

Using Oracle Solaris 11.4 StatsStore and System Web Interface



E56520-01
November 2020



Using Oracle Solaris 11.4 StatsStore and System Web Interface,

E56520-01

Copyright © 2018, 2020, Oracle and/or its affiliates.

Primary Author: Cathleen Reiher, Alta Elstad

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Copyright © 2018, 2020, Oracle et/ou ses affiliés.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, la documentation du logiciel, les données (telles que définies dans la réglementation "Federal Acquisition Regulation") ou la documentation qui l'accompagne sont livrés sous licence au Gouvernement des États-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des États-Unis, la notice suivante s'applique :

UTILISATEURS DE FIN DU GOUVERNEMENT É.-U. : programmes Oracle (y compris tout système d'exploitation, logiciel intégré, tout programme intégré, installé ou activé sur le matériel livré et les modifications de tels programmes) et documentation sur l'ordinateur d'Oracle ou autres logiciels Oracle Les données fournies aux utilisateurs finaux du gouvernement des États-Unis ou auxquelles ils ont accès sont des "logiciels informatiques commerciaux", des "documents sur les logiciels informatiques commerciaux" ou des "données relatives aux droits limités" conformément au règlement fédéral sur l'acquisition applicable et aux règlements supplémentaires propres à l'organisme. À ce titre, l'utilisation, la reproduction, la duplication, la publication, l'affichage, la divulgation, la modification, la préparation des œuvres dérivées et/ou l'adaptation des i) programmes Oracle (y compris tout système d'exploitation, logiciel intégré, tout programme intégré, installé, ou activé sur le matériel livré et les modifications de ces programmes), ii) la documentation informatique d'Oracle et/ou iii) d'autres données d'Oracle, sont assujetties aux droits et aux limitations spécifiés dans la licence contenue dans le contrat applicable. Les conditions régissant l'utilisation par le gouvernement des États-Unis des services en nuage d'Oracle sont définies par le contrat applicable à ces services. Aucun autre droit n'est accordé au gouvernement américain.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Oracle®, Java, et MySQL sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut être une marque appartenant à un autre propriétaire qu'Oracle.

Intel et Intel Inside sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Epyc, et le logo AMD sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité et excluent toute garantie expresse ou implicite quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Contents

Using This Documentation

Product Documentation Library	vii
Feedback	vii

1 Using Oracle Solaris Observability Tools

Features of Oracle Solaris StatsStore and System Web Interface	1-1
--	-----

2 Using Oracle Solaris System Web Interface

Opening System Web Interface	2-1
Using System Web Interface to Diagnose Problems	2-1
Showing More Information About a Data Point	2-3
Viewing Different Aspects of the Same Data	2-4
Using the Partition Menu	2-4
Using the Drill Down Menu	2-6
Comparing Multiple Statistics	2-7
Comparing Different Statistics Over the Same Time Period	2-7
Comparing the Same Statistic Over Different Time Periods	2-8
Oracle Database Sheet	2-9
Configuring an Oracle Database Sheet	2-10
Using the statcfg Utility	2-10
Additional User Configuration for Oracle Database Instances	2-11
Adding an Oracle Database Sheet	2-12
Removing an Oracle Database Sheet	2-12
Troubleshooting Configuring an Oracle Database Sheet	2-12

3 Using Command Line Interfaces

Command Line Interfaces for Retrieving Statistic and Event Data	3-1
Displaying Resource and Statistic Slices	3-2
Displaying Multiple Resources Using Wildcards and Slice Notation	3-2
Displaying Multiple Statistics Using Wildcards and Slice Notation	3-3

Displaying Values of Statistic Slices	3-3
Displaying Values of Statistic Partitions	3-4
Displaying Values of Static Partitions	3-4
Displaying Values of Dynamic Partitions	3-6
Formatting Event Information	3-7
Using the sstore Command in Interactive Mode	3-8
Using Collections	3-10

4 Performing Operations on Statistic Values

Minimum and Maximum Value	4-1
Average Value	4-3
Sum, Difference, Product, and Quotient	4-4
Rate and Delta	4-5
Resource Utilization	4-5
Utilization of a Resource that is Not a Partition	4-6
Utilization of a Partition Resource	4-7
Top Values	4-7
Top Average Values	4-8
Sub-Sort by Maximum Values	4-9
Sub-Sort by Value Variance	4-10
Top Values of a Partition Resource	4-10
Unit Conversion	4-12
Match Values	4-13
Number of Data Points	4-14
Value Change Points	4-14

A Statistics Store Identifiers

Statistics Store Identifier Structure	A-1
Representing Statistics	A-2
Resource Identifiers	A-2
Statistic Identifiers	A-3
Using Statistics Store Identifier Aliases	A-4
Representing Topology	A-5
Representing Events	A-6
Representing Sets of Statistics and Events	A-7

B Administering StatsStore and System Web Interface

Saving Data for Future Analysis	B-1
Customizing System Web Interface	B-1

Global Preferences Properties Files	B-2
Global Preferences Service Properties	B-2
Allowing Alternative System Web Interface URLs	B-2
Setting the Landing Page	B-3
Individual User Customizations	B-3
Troubleshooting Retrieving Statistics	B-4
Ensure Services Are Online	B-4
StatsStore Services	B-4
System Web Interface Service	B-4
Resolving System Web Interface Problems	B-4
Ensure the Certificate and Key are Installed	B-5
Degraded Performance	B-6
No Matches Found for Statistic	B-7
SSID Does Not Exist	B-7
SSID is Incorrectly Specified	B-7
SSID is Unbrowsable or Unstable	B-8
Size of the StatsStore Repository	B-9
Pruning Data from the StatsStore Repository	B-9
Changing the StatsStore Repository Size Limit	B-10
Run in Debug Mode	B-11
Statistics Store Authorizations and Administrative Profiles	B-11
Configuring System Web Interface Listening Ports	B-12
Redirecting Listen Requests	B-12
Adding Listening Ports	B-12

Index

Using This Documentation

Product Documentation Library

Documentation and resources for this product and related products are available at <http://www.oracle.com/pls/topic/lookup?ctx=E37838-01>.

Feedback

Provide feedback about this documentation at <http://www.oracle.com/goto/docfeedback>.

1

Using Oracle Solaris Observability Tools

Features of Oracle Solaris StatsStore and System Web Interface

StatsStore and System Web Interface provide statistic values and event information from a variety of system and application sources to help you diagnose system problems.

StatsStore unifies the broad set of Oracle Solaris observability technologies under one set of naming rules so that you can more easily compare and operate on the statistics.

System Web Interface guides you through health and performance analysis to problem diagnosis. System Web Interface enables you to view historical information about system performance, contrast that historical performance with current performance, and correlate statistics and events from multiple sources.

A common, historically preserved view of system statistics and events simplifies diagnosis of performance issues by relating varied performance data to system events such as Fault Management Architecture (FMA) faults and system configuration changes such as Service Management Facility (SMF) and audit events. A shared namespace across statistics and events enables you to easily explore all available information for a given system.

Built-in statistical operations enable you to easily manipulate the stored data. For example, you can request the maximum value or rate of change of a statistic value.

StatsStore provides the following capabilities:

- Provides a common namespace, CLI, and API to access consolidated performance data and events from a variety of system sources.
- Consolidates data from a variety of providers.
 - The `kstat` command captures kernel statistics.
 - The `proc` tools capture process statistics.
 - System problems are captured through FMA faults and alerts.
 - Administrative changes are captured through audit events.
 - System-provided DTrace scripts provide more detailed tracing information.
 - Applications included with Oracle Solaris and other Oracle applications provide their own statistics.
 - ISV and custom applications can provide their own statistics.
- Uses a common time source to provide a consistent view of that data.
- Enables you to view historical data in addition to current data.
- Automatically records commonly used system statistics and events.
- Enables you to record data persistently or for a specified period of time.

- Provides data sharing: Data recorded by anyone can be retrieved by everyone.
- Includes built-in statistical operations.

System Web Interface provides the following additional capabilities:

- Easily compare current and historical data to visualize anomalies and trends.
- Display multiple statistics and events on the same graph or the same page.
- View related statistics side-by-side.
- Overlay audit and system events on other data.
- Select a particular time period or data point and show more detail.
- View different aspects of the same data such as utilization rate of a resource and largest users of the resource.

2

Using Oracle Solaris System Web Interface

Opening System Web Interface

1. On an Oracle Solaris 11.4 system named *host-name*, make sure the `webui-server` package is installed:
2. Make sure the `svc:/system/webui/server:default` and `svc:/system/sstore:default` services are online:

```
$ pkg list webui-server
```

```
$ svcs webui/server sstore
```

3. Open the following location in a browser:

```
https://host-name:6787/solaris/
```

By default, System Web Interface listens on port 6787. To listen on a different port, see [Configuring System Web Interface Listening Ports](#). An administrator can allow the use of shorter URLs, as described in [Allowing Alternative System Web Interface URLs](#).

A self signed certificate is automatically generated at start time. Because the certificate is self signed, the certificate might fail validation. Follow the instructions from your browser to add an exception to complete this connection. To configure your own SSL certificate chain, see [Ensure the Certificate and Key are Installed](#).

4. Log in as any Oracle Solaris user that is configured to log into *host-name*.

By default you are on the Solaris Dashboard page. See [Setting the Landing Page](#) to set the default landing page to the Solaris Analytics page.

If this is the first time you have logged into System Web Interface for *host-name*, the Welcome page of the online help is displayed. Display the online help at any time by clicking the Help option at the top of the user name menu.

To set preferences, see [Customizing System Web Interface](#).

Using System Web Interface to Diagnose Problems

The System Web Interface Dashboard provides a single page overview of the status of a single Oracle Solaris instance. Select the name of the system to display information about that system configuration. Select the Faults & Activity button to show FMA faults and alerts and system configuration changes such as SMF and audit events.

The Dashboard displays visualizations for key components of the system. These visualizations show current and recent performance and system faults and other events that are related to the statistics on the visualization. Each visualization has four subsections that show the data values for the following time ranges: last 7 days, last 24 hours, last hour, and last minute. Hover your mouse pointer over the data to display a pop-up with the name, time stamp, and value of that data point. Related events are marked on the visualizations by blue triangular icons. Hover your mouse pointer over a triangle icon to display a pop-up with

specific information about that event or alert. For example, a graph that shows a spike in CPU utilization might have an event icon at that same point that shows a CPU faulting or being taken offline.

The information on the Dashboard helps you quickly identify any anomalies, which guides you to which system resources to examine to diagnose potential problems. Past system resource use is often a good indicator of whether the current behavior is anomalous.

If any graph looks anomalous, or if a resource in the graph is specifically interesting, select that graph to open a sheet with more visualizations with related statistics and events. These sheets have sections, one or more groups in each section, and one or more visualizations in each group. Sections, groups, and visualizations have descriptions of the information that is displayed, possible causes of any anomalies, and references to other sheets and visualizations that might be helpful.

To review data on other sheets, select Oracle Solaris Analytics from the Applications menu, and select the thumbnail of the sheet you want to view. The online help explains how to use the filtering options at the top of the page or mark a sheet as a favorite to more quickly locate a sheet.

If the time of the data values that you want to investigate is not shown on the new sheet, use the arrow buttons or time range button at the top of the sheet to adjust the view. To hold the display at that time range while you investigate, select the Pause All button at the top of the sheet.

Many of the predefined visualizations report utilization, saturation, and errors of key system resources.

- Utilization is expressed as the currently achieved percentage of theoretical maximum throughput.
- Saturation is the amount of work queued: work that is not being actively serviced. Saturation is sometimes expressed as a multiple of the maximum utility (capacity).
- Errors are expressed as occurrences per second.

Other visualizations show these key metrics across all available resources for that measurement. Viewing all resources can provide a greater appreciation of the state of a particular resource. For example, high CPU utilization might be correlated with a backup event; you might see from historical data that similar utilization occurs every day at this time. Alternatively, high CPU utilization might be correlated with a fault event or other error, resulting in overload of remaining resources.

In addition to averages across resources, individual resource utilization is shown for many resources, enabling you to more effectively evaluate resources that are allocated directly to particular workloads. For example, an individual CPU might be allocated directly to a zone, and individual NICs are allocated to distinct networks.

Visualizations also provide partitioned views of statistics to quickly identify whether individual resources are seeing excessive utilization, saturation, or errors. [Viewing Different Aspects of the Same Data](#) shows an example of using a partitioned statistic to determine which applications are responsible for most of the network traffic on a system. Similarly, in addition to overall CPU usage data, you can select the `zone` partition of CPU usage data to determine which zones are using the most CPU. One CPU might be overused because of one particular zone.

A single graph can show event as well as statistic data. Events might include administrative changes to relevant resources such as configuring a new datalink, or FMA events such as a system service transitioning into the `maintenance` state.

Showing More Information About a Data Point

To show the value of a statistic at a particular time on a line graph visualization, hover over that point on the graph. The exact time and value display in a pop-up box.

The following actions enable you to more easily choose points of interest:

- Pause the data scrolling.
Value recording continues while scrolling is paused. When you resume data scrolling, you can use the time range arrow buttons to review data values that were recorded while the visualization was paused.
- Use the data cursor.
The data cursor uses a line and dot to highlight the selected point on the graph.
- If multiple lines are shown close together on the graph, show only the line of interest.

You can pause data scrolling and show the data cursor in one step: When you select Show Data Cursor from the Sheet Actions menu, all visualizations on the sheet are paused and the data cursor is displayed at the same time point on every visualization as you move your cursor across any visualization.

Selecting the left or right time range arrows pauses the data scrolling for only that visualization. The visualization stays fixed on the selected time range. You could also select Pause All at the top of the sheet.

To zoom the Y axis, click a data point.

To show just one line when more than one statistic or partition part is listed in the legend, do one of the following:

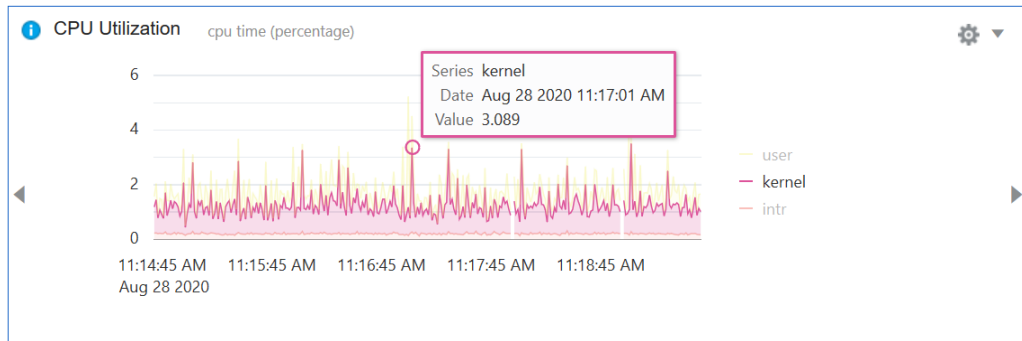
- Double-click on that line in the graph and select "Show only" from the pop-up menu.
- Click the item you want in the legend and select "Show only" from the pop-up menu.

For example, in the legend shown in the following figure, select "kernel," and then select "Show only kernel."

When you hover over a point on the graph, a pop-up displays that shows the name of the statistic or partition part that appears in the graph legend, the time, and the value of the statistic at that time.

The following figure shows the information that appears when you hover over a data point: statistic name, time, and value at that point.

Hovering Over a Data Point on a Graph



Viewing Different Aspects of the Same Data

If the statistic is a partitioned statistic, you can select a particular partition to view. You can continue to select a partition of a partition, also known as drilling down.

Using the Partition Menu

1. Select the Visualization Actions menu.
2. Select the Partition... option.
3. Click in "Click to show available partitions" to show the list of partitions.
4. Select the partition you want to view.

The following figure shows the Partition option on the Visualization Actions menu.

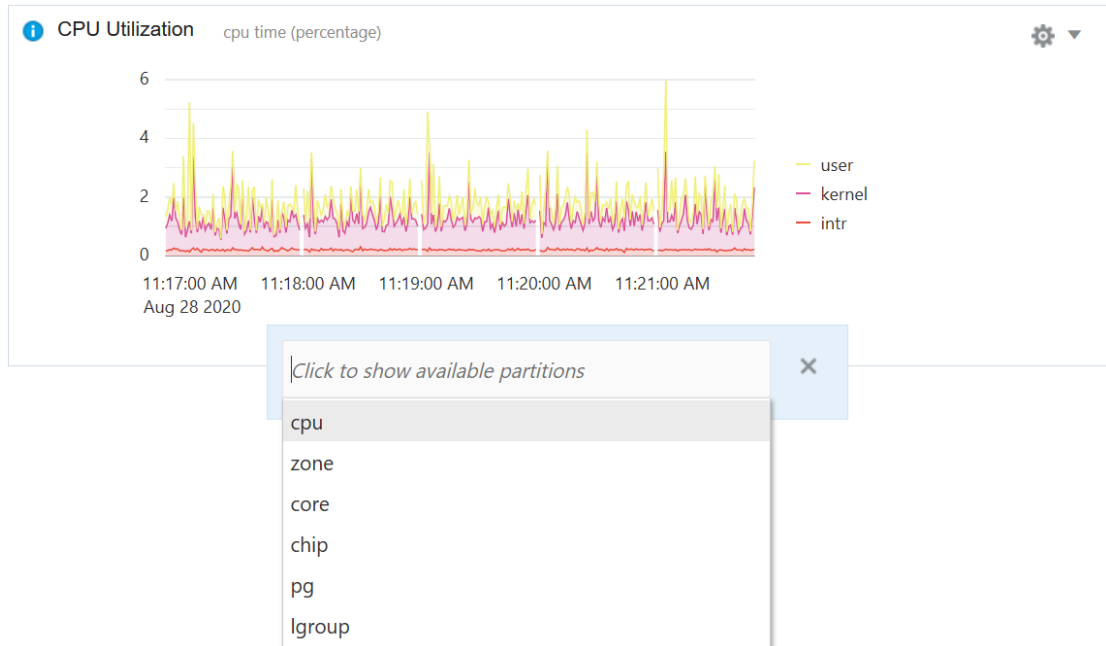
Partition Option on the Visualization Actions Menu



Selecting the Partition option causes the available partitions list to display in collapsed form. To expand the list, select the collapsed list where indicated.

