection_ids	String	MGCP Connection IDs
mgcp_mg_ip	String	MGCP Gateway IP
MOSlqe_avg	Numeric	Avg. MOS
MOSlqe_min	Numeric	Min. MOS
nlegs	Numeric	Segments
opc	String	OPC
out_devs	List	Out devices
pai	String	P-Asserted-ID
prefix_group	List	Prefix Group
q850_code	Numeric	Q.850 Code
q850_state_details	String	Q.850 Details
q850_state_msg	String	Q.850 State
realms_recordings	Boolean	Media For the data field of the filter, set type to **custom:streams_availability** and value to **false**, or **true** depending on which you want to filter for.
reason_hdr		Reason
rpid	String	Remote-Party-ID
rtcp_delay_avg	Numeric	Avg RTCP delay
rtcp_delay_max	Numeric	Max RTCP delay
rtcp_streams	Numeric	RTCP streams
setup_delay	Numeric	Setup Delay
setup_delay_type	List	Setup Delay Type With this column you have two options which you can filter for. You may filter for either one, or both options: ■ To filter for **Successfull Session Request Delay** include **1** (as a String) in the list you pass for <*value*>. ■ To filter for **Failed Session Request Delay** include the String **2** in the list you pass for <*value*>.
setup_start_ts	Datetime	Start timestamp
setup_time	Numeric	Setup time

Table 11–8 (Cont.) Fields and Types (in alphabetical order)

Field (<*field*>)	Type	Column in the Calls Table
src_codecs	String	Caller Codecs
src_initial_codecs	String	Caller Initial Codecs
src_ip	IP address	Caller IP address. For the data field of the filter, set type to **ipaddr** and value to the IP address string (e.g. 192.168.1.123 or 10.0.0.0/8).
src_ua	String	Caller User Agent
src_user	String	Caller
src_user_pref_tag	Boolean	Preferred Caller Number? For the data field of the filter: ■ To filter to **True**, set type to **string** and comparison to **notnull**. ■ To filter to **False**, set type to **string** and comparison to **null**.
state_details	String	State details
state_msg	List	State Options are as follows: ■ Blocked ■ Busy ■ Canceled ■ Error ■ Established ■ Failed ■ Finished ■ Not Found ■ Off-line ■ On hold ■ Proceeding ■ Redirected ■ Reset ■ Ringing ■ Setup ■ Terminate ■ Timed out ■ Unauthorized ■ Unequipped You may filter for one or more of the above options. Just include the desired value(s) from the above list as Strings in the array you pass for <*value*>.
term_devs	List	Terminator device

Table 11–9 lists fields that are not in the above table (cannot be filtered) but are part of the calls objects:

Table 11–9 Call Object Fields that Cannot be Filtered

Field	Description
pid	Process ID.
id	ID of the call. A call is uniquely identified through its **pid** and **id**.
id32	Internal only.
strictly_increasing	ID which can be used for sorting call events chronologically.
state	Numeric representation of state.
state_cls	Color of state.
end_ts	Timestamp of when the call was finished.
mos_cls	Color of the mos column.
overload	Whether during the processing of the call a system overload occurred.
number_prefix	Contains a comma separated list for all the number prefixes, that the respective call is matching.

Table 11–10 lists all available filter types. Please note that the table has been grouped by column data type (see above table) to show which filter options are available for the corresponding field type.

Table 11–10 Filter Types (grouped by column data types)

Type (<type>)	Comparators (<comparison>)	Description
String		
exactstring	None.	Matches exact string.
string	None.	Match string.
DateTime		
date	lt	less than (before given date) (<).
	gt	greater than (after given date) (>).
	eq	equal (on the given date) (=).
	neq	not equal (not on the given date) (!=).
	leq	less or equal (on or before the given date (<=).
	geq	greater or equal (on or after the given date) (>=).
datetime	(lt | gt | eq | neq | leq | geq)	Equals only compares dates.
datetime_utc	(lt | gt | eq | neq | leq | geq)	Equals only compares dates. Use this version if <value> is a UTC timestamp.
Numeric		
numeric	(lt | gt | eq | neq | leq | geq)	Does the job of any first grader only slightly faster.
List		
in_list	None.	Matches one value of the array given in <value>.
list	None.	Matches at least one element of the array given in <value>.
not_in_list	None.	Does not match a value of the array given in <value>.

Table 11–11 describes the **/me/getCallDetails** call object.

Table 11–11 /me/getCallDetails

Description	Method	Parameters	Returns
After you called *getPagedCalls* to retrieve a list of calls you can use *getCallDetails* to obtain further information about a specific call.	POST	*id* ID of call as returned in the **id** field of a call. *pid* As returned in the **pid** field of a call. *saved_id* (Optional) Instead of passing the previous two IDs you can just pass this ID in case you saved this call. *no_legs* (true | false).	An array with essentially the same fields as *getPagedCalls* but also a few additional fields: ■ *legs* - Array with the legs of the call. ■ *megaco_id* - Megaco ID. ■ *merged_into_call_id* ■ *total_legs* - Total amount of legs. Internal only fields: ■ *mgcp_id* ■ mgcp_mgc_ip ■ *mgcp_txids* ■ mgcp_verbs ■ traversing_devs

Table 11–12 describes the **/me/getCallMsgs** call object.

Table 11–12 /me/getCallMsgs

Description	Method	Parameters	Returns
Returns the SIP messages associated with a given call.	POST	*id* ID of call as returned in the **id** field of a call. *pid* As returned in the **pid** field of a call. *saved_id* (Optional) Instead of passing the previous two IDs you can just pass this ID in case you saved this call.	*proto* Protocol. *src_ip* Source IP. *src_mac* Source Mac. *opc* Source PC. *dst_ip* Destination IP. *dst_mac* Destination MAC. *dpc* Destination PC. *ts* Date and Time. *method* Message. ruri Details

Table 11–13 describes the **/me/getCallsTime** call object.

Table 11–13 /me/getCallsTime

Description	Method	Parameters	Returns
Returns the current time stamp (for use in for example *getPagedCalls* as value for the *older_than_ts* parameter) and the time stamp of the latest call.	POST	None.	*ts_first* Timestamp of the first call in the database. *ts_now* Current timestamp - can be used as a value for *older_than_ts*. For example in *getPagedCalls*.

Table 11–14 describes the **/me/getAvailablePcaps** call object.

Table 11–14 /me/getAvailablePcaps

Description	Method	Parameters	Returns
Get the available pcaps and the streams available in the respective pcap.	POST	*id* ID of call as returned in the **id** field of a call. *pid* As returned in the **pid** field of a call. *saved_id* (Optional) Instead of passing the previous two IDs you can just pass this ID in case you saved this call	*available* An object which contains all available recorded streams for the specified user grouped by stream type.

Table 11–15 describes the **/me/callPcap** call object.

Table 11–15 /me/callPcap

Description	Method	Parameters	Returns

Download a pcap file for a specific call.	POST	*id*
		ID of call as returned in the **id** field of a call.
		pid
		As returned in the **pid** field of a call.
		saved_id
		(Optional) Instead of passing the previous two IDs you can just pass this ID in case you saved this call.
		filename
		If this is not empty it is going to be the filename of the pcap file that is retrieved from the server.
		streams
		Array which includes which stream types you would like to retrieve. Include the string "EN10MB" in the array to include signaling messages, and for every stream for which you want to retrieve the RTP payload include a string of the format:
		`"<Media type>#<src_ip>:<src_port>#<dst_ip>:<dst_port>#<media_content_type>"`
		Example:
		`"MEDIA#62.220.31.225:19848#62.220.31.239:27998#rtp_full"`
		If the above fields are not familiar please make sure to experiment with *getAvailablePcaps* and play around with the RTP recording feature and the save PCAP feature in the regular Operations Monitor web interface.
			The requested pcap file.

Registrations

Table 11–16 describes the **/me/getRegsPaged** registrations object.

Table 11–16 /me/getRegsPaged

Description	Method	Parameters	Returns
Retrieves a list of registrations around a point in time or around another registration.	POST	*None* *older_than* Pass an ID of an element of the result set here to retrieve elements older than the given element. *older_than_ts* Same as the above, only here you pass the timestamp of an object- not its ID. *newer_than*, *newer_than_ts* Same as the above, just into the opposite direction.	*data* Array containing all registrations of the current result set. Fields contained in data: ■ *dev_id* – Registrar. ■ *code* – Code. ■ *contacts* – Contacts. ■ *ip* – Source IP address. ■ *ts* – Timestamp. ■ *type_msg* – Event. ■ *user* – User. ■ *dest_ip* – Destination IP address. *first_page* Whether or not the returned result set is the first page of results available (true | false). *last_page* Whether or not the returned result set is the last page of results available (true | false). *next_value* When you want to retrieve the next page of results using the same parameters, use *next_value* for the *older_than* field in the request header.

Table 11–17 describes the **/me/getRegsTime** registrations object.

Table 11–17 /me/getRegsTime

Description	Method	Parameters	Returns
Returns the current time stamp (for use in for example *getPagedRegs* as value for the *older_than_ts* parameter) and the timestamp of the latest registration.	POST	None.	*ts_first* Timestamp of the first registration result set using the passed filters. *ts_now* Current timestamp- can be used as a value for *older_than_ts* for example in *getPagedCalls*.

Table 11–18 describes the **/me/getRegEvent** registrations object.

Table 11–18 /me/getRegEvent

Description	Method	Parameters	Returns
Gets details about a registration event.	POST	*id* ID of the event for which you want to retrieve the SIP messages.	*reg* Array containing the SIP messages for the specified event.

Table 11–19 describes the **/me/getRegEvMsgs** registrations object.

Table 11–19 /me/getRegEvMsgs

Description	Method	Parameters	Returns
Gets the SIP messages for a specified event.	POST	*id* ID of the event for which you want to retrieve the SIP messages. *start* Starting offset for the results to be returned. *limit* Limits the amount of results to be returned.	*totalCount* Total count of the result set - not the amount of elements returned. *data* Array with the requested SIP messages.

KPI

Table 11–20 describes the **/me/getCounterValues** KPI object.

