Oracle® Communications Session Monitor

Developer Guide

Release 4.1

ORACLE®

Oracle Communications Session Monitor Developer Guide, Release 4.1

E99933-01

Contents

About This Guide

This guide describes how to extend Oracle Communications Session Monitor by using the Oracle Communications Session Monitor SAU Extension.

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

- Operations Monitor

- Enterprise Operations Monitor

- Fraud Monitor

- Control Plane Monitor

Revision History

Date	Description
November 2017	Initial release

1
Overview of SAU Extension

This chapter provides an overview of the Oracle Communications Session Monitor SAU Extension.

> **Note:**
>
> The SAU Extension feature requires a separate software license.

About SAU Extension

The Simultaneously Active Users (SAUs) KPI is defined as the number of users that are creating calls, sessions, or SMS messages during the busiest hour. It is computed as the number of users that creates a successful INVITE or a MESSAGE SIP request as active. Session Monitor counts how many distinct active users are in a given hour (for example, from 8:00 to 9:00, from 9:00 to 10:00, and so on).

The Simultaneously Attached Users (SAtUs) KPI is defined as the number of subscribers that are successfully attached to the LTE network during the busiest time of the day. It is computed as the number of subscribers (identified by IMSI) that are creating a successful Authentication Information S6a transaction during a given time interval. For the SAtU KPI, the time interval is also based on the clock hour, but can be a multiple of hours (for example,1h, 2h, 3h, 4h or 6h).

Session Monitor measures the KPIs for all the supported intervals, and it is only a matter of querying the interval length that is required. Both the RESTful API and the integrated GUI widgets have controls for selecting the interval lengths.

Using Session Monitor to View the Number of Simultaneous Active Users

A Session Monitor GUI widget is provided on both the ME's Operations Monitor product and in the AE's Mediation Engine Connector (MEC) product for visualizing the SAU and SAtU evolution and for displaying any errors in measuring. To add the widget, right click on an empty space from the dashboard and select Add a panel option. This is possible both in the ME's Operations Monitor and in the AE's MEC products.

The following shows the Add a panel dialog box on the dashboard.

In the Dashboard panel type dialog box, select Display Simultaneously Active Users for the SAU KPI or Display Simultaneously Attached Users for the SAtU KPI.

The following figure shows an example of Simultaneously Active Users panel.
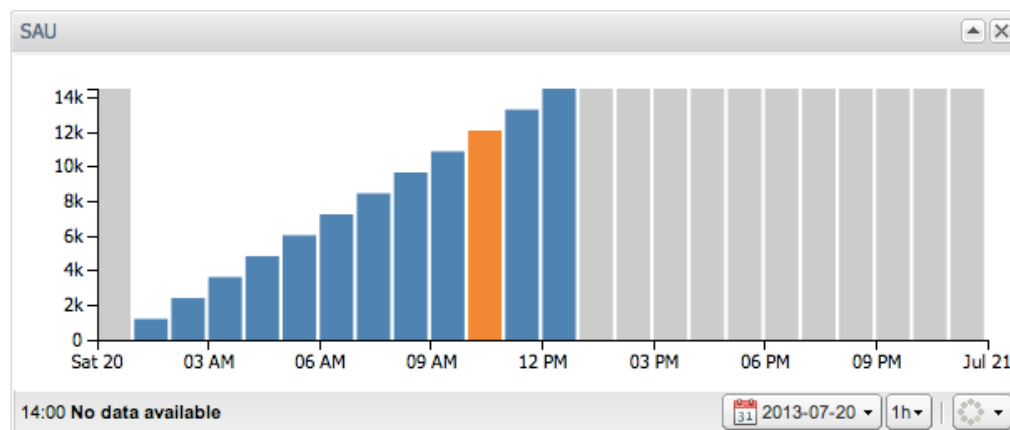
The SAU and SAtU panels have two configuration options:

- The first drop-down allows selecting the date for which to view the SAU and SAtU KPI evolution. Only the view of a 24 hours period is available.