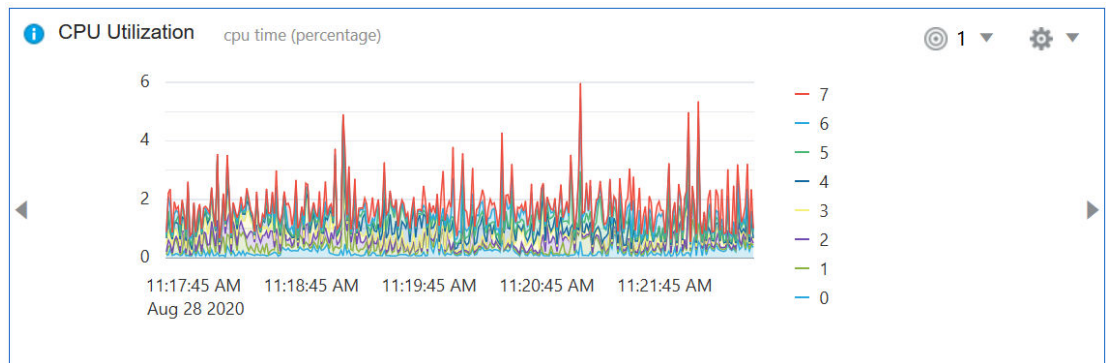
The following figure shows the selections on a sample Partition list.


Sample Partition List Selections



In the following figure, the user has selected the `cpu` partition from the list in the preceding figure. The graph shows one line for each CPU in the system. In both figures the statistic being shown is the percentage of CPU time used. The preceding graph showed the percentage of CPU time used by the kernel, by users, and by interrupts. The following graph shows the percentage of CPU time used by each CPU.

Viewing a Different Partition of the Data



The visualization also gained a new menu: the Undo History menu . The Undo History menu enables you to undo or save partitioning changes. The number in the Undo History menu title shows how many times you have partitioned this statistic. You can roll back to the previous partitioning, roll back to the original statistic, or roll back to an intermediary series of partitioning.

The Undo History menu contains the following options:

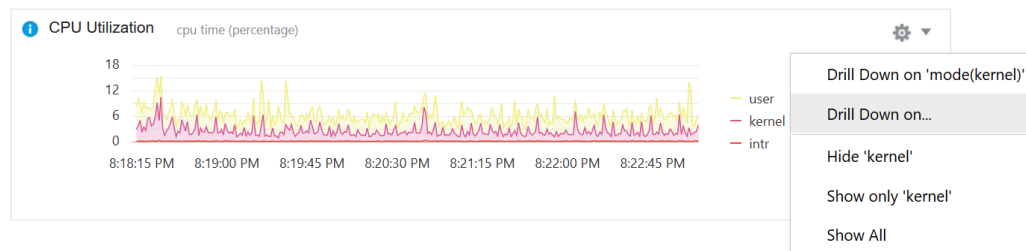
- Undo All. A tree structure of each partitioning is shown so that you can undo all or roll back to anywhere in the series of partitioning.
- Undo the current partitioning. A brief description of the previous partitioning is shown as the menu option so that you know what you are rolling back to. For example, in the previous visualization, you could roll back from reporting utilization per CPU to reporting CPU utilization for user, kernel, and interrupts.
- Save All to Visualization
- Save All as New Visualization

Using the Drill Down Menu

When you double-click on a line graph or click an item in a graph legend for a visualization that is showing a partitioned statistic, the drill-down menu is displayed.

In the following figure, the user selected "kernel" in the legend.

Drill Down Menu Options



In the previous figure, the user selected "Drill Down on" to display the choices shown in the dialog box in the following figure. If the list of choices is long, use the "Type to filter" field to show fewer choices. The Clear button clears the filter field and shows all choices. In this example, as soon as the user selected "user," the graph changed to show only the user part of the mode partition of the CPU utilization statistic. The dialog box remains displayed so that you can select additional partitions or change your selections. Click the Done button to close the dialog box.

Drill Down On Options

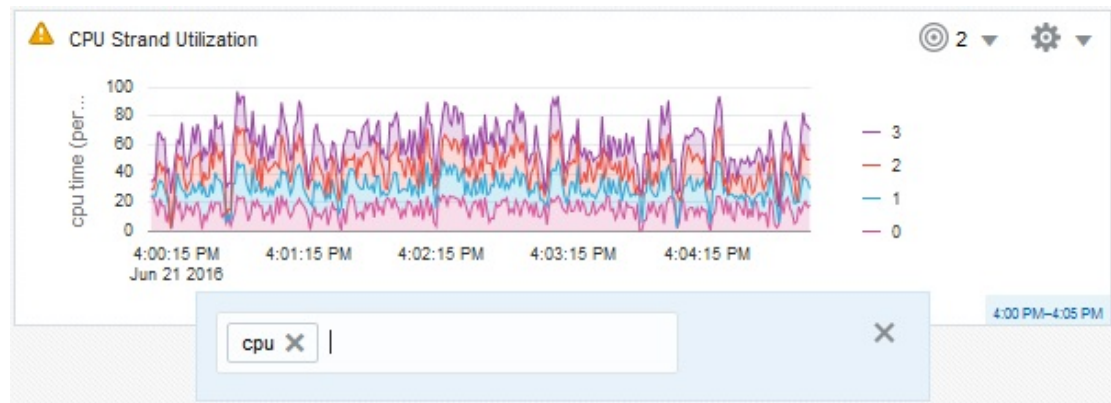


The user selected Done on the dialog box shown in the preceding figure, and then took one of the following paths to drill down further:

- Selected Partition from the Visualization Actions menu, and selected “cpu” from the list of available partitions.
- Selected "user" in the visualization legend, selected "Drill Down on" user from the Drill Down menu that displayed, and selected "cpu" from the list of available partitions.

The following figure shows the list of choices for drilling down another level.

Second Level of Drill Down



The Undo History menu changed to show a “2” because the user has drilled down two times and can undo two levels of drill down:

- Drilled down on `user` from the display of all parts of the `mode` partition.
- Drilled down on the `cpu` partition.

The preceding figure shows the percentage of CPU time that is `user` time on each CPU in the system.

Comparing Multiple Statistics

You might want to compare values for the same statistic at different time periods or compare different statistics over the same time period. In both cases you need to set the time range to display in each visualization being compared, and you might want to move the visualizations, for example to be vertically aligned. To compare the same statistic at different time periods, you need to create a copy of the visualization. When comparing different statistics over the same time period, you might want to show the data cursor.

Comparing Different Statistics Over the Same Time Period

To compare values of different statistics over the same time period, first set the time range for all visualizations to be compared.

To set the same time range for all visualizations on a sheet, select the time range shown between the arrows at the top of the sheet. Selecting that time range displays the Set Time Range dialog.

- Set the period. Use the Period menu to select the amount of time that will be shown in the width of the graph. Choices range from five minutes to one year.

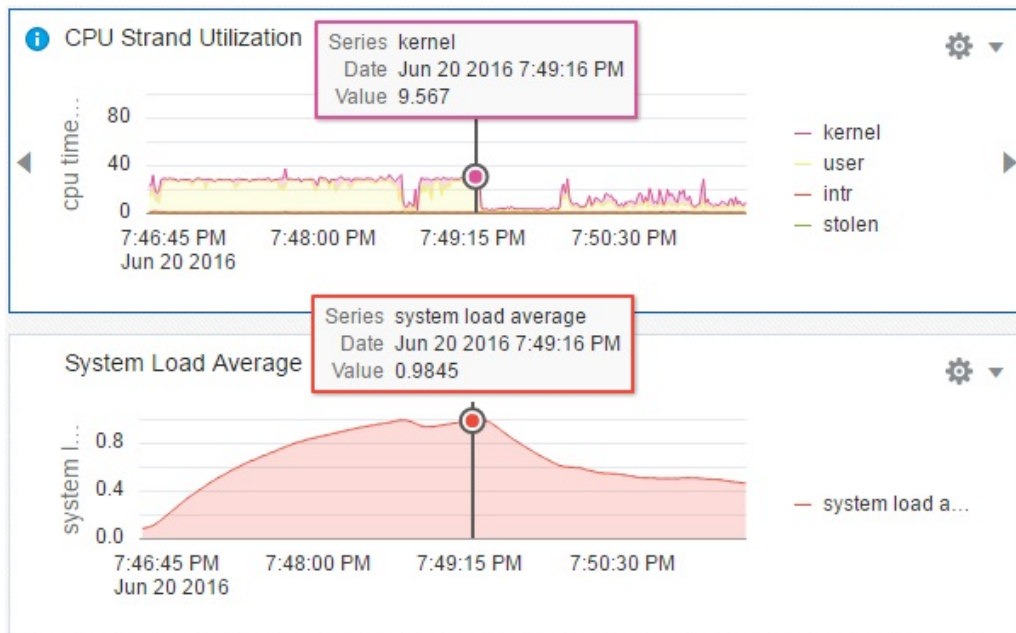
- Set the start or end time of the period.
 - If you select the Time button, select either Start Time or End Time and use the calendar and the clock to specify that start or end time for the selected period. Data scrolling is stopped.
 - Alternatively, select the Ending Now button. Data scrolling is not stopped.
- When you select Set, all visualizations on the sheet adjust to show the time range you specified.

If you want to rearrange the visualizations, select the Move option on a Visualization Actions menu, and select the left and right arrows to move the visualization.

You might want to select Show Data Cursor from the Sheet Actions menu, as shown in the following figure.

The following figure shows an example of two visualizations of different statistics with the same time period set and the data cursor displayed.

Comparison of Different Statistics Over the Same Time Period



Comparing the Same Statistic Over Different Time Periods

To compare values of the same statistic over different time periods, set the time range one visualization, and then set a different time range for a copy of the visualization.

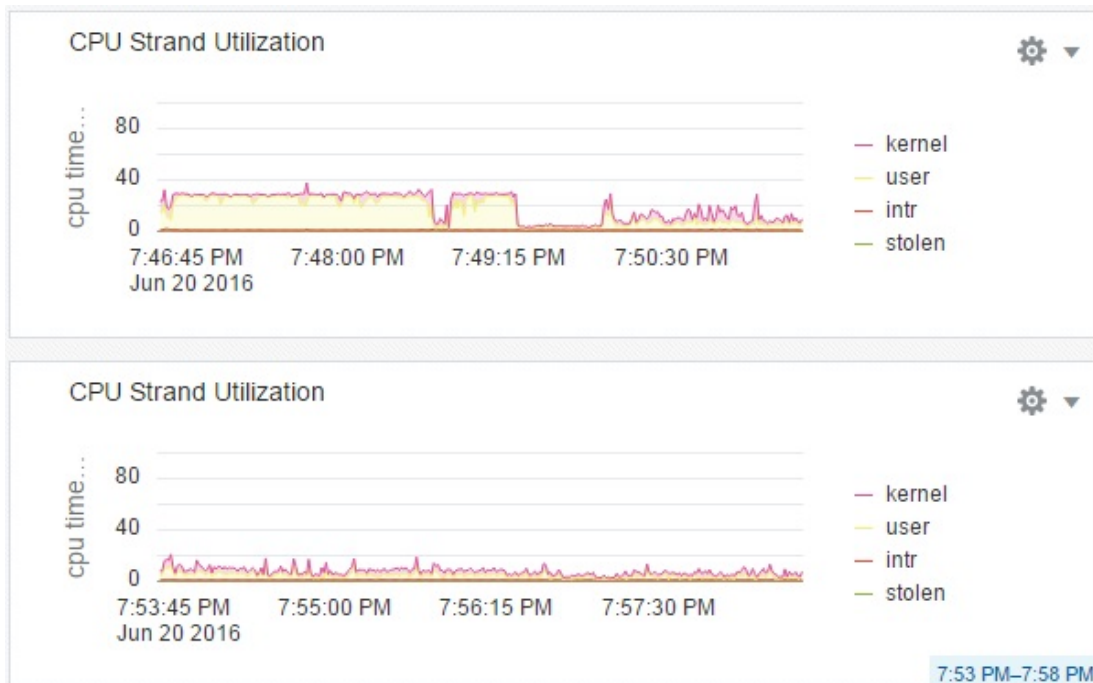
To set the time range for a single visualization, select the Set Time Range option on the Visualization Actions menu. The Set Time Range dialog is displayed. When you select Set, the visualization is adjusted to show the specified time period.

To create a copy of the visualization with a different time range displayed, select the Compare option on the Visualization Actions menu. The Set Time Range dialog is displayed. When you select Set, a copy of the visualization is created that shows the time range you selected.

If you want to rearrange the visualizations, select the Move option on a Visualization Actions menu, and select the left and right arrows to move the visualization.

The following figure shows an example of two visualizations of the same statistic with different time periods set.

Comparison of the Same Statistic Over Different Time Periods



Oracle Database Sheet

If the system is running the Oracle Database, you can use the Oracle Solaris System Web Interface to view a high-level overview of Oracle Database performance and problems on that system. The information provided enables an Oracle Solaris administrator to diagnose whether the source of a problem is in the operating system, network, or storage, or whether a database administrator is required to further diagnose the problem.

Use the Oracle Database sheet to examine data such as the following:

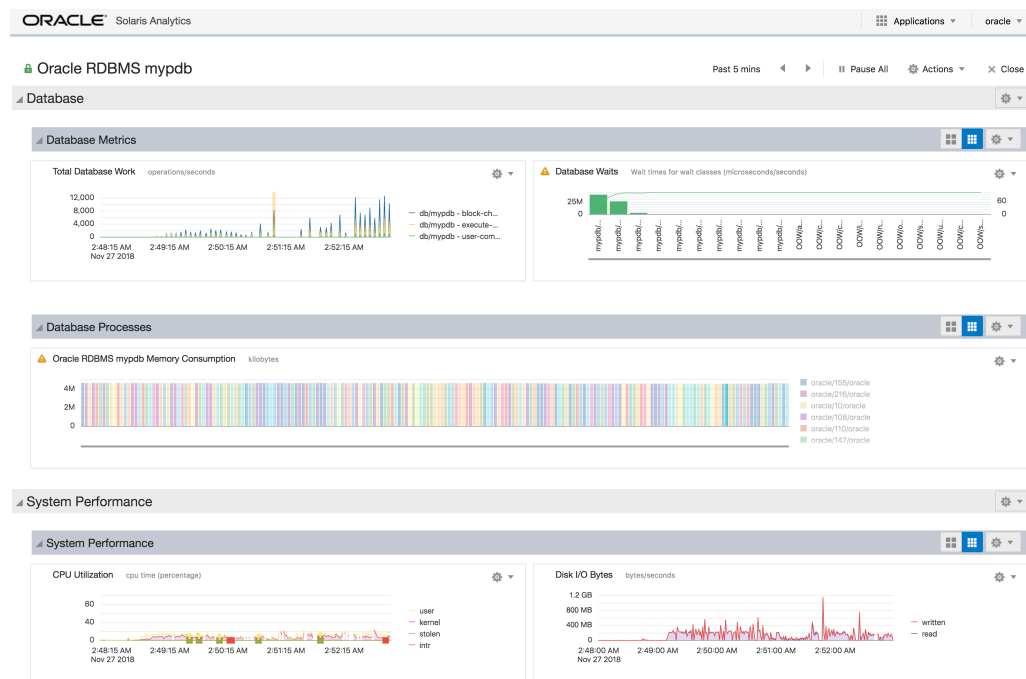
- Number of times per second that the database performs block changes, data retrievals, or user commits.
- Amount of time the database spends waiting due to system events, user application code, and system configuration.

See `V$SYSTEM_EVENT` and `V$SYSSTAT` in "Instance Tuning Using Performance Views" in the database performance tuning guide. For example, see [Oracle Database Performance Tuning Guide 20c](#). In particular, see "Table of Wait Events and Potential Causes."

- Memory consumption of running database processes. Note that memory consumption is in kilobytes. In the graph in the following figure, memory consumption is shown in millions of kilobytes so that the data fits on the graph. Thus "4M" on the y-axis means 4 million kilobytes, or 4 GB.

The following figure shows an example of an Oracle Database sheet. The Oracle Database instance in this example is `mypdb`.

Sample Oracle Database Sheet



The Oracle Database sheet also includes system performance visualizations that are not specific to Oracle Database but are included for convenience. For example, you could correlate a drop in database work or an increase in database wait times with an increase in disk reads and writes from other sources or with a CPU going offline.

Configuring an Oracle Database Sheet

To use the Oracle Database sheet, install the `service/oracle-rdbms-stats` package. The `service/oracle-rdbms-stats` package delivers the following:

- The Oracle Database sheet
- Oracle Instant Client
- The `application/stats/oracle-database-stats` SMF service
- The `statcfg` utility
- The `statcfg(1)` and `rdbms-stat(1)` man pages

Using the `statcfg` Utility

The `statcfg` utility enables the administrator to configure an Oracle Database sheet to report statistics for a specific Oracle Database instance. Until you run the `statcfg` utility, the `oracle-database-stats` SMF service is disabled, and you will not see an Oracle Database sheet.

The `statcfg` utility must be run as the `root` user or by assuming the `root` role.

Before you use the `statcfg` utility, export `ORACLE_HOME` as shown in the following example:

```
# export ORACLE_HOME=/u01/app/oracle/product/19.0.0.0/dbhome_1
```

The following syntax shows how to use the `statcfg` utility with an Oracle Database instance:

```
# /usr/bin/statcfg [add] oracle-rdbms -u user -g group -s sid -c connectstring
# /usr/bin/statcfg remove oracle-rdbms -u user -g group -s sid
```

The `add` subcommand adds an Oracle Database sheet for the `sid` database instance as described in [Adding an Oracle Database Sheet](#). The `remove` subcommand removes the Oracle Database sheet for the `sid` database instance as described in [Removing an Oracle Database Sheet](#).

user, group

The Oracle Solaris user and group that this `oracle-database-stats` service instance will run as. See [Additional User Configuration for Oracle Database Instances](#) for additional configuration required for the user specified by `user`.

sid

The name of the Oracle Database instance that this `oracle-database-stats:sid` service instance will monitor.

connectstring

The alias for access credentials for the `sid` database instance. See “Oracle Wallet” in [Additional User Configuration for Oracle Database Instances](#) for more information.

Use the following command to show a help message about using the `oracle-rdbms` service value:

```
# /usr/bin/statcfg oracle-rdbms -h
```

Additional User Configuration for Oracle Database Instances

Perform the following configuration for the user specified by `user`:

Authorizations

Ensure that `user` has the following authorizations assigned:

- `solaris.sstore.update.res`
- `solaris.sstore.write`

The following example shows how to add these authorizations for `user`:

```
# usermod -A +solaris.sstore.update.res,solaris.sstore.write user
```

SYSDBA

Ensure that `user` has the `SYSDBA` connect privilege for the specified `sid` database instance.