Table 11–20 /me/getCounterValues

Description	Method	Parameters	Returns
Returns counters marked as favorites and their recent values.	POST	None.	*counters* An array containing the requested counters.

Table 11–21 describes the **/me/addOrModifyCounter** KPI object.

Table 11–21 /me/addOrModifyCounter

Description	Method	Parameters	Returns
Add or modify a counter.	POST	*action* Whether to edit or add a counter ("add" | "edit"). *snmp_export* (Optional) Set this to "on" to enable snmp export, or do not use this parameter to disable snmp export. *maintype* Numeric representation of the counter. Use *getCounterTypes* to find out the main types of counters. *subtype* Numeric representation of the counter. Use *getCounterTypes* to find out the subtypes of counters. If you want to create an average counter include the three properties below: ■ *weekdays* (Optional) Set weekdays to 1 if you want the average to be taken only over weekdays otherwise set it to 0. ■ *days* (Optional) The average will be taken over the amount of days specified here. ■ *name* (Optiona) The name for the average counter.	*id* ID of the added or changed counter.

Table 11–22 describes the **/me/deleteCounter** KPI object.

Table 11–22 /me/deleteCounter

Description	Method	Parameters	Returns
Removes a counter.	POST	*id* ID of the counter to be removed.	None.

Table 11–23 describes the **/me/toggleSNMPExport** KPI object.

Table 11–23 /me/toggleSNMPExport

Description	Method	Parameters	Returns
Enables or disables the SNMP export of a counter.	POST	*ids* List of counter IDs. *action* ('enable' | 'disable').	None.

Table 11–24 describes the **/me/getCounterRealm** KPI object.

Table 11–24 /me/getCounterRealm

Description	Method	Parameters	Returns
Get a counter's realm.	POST	*id* ID of the counter.	*realm* Realm of the given counter.

Table 11–25 describes the **/me/getCounterConfig** KPI object.

Table 11–25 /me/getCounterConfig

Description	Method	Parameters	Returns
Retrieves the configuration of a user counter. You can modify the returned configuration and pass it to *addOrModifyCounter* to update/create a counter with the modified configuration.	POST	*id* ID of the user counter to be retrieved.	*data* An object with the configuration fields of the desired counter: ■ *id* – ID of the counter (numeric). ■ *name* – Name of the counter. ■ *service_id* – Internal only. ■ *maintype* – Numeric main type of the counter. ■ *subtype* – Numeric sub type of the counter. ■ *snmp_export* – Boolean if snmp export is enabled. ■ *src_cnt* – For average counters, the source counter for the average. ■ *weekdays* – For average counters, whether the average is calculated for each weekday separately. ■ *days* – For average counters, whether the average is calculated for each weekday separately. ■ *param1* – First counter parameter. ■ *param2* – Second counter parameter. ■ *param3* – Third counter parameter.

Table 11–26 describes the **/me/getSimpleCounters** KPI object.

Table 11–26 /me/getSimpleCounters

Description	Method	Parameters	Returns
Gets counters which are no average counters.	POST	*device_id* Device ID of the counter. *tag_id* Tag ID of the counter.	*counters* Object containing the following fields: ■ *id* – ID of the counter (numeric). ■ *name* – Name of the counter. ■ *subtype* – Numeric sub type of the counter. ■ *owner* – Owner of the counter. ■ *params* – List with the first, second and third parameter of the filter

Table 11–27 describes the **/me/getCounters** KPI object.

Table 11–27 /me/getCounters

Description	Method	Parameters	Returns
Gets all types of counters- even if they are average counters.	POST	*device_id* Device ID of the counter. *tag_id* Tag ID of the counter.	*counters* Object containing the following fields: ■ *id* – ID of the counter (numeric). ■ *maintype* – Main type of counter. ■ *name* – Name of the counter. ■ *subtype* – Numeric sub type of the counter. ■ *owner* – Owner of the counter. ■ *params* – List with the first, second and third parameter of the filter.

Table 11–28 describes the **/me/counters** KPI object.

Table 11–28 /me/counters

Description	Method	Parameters	Returns
Get all visible counter IDs.	POST	*alias* (true | false) Whether or not to include the user's aliases in the search for counters.	*counters* List with IDs of visible counters.

Table 11–29 describes the **/me/getCounterTotal** KPI object.

Table 11–29 /me/getCounterTotal

Description	Method	Parameters	Returns
Gets the current and maximum allowed amount of counters.	POST	None.	*total* Total count of current counters. *max_counters* Maximum possible amount of counters. *counters_lock* Whether there is currently an operation on the counters active.

Table 11–30 describes the **/me/getCounterTypes** KPI object.

Table 11–30 /me/getCounterTypes

Description	Method	Parameters	Returns
Gets the counter types with names and parameters.	POST	All parameters are optional: ■ *device_id* Device ID of counter. ■ *device_list* List of devices. ■ *iptag_list* List of IP tags. ■ *iptag_id* IP tag ID. ■ *tag_id* Tag ID. ■ *only_active* (true | false) Whether to return only active counters. ■ *remove_create_new* Description.	List with counters, fitting the given parameters.

Table 11–31 describes the **/me/MonitorGraph** KPI object.

Table 11–31 /me/MonitorGraph

Description	Method	Parameters	Returns
Draws and returns a counter graph.	POST	All parameters are optional: ■ *period* int - period in seconds. ■ *utcdate* date in format '2012-10-31T20:16:52Z'. ■ *format* Call returns a SVG image, if parameter is "svg"- else returns a PNG image (or JSON- see *raw_data*). ■ *raw_data* (true | false) If True, return json, no image. ■ *selected* A list of IDs to determine which metrics to render. ■ *tzoffset* Offset for the time zone.	*data* JSON object with the raw data in case *raw_data* was set to true. Else the requested image file.

Reverse Engineering of API Calls

As stated earlier, the above documentation only covers the most important features of the new REST API. With the new API Key however you have the same access to the features of Operations Monitor as with the regular web interface. So in the following section we will be using the Google Chrome browser's development tools to reverse engineer how to use API calls that are not documented here. You can also follow this guide using the Mozilla Firefox browser in conjunction with the Firebug extension. It is a fairly straightforward process though - so if you're impatient go ahead and use the web interface of Operations Monitor and watch the network activity while you're at it to figure out how to use parts of the API not covered here. For an in-depth guide how to achieve this goal please continue reading.

Note: When watching the network activity of Operations Monitor you will often see URLs that have a weird suffix.

For example:

```
getSystemSettings/?_dc=1398269727192
```

The suffix is pretty much a timestamp appended by our frontend framework. This is done to avoid any possible caching of HTTP requests by altering the URL with the timestamp. This is done to ensure that any results are always directly from the server and not any outdated, cached results.

This scenario is very unlikely however and is merely a workaround to bad network configuration. You can go ahead and ignore this parameter when analyzing backend calls and leave the parameter out when using backend calls from a custom script.

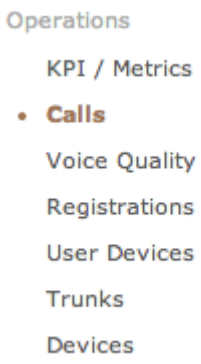
As you might recall Operations Monitor has a feature for saving calls (you might have noticed that in many Calls related API calls you can pass a **saved_id** for identifying a call instead of the **id** and the **pid**). If we were to use that functionality through a script an interesting use case might be to save certain calls in the course of an operation, in order to have a human user review these later.

So let's find out, how to save calls using the new REST API. In order to walk you through the process you need to have a good understanding of how the new REST API works. If you think you might need a refresher on that please head over to the **API Key** and the **Getting Started** sections and have a quick look at our use cases **Usecase Backup** and **Usecase High Availability**.

To get started, please retrieve your API Key, and have your tools ready for action.

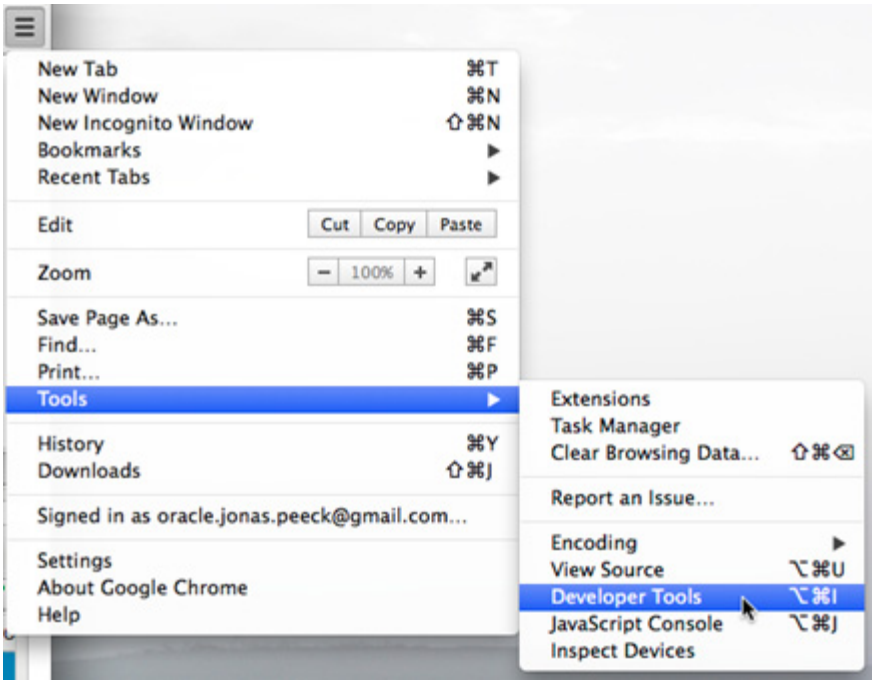
Next go to the **Calls** section in the web interface of Operations Monitor, which is shown in Figure 11–2:

Figure 11–2 The Calls Section in the Main Menu



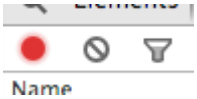
Go ahead and open the developer tools so that we know what's going on in the background. Click the Customize and control Google Chrome icon to access **Developer Tools** as shown in Figure 11–3:

Figure 11–3 Accessing the Google Chrome Developer Tools



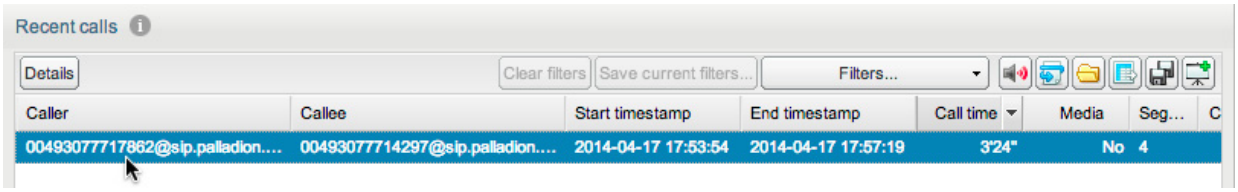
Please make sure that you have the **Network** tab open and that recording is enabled, as shown in Figure 11–4 , so that we can observe the network activities that are happening in the background while using the web interface of Operations Monitor.