- The interval drop-down allows the selection of one of the available intervals (1 hour, 2 hours, 3 hours, 4 hours or 6 hours). The values for each of the intervals is stored separately in Session Monitor's database. Simply adding the 1 hour intervals to create higher level intervals would not be correct because duplicates need to be eliminated. Therefore, only the pre-defined intervals can be selected.

The following figure shows an example of the SAUs in three hour intervals.
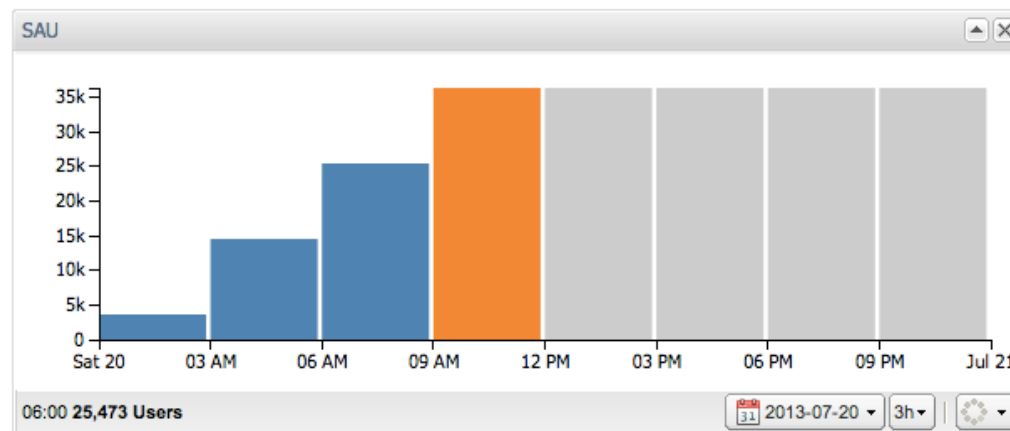
Hovering over an interval bar shows the value for that time interval. If the bar color is blue, the value is known to be correct. If the color is orange, some Session Monitor errors (for example, probes not reachable or overload situations) took place during that interval. In this case, the tooltip will show what error happened. If errors are shown for an interval, the computed KPI value can be smaller than in reality, but it is guaranteed to never be higher.

The following figure shows an example of an error in an interval.
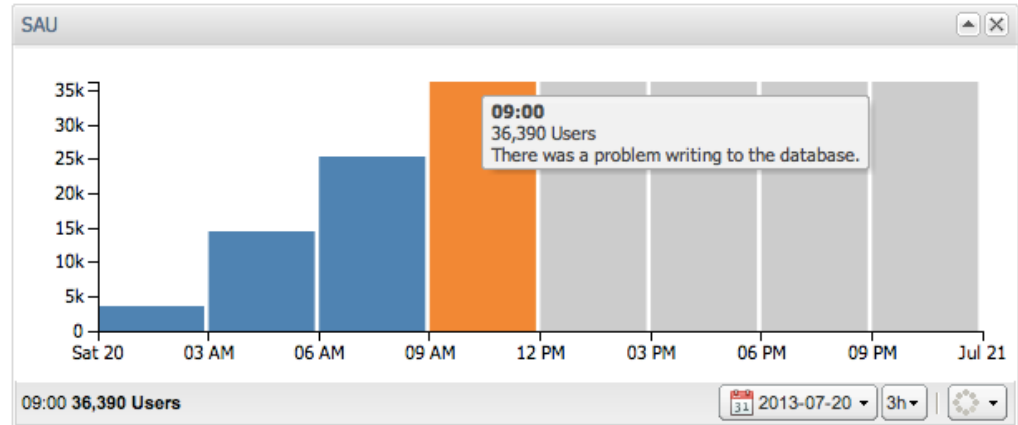
When adding or removing MEs and Probes to the Session Monitor system, they will not be reflected in the values of the KPIs for up to the selected interval length. For example, when selecting 6h intervals, and adding a new ME to the system, the values will be guaranteed to take the new ME into account only 6 hours after the addition.