ORACLE_HOME

The shell startup script for `user` must specify `ORACLE_HOME`, or specify `ORACLE_HOME` in the service as shown in the following example:

```
# svccfg -s application/stats/oracle-database-stats:sid \  
setenv ORACLE_HOME /u01/app/oracle/product/19.0.0.0/dbhome_1
```

Oracle Wallet

The `sqlnet.ora` parameter file must specify the Oracle Wallet location for `user`. The `tnsnames.ora` file must specify the Oracle Wallet `db_connect_string` alias for the `sid` database instance.

By default, both `sqlnet.ora` and `tnsnames.ora` are located in the `$ORACLE_HOME/network/admin` directory.

Follow the instructions in “Managing the Secure External Password Store for Password Credentials” in the database security guide to configure the user’s Oracle Wallet. For example, see [Oracle Database Security Guide 20c](#).

Adding an Oracle Database Sheet

The following command adds an Oracle Database sheet that reports statistics related to the `sid` Oracle Database instance.

```
# /usr/bin/statcfg add oracle-rdbms -u user -g group -s sid -c connectstring
```

Run this command again with a different value for `sid` to create an additional Database sheet that reports statistics for that `sid`.

Removing an Oracle Database Sheet

Use the `remove` subcommand to remove the database sheet for the specified database instance.

```
# /usr/bin/statcfg remove oracle-rdbms -u user -g group -s sid
```

Troubleshooting Configuring an Oracle Database Sheet

Make sure the database instance is up and running.

Make sure the user has sufficient privilege to run the `statcfg` command. Run the `statcfg` command as the `root` user or assume the `root` role.

Make sure `ORACLE_HOME` is set in the environment where you run `statcfg`.

Check the state of the `oracle-database-stats:sid` service.

```
$ svcs oracle-database-stats:sid
```

If any of the Oracle RDBMS connections fail to communicate with their respective databases, the service will enter the `degraded` state.

If the service is in `maintenance` or is otherwise not online, check the service log as shown in the following example:

```
$ svcs -xL oracle-database-stats:sid
```

To check the settings of the service instance, use the `svcpprop` command to show the values of the following properties:

- `user`
- `group`
- `db/sid`
- `db/db_connect_string`

For more information about service commands, see [Managing System Services in Oracle Solaris 11.4](#).

3

Using Command Line Interfaces

Command Line Interfaces for Retrieving Statistic and Event Data

The `sstore list` and `sstore info` commands show information about statistics and events. The `sstore capture` and `sstore export` commands record and display statistic values. The `sstore info` command shows metadata. The `sstore export` command shows data. See the [sstore\(1\)](#) man page for syntax and arguments.

Display identifiers

Use the `sstore list` command to list the currently available identifiers (SSIDs) that match the specified pattern. These SSIDs can then be used with other `sstore` subcommands. You can specify a time to list identifiers that are not currently available but were available in the past. If you specify a time, then the output shows the time ranges when the identifier was available for capture. One use of time specification is to determine when resources were available on the system in the past.

Display metadata

Use the `sstore info` command to display metadata information about the specified identifier such as description and data type. As with the `sstore list` command, you can specify a time range to display metadata that was available in the past.

Record and display values

Use the `sstore capture` command to record and display the statistics and event information for the specified identifiers. Specify an interval and count to display values every *interval* seconds for *count* iterations. If you do not specify a count, values are displayed until you enter Ctrl-C or otherwise terminate the process. The default interval is one second. To specify a count, you must also specify an interval.

Values are displayed from now. Values continue to be recorded and stored in the statistics store for five minutes after the most recent request (`sstore capture`) for those values, even though the values might not be displayed. Use the `sstore export` command to retrieve those values.

Display values at a specified time

Use the `sstore export` command to display the statistic and event information for the specified identifiers from any time that information for those identifiers is available in the statistics store. If you do not specify a time range or a start time and a number of data points, all data in the statistics store for the given identifiers is displayed. Because system configuration changes, the identifiers for which you want information might not exist now. Use the `sstore list` command with a time range to find identifiers for resources that existed in the past, and then use `sstore export` with that same time range to retrieve the data for those resources.

The examples shown in the remainder of this section demonstrate these commands. Depending on your environment, you might need to escape some characters in an SSID, such as / or (.

Displaying Resource and Statistic Slices

Wildcards and slice notation enable you to specify multiple resources or statistics in one SSID.

Displaying Multiple Resources Using Wildcards and Slice Notation

To specify multiple resources with one SSID, use the * wildcard or the s (slice) keyword. Both of the following commands show the same result:

```
$ sstore list //:class.cpu//:res.id/*
$ sstore list //:class.cpu//:res.id//:s.[*]
IDENTIFIER
//:class.cpu//:res.id/0
//:class.cpu//:res.id/1
//:class.cpu//:res.id/2
//:class.cpu//:res.id/3
```

The following command selects a subset of resources:

```
$ sstore list //:class.cpu//:res.id//:s.[1,3]
IDENTIFIER
//:class.cpu//:res.id/1
//:class.cpu//:res.id/3
```

The slice notation is particularly helpful if the resource namespace is numeric and sequential. The following command selects a range of resources:

```
$ sstore list //:class.cpu//:res.id//:s.[1:3]
IDENTIFIER
//:class.cpu//:res.id/1
//:class.cpu//:res.id/2
//:class.cpu//:res.id/3
```

The following command shows resources with non-numeric names:

```
$ sstore list //:class.disk//:res.name/*
IDENTIFIER
//:class.disk//:res.name/sd0
//:class.disk//:res.name/sd1
//:class.disk//:res.name/sd10
//:class.disk//:res.name/sd11
...
```

The following command selects a subset of resources with non-numeric names:

```
$ sstore list //:class.disk//:res.name//:s.[sd2*,sd7*]
IDENTIFIER
//:class.disk//:res.name/sd2
//:class.disk//:res.name/sd20
...
//:class.disk//:res.name/sd29
//:class.disk//:res.name/sd7
```

A range cannot specify non-numeric resource names:


```
$ sstore list //:class.disk//:res.name//:s.[sd2:sd7]
Warning (//:class.disk//:res.name/sd2:sd5) - lookup error: no matches found
```

Displaying Multiple Statistics Using Wildcards and Slice Notation

Similar to resources, you can specify multiple statistics with one SSID by using the `*` wildcard or the `s` (slice) keyword. Statistic slice notation is useful for specifying the arguments for an operation as shown in [Performing Operations on Statistic Values](#).

The following example lists all statistics that have `block` in their name:

```
$ sstore list //:class.fs//:res.mountpoint//var/share//:stat.*block*
IDENTIFIER
//:class.fs//:res.mountpoint//var/share//:stat.available-blocks
//:class.fs//:res.mountpoint//var/share//:stat.block-size
//:class.fs//:res.mountpoint//var/share//:stat.free-blocks
//:class.fs//:res.mountpoint//var/share//:stat.total-blocks
```

The following example specifies statistic slices:

```
$ sstore list \
> //:class.fs//:res.mountpoint//var/share//:stat.//:s.[block-size,free-blocks]
IDENTIFIER
//:class.fs//:res.mountpoint//var/share//:stat.block-size
//:class.fs//:res.mountpoint//var/share//:stat.free-blocks
```

Displaying Values of Statistic Slices

The following `sstore capture` command shows the current value of the `in-bytes` and `out-bytes` statistics for the `net0` resource:

```
$ sstore capture //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-bytes] 1
1
TIME                VALUE          IDENTIFIER
2020-08-04T20:45:44 120744284 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:45:44 14960822  //:class.link/phys//:res.name/net0//:stat.out-bytes
```

Though the previous `capture` command only shows one data point, as requested, additional values are recorded. After data recording starts (by using the `capture` command or the `BUI`, for example), recording continues for several minutes.

The following `sstore export` command requests data values at 60-second intervals over a two-minute period:

```
$ sstore export -t 2020-08-04T20:46:00 -e 2020-08-04T20:48:00 -i 60 \
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-bytes]
TIME                VALUE          IDENTIFIER
2020-08-04T20:46:00 132264022.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:47:00 166141381.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:48:00 166147907.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:46:00 15581480.0  //:class.link/phys//:res.name/net0//:stat.out-bytes
2020-08-04T20:47:00 17407359.0  //:class.link/phys//:res.name/net0//:stat.out-bytes
2020-08-04T20:48:00 17415231.0  //:class.link/phys//:res.name/net0//:stat.out-bytes
```

If you use the `-p` option to specify a number of data points, you get the specified number of consecutive data points that were recorded from the specified start time.

```
$ sstore export -t 2020-08-04T20:46:00 -p 3 \
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-bytes]
```

TIME	VALUE	IDENTIFIER
2020-08-04T20:46:00	132264022	//:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:46:01	132828645	//:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:46:02	133476528	//:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-04T20:46:00	15581480	//:class.link/phys//:res.name/net0//:stat.out-bytes
2020-08-04T20:46:01	15611793	//:class.link/phys//:res.name/net0//:stat.out-bytes
2020-08-04T20:46:02	15942307	//:class.link/phys//:res.name/net0//:stat.out-bytes

If you use the `-p` option with a negative argument, you get the specified number of consecutive data points that were recorded prior to the specified start time.

```
$ sstore export -t 2020-08-04T20:46:00 -p -3 \  
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-bytes]
```

For data of type `counter` (the most common type), if values are recorded for several minutes, then recording is stopped, and then recording is started again, `sstore export` shows interpolated values for the period when no data was recorded.

Displaying Values of Statistic Partitions

A *partition* is a grouping of a set of statistic values. Partitions take arguments to specify which partition data to return. The two types of partitions are static and dynamic.

static

A static partition is a full enumeration of the exact names of the partitions. For example, CPU usage can be broken down by mode, which has component parts user, stolen, kernel, interrupt, and idle time. The components of a static partition are explicitly listed in the partition metadata.

dynamic

Dynamic partitions return a different list of constituents depending on the environment in which the statistic is requested. In many cases, dynamic partitions map to resource names, and the partition metadata describes that mapping. This enables zooming out to examine a partition itself, rather than exploring data by infinite repartitioning.

Use the `part` keyword to specify which partition data to return. If you do not specify a partition for a statistic that is partitioned, the aggregate value of all parts of the partition is returned.

Displaying Values of Static Partitions

The following command shows that CPU usage is partitioned by mode (`partitions: mode`):

```
$ sstore info //:class.cpu//:res.id/0//:stat.usage  
Identifier: //:class.cpu//:res.id/0//:stat.usage  
$schema: //:stat  
copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.  
description: cpu time  
id: //:class.cpu//:stat.usage  
multiplier: 1000000  
stability: stable  
type: counter  
units: milliseconds  
zeroes: False  
partitions: mode
```

The following command shows that the `mode` partition is not further partitioned:

```
$ sstore info //:class.cpu//:res.id/0//:stat.usage//:part.mode
Identifier: //:class.cpu//:res.id/0//:stat.usage//:part.mode
  $schema: //:stat
  copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
description: cpu time
  id: //:class.cpu//:stat.usage
multiplier: 1000000
stability: stable
  units: milliseconds
  zeroes: False
  type: partition
```

Example 3-1 Display the Aggregate Value of a Partition

The following command shows that the value of a partition is aggregated if you do not specify a component of the partition:

```
$ sstore capture //:class.cpu//:res.id/0//:stat.usage 1 1
TIME          VALUE          IDENTIFIER
2020-08-27T23:09:56 1503723065.775699 //:class.cpu//:res.id/0//:stat.usage
```

Example 3-2 Display the Value of Each Component of a Partition

The following `sstore export` command uses the time from the previous `sstore capture` command to show the values of the individual components of the `mode` partition:

```
$ sstore export -t 2020-08-27T23:09:56 -p 1 \
> //:class.cpu//:res.id/0//:stat.usage//:part.mode
TIME          VALUE IDENTIFIER
2020-08-27T23:09:56 //:class.cpu//:res.id/0//:stat.usage//:part.mode
                  idle: 1501048573.375829
                  intr: 1143137.286091
                  kernel: 1398988.483238
                  user: 132366.630541
```

The preceding command shows the arguments that the `mode` partition can take (`idle`, `intr`, `kernel`, `user`) and the CPU usage for each component. This output also shows that the aggregate value is equal to the sum of the values of all components.

Example 3-3 Display the Values of Selected Partition Components

The following command specifies the time output from the previous command as the start time and requests CPU usage values at 30-second intervals for just the user and kernel components:

```
$ sstore export -t 2020-08-27T23:09:56 -e 2020-08-27T23:10:56 -i 30 \
> //:class.cpu//:res.id/0//:stat.usage//:part.mode(user, kernel)
TIME          VALUE IDENTIFIER
2020-08-27T23:09:56 //:class.cpu//:res.id/0//:stat.usage//:part.mode(user, kernel)
                  kernel: 1398988.483238
                  user: 132366.630541
2020-08-27T23:10:26 //:class.cpu//:res.id/0//:stat.usage//:part.mode(user, kernel)
                  kernel: 1399010.830562
                  user: 132366.630541
2020-08-27T23:10:56 //:class.cpu//:res.id/0//:stat.usage//:part.mode(user, kernel)
                  kernel: 1399038.108177
                  user: 132366.630541
```

Displaying Values of Dynamic Partitions

The following command shows that CPU interrupt time is partitioned by level:

```
$ sstore info //:class.cpu//:res.id/0//:stat.interrupt-time
  Identifier: //:class.cpu//:res.id/0//:stat.interrupt-time
    $schema: //:stat
    copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
  description: interrupt time
    id: //:class.cpu//:stat.interrupt-time
  multiplier: 1000000
  stability: stable
    type: counter
    units: milliseconds
index-partition: level
partitions: level
```

The following command shows that the level partition is not further partitioned:

```
$ sstore info //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level
  Identifier: //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level
    $schema: //:stat
    copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
  description: interrupt time
    id: //:class.cpu//:stat.interrupt-time
  multiplier: 1000000
  stability: stable
    units: milliseconds
index-partition: level
  type: partition
```

CPU interrupt time levels are defined dynamically. These levels are not defined in metadata.

The following `sstore export` command shows the total interrupt time for all levels and the total broken down by level:

```
$ sstore export -t 2020-08-28T00:02:47 -p 1 \
> //:class.cpu//:res.id/0//:stat.interrupt-time \
> //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level
TIME          VALUE IDENTIFIER
2020-08-28T00:02:47 1145541.536654 //:class.cpu//:res.id/0//:stat.interrupt-time
2020-08-28T00:02:47 //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level
0: 17242.24916
1: 0.374476
2: 1.916197
3: 0.0
4: 8.525565
5: 1112.138469
6: 0.0
7: 0.0
8: 2.595788
9: 789717.736216
10: 0.0
11: 0.0
12: 1516.502436
13: 335933.168499
14: 6.329848
```

The following command shows specifying components of a partition:

```
$ sstore export -t 2020-08-28T00:02:47 -e 2020-08-28T00:03:47 -i 60 \
> //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level(1,9,13)
TIME                VALUE IDENTIFIER
2020-08-28T00:02:47 //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level(1,9,13)
                    1: 0.374476
                    9: 789717.736216
                    13: 335933.168499
2020-08-28T00:03:47 //:class.cpu//:res.id/0//:stat.interrupt-time//:part.level(1,9,13)
                    1: 0.374476
                    9: 789747.46965
                    13: 335946.036757
```

Formatting Event Information

In the following examples, an administrator has enabled the `svc:/network/smtp:sendmail` service. In the examples, the command output is shown 80 characters wide, even when the actual output is a single line.

The `fmt.label` format shows the label `AUE_smf_enable` in the value column of the output.

```
$ sstore export \
> //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action//:fmt.label
TIME                VALUE IDENTIFIER
2015-08-30T00:31:14 ('AUE_smf_enable',) //:class.svc//:res.fmri/network/smtp:sen
dmail//:event.adm-action//:fmt.label
```

The `fmt.description` format shows a description of the event in the value column, including that the action was enabling a service, the `solaris.smf.modify` authorization was required, and the value of the `general/enabled` property of the `svc:/network/smtp:sendmail` service was changed.

```
$ sstore export \
> //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action//:fmt.description
TIME                VALUE IDENTIFIER
2015-08-30T00:31:14 ('header, 177, 2, persistently enable service instance, , sy
stem1, 2015-08-30 00:31:14.027-07:00, subject, usr1, root, root, root, root, 158
706, 4293668494, 58180 22 dhcp.vpn.example.com, use of authorization, solaris.sm
f.modify, fmri, svc:/network/smtp:sendmail/:properties/general/enabled, return,
success, 0\n',) //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action
//:fmt.description
```

The `fmt.summary` format shows both the label and the description of the event.

```
$ sstore export \
> //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action//:fmt.summary
TIME                VALUE IDENTIFIER
2015-08-30T00:31:14 //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-a
ction//:fmt.summary
                    0:
                        ts: 1440919874027218
                        label: AUE_smf_enable
                        description: header, 177, 2, persistently enable service
instance, , system1, 2015-08-30 00:31:14.027-07:00, subject, usr1, root, root,
root, root, 158706, 4293668494, 58180 22 dhcp.vpn.example.com, use of authorizat
ion, solaris.smf.modify, fmri, svc:/network/smtp:sendmail/:properties/general/en
abled, return, success, 0
```

If you do not specify any formatting, or if you specify `fmt.raw`, the output is shown as a list of name-value pairs.

```
$ sstore export //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action
TIME                VALUE IDENTIFIER
2015-08-30T00:31:14 //:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-
action
                    0:
                      version: 2
                      event: AUE_smf_enable
                      host: system1
                      iso8601: 2015-08-30 00:31:14.027-07:00
                      subject/audit-uid: usr1
                      subject/uid: root
                      subject/gid: root
                      subject/ruid: root
                      subject/rgid: root
                      subject/pid: 158706
                      subject/sid: 4293668494
                      subject/tid: 58180 22 dhcp.vpn.example.com
                      use_of_authorization/value: solaris.smf.modify
                      fmri/value: svc:/network/smtp:sendmail/:properties/

general/enabled

                      return/errval: success
                      return/retval: 0
```

Using the `sstore` Command in Interactive Mode

Invoking the `sstore` command with no subcommand opens an interactive, shell-like environment that enables you to quickly explore the statistics store namespace. See the [`sstore\(1\)`](#) man page for the full descriptions of commands that are supported in the `sstore` shell.