Figure 11–4 Enable Recording



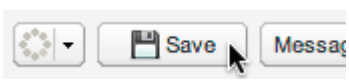
In the **Recent calls** section of the web interface of Operations Monitor, double click on any call to open up the call details dialog, as shown in Figure 11–5.

Figure 11–5 Double Click on a Call to See Details



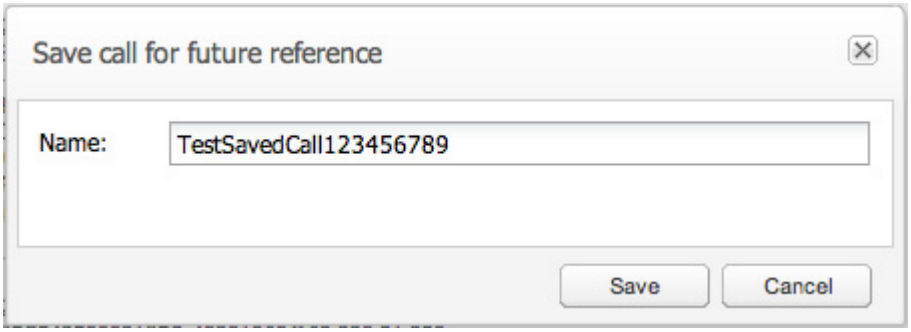
On the bottom of the now open dialog you will find a set of buttons, including a **Save** button as shown in Figure 11–6. Please place your mouse cursor above that button and use it by firmly pressing your left mouse button.

Figure 11–6 Save Button



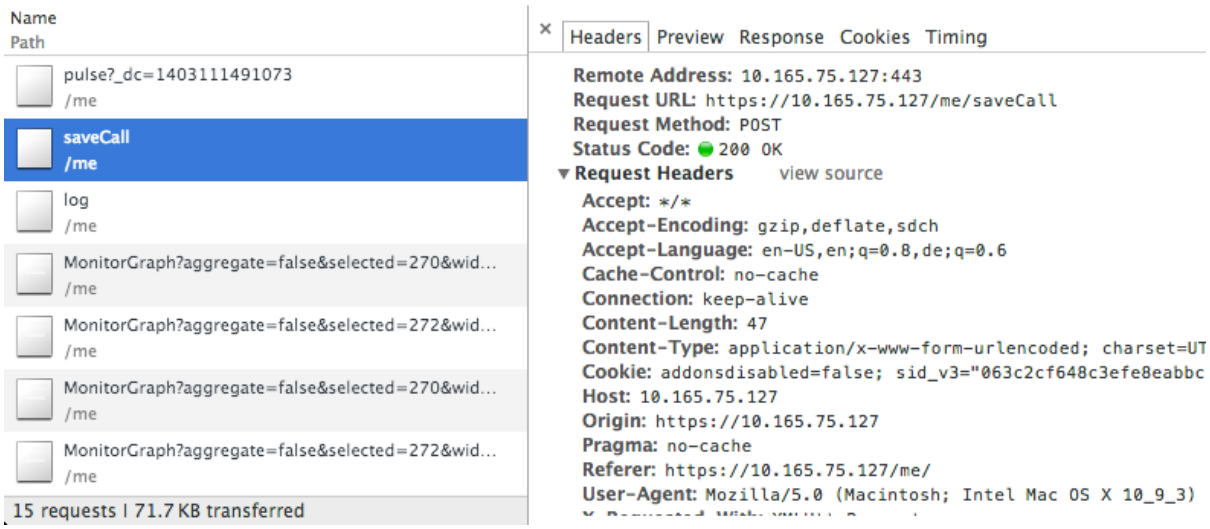
Now it's time to name the to be saved call. Please use the name **TestSavedCall123456789** or a name that will be easily recognizable as your custom name as shown in Figure 11–7.

Figure 11–7 Adding a Custom Name



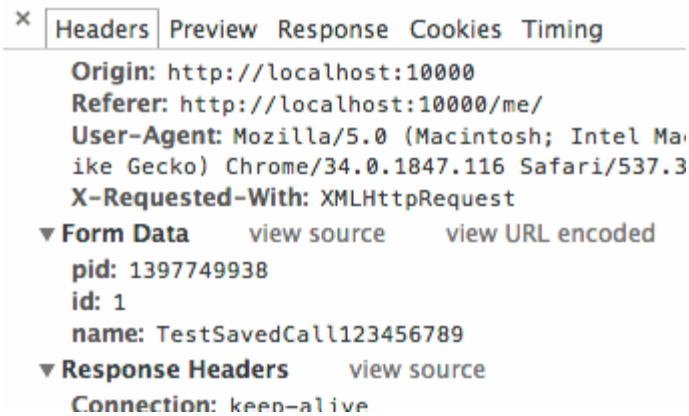
Next, if you click **Save**, you should see a **saveCall** https call popup in the network monitoring section of the developer tools as shown in Figure 11–8. You may go ahead and stop the recording of network activity at this point as we have all the info we need to find out how to use this call in the future.

Figure 11–8 saveCall https Call Popup



Scroll down to the section **Form Data** as shown in Figure 11–9.

Figure 11–9 The Form Data Section of the saveCall Request



Here we can easily read out the parameters for the **saveCall** request:

- *pid*

  The Process ID of the call we opened and saved.

- *id*

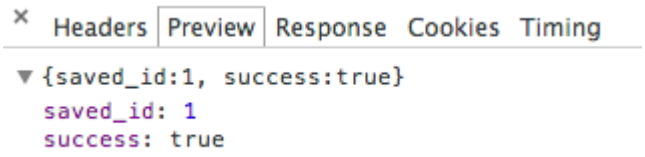
  The ID of the call we saved.

- *name*

  The name for the saved call as previously entered.

If you now switch to **Preview** tab of the request we can discover the returns of this call as shown in Figure 11–10:

Figure 11–10 Preview Tab



- *saved_id*

  The **saved_id** of the call we just saved. In the future we may use this to identify this call instead of using the **id** and **pid** parameter.

- *success*

  This field is always part of the return if no error occurred.

And with that you now know how to save calls using the new REST API. You can use the same approach for any other functionality in the web interface of Operations Monitor that you would like to automate.

All you have to do is to play around with the web interface a bit and see how requests behave differently based on changes you perform.

12

Client-Side Add-Ons

Apps provide an easy way to customise and extend Oracle Communications Operations Monitor with capabilities that are not currently present in the user interface using Javascript and CSS.

You can enable or disable add-ons at login time using the **Disable add-ons** check box from the login form. The check box label shows the number of enabled extensions that will be loaded (unless the check box is selected), and the tooltip shows the name and the description of each add-on.

Structure of an Add-On

A valid Operations Monitor add-on consists of a **manifest.json** file and a set of files that will be exposed to the client. The **manifest.json** file must be a valid JSON file and include the fields found in Table 12–1:

Table 12–1 manifest.json File Required Fields

Field	Description
name	Name of the extension. This name will be used as the extension files path in the server script.
description	Describes the functions of the add-on.
api_version	The current supported API version is "0.1.0".
css	(Optional) List of CSS stylesheets that are included.
js	(Optional) List of Javascript files included.
images	(Optional) List of image files included.

Example:

```
{
  "name": "customextension',
  "description": "A simple example extension.\n Additional comments can be added
here",
  "api_version": "0.1.0",
  "css": ["css1.css", "custom_css.css"],
  "js": ["custom.js", "extension.js"],
  "images": ["logo.png", "image.jpeg", "icon.ico"],
}
```

Note: The CSS and JS files will be loaded in the same order as declared in the **manifest.json** file. They will be loaded after all Operations Monitor libraries. The image files order is not meaningful as they are not pre-loaded.

Packaging

The add-on files must be packed in ZIP format and its filename must use the **.zip** extension. The resulting ZIP file can contain a root folder. The name of that folder is not meaningful. Such a ZIP archive can be uploaded to the Platform Setup Application using the provided form in the Add-ons page.