2

Deploying the SAU Extension

This chapter describes deploying the Oracle Communications Session Monitor SAU Extension.

Session Monitor Deployments with SAU Extension

The Session Monitor architecture consists of the probe layer, Mediation Engine layer, and the Aggregation Engine layer (see the discussion about Session Monitor architecture in Session Monitor Installation Guide for information about the functions performed in each layer).

The figure below depicts the Session Monitor architecture in a typical deployment that uses the Aggregation Engine layer (AE). The machines are deployed in high-availability (HA) pairs for redundancy reasons. Each Probe machine sends traffic to the two Mediation Engines, which are running as active-active from the point of view of the traffic.

The Mediation Engine parses the SIP and Diameter traffic, the list of the unique phone numbers (matching a regular expression), and IMSI numbers that respect the SAU and SAtU definition for each interval.

Once per hour, this list of users is sent from the Mediation Engine to the Aggregation Engine pair. The Aggregation Engines de-duplicate the lists and count the SAU and SAtU KPIs for each of the supported time intervals.
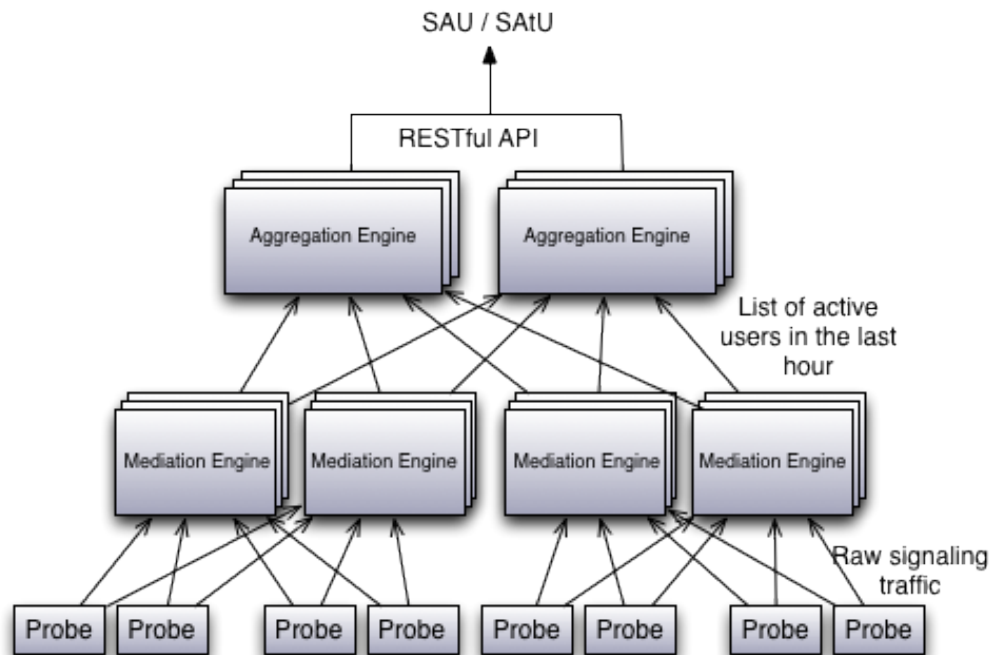
The figure below depicts the Session Monitor architecture in a smaller deployment that does not use the Aggregation Engine.
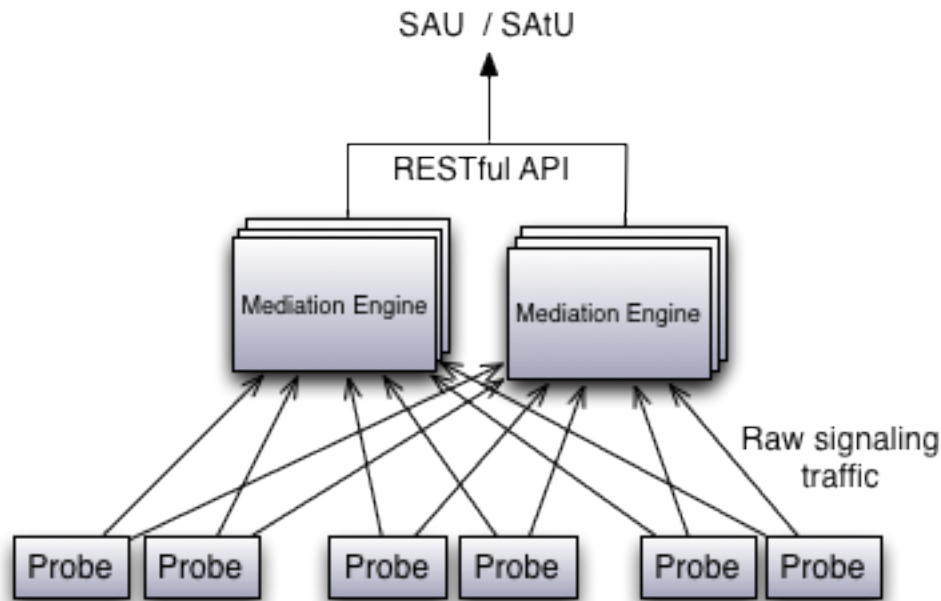
In this case, the RESTful API for exporting the KPIs is served directly by the ME machines (no aggregation step is needed). The same RESTful API is used by both the AE and the ME, so the application that interrogates it doesn't need to know if the Session Monitor setup is using an AE or not.