Example 3-4 Exploring Statistics in Interactive Mode

This example shows exploring the `usage` statistic in the `cpu` class in `sstore` interactive mode.

```
$ sstore
Interactive mode. Type help to see list of commands.
//: > help
sstore commands: capture export info list pwd select
sstore aliases:  cat cd ls
other commands:  alias help history exit quit shell !
//: > alias
alias cat=capture
alias cd=select
alias ls=list
//: > cd class.cpu
//:class.cpu//: > ls
event.adm-action
event.fault
res.id/0
res.id/1
stat.context-switches
stat.fpu-capacity
stat.fpu-usage
stat.integer-pipe-capacity
stat.integer-pipe-usage
```

```

stat.interrupt-count
stat.interrupt-time
stat.involuntary-context-switches
stat.system-calls
stat.thread-migrations-core
stat.thread-migrations-socket
stat.traps
stat.usage
stat.xcalls
//:class.cpu//: > ls -l
//:class.cpu//:event.adm-action CPU admin event
//:class.cpu//:event.fault CPU fault event
//:class.cpu//:res.id/0 cpu instance
//:class.cpu//:res.id/1 cpu instance
//:class.cpu//:stat.context-switches context switches
//:class.cpu//:stat.fpu-capacity max floating point instructions per second
//:class.cpu//:stat.fpu-usage floating point instructions
//:class.cpu//:stat.integer-pipe-capacity max integer instructions per second
//:class.cpu//:stat.integer-pipe-usage integer instructions
//:class.cpu//:stat.interrupt-count interrupt count
//:class.cpu//:stat.interrupt-time interrupt time
//:class.cpu//:stat.involuntary-context-switches involuntary context switches
//:class.cpu//:stat.system-calls system calls
//:class.cpu//:stat.thread-migrations-core threads migrated to another core
//:class.cpu//:stat.thread-migrations-socket threads migrated to another socket
//:class.cpu//:stat.traps traps
//:class.cpu//:stat.usage cpu time
//:class.cpu//:stat.xcalls cross calls
//:class.cpu//: > info stat.usage
Identifier: //:class.cpu//:stat.usage
  $schema: //:stat
  copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
description: cpu time
  id: //:class.cpu//:stat.usage
  localize: description
multiplier: 1000000
stability: stable
  type: counter
  units: milliseconds
  zeroes: False
partitions: cpu
partitions: mode
partitions: zone
partitions: core
partitions: chip
partitions: pg
partitions: lgroup
//:class.cpu//: > cd stat.usage
//:class.cpu//:stat.usage//: > cd part.mode
//:class.cpu//:stat.usage//:part.mode//: > cat
TIME VALUE IDENTIFIER
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.chip
0: 192981675680.16354
1: 192981627774.54575
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.core
0: 12061387159.680264
1: 12061352884.526672
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.cpu
0: 1507703307.25418
1: 1507669112.690865
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.lgroup

```

```

1: 192981675680.16354
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.pg
1: 12061387159.680264
2020-08-28T00:35:20 //:class.cpu//:stat.usage//:part.mode//:part.zone
global: 385963303454.70905
^C//:class.cpu//:stat.usage//:part.mode//: > history
 1 help
 2 alias
 3 cd class.cpu
 4 ls
 5 ls -l
 6 info stat.usage
 7 cd stat.usage
 8 cd part.mode
 9 cat
//:class.cpu//:stat.usage//:part.mode//: > exit
$

```

Using Collections

A collection is a convenient way to show multiple statistics using one SSID, as described in [Representing Sets of Statistics and Events](#), and to record all these statistics persistently.

Oracle Solaris includes several collections, which are owned by `root`. The following command lists all collections on this system that are owned by the user that issued the command, which is `root` in this case:

```

root# sstore list //:class.collection//:collection.*
IDENTIFIER
//:class.collection//:collection.name/root/apache-stats
//:class.collection//:collection.name/root/compliance-stat
//:class.collection//:collection.name/root/cpu-stats
//:class.collection//:collection.name/root/network-stats
//:class.collection//:collection.name/root/solaris-dashboard
//:class.collection//:collection.name/root/system

```

The same command issued by a different user, `user1` in this case, gives different results:

```

user1$ sstore list //:class.collection//:collection.*
IDENTIFIER
//:class.collection//:collection.name/user1/my-stats

```

You cannot use a wildcard in the *name* position, but you can see the collection if you specify the *name* explicitly. The following command issued by `user1` lists all collections on this system that are owned by `root` because `root` is named in the collection SSID:

```

user1$ sstore list //:class.collection//:collection.name/root/*
IDENTIFIER
//:class.collection//:collection.name/root/apache-stats
//:class.collection//:collection.name/root/compliance-stat
//:class.collection//:collection.name/root/cpu-stats
//:class.collection//:collection.name/root/network-stats
//:class.collection//:collection.name/root/solaris-dashboard
//:class.collection//:collection.name/root/system

```


You can use a wildcard in the *uuid* position to list all collections on the system. The UUID is the most reliable identifier. The collection name could be changed by an authorized user; the UUID remains constant through collection name changes.

```
user1$ sstore list //:class.collection//:collection.uuid/*
IDENTIFIER
//:class.collection//:collection.uuid/0ae3fbf0-927a-4f07-a569-c64f9b4bd769
//:class.collection//:collection.uuid/3999ac1f-7cce-4d9d-b1ae-b32525b61f43
//:class.collection//:collection.uuid/724adf52-57b0-430e-9c9a-8d21056c3a33
//:class.collection//:collection.uuid/7bdd7228-1d72-47f6-8d96-fa659d57ed2d
//:class.collection//:collection.uuid/8f174784-3613-4371-ad54-ff0c258e3671
//:class.collection//:collection.uuid/dd19683a-88b3-47c3-beff-c2c5d0e7e83f
//:class.collection//:collection.uuid/f465bec7-a20a-4b78-b5ed-fedb0610473e
//:class.collection//:collection.uuid/f615ff30-4e1c-41e1-91f7-b0e58651cde5
```

You cannot use the `sstore list` command to list the statistics that belong to the collection. Use the `sstore info` command instead.

```
$ sstore list //:class.collection//:collection.name/root/network-stats/*
Warning (//:class.collection//:collection.name/root/network-stats/*) -
Collection name network-stats/* cannot have ''
```

The `sstore info` command shows you which statistics and events are included in the collection, whether the collection is enabled, and the UUID, owner, and name of the collection. All of the following commands display the same output except for the value of Identifier. The user name is needed only if you are not the owner of the collection or if two users create collections with the same name (*cname*).

```
root# sstore info //:class.collection//:collection.name/network-stats
user1$ sstore info //:class.collection//:collection.name/root/network-stats
user1$ sstore info \
> //:class.collection//:collection.uuid/cd9df265-7f50-414f-b8b6-d70352d16092
Identifier: //:class.collection//:collection.uuid/0ae3fbf0-927a-4f07-a569-c64f9b4bd769
  ssid: //:class.link/phys//:stat.in-bytes
  ssid: //:class.link/phys//:stat.out-bytes
  ssid: //:class.link/phys//:stat.speed
state: disabled
  uuid: 0ae3fbf0-927a-4f07-a569-c64f9b4bd769
  owner: root
  cname: network-stats
crtime: 1480351396984968
```

The following command lists more than 20 statistics in the `//:class.link/phys` class:

```
$ sstore list -a //:class.link/phys//:stat.*
```

The `network-stats` collection includes only three of these statistics, as shown in the preceding `sstore info` example. Using a collection is a convenient way to record the statistics that interest you most. Though the statistics in the `network-stats` collection are not being recorded persistently (the collection state is `disabled`), you can conveniently use the single collection SSID to record these statistics on demand using the System Web Interface, the CLI, or APIs, as in the following example:

```
$ sstore capture //:class.collection//:collection.name/root/network-stats
```

To create your own collection, see [Creating a Collection in Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface](#).

When a collection is enabled (see the `state` in the `sstore info` output), the statistics in the collection are recorded persistently. When a collection is disabled, the statistics in the

collection are not being recorded unless they are being recorded in some other way, such as by the `sstore capture` command or by displaying those statistics in the System Web Interface.

Two of the collections shown in this section are enabled by default: `system` and `solaris-dashboard`.

- The basic system configuration collection (`system`) includes all `//:class.system//:*` statistics.
- The Dashboard collection (`solaris-dashboard`) records statistics and events from eight different classes, including all events and all `//:class.system//:*` statistics. The `solaris-dashboard` collection provides a good example of the convenience of using a collection to access a set of data from different classes.

The following output shows the statistics that are in the `solaris-dashboard` collection:

```
$ sstore info //:class.collection//:collection.name/root/solaris-dashboard
Identifier: //:class.collection//:collection.name/root/solaris-dashboard
  ssid: //:class.cpu//:event.*
  ssid: //:class.cpu//:stat.integer-pipe-usage
  ssid: //:class.cpu//:stat.usage
  ssid: //:class.disk//:stat.read-bytes
  ssid: //:class.disk//:stat.write-bytes
  ssid: //:class.event//:event.*
  ssid: //:class.link/phys//:stat.in-bytes
  ssid: //:class.link/phys//:stat.in-discards
  ssid: //:class.link/phys//:stat.in-drops
  ssid: //:class.link/phys//:stat.in-errors
  ssid: //:class.link/phys//:stat.out-bytes
  ssid: //:class.link/phys//:stat.out-discards
  ssid: //:class.link/phys//:stat.out-drops
  ssid: //:class.link/phys//:stat.out-errors
  ssid: //:class.link/phys//:stat.out-overflows
  ssid: //:class.link/phys//:stat.out-underflows
  ssid: //:class.nfs/client//:stat.calls
  ssid: //:class.nfs/server//:stat.calls
  ssid: //:class.system//:stat.*
  ssid: //:class.zpool//:res.name/*//:stat.capacity
state: enabled
  uuid: e34d4d00-9c04-426a-8058-bf1a085c132f
  owner: root
  cname: solaris-dashboard
  crtime: 1597081754855017
```

Instead of specifying each SSID separately, you can use the single collection SSID to examine the data. The following command uses the single collection SSID to show the current values of each of the statistics from several different classes:

```
$ sstore export -t now -p -1 \
> //:class.collection//:collection.name/root/solaris-dashboard
```

The following table describes the `system (//:class.system//:*)` statistics. This information is from the `description` metadata that you can see by using `sstore info`.

Table 3-1 Statistics in the Basic System Configuration Collection

Identifier <code>::class.system::*</code>	Description
<code>stat.available-memory</code>	Available system memory
<code>stat.available-memory-pages</code>	Memory pages not currently in use by this system
<code>stat.disk-swapspace</code>	Disk-based swap utilization
<code>stat.fqdn</code>	Host name and default domain name of this system
<code>stat.last-boot</code>	Time since this system was last booted, in seconds since epoch
<code>stat.load-average</code>	System load average
<code>stat.memory-page-size</code>	System memory page size
<code>stat.memory-reservation</code>	Physical memory reservations
<code>stat.memory-reservation-failures</code>	Count of failed attempts to reserve physical memory
<code>::class.system::stat.nodename</code>	System node name (<code>uname -n</code>)
<code>::class.system::stat.operatingsystem</code>	The operating system running in this image
<code>stat.page-ins</code>	Page ins
<code>stat.page-outs</code>	Page outs
<code>stat.physical-memory</code>	Physical memory usage
<code>stat.pkg-version</code>	Version of <code>pkg:/entire</code> installed in this image
<code>::class.system::stat.platform</code>	Name of the platform that the sstored daemon is running on (<code>uname -i</code>)
<code>::class.system::stat.processor</code>	Name of the processor or ISA that the sstored daemon is running on (<code>uname -p</code>)
<code>stat.processor-display-name</code>	Display model name of the processor that the sstored daemon is running on
<code>::class.system::stat.processors-physical</code>	Number of physical processors in this system
<code>::class.system::stat.processors-virtual</code>	Number of virtual processors in this system
<code>::class.system::stat.product</code>	System product name
<code>::class.system::stat.serial</code>	System serial number (only for physical systems, not for virtual systems)
<code>::class.system::stat.swap-allocated</code>	Swap space allocated
<code>::class.system::stat.swap-available</code>	Swap space available
<code>::class.system::stat.swap-reserved</code>	Swap space reserved
<code>::class.system::stat.total-memory</code>	Total system memory
<code>::class.system::stat.total-memory-pages</code>	Number of memory pages in this system
<code>::class.system::stat.uptime</code>	System uptime in seconds

Table 3-1 (Cont.) Statistics in the Basic System Configuration Collection

Identifier <code>::class.system::*</code>	Description
<code>::class.system::stat.virtual</code>	Type of virtualization, or physical
<code>stat.virtual-memory</code>	Total virtual memory usage
<code>stat.virtual-memory-reservation-failures</code>	Count of failed attempts to reserve virtual memory
<code>stat.virtual-memory-zone-usage</code>	Zones virtual memory usage

4

Performing Operations on Statistic Values

Minimum and Maximum Value

The `op.min` and `op.max` operations return the minimum or maximum value among the specified SSID data points. These operations can take an optional percentage argument and return the top or bottom of the specified percentage of values in the set of data points.

- `op.min(pct)` returns the values of all data points in the range from the smallest value to the smallest value plus the specified percentage of the difference between the largest and smallest values:

Return all values in the range from `min` to $(\text{min} + (\text{max} - \text{min}) * \text{pct}/100)$.

- `op.max(pct)` returns the values of all data points in the range from the largest value down to the smallest value plus the specified percentage of the difference between the largest and smallest values:

Return all values in the range from $(\text{min} + (\text{max} - \text{min}) * \text{pct}/100)$ to `max`.

Note:

Rate data is used in the following operations examples because values of statistics of type `count`, such as number of bytes read and written, are monotonically increasing and therefore not interesting for some operations, such as minimum, maximum, and average. However, the rate at which bytes are read or written increases and decreases. You might be interested in the minimum, maximum, or average rate. For information about the rate operation, see [Rate and Delta](#).

The following data shows the variance in the rate at which bytes are transmitted and received over the specified period:

```
$ sstore export -t 2020-10-13T10:01:18 -e 2020-10-13T10:07:18 \  
> -i 60 //:class.link//:stat.out-bytes//:op.rate  
TIME VALUE IDENTIFIER  
2020-10-13T10:01:18 36903.0 //:class.link//:stat.out-bytes//:op.rate  
2020-10-13T10:02:18 10641.083333333334 //:class.link//:stat.out-bytes//:op.rate  
2020-10-13T10:03:18 6216.933333333333 //:class.link//:stat.out-bytes//:op.rate  
2020-10-13T10:04:18 244361.9 //:class.link//:stat.out-bytes//:op.rate  
2020-10-13T10:05:18 720056.7 //:class.link//:stat.out-bytes//:op.rate  
2020-10-13T10:06:18 52420.95 //:class.link//:stat.out-bytes//:op.rate
```

Note that this data is a sampling: one data point every minute over six minutes. The sampling did not necessarily report the minimum or maximum values over that period. The following example shows the minimum and maximum rate at which bytes were sent over the specified period:

```
$ sstore export -t 2020-10-13T10:01:18 -e 2020-10-13T10:07:18 \  
> //:class.link//:stat.out-bytes//:op.rate//:op.min
```

```

TIME                VALUE IDENTIFIER
2020-10-13T10:05:18 509.8261492830945 //:class.link//:stat.out-
bytes//:op.rate//:op.min
$ sstore export -t 2020-10-13T10:01:18 -e 2020-10-13T10:07:18 \
> //:class.link//:stat.out-bytes//:op.rate//:op.max
2020-10-13T10:05:56 8483597.747689895 //:class.link//:stat.out-
bytes//:op.rate//:op.max

```

The following example shows the bottom two percent of rates at which bytes were received over the specified period:

```

$ sstore export -t 2020-10-13T10:04:41 -e 2020-10-13T10:05:19 \
> //:class.link//:stat.out-bytes//:op.rate//:op.min(2)
TIME                VALUE IDENTIFIER
2020-10-13T10:04:43 1221.7641995094946 //:class.link//:stat.out-
bytes//:op.rate//:op.min(2)
2020-10-13T10:04:50 1209.93708327167 //:class.link//:stat.out-
bytes//:op.rate//:op.min(2)
2020-10-13T10:05:12 1429.7855321701743 //:class.link//:stat.out-
bytes//:op.rate//:op.min(2)
2020-10-13T10:05:18 509.8261492830945 //:class.link//:stat.out-
bytes//:op.rate//:op.min(2)

```

The following `op.max(40)` example shows the top sixty percent of rates at which bytes were received over the specified period. According to the formula, the top sixty percent of data points are all data points in the specified range from the largest value down to the value that is the smallest value plus forty percent of the difference between the largest and smallest values. In this example that is all values in the following range:

```

$ sstore export -t 2020-10-13T10:05:43 -e 2020-10-13T10:05:57 \
> //:class.link//:stat.out-bytes//:op.rate//:op.min
TIME                VALUE IDENTIFIER
2020-10-13T10:05:44 1586.8238625512568 //:class.link//:stat.out-
bytes//:op.rate//:op.min
$ sstore export -t 2020-10-13T10:05:43 -e 2020-10-13T10:05:57 \
> //:class.link//:stat.out-bytes//:op.rate//:op.max
TIME                VALUE IDENTIFIER
2020-10-13T10:05:56 8483597.747689895 //:class.link//:stat.out-
bytes//:op.rate//:op.max

```

1586.8238625512568 + (8483597.747689895-1586.8238625512568) * 0.4 to
8483597.747689895
1586.8238625512568 + 3392804.36953093749728 to 8483597.747689895
3394391.19339348875408 to 8483597.747689895 bytes per second

```

$ sstore export -t 2020-10-13T10:05:43 -e 2020-10-13T10:05:57 \
> //:class.link//:stat.out-bytes//:op.rate//:op.max(40)
TIME                VALUE IDENTIFIER
2020-10-13T10:05:43 7005824.563157762 //:class.link//:stat.out-
bytes//:op.rate//:op.max(40)
2020-10-13T10:05:49 5156391.649592469 //:class.link//:stat.out-
bytes//:op.rate//:op.max(40)
2020-10-13T10:05:56 8483597.747689895 //:class.link//:stat.out-
bytes//:op.rate//:op.max(40)

```

The following example shows the top ten percent of rates at which bytes were received over the specified period. This example reports a value that was not reported in the previous example of the top sixty percent of values because the time range is more narrow in the previous example:

```
$ sstore export -t 2020-10-13T10:01:18 -e 2020-10-13T10:07:18 \
> //:class.link//:stat.out-bytes//:op.rate//:op.max(90)
TIME                VALUE IDENTIFIER
2020-10-13T10:04:20 8083752.515844949 //:class.link//:stat.out-
bytes//:op.rate//:op.max(90)
2020-10-13T10:05:56 8483597.747689895 //:class.link//:stat.out-
bytes//:op.rate//:op.max(90)
```

Average Value

The `op.avg` operation operates on one or more resources and can take an optional `agg` argument. Time stamps are not matched across SSIDs.