Here is a sample directory structure for the ZIP archive:

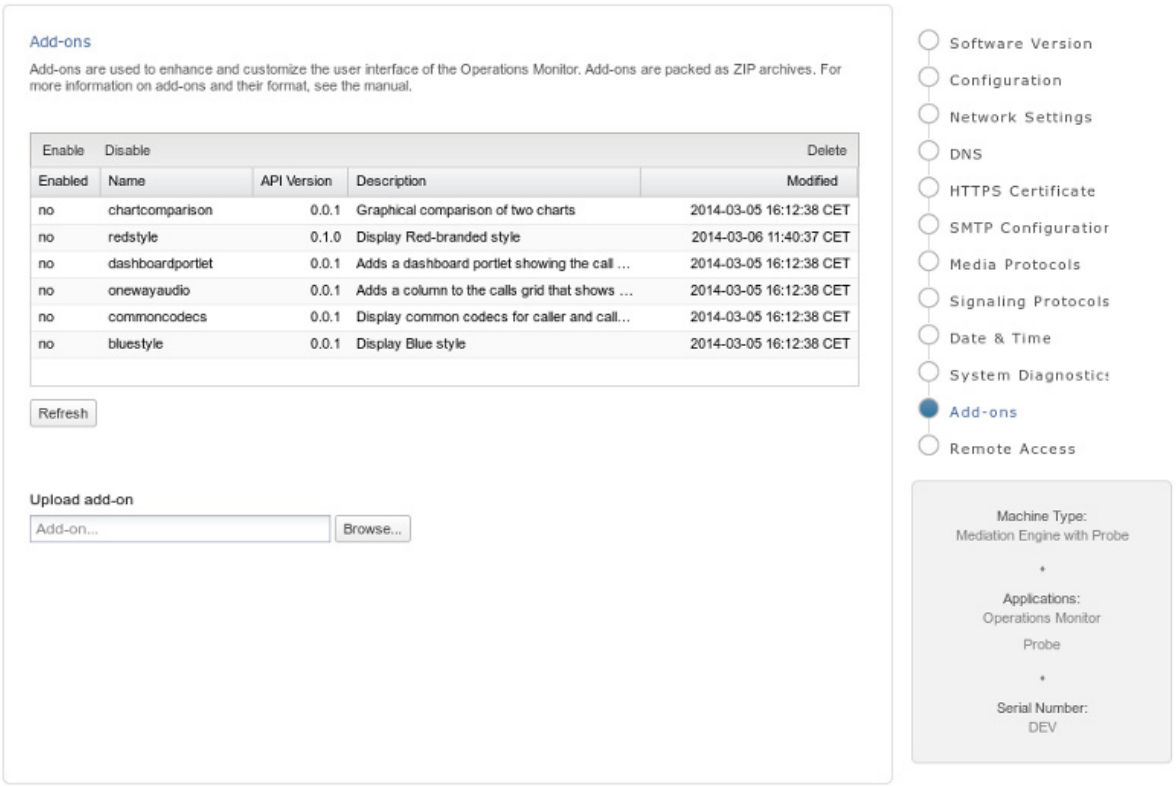
```
> tree dashboardportlet/
dashboardportlet/
   css/
      dashboardportlet.css
   js/
      dashboardportlet.js
   manifest.json

2 directories, 3 files
```

Management

In the Add-ons page of the Platform Setup Application, a grid is provided where the system administrators can enable, disable, and sort the extensions load order. Each extension operation in this page will be applied for all ME active instances immediately on the next page reload. Figure 12–1 shows the Add-ons page.

Figure 12–1 Add-ons Page



Code Examples

CSS file:

```
div.header {
    background-color: #EC0000 !important;
    background-repeat: repeat-x, repeat;
}

span#oracleProductName {
    background-image: url('/me/addons/redstyle/images/logo.png');
    height: 46px;
    background-repeat: no-repeat;
    background-position: right 20px top 10px;
    background-size: 45px 45px;
}
```

Javascript file:

```
Ext.onReady(function () {

  var log = function() {
    console.log("Page change")
  };

  Pld.Application.on("pagechange", log());
});
```

13

Diameter Transaction Records (TDRs)

Oracle Communications Operations Monitor creates Diameter Transaction Records (TDR files) in CSV format. The files are placed in the **tdr/** directory under the FTP/FTPS root. The TDR files have the following format:

`tdr-`*`unix_timestamp-sequence`*`.csv`

where:

- *unix_timestamp* is the Unix timestamp when the file was created.
- *sequence* is a number monotonically increasing.

The files are rotated when they reach their maximum size (10000 records). When a CSV file is finished, another empty file having the following format is created:

`tdr-`*`unix_timestamp-sequence`*`.csv.FIN`

Several times per day, old TDR files are compressed to files having the following format, while the corresponding uncompressed files are deleted:

`tdr-`*`unix_timestamp-sequence`*`.csv.gz`

The recommended way of gathering the TDR files is to connect via FTP/FTPS, copy and delete all the files ending in **csv.gz**. Alternatively, to get the TDR data in near real time, the MDR files that have a corresponding **.FIN** file can be copied and deleted.

Operations Monitor automatically limits the size of the **tdr/** directory to 5 GB, by deleting the oldest files.

Table 13–1 lists the fields present in the generated CSV files:

Table 13–1 TDR CSV Fields

Field	Description
application-id	The IANA assigned Application Identifier of the Diameter transaction. The possible values as defined in RFC 5516 are outlined below: ■ *0:* Diameter Common Messages ■ *1:* Nasreq ■ *2:* Diameter mobile ip ■ *3*: Diameter Base Accounting ■ *0xffffffff*: Diameter Relay ■ *16777216*: 3GPP Cx application ■ *16777251*: 3GPP S6a/S6d application ■ *16777252*: 3GPP S13/S13′ application ■ *16777272*: 3GPP S6b application ■ *16777310*: 3GPP S6m application
command	A human readable string describing the Diameter command-code. Can be one of the following: ■ Authentication-Information ■ Cancel-Location ■ Delete-Subscriber-Data ■ Insert-Subscriber-Data ■ Location-Info ■ ME-Identity-Check ■ Multimedia-Auth ■ Notify ■ Purge-UE ■ Push-Profile ■ Registration-Termination ■ Reset ■ Server-Assignment ■ Update-Location ■ User-Authorization
command-code	The Diameter command code of the transaction. For example for the command UPDATE_LOCATION the code would be 316. All the possible codes are defined in 3GPPTS 29.272 at the interface commands for S6a/S6d and S13.
destination-realm	Contains the realm the messages of a Diameter transaction should be routed to.
dst-dev-id	Identifier of the destination Diameter device.
dst-ip	Destination IP address of the Diameter transaction.
duration	The duration of the transaction in seconds (with a precision of microseconds), from the time the request is received to the time the transaction completes.
IMEI	IMEI value of the user.
internal-id	Unique internal identifier used by Operations Monitor to identify Diameter transactions.

Table 13–1 (Cont.) TDR CSV Fields

Field	Description
msisdn	The **msisdn** number of the Diameter transaction identifying the subscription to a mobile network.
network-access-mode	Describes the networks that the user is registered to. Values range from 0 to 2.
network-access-mode-text	Describes the networks that the user is registered to. Possible values can be one of the following: ■ PACKET_AND_CIRCUIT ■ Reserved ■ ONLY_PACKET
origin-realm	Contains the realm of the originator of the Diameter messages of this transaction.
rat-type	A numeric code used to identify radio access technology that is serving the UE. Defined in TS 29212 at section 5.3.31.
realm-ids	Diameter transaction realm ID - not currently exported.
result	Human readable string of the result code.
result-code	The numerical value of the Result-Code of the diameter transaction. For example 2001 for DIAMETER_SUCCESS. For all possible values refer to the standards.
session-id	A string with the Session-ID AVP of the Diameter transaction.
src-dev-id	Identifier of the source Diameter device.
src-ip	Source IP address of the Diameter transaction.
start_ts	Time value for the transaction request.
stop_ts	Time value for the transaction response.
timestamp	Unix timestamp of the start of the transaction.
user-name	IMSI used to identify the user of the Diameter transaction.
visited-plmn-id	The decoded visited PLMN ID found in the transaction with the 3 digit Mobile Country Code (MCC) and 2 or 3 digit Mobile Network Code (MNC) as defined in TS 123000 at section 12.1.

14

Public CDR Generation

Note: CDR file generation can be enabled and disabled using the system setting **Enable CDR Writer**.

Oracle Communications Operations Monitor creates call detail records (CDR files) in CSV format. The files are placed in the **cdr/** directory from the FTP/FTPS root. The CDR files have the following format:

`palladion-`*`unix_timestamp-sequence`*`.csv`

where:

- *unix_timestamp* is the Unix timestamp when the file was created.
- *sequence* is a number monotonically increasing.

The files are rotated when they reach their maximum size. The maximum size is configurable from the System Settings (by default, 50000 records). When a CSV file is finished, another empty file having the following format is created:

`palladion-`*`unix_timestamp-sequence`*`.csv.FIN`

Several times per day, old CDR files are compressed to files having the following format, while the corresponding uncompressed files are deleted:

`palladion-`*`unix_timestamp-sequence`*`.csv.gz`

The recommended way of gathering the CDR files is to connect via FTP/FTPS, copy and delete all the files ending in **csv.gz**. Alternatively, to get the CDR data in near real time, the CDR files that have a corresponding **.FIN** file can be copied and deleted.

Operations Monitor automatically limits the size of the **cdr/** directory to 1 GB, by deleting the oldest files.

One record (CDR) for each call seen is created (entries are created only for successfully established calls, failed calls do not appear in the CDR files). If the call is visible in different segments, all of them are taken into account for computing the call details, but only a single record will be written. The call details are always the same as presented in Operations Monitor's web interface.

Operations Monitor also allows the creation of periodic CDRs for a call while a call is still active. This functionality can be enabled and disabled by using the system setting **CDR Interim Update Interval**.

By default the value should be 0. A value of 0 means no periodic CDR. The legal values for the system setting are between 1 and 10. This value represents the number of minutes at which a periodic CDR entry will be added to the CDR file.

Note: Activating the periodic CDR functionality will have an adverse effect on performance.