In all cases, the historical SAU and SAtU values are stored at both the ME and AE level for a period of three months. The historical values are kept for all supported intervals.

About Installing SAU Extension

To install Session Monitor with the SAU and SAtU Extension enabled, follow the instructions in Session Monitor Installation Guide. On all Session Monitor machines, at the configuration step in the Session Monitor Platform Setup Application, select Simultaneously Active Users extension.
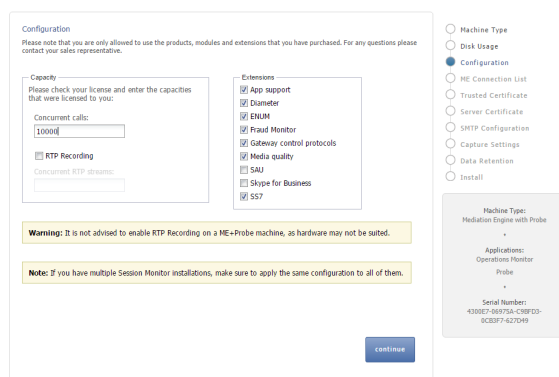
If your setup has an Aggregation Engine (AE) machine, you need to install the Mediation Engine Connector (MEC) on it and configure MEC to connect to all the installed Mediation Engines (MEs).This is because the SAU extension relies on the same communication path used by MEC to communicate with the MEs.

If the Simultaneously Attached Users (SAtU) KPI is required, which is based on Diameter traffic, the Control Plane Monitor (CPM) must be installed on all Mediation Engines. This is because the CPM is measuring the number of attached users. If only the SIP based Simultaneously Active Users KPI is required, CPM is not needed and should not be installed.

Configuring How SAU Phone Numbers Are Selected

For the Simultaneously Active Users KPI, Session Monitor has a flexible mechanism for selecting the phone numbers that are extracted from the SIP headers. This flexibility is needed to ensure that only the phone numbers that respect the SAU definition are taken into account.

This is done by the use of two system settings:

- **Headers to check for SAU KPI:** A space separated list of header field names (max 5) to check for a user part and phone number matching the configured pattern. Order of this list matters; if the first header field is a successful match, the rest of the fields are not checked.

- **Pattern for valid SAU numbers:**The user part of the URI (typically the phone number) of the selected header fields from the SIP INVITE and MESSAGE messages is matched against this regular expression and if there is a match the user part is counted into the SAU count.