- `op.avg` returns the average of the values of all of the specified statistics for each time stamp
- `op.avg(agg=all)` returns in a single value the average of all of the values of all of the specified statistics over all specified time stamps
- `op.avg(agg=time)` returns the average of values for all time stamps for each specified statistic

The following example shows the average of the rate of bytes received and the rate of bytes transmitted at each time stamp in the specified period:

```
$ sstore export -t 2020-08-26T20:27:28 -e 2020-08-26T20:27:32 \
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-
bytes]//:op.rate//:op.avg
TIME                VALUE IDENTIFIER
2020-08-26T20:27:28 1750.4253606931366 //:class.link/phys//:res.name/net0//:stat.//:s.
[in-bytes,out-bytes]//:op.rate//:op.avg
2020-08-26T20:27:29 1750.4253606931366 //:class.link/phys//:res.name/net0//:stat.//:s.
[in-bytes,out-bytes]//:op.rate//:op.avg
2020-08-26T20:27:30 1230.0049200196802 //:class.link/phys//:res.name/net0//:stat.//:s.
[in-bytes,out-bytes]//:op.rate//:op.avg
2020-08-26T20:27:31 1227.9418727689435 //:class.link/phys//:res.name/net0//:stat.//:s.
[in-bytes,out-bytes]//:op.rate//:op.avg
```

The following example shows the average rate at which bytes were either received or transmitted over the specified period:

```
$ sstore export -t 2020-08-26T20:27:28 -e 2020-08-26T20:27:32 \
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-
bytes]//:op.rate//:op.avg(agg=all)
TIME                VALUE IDENTIFIER
2020-08-26T20:27:31 1269.5934733713102 //:class.link/phys//:res.name/net0//:stat.//:s.
[in-bytes,out-bytes]//:op.rate//:op.avg(agg=all)
```

The following example shows the average rate at which bytes were received over the specified period and the average rate at which bytes were transmitted over the specified period:

```
$ sstore export -t 2020-08-26T20:27:28 -e 2020-08-26T20:27:32 \
> //:class.link/phys//:res.name/net0//:stat.//:s.[in-bytes,out-
bytes]//:op.rate//:op.avg(agg=time)
TIME                VALUE IDENTIFIER
2020-08-26T20:27:31 2502.187908717609 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate//:op.avg(agg=time)
2020-08-26T20:27:31 36.999038025011345 //:class.link/phys//:res.name/net0//:stat.out-
bytes//:op.rate//:op.avg(agg=time)
```

Sum, Difference, Product, and Quotient

The `op.sum`, `op.difference`, `op.product`, and `op.quotient` operations provide arithmetic operations across different SSIDs. Time stamps in the different SSIDs are matched.

The following example sums the number of interrupts for CPUs 0 and 2 at the specified time:

```
$ sstore export -t 2020-09-02T13:37:42 -e 2020-09-02T13:37:43 \
> //:class.cpu//:res.id//:s.[0,2]//:stat.interrupt-count
TIME                VALUE IDENTIFIER
2020-09-02T13:37:42 422910471.0 //:class.cpu//:res.id/0//:stat.interrupt-count
2020-09-02T13:37:43 422910685.0 //:class.cpu//:res.id/0//:stat.interrupt-count
2020-09-02T13:37:42 8792931.0 //:class.cpu//:res.id/2//:stat.interrupt-count
2020-09-02T13:37:43 8792936.0 //:class.cpu//:res.id/2//:stat.interrupt-count
$ sstore export -t 2020-09-02T13:37:42 -e 2020-09-02T13:37:43 \
> //:class.cpu//:res.id//:s.[0,2]//:stat.interrupt-count//:op.sum
TIME                VALUE IDENTIFIER
2020-09-02T13:37:42 431703402.0 //:class.cpu//:res.id//:s.
[0,2]//:stat.interrupt-count//:op.sum
2020-09-02T13:37:43 431703407.0 //:class.cpu//:res.id//:s.
[0,2]//:stat.interrupt-count//:op.sum
```

Different parts of a single partition are not multiple identifiers, as shown in the following example:

```
$ sstore export -t 2020-09-02T13:58:42 -e 2020-09-02T13:58:43 \
> //:class.cpu//:res.id/1//:stat.usage//:part.mode(user, kernel)
TIME                VALUE IDENTIFIER
2020-09-02T13:58:42 //:class.cpu//:res.id/1//:stat.usage//:part.mode(user, kernel)
)
kernel: 740559.213307
user: 173458.285965
$ sstore export -t 2020-09-02T13:58:42 -e 2020-09-02T13:58:43 \
> //:class.cpu//:res.id/1//:stat.usage//:part.mode(user, kernel)//:op.sum
Warning (//:class.cpu//:res.id/1//:stat.usage//:part.mode(user, kernel)//:op.sum)
-
//:op.sum does not support partition-type stats
```

The `op.difference` and `op.quotient` operations operate on the same statistic for exactly two different resources. Make sure you order the resource operands correctly.

```
$ sstore export -t 2020-09-02T13:37:42 -e 2020-09-02T13:37:43 \
> //:class.cpu//:res.id//:s.[0,2]//:stat.interrupt-count//:op.quotient
TIME                VALUE IDENTIFIER
2020-09-02T13:37:42 48.0966438835924 //:class.cpu//:res.id//:s.
[0,2]//:stat.interrupt-count//:op.quotient
2020-09-02T13:37:43 48.096616533999565 //:class.cpu//:res.id//:s.
[0,2]//:stat.interrupt-count//:op.quotient
$ sstore export -t 2020-09-02T13:37:42 -e 2020-09-02T13:37:43 \
> //:class.cpu//:res.id//:s.[2,0]//:stat.interrupt-count//:op.quotient
TIME                VALUE IDENTIFIER
2020-09-02T13:37:42 0.0207914714885364 //:class.cpu//:res.id//:s.
[2,0]//:stat.interrupt-count//:op.quotient
2020-09-02T13:37:43 0.0207914833113697 //:class.cpu//:res.id//:s.
[2,0]//:stat.interrupt-count//:op.quotient
```


Rate and Delta

The `op.rate` and `op.delta` operations return the change or rate of change between two consecutive data points of the same resource. For values `value1` and `value2` recorded at times `time1` and `time2`, rate and delta are calculated as follows:

- `op.rate` is calculated as $(value2 - value1) / (time2 - time1)$
- `op.delta` is calculated as $value2 - value1$

This result is provided for each pair of consecutive data points.

The following example shows the number of bytes received:

```
$ sstore export -t 2020-08-26T20:25:28 -e 2020-08-26T20:27:28 -i 40 \
> //:class.link/phys//:res.name/net0//:stat.in-bytes
TIME                VALUE IDENTIFIER
2020-08-26T20:25:28 2198658462.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-26T20:26:08 2198754940.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-26T20:26:48 2198837272.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
2020-08-26T20:27:28 2198957430.0 //:class.link/phys//:res.name/net0//:stat.in-bytes
```

The following example shows the difference between each pair of values taken 40 seconds apart:

```
$ sstore export -t 2020-08-26T20:25:28 -e 2020-08-26T20:27:28 -i 40 \
> //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.delta
TIME                VALUE IDENTIFIER
2020-08-26T20:25:28 96478.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.delta
2020-08-26T20:26:08 82332.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.delta
2020-08-26T20:26:48 120158.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.delta
```

The following example shows the difference between each pair of values divided by the difference between time stamps for those values (40):

```
$ sstore export -t 2020-08-26T20:25:28 -e 2020-08-26T20:27:28 -i 40 \
> //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate
TIME                VALUE IDENTIFIER
2020-08-26T20:25:28 2411.95 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate
2020-08-26T20:26:08 2058.3 //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate
2020-08-26T20:26:48 3003.95 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate
```

Delta and rate are not necessarily exactly the same when the time difference is one second because the actual time the data was posted in the statistics store is not necessarily exactly at the reported second.

Resource Utilization

The `op.util` operation compares the current value of the named statistic to the total capacity of the named statistic to return the percent utilization of that resource at each time stamp.

Utilization of a Resource that is Not a Partition

The following commands show how much memory the system has and how much of that memory is available:

```
$ sstore export -t 2020-09-02T15:00:00 -p 1 //:class.system//:stat.total-memory
TIME                VALUE IDENTIFIER
2020-09-02T15:00:00 273804165120 //:class.system//:stat.total-memory
$ sstore export -t 2020-09-02T15:00:00 -e 2020-09-02T18:00:00 -i 3600 \
> //:class.system//:stat.available-memory
TIME                VALUE IDENTIFIER
2020-09-02T16:00:00 17605545259.563152 //:class.system//:stat.available-memory
2020-09-02T17:00:00 17127931904.0 //:class.system//:stat.available-memory
2020-09-02T18:00:00 16594131156.955187 //:class.system//:stat.available-memory
```

When you apply the `util` operation to a statistic that is not a partition, the metadata is searched to determine whether the statistic has a known capacity. In this example, `total-memory` is the known capacity of `available-memory`, as shown by the following `sstore info` command:

```
$ sstore info //:class.system//:stat.available-memory
Identifier: //:class.system//:stat.available-memory
$schema: //:stat
copyright: Copyright (c) 2014, 2020, Oracle and/or its affiliates.
description: Available system memory
id: //:class.system//:stat.available-memory
max-capacity: //:stat.total-memory
stability: stable
type: scalar
units: bytes
```

Applying the `util` operation to `available-memory` returns the percent of memory that is available at each time stamp: `available-memory` divided by `total-memory`.

```
$ sstore export -t 2020-09-02T15:00:00 -e 2020-09-02T18:00:00 -i 3600 \
> //:class.system//:stat.available-memory//:op.util
TIME                VALUE IDENTIFIER
2020-09-02T15:00:00 NO_DATA //:class.system//:stat.available-memory//:op.util
2020-09-02T16:00:00 6.429977152410074 //:class.system//:stat.available-
memory//:op.util
2020-09-02T17:00:00 6.255541034773285 //:class.system//:stat.available-
memory//:op.util
2020-09-02T18:00:00 6.060583902981347 //:class.system//:stat.available-
memory//:op.util
```

In the following example, the `difference` operation shows the amount of memory that is being used. When the `util` operation is applied to the new synthetic statistic, the result is the percent of memory that is being used: `total-memory` less `available-memory` divided by `total-memory`.

```
$ sstore export -t 2020-09-02T15:00:00 -e 2020-09-02T18:00:00 -i 3600 \
> //:class.system//:stat.//:s.[total-memory,available-memory]//:op.difference
TIME                VALUE IDENTIFIER
2020-09-02T15:00:00 NO_DATA //:class.system//:stat.//:s.[total-memory,available-
memory]//:op.difference
2020-09-02T16:00:00 256198619860.43686 //:class.system//:stat.//:s.[total-
memory,available-memory]//:op.difference
2020-09-02T17:00:00 256676233216.0 //:class.system//:stat.//:s.[total-
```

```

memory,available-memory]//:op.difference
2020-09-02T18:00:00 257210033963.0448 //:class.system//:stat.//:s.[total-
memory,available-memory]//:op.difference
$ sstore export -t 2020-09-02T15:00:00 -e 2020-09-02T18:00:00 -i 3600 \
> //:class.system//:stat.//:s.[total-memory,available-memory]//:op.difference//:op.util
TIME                VALUE IDENTIFIER
2020-09-02T15:00:00 NO_DATA //:class.system//:stat.//:s.[total-memory,available-
memory]//:op.difference//:op.util
2020-09-02T16:00:00 93.57002284758993 //:class.system//:stat.//:s.[total-
memory,available-memory]//:op.difference//:op.util
2020-09-02T17:00:00 93.74445896522671 //:class.system//:stat.//:s.[total-
memory,available-memory]//:op.difference//:op.util
2020-09-02T18:00:00 93.93941609701865 //:class.system//:stat.//:s.[total-
memory,available-memory]//:op.difference//:op.util

```

Utilization of a Partition Resource

When you apply the `util` operation to a partition, the sum of all parts of the partition is used as the capacity. The following command shows the milliseconds used by each part of the partition:

```

$ sstore export -t 2020-09-08T10:08:00 -p 1 //:class.cpu//:stat.usage//:part.mode
TIME                VALUE IDENTIFIER
2020-09-08T10:08:00 //:class.cpu//:stat.usage//:part.mode
                    idle: 48729451434.882675
                    intr: 58156498.928085
                    kernel: 305427215.107275
                    user: 81600688.06611

```

The following command shows the percent of total usage accounted for by each part of the partition:

```

$ sstore export -t 2020-09-08T10:08:00 -p
1 //:class.cpu//:stat.usage//:part.mode//:op.util
TIME                VALUE IDENTIFIER
2020-09-08T10:08:00 //:class.cpu//:stat.usage//:part.mode//:op.util
                    idle: 99.09468693661124
                    intr: 0.11826523560005732
                    kernel: 0.621107223080918
                    user: 0.16594060469834854

```

Top Values

The `op.top` operation returns the average value of each SSID over each interval of the specified period. By default, the SSIDs are listed in order of descending average value over the entire specified period. Use the `limit` argument to specify how many SSIDs to list. The default value of `limit` is 5. Fewer than `limit` values might be returned because zero values and `NO_DATA` are not included in the results.

The `sstore capture` and `sstore export` commands might show different top SSIDs over the same period of time:

- For `sstore capture`, each *interval* is the entire specified time period. The top value SSIDs could be different at each interval. Thus, over multiple intervals, results might be reported for more than *limit* SSIDs.
- The `sstore export` command always shows no more than *limit* SSIDs. The `sstore export` command compares all available data over the entire period, from the start time

to the end time, to determine which *limit* SSIDs are the top in average value over the whole period. Those same SSIDs are shown in the same order at each interval, even if the average value of a different SSID is higher over that particular interval.

The `sstore export` command presents all the data for one SSID and then all the data for the next SSID, not all the data for one data point and then all the data for the next data point.

Use the `metric` argument to specify a second layer of sorting for the top average value SSIDs:

- The `op.top(limit)` operation returns the *limit* SSIDs with the largest average values over the entire specified period, in descending order from the SSID with the largest average value.
- The `op.top(limit, metric=max)` operation returns the *limit* SSIDs with the largest average values over the entire specified period, in descending order from the SSID with the largest maximum value to the SSID with the smallest maximum value.
- The `op.top(limit, metric=variance)` operation returns the *limit* SSIDs with the largest average values over the entire specified period, in descending order from the SSID with the largest difference between its maximum and minimum values to the SSID with the smallest difference between its maximum and minimum values.

Top Average Values

The `proc` class represents process statistics. The `op.top` operation is useful for reducing the process data to information about only the most active of hundreds of processes on the system. The following command lists the five processes that are using the largest percentage of CPU resources at each one-second interval:

```
$ sstore capture //:class.proc//:res.*//:stat.cpu-percentage//:op.top
TIME          VALUE IDENTIFIER
2020-09-09T15:21:30 3.997802734375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top
2020-09-09T15:21:30 0.4180908203125 //:class.proc//:res.zpool-rpool/6/
root//:stat.cpu-percentage//:op.top
2020-09-09T15:21:30 0.2288818359375 //:class.proc//:res.kmem_task/7/
root//:stat.cpu-percentage//:op.top
2020-09-09T15:21:30 0.1556396484375 //:class.proc//:res.sstored/1046/
root//:stat.cpu-percentage//:op.top
2020-09-09T15:21:30 0.067138671875 //:class.proc//:res.fsflush/3/root//:stat.cpu-
percentage//:op.top
2020-09-09T15:21:31 4.1351318359375 //:class.proc//:res.pkg/18905/
root//:stat.cpu-percentage//:op.top
...
```

The following `sstore export` command specifies the start time from the preceding `sstore capture` command and requests two results for each of the top four SSIDs. The values in the first result (the start time) are exactly the same as the values for those SSIDs in the preceding `sstore capture` example. The values in the second result are the average of the next 60 values for each SSID; they are not just the values at the end time.

Notice that if you had requested the top three SSIDs, the `kmem_task` process would not be listed. The `kmem_task` process used the third most CPU time at the start time, but

the `sstore` process used more CPU time on average over the entire requested 61-datapoint period than the `kmem_task` process used over that same period. The `sstore` process is shown in the third position at both time stamps because the `sstore export` command shows only *limit* SSIDs: the top *limit* SSIDs over all the data points in the entire requested period, not the top *limit* SSIDs at each point reported.

```
$ sstore export -t 2020-09-09T15:21:30 -e 2020-09-09T15:22:30 -i 60 \
> //:class.proc//:res.*//:stat.cpu-percentage//:op.top(4)
TIME                VALUE IDENTIFIER
2020-09-09T15:21:30 3.997802734375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:22:30 3.94287109375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:21:30 0.4180908203125 //:class.proc//:res.zpool-rpool/6/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:22:30 0.4058837890625 //:class.proc//:res.zpool-rpool/6/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:21:30 0.1556396484375 //:class.proc//:res.sstore/1046/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:22:30 0.177001953125 //:class.proc//:res.sstore/1046/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:21:30 0.2288818359375 //:class.proc//:res.kmem_task/7/root//:stat.cpu-
percentage//:op.top(4)
2020-09-09T15:22:30 0.09765625 //:class.proc//:res.kmem_task/7/root//:stat.cpu-
percentage//:op.top(4)
```

Sub-Sort by Maximum Values

The `op.top(limit, metric=max)` operation returns the *limit* SSIDs with the largest average values over the entire specified period, in descending order from the SSID with the largest maximum value to the SSID with the smallest maximum value.

The following example shows the same four SSIDs as are shown in the preceding example because these are the four resources that used the most average CPU time over the entire period. However, in the following example, these same four SSIDs are listed in a different order than in the preceding example.