Table 14–1 lists the fields present in the generated CSV files:

Table 14–1 CDR CSV Fields

Field	Description
acct_status_type	Used for the periodic CDR. Can be one of the following: ■ start. This is the first CDR for an established call. ■ update. This is a periodic CDR update for an already established call. ■ stop. This is either the last CDR that closes an already established call or it is not a periodic CDR or it is a canceled/failed call. To determine which of the above it is read the **sequence_number** CDR field and also the **state_msg**.
avg_mos	The average of MOS estimation values according to the E-model, computed by Operations Monitor either from the RTP stream or from the end point messages, depending on availability.
avg_rtcp_delay	Average RTCP delay in milliseconds: ■ If there is RTCP-XR with a delay value, this value is used. If there are multiple streams with RTCP-XR delay values, the average of those values is used. ■ If there is no RTCP-XR delay value, the RTCP delay is calculated using pairs of RTCP Sender Reports (see RFC 3550)
callee_ip	IP address of the called user that connected first. This field may be empty if the call was not successful.
caller_ip	IP address of the device initiating the call.
callid	The **Call-ID** header value from the initial INVITE.
call_time	The duration in milliseconds of the call. The time is measured from the final successful message until the first BYE message.
diversion_params	This field contains a "**;** " (semicolon) separated list of the parameters of the first **Diversion** header we see in the call.
diversion_uri	This field contains the uri of the first **Diversion** header we see in the call.
dst_codecs	The codecs, in order, offered by the callee. In case the codecs are changed during the call by using re-INVITEs, this field will contain the last offer of the callee.
dst_ip	Destination IP address of the first INVITE message.
dst_mac	Destination hardware address of the first INVITE message.
dst_port	Transport layer port of the first INVITE message.
dst_ua	User agent string of the callee.
dst_uri	SIP URI of the callee as present in the **In** header field.
dst_user	The user to which the call is addressed. This is usually taken from the **To** header field of the first *call leg*, or as defined by the **Number Determination** algorithm.

Table 14–1 (Cont.) CDR CSV Fields

Field	Description
dst_user_pref_tag	If the called user string is defined as the result of the **Number Determination** algorithm, this field is set to the numerical ID of the matching Number Determination rule.
dtg	The value of the **dtg** URI parameter from the **Request-URI** of the initial INVITE.
egress_devs	Comma-separated list of numerical device IDs of the egress devices for the call, that is, through which the call leaves the platform.
from_tag	The value of the **tag** parameter from the **From** header of the initial INVITE.
id	Unique identifier of the call within the Operations Monitor core instance.
ingress_devs	Comma-separated list of numerical device IDs of the ingress devices for the call, that is, through which the call enters the platform.
init_devs	Comma-separated list of numerical device IDs of the initiator devices for the call, that is, of devices that initiated a call segment without incoming segments.
max_rtcp_delay	Maximum RTCP delay in milliseconds. This is the maximum value of RTCP delay values seen across all streams for this call.
mec_ids	Comma-separated list of hexadecimal strings that can be used to correlate CDRs found on multiple Mediation Engines.
media_leg_locations	Comma-separated list of storage locations for Media Leg details. Voice quality records are kept in separate CSV files with Media Detail Records. For more information, see "Voice Quality Records (MDRs)". If an MDR is related to this call, its **media_leg_location** field will match one of the CDR **media_leg_locations**. **Note:** There may be locations without MDRs if no RTP stream was seen by Operations Monitor.
media_types	Indicates the media types that were negotiated in the call. Multiple media types are separated by a comma (for example: audio, video).
megaco_gateway	String with the IP address of the H.248/Megaco Media Gateway.
mgcp_gateway	String with the IP address of the MGCP Media Gateway.
MOS	The minimum MOS estimation according to the E-model, computed by Operations Monitor either from the RTP stream or from the end point messages, depending on availability.
otg	The value of the **otg** URI parameter from the **From** header of the initial INVITE.
pai	The uri of the **P-Asserted-Identity** of the initial INVITE.
pid	Identifier of the Operations Monitor core instance that created the record.
privacy	The value of the **Privacy** header of the initial INVITE.
q850_cause	This header contains the Q.850 cause code learned from native Q.850/ISUP signaling and NOT SIP.

Table 14–1 (Cont.) CDR CSV Fields

Field	Description
realm_ids	Comma-separated list of numerical IDs of the realms the call belongs to.
ruri	The **Request-URI** from the initial INVITE.
sequence_number	A sequence number for each periodic CDR update entry. Is unique in the context of each call. For non-Periodic CDR it should always have the value of **1**.
setup_delay	The duration for the call setup in milliseconds. The time is measured from the first INVITE messages until the last bit of the first provisional response is received.
setup_delay_type	The setup delay type used for the setup delay. Can be one of the following: ■ Successful Session Request Delay ■ Failed Session Request Delay
setup_start_ts	UNIX timestamp of the initial INVITE message.
setup_time	The duration in milliseconds of the call setup. The time is measured from the first INVITE message until the final successful answer.
sip_code	The SIP response code of the last received message from the INVITE transaction. For **Failed** calls, this represents the SIP error code.
sip_reason_cause	The code value of the SIP **Reason** header.
sip_reason_protocol	If there is a **Reason** header then this field shall contains the protocol of that header. Can be one of the following: ■ SIP ■ Q.850
sip_reason_text	The text explanation contained in the SIP **Reason** header. If the protocol is Q.850 then this field will be empty.
src_codecs	The codecs, in order, offered by the caller. In case the codecs are changed during the call by using re-INVITEs, this field will contain the last offer of the caller.
src_ip	Source IP address of the first INVITE message.
src_mac	Source hardware address of the first INVITE message.
src_port	Transport layer port of the first INVITE message.
src_ua	User Agent string of the caller.
src_uri	SIP URI of the caller as present in the **From** header field.
src_user	The user making the call. This is usually taken from the **From** header field of the first *call leg* or as defined by the **Number Determination** algorithm.
src_user_pref_tag	If the calling user string is defined as the result of the **Number Determination** algorithm, this field is set to the numerical ID of the matching Number Determination rule.
state_details	Operations Monitor can add details about the call state. Example: BYE seen but no 200 OK.

Table 14–1 (Cont.) CDR CSV Fields

Field	Description
state_msg	Call state. It can be one of the following: ■ Established. The call was successfully established. ■ Redirected. The call was not established but was redirected. ■ Finished. The call was successfully established and closed. ■ Timed out. The call was successfully established but the BYE message was never received, so it expired. ■ Error. The call was not recognized properly. ■ Unauthorized. ■ Canceled. ■ Failed. The call failed to get established, or there was some other reason for not being successful. ■ Not found. 404 User not found, 604. ■ Off-line. 480. ■ Busy. 486, 600. ■ Terminated. 487 If a call is ISUP based, state message might differ.
term_devs	Comma-separated list of numerical device IDs of the terminator devices for the call, that is, of devices that terminated a call segment without outgoing segments.
to_tag	The value of the **tag** parameter from the **To** header of the initial INVITE.
trav_devs	Comma-separated list of numerical device IDs of traversed devices for the call, that is, of devices that have both incoming and outgoing segments.
uid	Unique identifier for the call, generated from the *Call-ID*, *to_tag*, and *from_tag*. Can be used for merging the CDRs generated by different Operations Monitor instances.

Note: The fields **ts**, **setup_time**, and **call_time** are time related.

- Because the hardware clock of the Operations Monitor server has limited precision over time, we recommend configuring NTP servers.
- The timestamp precision may be affected by the delays introduced by tapping devices. The timestamp of the message is always saved the moment it reaches Operations Monitor's network interface.
- The fields **setup_time** and **call_time** are computed by subtracting the timestamps of the messages INVITE, 200 OK, and BYE with a precision of milliseconds.

The field **sip_code** in the final CDR of periodic CDRs of a canceled call may be 200 (from the response to CANCEL) instead of 487 because writing the CDR is triggered after the end of the CANCEL transaction.

15

Voice Quality Records (MDRs)

Oracle Communications Operations Monitor creates media detail records (MDR files) in CSV format. The files are placed in the **mdr/** directory under the FTP/FTPS root. The MDR files have the following format:

`mdr-`*`unix_timestamp-sequence`*`.csv`

where:

- *unix_timestamp* is the Unix timestamp when the file was created.
- *sequence* is a number monotonically increasing.

The files are rotated when they reach their maximum size (10000 records). When a CSV file is finished, another empty file having the following format is created:

`mdr-`*`unix_timestamp-sequence`*`.csv.FIN`

Several times per day, old MDR files are compressed to files having the following format, while the corresponding un-compressed files are deleted:

`mdr-`*`unix_timestamp-sequence`*`.csv.gz`

The recommended way of gathering the MDR files is to connect via FTP/FTPS, copy and delete all the files ending in **csv.gz**. Alternatively, to get the MDR data in near real time, the MDR files that have a corresponding **.FIN** file can be copied and deleted.

Operations Monitor automatically limits the size of the **mdr/** directory to 5 GB, by deleting the oldest files.

There are two types of data sources: Operations Monitor probe machines, which see and analyze the RTP stream and the VQ collector of Operations Monitor, which receives SIP PUBLISH messages with Voice Quality reports. Operations Monitor writes one MDR per RTP stream per data source. That means for one RTP stream there may be multiple MDRs. For example, if the stream was seen by two Operations Monitor probes or if it was seen by a probe and a VQ report was received.

To correlate a RTP stream record (MDR) with a call (CDR), you may use Media Leg information. A media leg is defined by a UDP address pair found by an SDP offer/answer exchange. Depending on the call scenarios there may be multiple media legs per call (for example, from multiple segments or after a re-negotiation) or multiple calls per media leg (for example, if the same SDP appears in different calls). Usually each media leg will contain two RTP streams, one per direction. In an MDR the media leg the stream belongs to can be identified using the **media_leg_location** field. In a CDR the **media_leg_locations** field lists the media legs correlated with the call. To find the CDR(s) related to an MDR, use the MDR **media_leg_location** and look for the CDRs where this string is one of the **media_leg_locations**. To find the MDRs related to a call, use all entries in the **media_leg_locations** field of the CDR and look for the

corresponding **media_leg_location** in an MDR. Note that there may be media legs which do not have any MDRs, for example, because no RTP was seen by Operations Monitor.

Table 15–1 lists the fields present in the generated CSV files.