> **Note:**
>
> In addition to matching the pattern as defined above, the phone number must be registered from the SIP point of view.
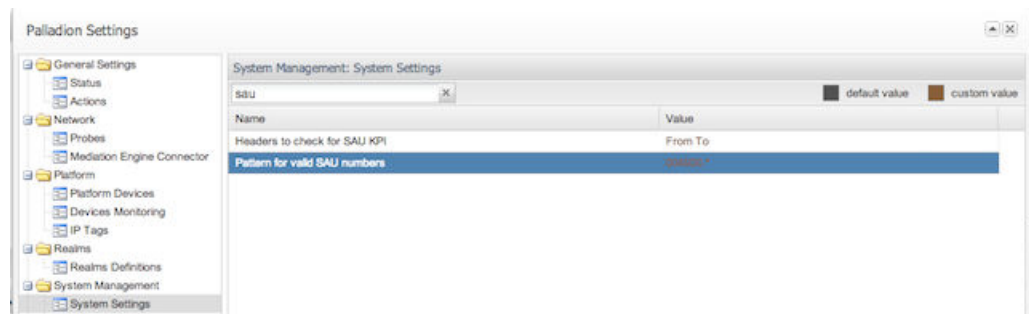
3

RESTful API

This chapter describes using the RESTful APIs to export historical Simultaneously Active User (SAU) and Simultaneously Attached User (SAtU) values.

Using RESTful API to Export SAU and SAtU Values

The historical SAU and SAtU values can be exported by the use of RESTful APIs published at both the Mediation Engine (ME) layer and at the Aggregation Engine (AE) layer (by the Mediation Engine Connector product). The ME and the AE and MEC interfaces are identical, only the prefix is different.

For the ME, the API is found under: `http://<ip-address>/me/r/sau/`

For the MEC (on the Aggregation Engine), the API is found under: `http://<ip-address>/mec/r/sau/`

> **Note:**
>
> Session Monitor RESTful APIs are available both over HTTP and HTTPS. Oracle recommends always using the HTTPS version, except for the development and testing phase.

The Session Monitor RESTful APIs uses digest authentication with the admin user credentials. See Operations Monitor User's Guide for the default value of the admin password and for how to change the password for production use.

1. Query with a GET request the /r/sau resource returns the URIs for the two types of KPIs (Simultaneous Active Users and Simultaneous Attached Users).

   For example, on an ME machine.

   ```
   curl --digest -u admin:secret 'http://<ip-address>/me/r/sau'
   ```

   returns:

   ```
   {
     "supported":
     {
       "sim_active_users": "r/sau/active",
       "sim_attached_users": "r/sau/attached"
     }
   }
   ```

Retrieving the SAU and SAtU Values

SAU KPI values are available under the **r/sau/active/** URL path. The SAtU KPI values are available under the **r/sau/attached/** URL path. Querying these HTTP resources with a GET request without any parameters will return the available one hour intervals for the last 24 hours.

```
curl --digest -u admin:secret 'http://<ip-address>/me/r/sau/active'
```

returns:

```
{
    "data": [
        {
            "duration": 3600,
            "errors": {
                "-1": []
            },
            "kpi_name": "sim_attached_users",
            "start_interval": 1374303600,
            "value": 210888
        },
        {
            "duration": 3600,
            "errors": {
                "-1": [
                    {
                        "error_code": "write_fails",
                        "failed_writes_count": 10
                    }
                ]
            },
            "kpi_name": "sim_attached_users",
            "start_interval": 1374307200,
            "value": 234320
        }
    ]
}
```

> **Note:**
>
> If a "404 Not supported" error is returned when calling this URL, it means that the SAU module is not enabled. It needs to be enabled in the setup configuration wizard (see "About Installing SAU Extension").

In the response, the success key indicates whether the API call was successful or if there were any exceptions while retrieving the values.

The data key is an array of JSON objects, each representing a measured interval and its properties and its value. Each of the JSON objects has the following keys:

- **kpi_name:** Valid values are **sim_active_users** for the Simultaneous Active Users KPI and **sim_attached_users** for the Simultaneous Attached Users KPI.

- **start_interval:** The Unix timestamp (UTC) of the start of the interval.