The `sstore` process used more CPU time over the entire period than the `kmem_task` process did, but the `kmem_task` process has a larger maximum value over the entire period than the `sstore` process has:

```
$ sstore export -t 2020-09-09T15:21:30 -e 2020-09-09T15:22:30 -i 60 \
> //:class.proc//:res.*//:stat.cpu-percentage//:op.top(4,metric=max)
TIME                VALUE IDENTIFIER
2020-09-09T15:21:30 3.997802734375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:22:30 3.94287109375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:21:30 0.4180908203125 //:class.proc//:res.zpool-rpool/6/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:22:30 0.4058837890625 //:class.proc//:res.zpool-rpool/6/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:21:30 0.2288818359375 //:class.proc//:res.kmem_task/7/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:22:30 0.09765625 //:class.proc//:res.kmem_task/7/root//:stat.cpu-
percentage//:op.top(4,metric=max)
2020-09-09T15:21:30 0.1556396484375 //:class.proc//:res.sstore/1046/root//:stat.cpu-
percentage//:op.top(4,metric=max)
```

```
2020-09-09T15:22:30 0.177001953125 //:class.proc//:res.sstored/1046/
root//:stat.cpu-percentage//:op.top(4,metric=max)
```

The values at the reported data points might not include the maximum values. Larger and smaller values might occur at points that are not reported.

Sub-Sort by Value Variance

The `op.top(limit, metric=variance)` operation returns the *limit* SSIDs with the largest average values over the entire specified period, in descending order from the SSID with the largest difference between its maximum and minimum values to the SSID with the smallest difference between its maximum and minimum values.

The following example shows the same four SSIDs as are shown in the preceding two examples because these are the four resources that used the most average CPU time over the entire period. In the following example, these same four SSIDs are listed in a different order than in either of the two preceding examples.

The `kmem_task` process has the greatest variance in its values over the entire period:

```
$ sstore export -t 2020-09-09T15:21:30 -e 2020-09-09T15:22:30 -i 60 \
> //:class.proc//:res.*//:stat.cpu-percentage//:op.top(4,metric=variance)
TIME VALUE IDENTIFIER
2020-09-09T15:21:30 0.2288818359375 //:class.proc//:res.kmem_task/7/
root//:stat.cpu-percentage//:op.top(4,metric=variance)
2020-09-09T15:22:30 0.09765625 //:class.proc//:res.kmem_task/7/root//:stat.cpu-
percentage//:op.top(4,metric=variance)
2020-09-09T15:21:30 3.997802734375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4,metric=variance)
2020-09-09T15:22:30 3.94287109375 //:class.proc//:res.pkg/18905/root//:stat.cpu-
percentage//:op.top(4,metric=variance)
2020-09-09T15:21:30 0.1556396484375 //:class.proc//:res.sstored/1046/
root//:stat.cpu-percentage//:op.top(4,metric=variance)
2020-09-09T15:22:30 0.177001953125 //:class.proc//:res.sstored/1046/
root//:stat.cpu-percentage//:op.top(4,metric=variance)
2020-09-09T15:21:30 0.4180908203125 //:class.proc//:res.zpool-rpool/6/
root//:stat.cpu-percentage//:op.top(4,metric=variance)
2020-09-09T15:22:30 0.4058837890625 //:class.proc//:res.zpool-rpool/6/
root//:stat.cpu-percentage//:op.top(4,metric=variance)
```

The values at the reported data points might not demonstrate the variance. Larger and smaller values might occur at points that are not reported.

Top Values of a Partition Resource

The following examples show how the `op.top` operation handles partitioned SSIDs. The following example shows the number of bytes sent by each listed application.

```
$ sstore capture //:class.net/ip//:stat.out-bytes//:part.application
2016-05-18T12:18:55 //:class.net/ip//:stat.out-bytes//:part.application
ssh: 332648.0
httpd: 0.0
sched: 79741.0
nscd: 604.0
pkg: 0.0
extract_boot_fil: 0.0
grub-mkconfig: 0.0
00_header: 0.0
nfsmapid: 399.0
```

```

                grub-install: 0.0
                ntpd: 448.0
                ldap_cachemgr: 3051.0
2016-05-18T12:18:56 //:class.net/ip//:stat.out-bytes//:part.application
                sshd: 348264.0
                httpd: 0.0
                sched: 86397.0
                nscd: 604.0
                pkg: 5302.0
                extract_boot_fil: 0.0
                grub-mkconfig: 0.0
                00_header: 0.0
                nfsmapid: 399.0
                grub-install: 0.0
                ntpd: 448.0
                ldap_cachemgr: 3051.0

```

The following `sstore capture` command reduces the data to just the largest three partition key values at each data point.

```

$ sstore capture //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
2016-05-18T12:18:55 //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
                sshd: 332648.0
                sched: 79741.0
                ldap_cachemgr: 3051.0
2016-05-18T12:18:56 //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
                sshd: 348264.0
                sched: 86397.0
                pkg: 5302.0

```

The `sstore export` command determines the top values of each partition key over the entire period for which data are available and then compares those top values for each key to determine the top keys over the entire period. Thus the `sstore export` command shows the same partition keys at each time stamp, even if the key has a zero value at some points. In the preceding `sstore capture` results, the `ldap_cachemgr` key has the third largest value of all the partition keys at time 12:18:55. The `sstore export` results at that same time report the value of the `pkg` key because over the entire time period, the `sshd`, `sched`, and `pkg` keys have the highest values. The `sstore export` results show these same three keys in this same order at all data points.

```

$ sstore export //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
2016-05-18T12:18:55 //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
                sshd: 332648.0
                sched: 79741.0
                pkg: 0.0
2016-05-18T12:18:56 //:class.net/ip//:stat.out-bytes//:part.application//:op.top(3)
                sshd: 348264.0
                sched: 86397.0
                pkg: 5302.0

```

If a single SSID specifies both partitioned SSIDs and non-partitioned SSIDs, partitioned SSIDs are ignored. The following command records only values for `//:class.proc//:res.*//:stat.cpu-percentage`; no values are recorded for `//:class.net/ip//:stat.out-bytes//:part.application`:

```

$ sstore capture \
> //:s.[//:class.proc//:res.*//:stat.cpu-percentage,//:class.net/ip//:stat.out-
bytes//:part.application]//:op.top

```

If you specify multiple non-partitioned SSIDs in a single SSID for `op.top`, all data values are compared together to determine the top values. Possibly the values of one of the component SSIDs will not appear in the `op.top` results.

Unit Conversion

The unit conversion operation supports conversion for size, time, and temperature statistics. The statistics can be scalar, counter, or rate types. The following table lists the keywords that can be used as arguments of the `convert` operation.

Table 4-1 Keywords Supported for the `convert` Operation

Unit Type	Keywords	Keywords	Keywords	Keywords
Size	Powers of 1000 bits: <ul style="list-style-type: none"> bits kilobits megabits gigabits terabits petabits exabits 	Powers of 1000 bytes: <ul style="list-style-type: none"> bytes kilobytes megabytes gigabytes terabytes petabytes exabytes 	Powers of 1024 bits: <ul style="list-style-type: none"> kibibits mebibits gibibits tebibits pebibits exbibits 	Powers of 1024 bytes: <ul style="list-style-type: none"> kibibytes mebibytes gibibytes tebibytes pebibytes exbibytes Pages: pages
Time	<ul style="list-style-type: none"> nanoseconds microseconds milliseconds 	<ul style="list-style-type: none"> seconds minutes hours 	<ul style="list-style-type: none"> days weeks years 	–
Temperature	Celsius	Fahrenheit	–	–

You can use the `sstore info` command on a statistic to determine the type of the statistic. For example, memory statistics are in bytes and could be converted to a different size unit.

```
$ sstore info //:class.system//:stat.total-memory
  Identifier: //:class.system//:stat.total-memory
  $schema: //:stat
  copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
  description: Total system memory
    id: //:class.system//:stat.total-memory
    localize: description
  max-capacity: //:stat.total-memory
  stability: stable
  type: scalar
  units: bytes
```

The following command records available memory in kilobytes:

```
$ sstore capture //:class.system//:stat.available-memory//:op.convert(kilobytes)
```

The rate at which bytes are transmitted and received is bytes per second.

```
$ sstore info //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate
  Identifier: //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate
```



```

    $schema: //:stat
    copyright: Copyright (c) 2014, 2016, Oracle and/or its affiliates.
    description: bytes received
      id: //:class.link/phys//:stat.in-bytes
    localize: description
    max-rate: //:stat.speed
    stability: stable
      units: bytes/seconds
    type: rate

```

The following command records the rate at which bytes are received as kilobytes per minute:

```

$ sstore capture \
> //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate//:op.convert(kilobytes/
minutes)

```

Match Values

The `op.filter` operation returns only data values that match the filter argument. The filter argument is one of the following types:

- Numeric
- String
- Key=value pair

The following example shows times when no bytes were received:

```

$ sstore export \
> //:class.link/phys//:res.name/net0//:stat.in-bytes//:op.rate//:op.filter(0)
TIME          VALUE IDENTIFIER
2020-08-04T20:47:07 0.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate//:op.filter(0)
2020-08-04T20:47:12 0.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate//:op.filter(0)
2020-08-04T20:47:24 0.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate//:op.filter(0)
2020-08-04T20:47:33 0.0 //:class.link/phys//:res.name/net0//:stat.in-
bytes//:op.rate//:op.filter(0)

```

The following example uses two key=value pairs to show a summary of an event in which user `usr1` successfully restarted the `svc:/system/webui/server:default` service on the system named `host1`:

```

$ sstore export \
> //:class.event//:event.adm-action//:op.filter(fmri/value=svc:/system/webui/
server,event=AUE_smf_restart)//:fmt.summary
TIME          VALUE IDENTIFIER
2016-06-11T19:09:01 //:class.event//:event.adm-action//:op.filter(fmri/value=svc:/
system/webui/server, event=AUE_smf_restart)//:fmt.summary
0:
    ts: 1465697341080424
    label: restart service instance
    description: usr1 (usr1:staff) success restart service
instance on host1
    tags: ('solaris/sys',)
    status: success

```

Number of Data Points

Use the `op.count` operation to show the number of occurrences of an SSID.

The following example shows that this system experienced 20 faults between the time specified by the `-t` option and the time shown in the `TIME` column:

```
$ sstore export -t 2016-06-01T08:00:00 //:class.event//:event.fault//:op.count
TIME                VALUE IDENTIFIER
2016-06-11T20:33:00 20 //:class.event//:event.fault//:op.count
```

The following example shows that the `svc:/system/webui/server:default` service has been restarted by an administrator four times since the time given in the `-t` option, with the last time being the time in the `TIME` column.

```
$ sstore export -t 2016-06-01T08:00:00 \
> //:class.event//:event.adm-action//:op.filter(fmri/value=svc:/system/webui/
server,event=AUE_smf_restart)//:op.count
TIME                VALUE IDENTIFIER
2016-06-11T19:09:38 4 //:class.event//:event.adm-action//:op.filter(fmri/
value=svc:/system/webui/server, event=AUE_smf_restart)//:op.count
```

Value Change Points

The `op.change` operation shows the initial value of a statistic and the timestamp and value of each subsequent change. This operation does not support event statistics because each event statistic value is by definition a change.

The following example shows when link errors occurred:

```
$ sstore export //:class.link/phys//:stat.in-errors//:op.changed
TIME                VALUE IDENTIFIER
2017-07-07T15:54:53 0.0 //:class.link/phys//:stat.in-errors//:op.changed
2017-07-10T20:34:12 1.0 //:class.link/phys//:stat.in-errors//:op.changed
2017-07-12T16:24:12 0.0 //:class.link/phys//:stat.in-errors//:op.changed
2017-07-17T12:06:29 5.0 //:class.link/phys//:stat.in-errors//:op.changed
2017-07-17T12:33:25 0.0 //:class.link/phys//:stat.in-errors//:op.changed
2017-07-18T13:11:49 2.0 //:class.link/phys//:stat.in-errors//:op.changed
```

A

Statistics Store Identifiers

Statistics Store Identifier Structure

Many identifiers are defined by default and added automatically when associated resources are added such as a disk or other device. Other identifiers are added when you run an application because the application uses statistics store APIs. You can add new identifiers to the statistics store and specify how data will be collected for them as described in [Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface](#).

A statistics store identifier is also called an SSID. An *SSID* has different formats for statistics, events, and collections. All SSIDs specify a class. SSIDs can optionally specify a system.

Each component of an SSID is preceded by the three-character `//:` separator. While names in an SSID can include `/` and `:`, the three-character `//:` sequence is reserved.

Each component of an SSID consists of a keyword, followed by a period (`.`), followed by a name that identifies one or more instances of that component. Each component can have its own special characters or namespace restrictions for instance names. To represent multiple SSIDs in one SSID, you can use slice and wildcard notation. The `*` character can appear anywhere and matches to the next `//:` separator. See [Displaying Resource and Statistic Slices](#) for information about the `s` (slice) keyword.

```
//:system.system-name//:class.class-name
```

system-name

The `system` component distinguishes statistics from different systems. The *system-name* is the name of the system that produced this statistic. Some classes have no resources on a particular system. An example of a *system-name* is `example.com/localhost`, where `example.com` is the name of the system. When no `system` component is specified, the local system is assumed.

class-name

A `class` is a type of object such as a device, a kernel statistic, or a service. A *class-name* represents all statistics for objects of that type. These statistics can be optional or required for that class. Some classes have no required statistics.

A class can specify that it is a subclass of other classes and inherit the namespace and statistics of the parents.

Examples of *class-name* values include `app`, `cpu`, `dev`, `io`, `link`, `pg`, `pset`, `svc`, `system`, and `zone`. Use the following command to list all the statistics store classes on the system:

```
$ sstore list //:class.*
```

For more information about the `sstore list` command, see [Using Command Line Interfaces](#).

Representing Statistics

A statistic is a piece of information about a resource or class. A statistic SSID includes a class component and a statistic component and usually includes a resource (class instance) component. A statistic SSID has the following format:

```
//:class.class-name//:res.resource-instance//:stat.statistic
```

The following example shows a statistic with no resource:

```
//:class.system//:stat.available-memory
```

The following example shows a statistic for a particular resource:

```
//:class.cpu//:res.id/0//:stat.usage
```

Resource Identifiers

A *resource-instance* is a specific object within a *class-name* class. The *resource-instance* includes the resource type and the name of the specific resource.

In the following SSID that identifies CPU instance 0, *id/0* is the *resource-instance*, *id* is the resource type, and 0 is the name of the specific *cpu* resource:

```
//:class.cpu//:res.id/0
```

If *class-name* is *svc*, *resource-instance* is a service instance FMRI. The following SSID identifies the *system/identity:node* SMF service instance, where *fmri* is the resource type, and *system/identity:node* is the name of the specific *svc* resource:

```
//:class.svc//:res.fmri/system/identity:node
```

If *class-name* is *app*, *resource-instance* is a running instance of the application.

Both *class* and *res* components are required to identify any system resource. Identical resource names can identify completely different resources and are differentiated by class. For example, in `//:class.cpu//:res.id/0`, *id/0* is CPU 0; in `//:class.lgroup//:res.id/0`, *id/0* is locality group 0; in `//:class.pset//:res.id/0`, *id/0* is processor set 0; and in `//:class.zone//:res.id/0`, *id/0* is non-global zone 0. The following command shows that the statistics available for each of these identically-named resources are completely different because the resources are not the same kind (class) of object:

```
$ sstore list //:class.*//:res.id/0//:stat.*
IDENTIFIER
//:class.cpu//:res.id/0//:stat.fpu-usage
//:class.cpu//:res.id/0//:stat.integer-pipe-usage
//:class.cpu//:res.id/0//:stat.interrupt-count
//:class.cpu//:res.id/0//:stat.interrupt-time
//:class.cpu//:res.id/0//:stat.usage
//:class.cpu//:res.id/0//:stat.xcalls
//:class.pset//:res.id/0//:stat.load-average
//:class.pset//:res.id/0//:stat.ncpus
//:class.pset//:res.id/0//:stat.runnable-threads
//:class.pset//:res.id/0//:stat.waiting-threads
```

As much as possible, resource names are the same as names used in the related administrative commands. The following example shows that the names of resources for `disk` and `nfs` class objects match the names of the devices in `iostat` output:

```
$ iostat -x
              extended device statistics
device      r/s      w/s      kr/s      kw/s  wait  actv  wsvc_t  asvc_t   %w   %b
sd0          0.0      0.0      0.0      0.0   0.0   0.0    0.0    0.0    0   0
sd1          0.0      0.0      0.0      0.0   0.0   0.0    0.0    0.0    0   0
sd2          0.4      1.0     66.9    223.8   0.0   0.0    0.0    4.8    0   0
sd3          1.9      7.7     20.9    86.1   0.0   0.1    0.0    6.4    0   2
...
nfs85        0.0      0.0      0.0      0.1   0.0   0.0   183.3  480.3   0   0
nfs1314      0.0      0.0      0.0      0.1   0.0   0.0    0.0    1.2    0   0
nfs2719      0.0      0.0      0.0      0.0   0.0   0.0    0.0    1.0    0   0
nfs3102      0.0      0.0      0.0      0.0   0.0   0.0    0.0    0.9    0   0
$ sstore list //:class.disk//:*
IDENTIFIER
//:class.disk//:event.fault
//:class.disk//:res.name/sd0
//:class.disk//:res.name/sd1
//:class.disk//:res.name/sd10
...
$ sstore list //:class.nfs/client//:*
IDENTIFIER
//:class.nfs/client//:res.name/nfs1314
//:class.nfs/client//:res.name/nfs2719
//:class.nfs/client//:res.name/nfs3102
```

The SSID component that follows the `res` component is usually a statistic (`stat`). This component could also be `class` or `event`, as shown in the following example.

```
$ sstore list //:class.cpu//:res.id/0//:*
IDENTIFIER
//:class.cpu//:res.id/0//:class.chip
//:class.cpu//:res.id/0//:class.core
//:class.cpu//:res.id/0//:class.pset
//:class.cpu//:res.id/0//:event.adm-action
//:class.cpu//:res.id/0//:event.fault
//:class.cpu//:res.id/0//:stat.fpu-usage
//:class.cpu//:res.id/0//:stat.integer-pipe-usage
//:class.cpu//:res.id/0//:stat.interrupt-count
//:class.cpu//:res.id/0//:stat.interrupt-time
//:class.cpu//:res.id/0//:stat.usage
//:class.cpu//:res.id/0//:stat.xcalls
```

A single system resource can be available through multiple classes or through multiple resource identifiers in the same class. See [Using Statistics Store Identifier Aliases](#) for examples. On a particular system, some classes might have no resources.