Table 15–1 MDR CSV Fields

Field	Description
avgPDV	Average packet delay variation in milliseconds.
dst_ip	Destination IP address of the RTP stream.
dst_port	Destination UDP port of the RTP stream.
end_ts	UNIX timestamp when the last RTP packet was seen (floating point number). For a VQ report from a SIP PUBLISH message, this is the timestamp when the report was received (start_ts and end_ts will have the same value).
expected	Number of RTP packets expected.
lost	Number of lost RTP packets.
maxPDV	Maximum of the packet delay variation values of all 10 second intervals.
media_leg_id	Unique identifier of the media leg (unique if combined with pid). Note, there may be multiple records per media leg, one per RTP stream.
media_leg_location	Internal identifier which can be used to find the media leg inside the Operations Monitor system.
minMOSlqe	Minimum MOSlqe value of all 10 sec intervals of the stream. Available only for the Operations Monitor source.
MOSlqe	MOS Listening Quality score for the whole stream, based on the E-Model, if available.
packet_loss_rate	Percentage of lost packets (100 * lost/expected).
payloads	Codecs used by the stream: format names like PCMA, G729, telephone-event.
pid	Identifier of the Operations Monitor process instance that created the record.
source	The source of the measurement: ID of the Operations Monitor probe machine where the RTP stream was seen or “RTCPXR” if the Operations Monitor VQ collector received a SIP PUBLISH message with a Voice Quality report.
src_ip	Source IP address of the RTP stream.
src_port	Source UDP port of the RTP stream.
ssrc	RTP SSRC identifier.
start_ts	UNIX timestamp when the first RTP packet was seen (floating point number). For a VQ report from a SIP PUBLISH message, this is the timestamp when the report was received (start_ts and end_ts will have the same value).
vlan	VLAN tag of the RTP stream.

16

Troubleshooting

This chapter describes how to fix common Oracle Communications Operations Monitor problems.

First Aid

In case you have a problem, please follow this checklist:

- Re-read the corresponding passages in the manual. Some features have special requirements, so make sure to check out these as well. For example, previous configurations, which may be covered in a different context.
- Try to find your problem in the list of suggestions in "Suggestions for Specific Error Cases".
- If you can't find a solution for your problem, you should create a system diagnostic report from Platform Setup Application and notifying your Operations Monitor support contact.

Suggestions for Specific Error Cases

This section lists problems you might encounter and provides information on how to fix them.

All calls are in the Proceeding state

Please check that both directions of the traffic are correctly mirrored to Operations Monitor. You can verify if both directions are correctly seen by using the diagram from the **Call Details** window.

A sudden drop shows in all charts

This usually means that the Operations Monitor core process was restarted. This can have two causes:

- A configuration change that requires the core to restart. We are continuously trying to reduce the number of configuration changes that require the core to restart. There are a few, however, that still require restarting, for example the **Use Usr Domains** system option.

  The option's changes that require a core restart are generally marked with a warning.
- Due to a system error, the core was restarted in order to return to the functional state as soon as possible.

'Sorry, an error occurred while processing your request'

This is the default error message shown by Operations Monitor when an unexpected exception occurs. It can be due to unexpected user input or due to a software regression. In these cases, creating an error report and sending it to Oracle or your system integrator will help us to fix the problem as soon as possible.

Domain realms do not work

Even though you have configured domain based realms, they do not work unless you enable the Use User Domains system option. For more information, see "Use User Domains".

After updating, some of the users don't see the new pages

After an update of Operations Monitor that adds functionality, you need to explicitly add the user rights to see and use new pages. To make this process as easy as possible, you have the option to add rights at once to multiple users in the **User Management** section.

No voice quality is displayed

Please make sure that the RTP traffic is mirrored to Operations Monitor, and the RTP module is enabled by the license you currently use.

All registration contacts are marked as Gone soon after the registration refresh

This is usually caused by the fact that the network has a SIP device (for example, SBC) that forwards the registration attempts and also alters the contact requested by the user. In this case, the issue can be solved by changing the values of the **Registrations Gone Events** and the **Ignore Internal Registrations** system options. For more information, see "Registrations Gone Events" and "Ignore Internal Registrations".

Created traces are empty

In the **Traces** page, please check that the requested trace period is covered by the raw buffer content.

A

System Settings Summary

This chapter provides a summary of the Oracle Communications Operations Monitor system settings.

302 Redirected: To-URI suffix length

How many characters of the user part of the Contact: and Request URI must match to merge a call. Use 0 to disable this option.

APID request interval for querying SBCs

Number of seconds interval for retrieving informations about autodiscovered SBCs. It is not recommended to alter this value.

Alerting delay after core restart

The length of time alerts are suppressed after a core restart, in minutes.

Allow regeneration of registration events for user updates

If a user is registered in core once, its realm will not be re-checked until the registration expires and gets renewed. If realms are changed continuously and/or customer needs to have registrations always matched to the current realms configuration, enable this option. Be aware that this has some performance impacts on core registration processing.

Append a string at the end of User Tracking searches

When using User Tracking, also look for aliases that have this suffix appended to the number.

Authentication token (shared secret between MEC and ME)

Secret used by the Mediation Engine Connector to sign requests proxied to this Mediation Engine.

Auto-refresh interval for grids

Number of seconds between auto-refresh (if enabled) for grids with this feature.

B2B incoming backlog search for merging

The number of seconds to search the outgoing call leg in the past when an incoming leg is received.

B2B outgoing backlog search for merging

The number of seconds to search the incoming call leg in the past when an outgoing leg is received.

Bind user sessions to IP addresses

Bind each session to an IP address.

Bulk counters/KPIs maximum limit

The maximum limit of counters being added in bulk.

CDR Interim Update Interval

This value represents the interval in minutes for which partial CDR data will be created. The allowed range of this interval is from 0 - 10 minutes. If 0 is given then the partial CDR generation feature is turned off. Warning: Turning on this feature has an impact in performance.

CPM Transaction time range

The default limit in minutes to cutoff CPM transaction queries. A value of 600 will result in the transactions grid showing transactions in the range of [NOW(), NOW() - 10 Hours].

Call Report Maximum Messages

Maximum number of messages to include in the call report messages section.

Call Report Style

Additional Cascading Style Sheet (CSS) rules can be defined here that will be included when generating a call report PDF.

Call Report Theme Color

Color of the headings and footer on PDF call reports. Can be the name of a color, or a hexadecimal representation of the color: #ff0000.

Call Transfer: Seconds for correlating calls using REFER

The number of seconds the Mediation Engine looks back for calls containing a REFER causing the current call. If 0, correlating the referred and referring call is disabled for the Call Transfer information.

Call Transfer: correlate using Replaces

Set to true to match calls involved in a call transfer using the Replaces header.

Call flow max height

The maximum height of the call flow diagram in pixels.

Call flow max width

The maximum width of the call flow diagram in pixels.

Call flow messages

The maximum number of messages displayed in a call flow.

Call flow parallel loading

When set, the in-browser call flow diagrams will be loaded in several stages, with progressively more information for some messages. This might improve rendering performance.

Call flow timestamp for fragmented messages
Use last fragment's timestamp as the timestamp for the message.

Call flow tree's expanded levels
Expanded levels in the popup for a message.

Cleanup Actions Syslog logging facility
Which logging facility should be used for the Mediation Engine's cleanup actions. Leave negative if this should be disabled. Allowed values are zero through to seven.

Custom header for realm definition
Specify a SIP header field name that is used to match the "Custom Pattern" definition for a realm.

Day on which the week should begin
Defines the day at which the week should begin. A page reload is required for this setting to take effect.

Days until inactive account expires
Expire inactive accounts after the given number of days.

Device Map Limit
Above this number of devices, device counters will be removed.

Digest delay time
The time between sending out digests of recurring alerts in minutes.

Disable logo changes
If enabled, users cannot make logo changes.

Disable user changes
If enabled, users cannot make password or logo changes.

Dumper raw buffer size
The size of the raw buffer in MB. This buffer is used for creating traces starting in the past. Warning: Increasing this setting might require adjusting memory quotas. Please contact Oracle or your system integrator.

Enable CDR writer
Enable if you want the Mediation Engine to generate CDR files. Please see the user manual for details about the format used and how to get the files.

Enable MEGACO BGF
Enables the processing of BGF messages in MEGACO.

Enable sharing saved calls
If enabled, saved calls can be shared globally. If you run a multi-tenancy setup, this feature should be DISABLED.

Enabled Device Map

If enabled (the default), Operations Monitor can display a real-time map of the configured platform devices and of the interactions between them. Please note that due to browser and backend performance reasons, this map cannot work for a large number of configured device counters. It is therefore recommended to disable this setting when configuring more than 20 devices or realms. In order to configure the maximum number of devices supported, adjust the **Device Map Limit** setting.

Enforce stringent password rules

A set of higher security rules for user passwords.

Expire passwords periodically

If enabled, non-admin users will be required to change their passwords periodically. This option can be overridden from the Mediation Engine Connector.

Export metrics to OCDM

Enable to export selected KPIs via the Oracle Communications Data Model file based integration.

Extend calls by hash inspection

During call legs discovery across multiple sites, a neighboring Mediation Engine can be interrogated multiple times, rather than only once.

Find similar aliases in User Tracking by using a suffix of this many characters

If this value is non-zero, the User Tracking page first searches the alias list for similar usernames/phone numbers and then displays the calls and registration events for all of them. To usernames/phone numbers are considered similar if the last N number of characters are identical, where N is the value of this setting.

Get updated call states

If enabled, the fronted will query the core process on each request to get the very latest state information. This can increase the delay before the answer is made, but generally makes the information more up to date.

Global partitions size

How many hours a database partition should hold. There can be at most 1024 partitions.

Grey out non-preferred numbers

Numbers that haven't been derived from a configured number determination source will be displayed in gray in the calls table.

Group new registrations from the same user

Group new registrations from the same user.

Guarantee sorting of VSI traces

Due to the distributed nature of the Mediation Engine, it is possible that the packets in the traces buffer are out of order. If this option is enabled, the Mediation Engine will re-sort the packets by their timestamp.

HTTP address of the Mediation Engine Connector

HTTP address where to reach the Mediation Engine Connector. This is automatically set from the Mediation Engine Connector.

Headers in which to look for realm URIs

A space separated list of SIP header fields, for example, **Diversion P-Asserted-Identity**. Realm matching will include these fields (in addition to To and From).

Hide Megaco messages before off-hook

This option helps if the MEGACO transactions IDs are reused very quickly. It will hide the messages that are assigned to the leg before the off-hook event.

Hide active user sessions

Hide active user session info when limit is reached.

Hide hardware addresses input from devices configuration

If set to true, the hardware addresses field will be hidden from the platform devices wizard.