- **duration:** The interval duration. Valid values are the following multiples of 3600: 3600, 7200, 10800, 14400, 21600.

- **value:** The value of the KPI for the given interval in number of users.

- **errors:** A JSON object that has as keys the IDs of all the Mediation Engines from the setup and as values arrays with the errors objects detected during the interval. In case the query is executed against a Mediation Engine (typically in a setup with the Aggregation Engine), a single key is present "-1", representing the local ME. The error objects always contain a key called error_code with one of the following values:

- **discontinuous:** There was an error calculating this interval. The result might be smaller than in reality.

- **unfinished:** There was an error calculating this interval. The result might be smaller than in reality.

- **overload:** The Session Monitor system was overloaded during this interval. Results may be inaccurate.

- **write_fails:** There was a problem writing the KPI to the database. The result can be missing.

- **late_packets:** The packets from the interval have arrived after the interval was finished and were not included in the KPI calculation. The KPI value might be slightly smaller than in reality. Session Monitor includes in the current interval packets that are late at most 20 seconds. Packets that are processed by the Session Monitor system more than 20 seconds after the interval has finished but less than 5 minutes after the interval is finished are not taken into account for the SAU calculation and will result in this error. Packets that are processed more than 5 minutes after the interval is finished are completely ignored.

The **r/sau/active** and **r/sau/attached** resources support the following filtering parameters, which are passed as GET query parameters. If more parameters are specified in the query, they are combined with a logical AND.

- **start:** Only interval objects with a **start_interval** value bigger than or equal to this parameter will be returned. The accepted format is UTC unix timestamp (seconds since the epoch).

  Default value is the current UTC timestamp minus 24 hours.

- **stop:** Only interval objects with a **start_interval** value strictly smaller than this parameter will be returned. The accepted format is UTC unix timestamp (seconds since the epoch).

  Default value is the current UTC timestamp.

- **interval_length:** Only interval objects with the duration key equal with this parameter will be returned. Only the following values are supported: 3600, 7200, 10800, 14400, 21600.

  Default value is 3600.

> **Note:**
>
> The currently measured interval is available by use of the RESTful API at most 30 minutes after it is finished.

Example Queries

The following are examples of queries to retrieve SAU and SAtU values.

Query the One Hour Intervals

Query URL:

```
http://<ip-address>/me/r/sau/active?
interval_length=3600&start=1375221600&stop=1375232400
```

Response:

```
{
    "data": [
        {
            "kpi_name": "sim_active_users",
            "duration": 3600,
            "start_interval": 1375221600,
            "errors": {
                "-1": []
            },
            "value": 310
        },
        {
            "kpi_name": "sim_active_users",
            "duration": 3600,
            "start_interval": 1375225200,
            "errors": {
                "-1": []
            },
            "value": 422
        },
        {
            "kpi_name": "sim_active_users",
            "duration": 3600,
            "start_interval": 1375228800,
            "errors": {
                "-1": []
            },
            "value": 731
        }
    ]
}
```

Query the Three Hour Intervals from the Same Time Period

Query URL:

```
http://<ip-address>/me/r/sau/active?
interval_length=10800&start=1375221600&stop=1375232400
```

Response:

```
{
    "data": [
        {
            "kpi_name": "sim_active_users",
            "duration": 10800,
            "start_interval": 1375221600,
            "errors": {
                "-1": []
            },
            "value": 1463
        },
        {
            "kpi_name": "sim_active_users",
            "duration": 10800,
            "start_interval": 1375225200,
            "errors": {
                "-1": []
            },
            "value": 1273
        },
```

```
            {
                "kpi_name": "sim_active_users",
                "duration": 10800,
                "start_interval": 1375228800,
                "errors": {
                    "-1": []
                },
                "value": 1101
            }
        ],
        "success": true
}
```

> **Note:**
>
> When querying intervals larger than one hour, the API returns one interval for each hour. For example, in the response above, the interval acts as a rolling window of 3 hours, moving in steps of an hour at a time. This is expected and meant to give the application more flexibility in selecting which hour boundaries to select.