Statistic Identifiers

A statistic can be information about a class or about a resource (a `class, res` pair). The most common statistic type is a counter.

The following resource statistic SSID holds the cumulative interrupt count for CPU instance 0:

```
//:class.cpu//:res.id/0//:stat.interrupt-count
```

To specify multiple statistics with one SSID, use the * wildcard as shown in the following example, or use the s (slice) keyword as shown in [Displaying Resource and Statistic Slices](#).

The following command lists all statistics available for each CPU resource:

```
$ sstore list //:class.cpu//:res.id/0//:stat.*
IDENTIFIER
//:class.cpu//:res.id/0//:stat.fpu-usage
//:class.cpu//:res.id/0//:stat.integer-pipe-usage
//:class.cpu//:res.id/0//:stat.interrupt-count
//:class.cpu//:res.id/0//:stat.interrupt-time
//:class.cpu//:res.id/0//:stat.usage
//:class.cpu//:res.id/0//:stat.xcalls
```

Only statistics can be partitioned. See [Displaying Values of Statistic Partitions](#) for examples.

Only statistic values can be used in mathematical operations. Mathematical operations are expressed as a component of the SSID. See [Performing Operations on Statistic Values](#) for examples.

Using Statistics Store Identifier Aliases

Sometimes you can access the same statistic, resource, or event by using different statistics store identifiers, known as aliases.

- **Statistic operations.** An SSID that specifies a statistical operation is an alias for that statistic combination or result. See [Performing Operations on Statistic Values](#).
- **System topology.** An SSID can include resources in multiple classes to show system topology.
- **Events.** An event identifier can be an alias for the resource associated with the event.

Both of the following identifiers represent available system memory. In this example, two different classes represent the same statistic.

```
//:class.system//:stat.available-memory
//:class.app/solaris/sysstat/sysconf//:stat.available-memory
```

In the following example, all of the following identifiers represent CPU instance 0:

```
//:class.chip//:res.id/0
//:class.core//:res.id/0
//:class.cpu//:res.id/0
```

Both of the following identifiers represent the same non-global zone. In this example, the same class provides two different resource names – the id and the zonename – for the same non-global zone:

```
//:class.zone//:res.id/0
//:class.zone//:res.zonename/zone1
```

Representing Topology

Topology shows the structure of the system by expressing relationships between system resources. The statistics store represents system topology by combining multiple canonical resource names, where a canonical resource name is a `class, res` pair. The last canonical name in the SSID is the name of the resource whose topology is described by the SSID. A `//:class` after a `//:res` is a topological link to related resources. If you are not familiar with the structure of the system, these SSIDs can help you explore the topology.

Topology mapping can also provide a more user-friendly SSID by omitting implementation details. For example, statistics from the `kstat` or `dtrace` providers might be mapped to providers such as `cpu` or `disk`.

Often a topology is valid only for a specific time period because events such as failures and hotplug operations change the configuration of a system. To explore past topology, specify the time range with the `sstore export` command as shown in [Using Command Line Interfaces](#).

The following SSIDs show that this system has only one processor set and one chip but has four cores and four CPUs. All `pset` and `chip` resource values are `id/0`, while `core` and `cpu` resources values are `id/0`, `id/1`, `id/2`, and `id/3`.

```
$ sstore list //:class.cpu//:res.*//:class.*//:res.*
IDENTIFIER
//:class.cpu//:res.id/0//:class.chip//:res.id/0
//:class.cpu//:res.id/0//:class.core//:res.id/0
//:class.cpu//:res.id/0//:class.pset//:res.id/0
//:class.cpu//:res.id/1//:class.chip//:res.id/0
//:class.cpu//:res.id/1//:class.core//:res.id/1
//:class.cpu//:res.id/1//:class.pset//:res.id/0
//:class.cpu//:res.id/2//:class.chip//:res.id/0
//:class.cpu//:res.id/2//:class.core//:res.id/2
//:class.cpu//:res.id/2//:class.pset//:res.id/0
//:class.cpu//:res.id/3//:class.chip//:res.id/0
//:class.cpu//:res.id/3//:class.core//:res.id/3
//:class.cpu//:res.id/3//:class.pset//:res.id/0
```

The following command shows an alternative way to explore relationships among cores, chips, and CPUs. The previous example shows that CPUs have cores, chips, and processor sets. The following example shows that cores have chips and CPUs.

```
$ sstore list //:class.core//:res.id/*/*/*/*
IDENTIFIER
//:class.core//:res.id/0//:class.chip//:res.id/0
//:class.core//:res.id/0//:class.cpu//:res.id/0
//:class.core//:res.id/1//:class.chip//:res.id/0
//:class.core//:res.id/1//:class.cpu//:res.id/1
//:class.core//:res.id/2//:class.chip//:res.id/0
//:class.core//:res.id/2//:class.cpu//:res.id/2
//:class.core//:res.id/3//:class.chip//:res.id/0
//:class.core//:res.id/3//:class.cpu//:res.id/3
//:class.core//:res.id/2//:class.cpu//:res.id/2
```

The following command confirms that these SSIDs are topology links:

```
$ sstore info //:class.core//:res.id/2//:class.cpu//:res.id/2
Identifier: //:class.core//:res.id/2//:class.cpu//:res.id/2
stability: stable
```

```

description: topology link
$ sstore info //:class.cpu//:res.id/3//:class.core//:res.id/3
  Identifier: //:class.cpu//:res.id/3//:class.core//:res.id/3
  stability: stable
description: topology link

```

You can query statistics and events by specifying only the last canonical resource name in the SSID, as shown by the following comparison:

```

$ sstore list //:class.core//:res.id/2//:class.cpu//:res.id/2//:*
IDENTIFIER
//:class.core//:res.id/2//:class.cpu//:res.id/2//:class.chip
//:class.core//:res.id/2//:class.cpu//:res.id/2//:class.core
//:class.core//:res.id/2//:class.cpu//:res.id/2//:class.pset
//:class.core//:res.id/2//:class.cpu//:res.id/2//:event.adm-action
//:class.core//:res.id/2//:class.cpu//:res.id/2//:event.fault
//:class.core//:res.id/2//:class.cpu//:res.id/2//:stat.usage
//:class.core//:res.id/2//:class.cpu//:res.id/2//:stat.interrupt-count
//:class.core//:res.id/2//:class.cpu//:res.id/2//:stat.interrupt-time
//:class.core//:res.id/2//:class.cpu//:res.id/2//:stat.xcalls
$ sstore list //:class.cpu//:res.id/2//:*
IDENTIFIER
//:class.cpu//:res.id/2//:class.chip
//:class.cpu//:res.id/2//:class.core
//:class.cpu//:res.id/2//:class.pset
//:class.cpu//:res.id/2//:event.adm-action
//:class.cpu//:res.id/2//:event.fault
//:class.cpu//:res.id/2//:stat.usage
//:class.cpu//:res.id/2//:stat.interrupt-count
//:class.cpu//:res.id/2//:stat.interrupt-time
//:class.cpu//:res.id/2//:stat.xcalls

```

Representing Events

An *event* is a time-specific change to a resource or class. Fault Manager and SMF services dynamically record event data. A device faulting is a system event. An administrative change to a service is a system event.

The following SSID identifies any system fault event:

```
//:class.event//:event.fault
```

The following SSID identifies a fault of CPU 0:

```
//:class.cpu//:res.id/0//:event.fault
```

The following SSID identifies any administrative action event:

```
//:class.event//:event.adm-action
```

The following SSID identifies an administrative action on the `svc:/network/smtp:sendmail` service:

```
//:class.svc//:res.fmri/network/smtp:sendmail//:event.adm-action
```

The statistics store can capture system fault, defect, and alert data as described in [Fault Management Overview in *Managing Faults, Defects, and Alerts in Oracle Solaris 11.4*](#) and some administrative action data. The following example lists all events for CPUs:


```
$ sstore list //:class.cpu//:res.id/0//:event.*  
IDENTIFIER  
//:class.cpu//:res.id/0//:event.adm-action  
//:class.cpu//:res.id/0//:event.fault
```

Event information is stored only if you requested the information. The statistics store is not populated with events that existed before event recording started. See the `sstore capture` command in [Using Command Line Interfaces](#).

The statistics store includes a predefined set of formatting operations for events. Use the `fmt` keyword in the event SSID to format event information as shown in [Formatting Event Information](#).

In some system object classes, events can be used as aliases to the resources with which they are associated.

Representing Sets of Statistics and Events

To easily access a specified set of statistics and events, you can create a *collection* of those statistics store identifiers. These statistics and events might all be useful for monitoring a particular behavior or diagnosing a particular problem.

- A collection is a list of statistic and event identifiers and metadata for the collection.
- When a collection is enabled, the SSIDs in the collection are persistently recorded.
- When you record or display a collection, values of all the identifiers in the collection are recorded or displayed.
- You can access a collection by using the unique ID of the collection or the name of the collection. If you use the collection name and do not specify a user name, your user name is implied.

```
//:class.collection//:collection.uuid/uuid  
//:class.collection//:collection.name/[user_name/]collection_name
```

The UUID is the most reliable identifier. The collection name could be changed by an authorized user; the UUID remains constant through collection name changes.

- Any user can read the data values and metadata of any collection.
- Only an authorized user can update a collection that was created by another user. See [Authorizing Access to Resources and Statistics in Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface](#) for information about authorizations.
- Collections can be used with wildcards.
- Collections cannot be used with slices.
- Collections cannot be used with operations.

See [Using Collections](#) for information about how to list and record the statistics and events in a collection.

B

Administering StatsStore and System Web Interface

Saving Data for Future Analysis

You might want to review data days or weeks from now, or you might want to share data with an ISV or support organization that does not have access to the system. To do this, take the following steps:

1. Ensure the data is being recorded.

Data is recorded when you view that data in the System Web Interface, but you probably do not always have that data open in the System Web Interface. Data is recorded when you use the `sstore capture` command or library routines that read the data, but you are not always capturing or reading this particular data.

To ensure the data you want is recorded, create a collection for those statistic and event identifiers (SSIDs). SSIDs that are in an enabled collection are recorded persistently. See [Representing Sets of Statistics and Events](#) and [Using Collections](#) for more information about collections. See [Creating a Collection in Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface](#) to learn how to persistently record the specific data you want.

2. Ensure the recorded data is saved.

You can use the `sstore export` command to retrieve data that you recorded from any time in the past. Over time, data can age out of the system so that you have fewer data points per time period. If you think you might want current data weeks or months in the future, you can save the `sstore export` output in a file.

In the System Web Interface, you can view data from any time period for which the data is available. You can select a longer time view from the Set Time Range dialog, and you can use the Set Time Range dialog and the back and forward arrows to view data values in other time periods.

Customizing System Web Interface

Customizations include preferences properties and customized sheets and visualizations. Administrators can set global system preferences and users can set their own individual user preferences.

To customize global preferences properties, see the following sections:

- [Global Preferences Properties Files](#)
- [Global Preferences Service Properties](#)

To customize sheets and visualizations or create new sheets, visualizations, and statistics for all users to see, see the following resources:

- [Adding Custom Data to the Oracle Solaris 11.4 StatsStore and System Web Interface](#)

- [oraclesolaris-contrib Sharing Sheets](#) on GitHub

To customize or add sheets just for yourself, see [Individual User Customizations](#).

Global Preferences Properties Files

The following file contains the default system preferences for the System Web Interface. These are mostly accessibility properties:

```
/usr/lib/webui/htdocs/solaris/preferences/global-prefs.json
```

Do not modify this file directly. Any modifications will be overwritten when the system is upgraded or as a result of running `pkg fix`.

To customize the preferences in `global-prefs.json`, deliver your customizations in one or more files in the following location:

```
/usr/lib/webui/htdocs/solaris/site/preferences
```

The files can have any names, but they must be in the same format as the `global-prefs.json` file and have the `.json` extension. See the [webui-preferences-json\(5\)](#) man page. Files in the `htdocs/solaris/site/preferences` directory are loaded after files in the `htdocs/solaris/preferences` directory. Files in the `htdocs/solaris/site/preferences` directory might be loaded in any order, so ensure that no property is set to different values in different files.

Create an Image Packaging System (IPS) package to deliver files to the `htdocs/solaris/site/preferences` directory. Installing a custom package is the easiest way to preserve your preferences across system upgrades. See [Packaging and Delivering Software With the Image Packaging System in Oracle Solaris 11.4](#).

Global Preferences Service Properties

The `webui/server` service delivers properties that affect the System Web Interface for all users of this host, such as the default landing page or absolute timeout. The following command lists these properties:

```
$ svcprop -p conf -p coordinator webui/server:default
```

See the [webui-service\(7\)](#) man page for descriptions of these properties.

Allowing Alternative System Web Interface URLs

By default, users need to enter the following URL to get to the System Web Interface log-in page on *host-name*:

```
https://host-name:6787/solaris/
```

If the `conf/redirect_root_url` property of the `webui/server:default` service is set to `true` on *host-name*, then users can get to the System Web Interface log-in page by entering only the following URL:

```
https://host-name:6787
```

The value of the `redirect_root_url` property is `true` by default.

If the `conf/redirect_from_https` property of the `webui/server:default` service is set to `true` on *host-name*, then users can get to the System Web Interface log-in page by entering only *host-name* for the URL.

The default value of the `redirect_from_https` property is `false`. Use the following steps to set the `redirect_from_https` property to `true` on *host-name*:

```
$ svcprop -p conf/redirect_from_https webui/server:default
false
$ svccfg -s webui/server:default setprop conf/redirect_from_https=true
$ svccfg -s webui/server:default refresh
$ svcprop -p conf/redirect_from_https webui/server:default
true
```

Setting the Landing Page

By default, the landing page is the Solaris Dashboard. To change the landing page to a different page for all users on *host-name*, set the `conf/default_landing_page` property of the `webui/server:default` service. The following example changes the landing page to the Solaris Analytics page:

```
$ svcprop -p conf/default_landing_page webui/server:default
/solaris
$ svccfg -s webui/server:default setprop conf/default_landing_page=/solaris/apps/
analytics
$ svccfg -s webui/server:default listprop conf/default_landing_page
conf/default_landing_page astring      /solaris/apps/analytics
$ svccfg -s webui/server:default refresh
$ svcprop -p conf/default_landing_page webui/server:default
/solaris/apps/analytics
```

Individual User Customizations

Whenever a user of the System Web Interface modifies the application behavior or adds new sheets, visualizations, or statistics that are stored persistently, these changes are stored as user preferences in the following file:

```
/var/user/user-name/webui/preferences/solaris.json
```

These user preferences are owned and readable only by that user or by administrators. Do not modify this file directly.

For example, when you use System Web Interface to change your home sheet in Solaris Analytics (right-click on the thumbnail of the sheet and select "Your Home Sheet"), an entry similar to the following is written to your preferences file:

```
$ cat /var/user/user-name/webui/preferences/solaris.json
{
  "_lastUpdated": 1603306041264,
  "apps": {
    "analytics": {
      "v1": {
        "homeSheet": "9af5e11f-6804-5a79-8918-39cbc85b57ac"
      }
    }
  }
}
```

Troubleshooting Retrieving Statistics

The following sections describe how to diagnose and fix certain issues.

Ensure Services Are Online

Verify that the system services associated with the statistics store and System Web Interface are online and not in maintenance.

If a service is not online, use the `svcs -Lx` command to investigate. See [Troubleshooting Services Problems in *Managing System Services in Oracle Solaris 11.4*](#).

StatsStore Services

The Oracle Solaris 11.4 OS includes StatsStore; you do not need to install it. The statistics store, CLI, and APIs are delivered by the `sstore` package. The `sstored` daemon is invoked automatically during system startup and restarted if any failures occur.

Verify that both of the following services are online:

- `sstore` – The `sstored` daemon captures statistics and events for the OS and applications.
- `sysstat` – The `sysstatd` daemon provides common OS statistics to the statistics store.

```
$ svcs sstore sysstat
STATE          STIME    FMRI
online         13:55:28 svc:/system/sstore:default
online         13:55:28 svc:/system/sysstat:default
```

System Web Interface Service

The System Web Interface is delivered by the `webui-server` package. You might need to install the `webui-server` package:

```
$ pkg install webui-server
```

Make sure the `svc:/system/webui/server:default` service is online.

Resolving System Web Interface Problems

Check the following log files:

- System Web Interface log files in `svcs -L webui/server` and `/var/webui/logs`
- Apache web server log files in `svcs -L http:apache24` and `/var/apache2/2.4/logs`

Ensure the Certificate and Key are Installed

A certificate, key, and certificate signing request (CSR) are generated at system start from properties of the `svc:/system/identity:cert` service.

The generated certificate is self signed and might fail validation. Users can follow the instructions from their browser to add an exception to accept this certificate.

To configure your own SSL certificate chain, see [How to Deliver a Custom Certificate and Key](#).

The following command shows that the `identity:cert` service has a `certificate` property group, the `certificate` property group has `ca` and `cert` child property groups, and the `cert` property group has a `private_key` child property group.

```
$ svccfg -s identity:cert listprop certificate/*
certificate/generate                boolean      false
certificate/ca                      application
certificate/ca/pem_value            astring     "pem_value"
certificate/ca/uri                  astring     /etc/certs/localhost/host-
ca/hostca.crt
certificate/cert                    application
certificate/cert/hash               astring     sha256
certificate/cert/keylen             integer     2048
certificate/cert/keytype            astring     rsa
certificate/cert/lifetime           astring     10-year
certificate/cert/pem_value          astring     "pem_value"
certificate/cert/subject            astring
certificate/cert/uri                astring     /etc/certs/localhost/
host.crt
certificate/cert/private_key        application
certificate/cert/private_key/pem_value astring     "pem_value"
certificate/cert/private_key/read_authorization astring     solaris.smf.read.identity
certificate/cert/private_key/uri    astring     /etc/certs/localhost/
host.key
```

When the value of the `certificate/generate` property is `true`, the properties in the `certificate/cert` property group are used to set the certificate and CSR metadata. The default value of `certificate/cert/subject` is `CN=fqdn`.

A host CA is generated (`certificate/ca`), and the host certificate is signed by that host CA, which ensures that the services can come online initially. A CSR also is generated, which you can use to get a certificate signed by your own CA service and replace the generated certificate. The `/etc/certs/localhost/host-ca/hostca.crt` host CA certificate is linked into `/etc/certs/CA` so that it is trusted by at least local TLS clients. The host and CA certificates get a randomly generated serial number.