High threshold for MOS

The high threshold for determining the MOS.

Ignore internal registrations

If disabled, registrations from inside the platform will be recorded. This might make some REGISTER transactions to be incorrectly considered unauthorized because the 401/407 answer is seen more than once.

Ignore internal registrations for User Devices statistics

If enabled, the internal registrations (starting from a platform device) will be ignored when computing the User Devices statistics.

Ignore mismatch on contact URI (registrations)

Some registrars change the Contact URI in a reply to a REGISTER. This tells the Mediation Engine to ignore mismatching URIs, but might lead to merging multiple registrations into one.

Ignore non RFC 3261 params in SIP URI username

If set to true, the Mediation Engine will remove the parameters appended to the username before matching the SIP URI username.

Ignore plus prefix number when matching realms

By default the "+" number prefix is ignored in realm matching, when turned on, this option will assign numbers containing + only to the default realm.

Initial search range

The initial search range (in seconds) for calls, registrations etc to search before asking if the search should continue. If 'Paging query range' is bigger, that value will be used.

KPI - RTCP Jitter threshold

Defines a threshold for jitter KPIs (in ms) based on RTCP. Any jitter value above this threshold is ignored for the KPI computation. 0 means no threshold.

Limit select queries to this many seconds

Stop SELECT queries if they take more than this many seconds. Set to 0 to disable.

Log user authentication failures

Log information when a user fails to login.

Login Splash Page

A splash page which is shown after login. Removing all content will disable the splash page.

Loose user searching

If enabled, when searching users by using the live search feature, the last digits of the number act as wildcards to suggest similar numbers from the same range.

Low threshold for MOS

The low threshold for determining the MOS.

Match registration events by comparing a suffix of the username

The Mediation Engine matches the related registration attempts by the SIP username. If this value is zero, then the whole username is used. If more than zero, the number defines how many digits/characters must match.

Max concurrent sessions

The maximum number of times a user can login in parallel.

Max concurrent users

The maximum number of users that can be logged in at the same time.

Maximum concurrent traces

The maximum number of traces that can be running at the same time.

Maximum frequency for raising alerts

Determines at what point raised alerts that are not read can be raised again. If an alert has been continuously raised for more than this amount of seconds, it is eligible for being raised again.

Maximum lines of a CDR file

The maximum number of lines that a CDR file can contain. When this limit is reached, the Mediation Engine automatically creates a new CDR file.

Maximum number of calls exported with bulk export

The maximum number of calls that can be exported with one bulk export.

Maximum number of items returned by each request to the REST interface

Maximum number of items returned by each request to the REST interface.

Maximum number of legs in one call

If a call already has this many calls, the Mediation Engine won't merge new legs in the same call, even if they match according to the merging algorithm. The purpose of setting a maximum is to avoid infinite calls.

Maximum number of lines exported with CSV export

The maximum number of lines that can be exported with one CSV export.

Maximum number of messages per leg

Configure maximum number of messages to be stored per leg.

Maximum number of number alert definitions per user

Maximum number of number alert definitions that a user can create.

Maximum script output size

The maximum output size by script run in bytes that is shown. It's stored in memory.

Maximum simultaneous script runs

The maximum number of script runs that can run simultaneously.

Maximum stored script runs

The maximum number of script runs that are stored. If this number is exceeded the oldest script runs are removed as needed to stay under this limit.

Mediation Engine Connector hash on P-Charging-Vector icid-value parameter

To enable call merging in Mediation Engine Connector, you hash the INVITE message of the corresponding leg. By default, the Mediation Engine hashes the full INVITE message. When enabled, and the INVITE message contains a **P-Charging-Vector** header with an **icid-value** parameter, the value of the **icid-value** parameter is used to find a match instead of a hashing the full INVITE message.

Field	Value
type	Boolean
flags	Browse
default	False

Mediation Engine Connector hash search on all external legs

To enable call merging in Mediation Engine Connector, you hash the INVITE message of the corresponding leg. By default, the mediation engine hashes **all** the call legs that traverse between neighboring mediation engines. When enabled, a hash is added to **all** call legs to or from unknown IP addresses. This option exists for backward compatibility: enabling this option is not recommended if you require call correlation.

Field	Value
type	Boolean
flags	Browse
default	False

Note: Oracle recommends that the **Mediation Engine Connector hash search on all external legs** system setting is set to disabled.

Merge globally by Call-ID

If this setting is enabled, Mediation Engine will merge two legs that have the same Call-ID even if they are not incoming/outgoing to the same device.

OCDM connection string

The Oracle database connection string to be used for querying Oracle Communications Data Model.

Omit Gray Bars

Omit the gray bars in charts.

Paging query range

Time range a query covers in a single paging request (in seconds).

Partitions to display in Recent calls

Number of partitions to display at once in Recent calls.

Partitions to display in User tracking

Number of partitions to display at once in User tracking calls.

Password strength

If enabled, the users will be required to respect a set of rules for their passwords (minimum 8 characters, one non-lowercase letter character, no reference to the user name).

Passwords expire duration for non-sensitive enabled users

Number of days after which users without access to sensitive information are required to change their password.

Passwords expire duration for sensitive enabled users

The number of days after which users with access to sensitive information are required to change their password.

RUDP quick push

Choose the RUDP decoding mode: when 'quick push' mode is activated, the data chunks will be processed as soon as they are read, rather then waiting for the corresponding ACK. This allows better operation of the Mediation Engine in cases where ACKs are delayed excessively, but it will also prevent the detection of some malfunctions in the protocol (like lost packets).

Recurrent alerts threshold

Days during which a new alert identical to a raised unread one will not be created.

Refer: Merging calls requires full Contact match

If call merging on REFER is enabled this option controls whether the full Contact URI must match between the INVITEs of the original call and the referred one. If enabled

the complete Contact URIs of both INVITEs are compared, if disabled only the user name parts are compared.

Refer: Seconds for merging calls

The number of seconds the Mediation Engine looks back for calls containing a REFER causing the current call. If 0, merging the referred and referring call is disabled.

Refer: To-URI suffix length

How many characters of the user part of Refer-To: and To: must match to merge a call. Set to 0 to compare the full user part.

Refresh interval for portlets

The refresh interval in seconds that overrides the default one in each portlet when the user reloads the dashboard. Set to 0 to use each portlet's default interval.

Registrations gone events

If enabled, a REGISTER transaction in which not all previous known contacts of the user are seen in the successful answer generates gone events for the missing contacts. Leave disabled if this is expected in your network.

Rotate CDR files every N seconds

If this option is non-zero, the CDR files are rotated every N seconds, where N is the value of the setting.

SCTP quick push

Choose the SCTP decoding mode: when **quickpush** mode is activated, the data chunks will be processed as soon as they are read, rather then waiting for the corresponding SACK. This allows better operation of the Mediation Engine in cases where SACKs are delayed excessively, but it will also prevent the detection of some malfunctions in the protocol (like lost packets).

SIP packet deduplication

This option sets & enables packet deduplication for SIP on the ME. The value represents the time limit (in ms) for which a packet is considered duplicate. 0 means off. Warning: enabling deduplication has performance impacts.

SS7 default flavor

Choose the default standard for the MTP3/ISUP protocols. Supported are ITUT and ANSI.

SS7: Prefer ported number to callee (in IAM)

If IAM contains GAP for Ported number, display this as callee number. The matching algorithm is not affected by this setting.

Search for matching registration segments

When viewing the diagram of a Registration event, Mediation Engine will search for other registration events that are related. This option controls the time span in which it searches, in seconds. If set to 0, the searching of related registration events is disabled.

Seconds for merging 302 calls

The number of seconds the Mediation Engine looks back for related 'redirected by 302' calls. If 0, merging the redirected calls is disabled.

Seconds for remembering INVITE when merging 302 and REFER calls

The number of seconds the Mediation Engine looks forward for an INVITE if merging calls on 302 and REFER messages. Keep this value low. 0 disables just the forward-looking functionality.

Session-timeout for calls

The number of seconds after which the Mediation Engine will consider a call as Timed-Out if no SIP messages are received.

Set customer id for VSI FDP module

Specify a customer ID for VSI FDP module.

Show ISUP abnormal termination in details

For certain Q850 codes, add '- abnormal termination' on the details field.

Show OCDM widget

Display data gathered from the Oracle Communications Data Model in the User tracking page.

Strict SIP CSeq handling for unauthorized calls

When unset, some non-RFC-conformant but successful calls will be treated as successful instead of failed.

Stringent passwords expiration duration

Number of days after which passwords expire when the stringent password policy is enabled.

Suffix length (H.248, Diameter, ENUM correlation)

Define the suffix length for correlation on DIAMETER Public ID, H.248/Megaco and ENUM. Set to 0 to only remove leading + from the Request URI/the Public ID.

TCP quick push

Choose the TCP decoding mode: when 'quick push' mode is activated, the data chunks will be processed as soon as they are read, rather then waiting for the corresponding ACK. In normal operation, you should not activate this option. If Agents see only one half of TCP traffic (and hence no ACKs are received), enable this option.

TDR enabled

Enable the Transaction Data Record.

Time difference in probes that will trigger an alert

Time difference in milliseconds between a probe and the ME that will trigger an alert.

Time window for each chunk of records when exporting

Number of seconds defining the interval for sql-querying a chunk of records to be exported.

Time window for records when exporting

Number of seconds (back from the last record) used for limiting the records to export.

Timeout for Mediation Engine querying

Number of seconds to wait for an answer from the Mediation Engine peers. If one of them becomes unavailable, the timeout ensures that the local results are still available.

Trace delay time

The time to wait between starting a new trace from the same alert definition, in minutes.

Transport MAC aware

This setting is used when matching a message to an existing call leg. If this setting is 'true', in case the source IP or destination IP of a new message is not configured as a device, Operations Monitor checks the MAC address before matching this packet to an existing call leg. However, if both source and destination IPs of the new packets can be matched to an existing device, this setting will be ignored and the device MAC address configuration will have precedence when matching the packets to the call leg.