The CA, certificate, and private key are stored in the `pem_value` properties and in the locations given by the `uri` properties. Do not modify the `uri` properties.

How to Deliver a Custom Certificate and Key

Use this procedure to replace the generated self-signed certificate configuration with your own properly signed certificate from a certificate authority (CA).

The certificate and key are single PEM-encoded X.509 strings. The specified certificate and key are installed during system installation or `sysconfig` when the value of the `certificate/generate` property is `false`.

Follow this procedure carefully, noting the following cautions:

- Modifying any of the `uri` property values will cause the system to fail.
- Setting only the certificate `pem_value` or only the key `pem_value`, but not both, will cause the system to fail.
- Setting both the certificate `pem_value` and the key `pem_value` when the value of the `certificate/generate` property is `true` will cause the system to fail.

1. Verify that the value of the `certificate/generate` property is `false`.

```
$ svcprop -p certificate/generate identity:cert
true
```

If necessary, set the value to `false`:

```
$ svccfg -s identity:cert
svc:/system/identity:cert> setprop certificate/generate=false
certificate/generate boolean      false
```

2. Set the certificate PEM string.

```
svc:/system/identity:cert> setprop certificate/cert/pem_value = astring: \
"${cat /path/to/signed/certificate.crt }"
```

3. Set the key PEM string.

```
svc:/system/identity:cert> setprop certificate/cert/private_key/pem_value =
astring: \
"${cat /path/to/signed/certificate.key }"
```

4. Set the CA PEM string.

```
svc:/system/identity:cert> setprop certificate/ca/pem_value = astring: \
"${cat /path/to/issuer/certificate.crt }"
```

5. Refresh the service.

```
svc:/system/identity:cert> refresh
svc:/system/identity:cert> exit
```

6. Restart the service.

```
$ svcadm restart -sr svc:/system/identity:cert
```

Because of SMF dependencies, restarting the `svc:/system/identity:cert` service causes the `svc:/system/webui/server` service to be restarted also.

Degraded Performance

If your System Web Interface performance is degraded, try one of the following steps:

- Close some instances of the System Web Interface if you have multiple tabs or browsers open.
- Check the performance impact of data you are showing. Select the Properties option from the Visualization Actions menu, and note the value of "Performance impact" at the bottom of the panel. If you can, reduce the number of expensive statistics you are monitoring.

You can also check for expensive statistics at the command line:

```
$ sstore info //:class.net/ip//:stat.out-bytes
Identifier: //:class.net/ip//:stat.out-bytes
  agg-name: bytes
description: bytes sent
  partitions: link
  partitions: protocol
  partitions: local-address
  partitions: remote-address
  partitions: application
    ssid: //:class.net/ip//:stat.out-bytes
  units: bytes
  stability: unstable
expensive: True
  type: counter
```

- Flush the browser cache.
- Restart the browser.

No Matches Found for Statistic

A “no matches found” error message is shown in the following cases:

- The class, resource, or statistic does not exist.
- The SSID is specified incorrectly.
- You used a wildcard in an SSID that is classified as not browsable or not stable.

SSID Does Not Exist

In the following example, resource identifiers 0-3 exist, but the specified identifier, 4, does not exist:

```
$ sstore list //:class.cpu//:res.id/4
Warning (//:class.cpu//:res.id/4) - lookup error: no matches found
$ sstore list //:class.cpu//:res.id/*
IDENTIFIER
//:class.cpu//:res.id/0
//:class.cpu//:res.id/1
//:class.cpu//:res.id/2
//:class.cpu//:res.id/3
```

SSID is Incorrectly Specified

In the following example, a colon character was omitted from the delimiter in front of the operation:

```
$ sstore capture //:class.proc//:res.*//:stat.cpu-percentage//op.top
TIME          VALUE IDENTIFIER
Warning (//:class.proc//:res.*//:stat.cpu-percentage//op.top) - lookup error: no
matches found
$ sstore capture //:class.proc//:res.*//:stat.cpu-percentage//:op.top
TIME          VALUE IDENTIFIER
2016-05-16T17:33:22 0.20751953125 //:class.proc//:res.sstored/1696/root//:stat.cpu-
percentage//:op.top
...
```


SSID is Unbrowsable or Unstable

If an SSID is classified as unbrowsable or unstable, use one of the following methods to get information about that SSID:

- Specify the full SSID explicitly
- If you want to use a wildcard in the SSID to match multiple SSIDs, use the `-a` option with the `sstore` subcommand.

The following example shows that the `kstat` class is not browsable, though you can get information when you specify the SSID without the wildcard.

```
$ sstore list //:class.kstat*
Warning (//:class.kstat*) - lookup error: no matches found
$ sstore info //:class.kstat
  Identifier: //:class.kstat
description: kstat free-for-all namespace
  stability: stable
browsable: False
```

All resources in the `kstat` class are classified as unstable.

```
$ sstore list //:class.kstat//:res.*
Warning (//:class.kstat//:res.*) - lookup error: no matches found
```

You can get information or record data if you fully specify the SSID.

```
$ sstore info //:class.kstat//:res.system/cpu/0/intr
  Identifier: //:class.kstat//:res.system/cpu/0/intr
stability: unstable
  type: Untyped
description: CPU Interrupt statistics
```

If you need to use a wildcard, use the `-a` option to get information or record data.

```
$ sstore list -a //:class.kstat//:res.system/cpu/0/intr//:*
IDENTIFIER
//:class.kstat//:res.system/cpu/0/intr//:stat.crtime
//:class.kstat//:res.system/cpu/0/intr//:stat.level-count
//:class.kstat//:res.system/cpu/0/intr//:stat.level-time
//:class.kstat//:res.system/cpu/0/intr//:stat.snaptime
$ sstore info //:class.kstat//:res.system/cpu/0/intr//:stat.snaptime
  Identifier: //:class.kstat//:res.system/cpu/0/intr//:stat.snaptime
stability: unstable
  type: counter
kstat name/value type: Relative time
  description: kstat snapshot time
  kstat scale: 1000000000
```

Some, but not all, `//:class.net/ip` statistics are classified as unstable.

The following command records values for eight statistics:

```
$ sstore capture //:class.net/ip//:stat.*
```

The following command records values for 26 statistics:

```
$ sstore capture -a //:class.net/ip//:stat.*
```

If you name the statistic explicitly, you can get information or record data without using the `-a` option.

```
$ sstore info //:class.net/ip//:stat.out-bytes
  Identifier: //:class.net/ip//:stat.out-bytes
    agg-name: bytes
description: bytes sent
  partitions: link
  partitions: protocol
  partitions: local-address
  partitions: remote-address
  partitions: application
    ssid: //:class.net/ip//:stat.out-bytes
    units: bytes
stability: unstable
  expensive: True
  type: counter
$ sstore capture //:class.net/ip//:stat.out-bytes
TIME          VALUE IDENTIFIER
2016-05-18T15:15:06 1201563.0 //:class.net/ip//:stat.out-bytes
```

The `sstore export` command does not require the `-a` option because that data is already available.

```
$ sstore export //:class.net/ip//:stat.*
TIME          VALUE IDENTIFIER
2016-05-18T14:10:41 34731 //:class.net/ip//:stat.in-delivers
```

Size of the StatsStore Repository

An important feature of the StatsStore repository is that the data you request (whether through the System Web Interface, the CLI, or APIs) is saved, enabling you to later compare current and historical values. To prevent the StatsStore repository from growing too large, a size limit is set, and older data is pruned from the repository to stay within the defined limit.

You can prune data, and you can change the size limit of the repository.

- [Pruning Data from the StatsStore Repository](#)
- [Changing the StatsStore Repository Size Limit](#)

Pruning Data from the StatsStore Repository

The `sstored` daemon prunes data by saving fewer and fewer samples of older data. The same set of data might be pruned multiple times as necessary to keep the repository within the size limit as new data is added. For example, if you had recorded some data every second, `sstored` might prune older data to one value every five seconds, then one value every minute, then one value every hour and so forth as needed.

You can do some data pruning yourself by using the `sstoreadm purge` command.

The following command deletes all data for both `ssid-1` and `ssid-2`. If an SSID does not explicitly name statistics (for example the SSID only names a resource), then all statistic data associated with that SSID is deleted:

```
$ sstoreadm purge ssid-1
ssid-2
```

The following command deletes all data for *ssid-1* that was collected between *start-time* and *end-time*. Data collected before *start-time* or after *end-time* is retained:

```
$ sstoreadm purge -t start-time -e end-time
ssid-1
```

Specify only an end time and not a start time to delete data up to the end time and retain data after the end time. Specify only a start time and not an end time to delete data from the start time to the current time and retain data before the start time.

The following command deletes all data for *ssid-1* that was collected between *start-time* and *end-time* and that is in increments of 60 seconds or less. Data collected in increments of 61 seconds or larger is retained:

```
$ sstoreadm purge -t start-time -e end-time -i 60 ssid-1
```

You can also use *i* with just an end time and no start time, with just a start time and no end time, and with no start or end time.

See also the [sstoreadm\(1\)](#) man page.

Changing the StatsStore Repository Size Limit

The default maximum size of the StatsStore repository is 2 gigabytes. You can specify a higher or lower maximum size by setting the `config/max-repo-size` property in the `svc:/system/sstore` service.

Use the `zfs list` command to check the current usage:

```
$ zfs list rpool/VARSHARE/sstore
NAME                               USED AVAIL REFER MOUNTPOINT
rpool/VARSHARE/sstore             26.8M  245G  26.8M  /var/share/sstore/repo
```

Like the `config/repo-path` property, the `config/max-repo-size` property is a global property of the `sstore` service that will be inherited by the `sstore:default` service instance. The `max-repo-size` property is not defined in the `sstore` service manifest. The first time you set the `max-repo-size` property, you must include the type of the property. The type of the `max-repo-size` property is integer and the value is in megabytes.

The following command shows that the `max-repo-size` property has never been set:

```
$ svccfg -s sstore
svc:/system/sstore> listprop config
config/repo-path astring    /var/share/sstore/repo
```

The following command sets the `max-repo-size` property:

```
svc:/system/sstore> setprop config/max-repo-size = integer: 4096
svc:/system/sstore> listprop config
config/max-repo-size integer    4096
config/repo-path    astring    /var/share/sstore/repo
```

By default, the `svccprop` command shows the composed view of properties. The `max-repo-size` property will not show until you refresh the service instance:

```
$ svccprop -p config sstore:default
config/repo-path astring /var/share/sstore/repo
$ svcadm refresh sstore:default
```

```
$ svcprop -p config sstore:default
config/max-repo-size integer 4096
config/repo-path astring /var/share/sstore/repo
```

For the new `max-repo-size` value to take effect, you must restart the service instance:

```
$ svcadm restart sstore:default
```

Run in Debug Mode

Another way to get more information about the statistics store is to run the statistics store daemon with the `--debug` and `--log` options:

```
# /usr/lib/sstored --debug --log --log-path=/var/tmp/log --no-cache --events --no-auth
```

Statistics Store Authorizations and Administrative Profiles

By default, any user can read and record any data in the statistics store. Some other operations, such as reading sensitive data, require additional authorization. The following authorizations are available for operations on the statistics store. See the [sstore-security\(7\)](#) and [sstore-authorized-user\(7\)](#) man pages for more information. The `root` user or role has all `solaris` authorizations. See [Securing Users and Processes in Oracle Solaris 11.4](#) for more information about authorizations, roles, and rights profiles.

Table B-1 Statistics Store Operation Authorizations

Authorization	Authorized Operation
<code>solaris.sstore.read.sensitive</code>	Read a sensitive statistic or event.
<code>solaris.sstore.capture.sensitive</code>	Record a sensitive statistic or event.
<code>solaris.sstore.capture.expensive</code>	Record an expensive statistic or event.
<code>solaris.sstore.update.res</code>	Add a resource to a class or remove a resource from a class.
<code>solaris.sstore.write</code>	Provide statistic or event data.
<code>solaris.sstore.delete</code>	Purge statistic or event data.
<code>solaris.sstore.configure</code>	Update a collection created by another user.
<code>solaris.smf.manage.sstore</code>	Manage statistics store states.
<code>solaris.smf.value.sstore</code>	Change values of statistics store properties.

The following Oracle Solaris administrative profiles are available for operations on the statistics store. The System Administrator profile includes the Stat Store Management profile. The System Observability profile includes the Stat Store Read All profile.

Table B-2 Oracle Solaris Statistics Store Administrative Profiles

Administrative Profile	Authorizations
Stat Store Management	<code>solaris.smf.manage.sstore</code> , <code>solaris.smf.value.sstore</code> , <code>solaris.sstore.*</code>

Table B-2 (Cont.) Oracle Solaris Statistics Store Administrative Profiles

Administrative Profile	Authorizations
Stat Store Read All	solaris.sstore.read.sensitive, solaris.sstore.capture.sensitive, solaris.sstore.capture.expensive

Configuring System Web Interface Listening Ports

By default, System Web Interface listens on port 6787. You can add listening ports, and you can redirect listen requests from port 443 to port 6787.

Redirecting Listen Requests

By default, System Web Interface listens on port 6787. To listen on port 443 and then redirect requests to port 6787, set the value of the `conf/redirect_from_https` property to `true` on the `webui/server:default` service, as shown in the following example. Notice that neither port 6787 nor port 443 is explicitly mentioned:

```
$ $ svccfg -s webui/server:default
svc:/system/webui/server:default> listprop conf/redirect_from_https
conf/redirect_from_https boolean    false
svc:/system/webui/server:default> setprop conf/redirect_from_https=true
svc:/system/webui/server:default> listprop conf/redirect_from_https
conf/redirect_from_https boolean    true
svc:/system/webui/server:default> refresh
svc:/system/webui/server:default> exit
$ svcprop -p conf/redirect_from_https webui/server:default
true
```

Adding Listening Ports

The following example shows how to configure System Web Interface to listen on multiple ports. In this example, two ports, 6787 and 8080, are specified. Always include port 6787, which is the default:

```
$ svccfg -s webui/server:default
svc:/system/webui/server:default> setprop conf/listen_addresses = astring:
("6787" "8080")
svc:/system/webui/server:default> listprop conf/listen_addresses
conf/listen_addresses astring      "6787" "8080"
svc:/system/webui/server:default> refresh
svc:/system/webui/server:default> exit
$ svcprop -p conf/listen_addresses webui/server:default
6787 8080
```

Index

Symbols

//:class, [A-1](#)
//:event, [A-6](#)
//:fmt, [3-7](#)
//:part, [3-4](#)
//:res, [A-2](#)
//:s, [3-2](#)
//:stat, [A-2](#)
//:system, [A-1](#)

A

aliases, [A-4](#)

C

certificates and keys, [B-5](#)
collections, [3-10](#), [A-7](#), [B-1](#)
comparing statistics, [2-7](#)

D

drill down, [2-4](#), [2-6](#)

E

events, [A-6](#)
 formatting, [3-7](#)

F

formatting events, [3-7](#)

G

global-prefs.json global preferences file, [B-2](#)

I

IPS packages, [B-4](#)

L

listening ports, [B-12](#)

O

operations

- op.avg, [4-3](#)
- op.change, [4-14](#)
- op.convert, [4-12](#)
- op.count, [4-14](#)
- op.delta, [4-5](#)
- op.difference, [4-4](#)
- op.filter, [4-13](#)
- op.max, [4-1](#)
- op.min, [4-1](#)
- op.product, [4-4](#)
- op.quotient, [4-4](#)
- op.rate, [4-5](#)
- op.sum, [4-4](#)
- op.top, [4-7](#)
- op.util, [4-5](#)

P

packages

- service/system/sstore, [B-4](#)
- system/management/webui/webui-server,
 [B-4](#)

partitions, [3-4](#)

- drilling down, [2-6](#)
- showing, [2-4](#)

ports, [B-12](#)

preferences

global

- global-prefs.json file, [B-2](#)
- system/webui/server service, [B-2](#)
- webui/htdocs/solaris/site/
 preferences files, [B-2](#)

user

- solaris.json file, [B-3](#)

R

resources
 multiple, [3-2](#)
 SSID, [A-2](#)
 topology, [A-5](#)

S

service/system/sstore package, [B-4](#)
 services, [B-4](#)
 slices, [3-2](#)
 SMF services
 system/identity:cert, [B-5](#)
 system/sstore:default, [B-4](#), [B-9](#)
 system/sysstat:default, [B-4](#)
 system/webui/server:default, [B-2](#), [B-4](#),
[B-5](#), [B-12](#)
 solaris.json user preferences file, [B-3](#)
 SSID
 class-name, [A-1](#)
 resource-instance, [A-2](#)
 statistic, [A-2](#), [A-3](#)
 system-name, [A-1](#)
 sstore command, [3-1](#)
 capture subcommand, [3-1](#)
 multiple statistic values, [3-3](#)
 export subcommand, [3-1](#)
 multiple statistic values, [3-3](#)
 info subcommand, [3-1](#)
 interactive mode, [3-8](#)
 list subcommand, [3-1](#)
 multiple resources, [3-2](#)
 multiple statistics, [3-3](#)
 sstore shell, [3-8](#)
 sstored daemon, [B-4](#)
 statistics
 multiple, [3-3](#)

statistics (*continued*)
 multiple values, [3-3](#)
 partitions, [3-4](#)
 recording values persistently, [3-10](#)
 SSID, [A-2](#), [A-3](#)
 unit conversion, [4-12](#)
 utilization, [4-5](#)
 statistics store identifiers, [A-1](#)
 aliases, [A-4](#)
 collections, [3-10](#), [A-7](#), [B-1](#)
 events, [A-6](#)
 partitions, [3-4](#)
 slices, [3-2](#)
 topology, [A-5](#)
 wildcards, [3-2](#)
 sysstatd daemon, [B-4](#)
 system/identity:cert service, [B-5](#)
 system/management/webui/webui-server
 package, [B-4](#)
 system/sstore:default service, [B-4](#), [B-9](#)
 system/sysstat:default service, [B-4](#)
 system/webui/server:default service, [B-4](#), [B-5](#),
[B-12](#)
 global preferences properties, [B-2](#)

T

topology, [A-5](#)

V

visualizations
 comparing, [2-7](#)

W

webui/htdocs/solaris/site/preferences
 global preferences files, [B-2](#)
 wildcards, [3-2](#)