Transport VLAN aware

This setting is used when matching a message to an existing call leg. If this setting is **true**, in case the source IP or destination IP of a new message is not configured as a device, Operations Monitor checks that both source and destination MAC addresses are the same before matching this packet to an existing call leg (if one of the endpoints was already matched via device lookups, the MAC address check for that endpoint is skipped). However, if both source and destination IPs of the new packets can be matched to an existing device, this setting will be ignored and the device VLAN configuration will have precedence when matching the packets to the call leg.

Transport ports aware

Consider ports when comparing packet and device addresses.

Use the hash of preferred number config to link it to Caller/Callee

If enabled the hash of a Preferred Number config will be used to link it to numbers in calls. If you delete a config, and then re-create it with the same parameters, the link between the Caller or Callee and the Device+HeaderType will be re-established in the UI. If this option is disabled the Database ID of the config will be used to link. In this case if the Preferred Number source config is modified at all the relation will be lost for all historical call data.

Use user domains

Set to true to match users based on the SIP URI (domain name), in addition to the user name (number). Changing this value restarts the core process.

User Actions File logging

Whether or not the Mediation Engine's user action should be logged into the dedicated file logging facility.

User Actions Syslog logging facility

Which logging facility should be used for the Mediation Engine's user action. Leave negative if this should be disabled. Allowed values are zero through to seven.

User Roles

Assign users roles containing predefined permissions.

User default locale

Assign newly created users this locale.

User devices chart history limit

Fetches only calls not older than specified number of seconds.

User tracking paging query range for

Time range a query covers in a single paging request (in seconds) for user tracking. Should typically be larger than the normal paging query range.

Voice Quality Chart scale in minutes/minute

If true, the Voice Quality Chart scale will be minutes/minute instead of minutes/15 minutes.

Glossary

AOR

Address of Record.

API Key

Key that is used for authentication of calls to the new REST API. It represents the user account it was created with. It can be enabled and disabled in **Settings**.

B2BUA

Back-to-back User Agent. A logical entity that receives a request and processes it as a UAS. In order to determine how the request should be answered, it acts as a UAC and generates requests. For more information, see **RFC 3261** on the IETF Tools website at:

http://tools.ietf.org/html/rfc3261.html

BSS

Business Support System.

Call leg

A call leg is the portion of the call between two SIP devices.

CDR

Call Detail Record.

Codec

Algorithms for compressing and decompressing data.

CSV

Comma Separated Values. An exchange format for tabular data understood by Microsoft Excel, OpenOffice.org, and many other applications.

DHCP

Dynamic Host Configuration Protocol. Used for automatically assigning network addresses.

Diameter

Network protocol for data exchange, database access, accounting and policy control, successor of RADIUS.

DNS

Domain name service.

DoS

Denial of Service.

DTMF

Dual-Tone Multifrequency Signaling. Known in the UK as MF4. A protocol for transmitting signaling tones, typically originating from the user pressing buttons on a physical hand-set but also used without use interfaction for signaling by appliances such as fax machines. Defined in **RFC 2833**, obsoleted by **RFC 4733**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc2833.html

ENUM

Protocol based on DNS used within IMS for routing decisions.

E-Model

Computational model for use in transmission planning. Defined by the ITU in recommendation G.107.

Egress device

An egress device is the SIP device through which the call leaves the platform.

A more formal definition: An egress call leg is one which has as source a device from the platform, and the destination IP address is from outside the platform. A device is an egress device from a call if it is the source device of an egress call leg. If the call is terminated by a gateway device, this device is also considered an egress device.

Ethernet

Family of frame-based computer networking technologies for local area networks (LANs).

FDP

Fraud Monitor. A product within Oracle Communications Session Monitor that is capable of detecting toll fraud and which provides measures to actively prevent toll fraud related attacks from being successful.

FTP

File Transfer Protocol.

HA

High Availability.

HTML

HyperText Markup Language.

HTTP cookie

Information unit from a web server for purposes of identification and customization. It is stored by the web browser and accessed by the server during subsequent visits.

H.248

Gateway control protocol.

H.323

VoIP protocol defined by ITU-T.

ICMP

Internet Control Message Protocol. Defined in **RFC 792**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc792.html

Ie-eff

Effective equipment impairment factor. See ITU recommendation G.107.

IMS

IP Multimedia Subsystem.

Ingress device

An ingress device is the SIP device through which the call enters the platform.

A more formal definition: An ingress call leg is one which has as destination a device from the platform, and the source IP address is from outside the platform. A device is an ingress device from a call if it is the destination device of an ingress call leg. If the call is created by a gateway device, this device is also considered an ingress device.

IP

Internet Protocol. Defined in **RFC 791**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc791.html

ISUP

ISDN user part.

ITU

International Telecommunication Union.

Javascript

A scripting programming language most commonly used to add interactive features to web pages.

Jitter

A measure of the variability over time of the latency across a network. Term generally used in the VoIP environment describing the variation in delay between packets.

JSON

JavaScript Object Notation, a lightweight computer data interchange format.

LISP

Mature high-level programming language based on lambda calculus.

MDR

Media Detail Record, record containing voice quality details.

ME

Mediation Engine. The ME is the core of the Operations Monitor software running the real-time data processing and serves the frontend and interfaces.

Media Leg

Defined by the media address pair of a conversation, usually the IP addresses/ports of two RTP endpoints.

Megaco

Gateway control protocol.

MEGACO ContextID

A Context is an association between a number of Terminations. The Context describes the topology (who hears/sees whom) and the media mixing and/or switching parameters if more than two Terminations are involved in the association.

MEGACO TerminationID

Termination IDs of physical Terminations are provisioned in the Media Gateway.

MEGACO Transaction

MEGACO Commands between the Media Gateway Controller and the Media Gateway are grouped into Transactions.

MGCP

Media Gateway Control Protocol.

MIB

Management Information Base.

MOS

The Mean Opinion Score (MOS) provides a numerical indication of the perceived quality of the received media. The MOS is expressed as single number in the range of 1 to 5. MOS is always measured by humans. Software products and devices like Operations Monitor can only estimate it, the result being MOS-LQE (listening quality estimate).

The estimation is done based on a set of static parameters and taking into account a set factors related to the flow of the voice packets throughout the network. The content of the packets is *not* deeply inspected. This can have an impact in such cases where a call is hopping over multiple media processors, resulting in multiple legs, some of which not available to Operations Monitor (like foreign network segments, TDM etc). In this cases, only the maximum possible voice quality over the inspected segments is provided, rather than an absolute estimate, end-to-end.

In other words, if one of the media legs not accessible to Operations Monitor will degrade the quality, a processor downstream will decode the signal and re-packetize it to good parameters, but without enhancing it back, Operations Monitor might rate the call higher than the human listener will actually perceive it.

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bade	Very annoying

NIC

Network Interface Card.

NTP

Network Time Protocol.

OID

Object Identifiers.

OSI

Open Systems Interconnection. A joint ISO and ITU-T standard for computer networks and communication protocols.

OSS

Operations Support System.

PCAP

Packet Capture file format. Used by many network analyzers including the open source tool Wireshark. The stored messages contain TCP/UDP headers, IP header and Layer 2 headers, plus the timestamp at which the message was received.

PDF

Portable Document Format. PDF is used for representing two-dimensional documents in a manner independent of the application software, hardware, and operating system.

Probe

A probe is software that collects raw signaling data and media traffic. You can configure probes to run locally within the Mediation Engine (embedded probe), or integrated with Oracle Communications Session Border Controller (embedded probe), or run on dedicated machines (standalone probe).

Proxy

An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing. For more information, see the **RFC 3261** at the IETF Tools website at:

http://tools.ietf.org/html/rfc3261.html

PSTN

Public Switched Telephone Network.

R-Factor

Voice quality score on a scale from 0 (worst) to 100 (best).

RADIUS

Remote Authentication Dial-In User Service is a networking protocol that provides centralized Authentication, Authorization, and Accounting (AAA) management for computers to connect and use a network service.

REST

Representational State Transfer. A convention for web services.

RSPAN

Remote SPAN.

RTCP

Real-time Transport Control Protocol. Used for reporting end point media quality information.

RTP

Real-time Transport Protocol. Used for transporting media. Defined in **RFC 3550**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc3550.html

RTT

Round-Trip Time. The time elapsed for a message to a remote target and back again.

SBC

Session Border Controller. Used in some VoIP networks to offer decoupling, interoperability, and to hide the internal topology. They are typically involved in the signalling and often also relay the media streams. From the SIP point of view, they are usually B2BUAs.

SCTP

Stream Control Transmission Protocol. Is a Transport Layer protocol ensuring reliable, in-sequence transport of messages with congestion control.

SDP

Session Description Protocol. Defined in **RFC 4566**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc4566.html

SIGTRAN

Suite of protocols to enable the use of SS7 over IP networks.

SIP

Session Initiation Protocol. Defined in **RFC 3261**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc3261.html

SNMP

Simple Network Management Protocol.

SPAN

Switched Port Analyzer.

SPIT

SPAM over Internet Telephony.

SS7

Signaling System 7.

TCP

Transmission Control Protocol.

TDR

Diameter transaction records that are created by Operations Monitor and exported in a CSV file.

UAC

User Agent Client. The SIP element that creates a new request; usually the caller's SIP device in case of calls, or the user's SIP device in case of registrations. For more information, see RFC 3261, Section 6 on the IETF Tools website at:

http://tools.ietf.org/html/rfc3261.html

UAS

User Agent Server. The SIP element answering the request; usually the callee's SIP device, or a SIP server. For more information, see **RFC 3261**, Section 6 on the IETF Tools website at:

http://tools.ietf.org/html/rfc3261.html

UDP

User Datagram Protocol.

URI

Uniform Resource Identifier.

VLAN

Virtual Local Area Network.

VRRP

Virtual Router Redundancy Protocol. A redundancy protocol described in **RFC 3768**. For more information, see the IETF Tools website at:

http://tools.ietf.org/html/rfc3768.html

x86-64

A 64-bit superset of the popular x86 instruction set architecture.

XML

The Extensible Markup Language is a flexible text format for creating structured computer documents.