System Environment Data Collections (SEDC) Guide

Contents

About System Environmental Data Collections (SEDC)

SEDC is a tool that collects and reports in real time the environmental data on all Cray systems. Data includes information from sensors located on significant hardware components at the cabinet and blade level, such as power supplies, processors, memory and fans. SEDC refers to these sensors as *scan IDs*. Examples of collected data include cabinet and blade temepratures, cooling system air pressure, voltage, current, power from a variety of internal cabinet temperatures, and cooling system air pressures.

Release Information

This release supports the 7.2 UP04 release of the SMW system software. Changes to this document are limited to new organization and formatting, and edits to previous content. There are no new software features for this release.

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, key strokes (e.g., `Enter` and `Alt-Ctrl-F`), and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a graphical user interface window or element.
\ (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line). Do not type anything after the backslash or the continuation feature will not work correctly.

Scope and Audience

This publication is written for System Administrators.

Feedback

Visit the Cray Publications Portal at *http://pubs.cray.com* and make comments online using the *Contact Us* button in the upper-right corner or Email *pubs@cray.com*. Your comments are important to us and we will respond within 24 hours.

Use Group Log Files for Data Collection

By default, SEDC data is collected and stored in automatically rotated flat text files (called *group log files*) with the location, file size, and number of file rotations being specified in the SEDC configuration file. When using group log files to collect data, SEDC has three major components: the SMW SEDC server (`sedc_manager`), blade and cabinet SEDC daemons, and the SEDC UI client.

- The `sedc_manager` is the System Environment Data Collections (SEDC) server. The `sedc_manager` manages SEDC data collection. Control of sedc_manager and definition of the types of environmental data to be collected is acomplished by means of configuration parameters in the SEDC configuration file, `sedc_srv.ini`.
- The `sedc_manager` sends out the scanning configuration for specific groups to the cabinet controllers and blade controllers and records the incoming data by group. The SEDC server saves all collected data coming from blade and cabinet SEDC daemons in group log files that are kept in the location specified in the SEDC `sedc_srv.ini` configuration file. For more information, see *Using SEDC Log Files*.
- Blade and cabinet SEDC daemons scan the hardware to provide the detailed system environment data, such as fan speed, temperatures, and voltages, per requests from the SMW SEDC server.
- SEDC UI clients subscribe to the scanning result events from blade and cabinet SEDC daemons and present data in a readable format.

  Cray provides a default SEDC configuration file, `/opt/cray/hss/default/etc/sedc_srv.ini`. This file contains parameters that configure the SEDC server and parameters that configure data collections. Cray software manages the `sedc_srv.ini` file as a symbolic link to one of the following files:

  - On Cray XC series systems, `/opt/cray/hss/default/etc/sedc_srv.ini.cascade`
  - On Cray XE and Cray XK systems, `/opt/cray/hss/default/etc/sedc_srv.ini.xtek`

  The default configuration as delivered with the released system software enables continuous data collection and includes basic definitions for scanning groups. This configuration is customizable for any system and sites may choose to create their own copies of configuration file for different purposes. For example, a system administrator may create groups that better match site-specific hardware or that increase/decrease scan frequencies for specific group.

  The `sedc_manager` reads the configuration file upon startup and sends configuration information, such as which sensors to scan, to the cabinet and blade controllers. When the contents of the configuration file are modified, the `sedc_manager` must be directed to re-read this file and send new configuration to controllers; this is done by sending a SIGHUP signal to the `sedc_manager`.

  The SEDC Warning and Control System (WACS)/Environmental Monitoring feature issues a warning notification if the collected value for a measurable scan ID falls outside of the configured limits. The warning event is generated and the occurrence is logged to the event log file.

Display SEDC Data

To display System Environmental Data Collections (SEDC) data or to view server configurations (groups), use the `xtsedcviewer` command-line interface. The `xtsedcviewer` command displays the data from sensors (temperature, voltage, health/status) on blade and cabinet controllers in real time. SEDC reports values of cabinet and blade health status bit-field scan IDs as hexadecimal numbers; the status scan IDs that are not bit fields are reported as decimal numbers.

> **NOTE:** SEDC scan IDs that apply to nodes reflect naming for the logical nodes, not physical nodes.
>
> When the `xtsedcviewer` command is executed, the following navigation and information display options are available (also see the `xtsedcviewer(8)` man page):

↑ **(up arrow) or** `k`	Scrolls up
↓ **(down arrow) or** `j`	Scrolls down
→ **(right arrow) or** `l`	Scrolls right
← **(left arrow) or** `h`	Scrolls left
`a`	Displays the SEDC address map screen
`c`	Displays the SEDC config screen
`d`	Displays the SEDC data screen
`g`	Resets the display (goto origin)
`H`	Displays a help summary
`q`	Exits the program
`u`	Refreshes the display

Group Log Files

By default, the `sedc_manager` application saves all collected data in the log files (also called *group log files*). To log SEDC data, a file writer plugin must be defined in the SEDC configuration file, `/opt/cray/hss/default/etc/sedc_srv.ini`. The default file writer saves collected data in `.CSV` format. For more information, see *Directives That Apply to All Configurations* on page 12.

The sedc_manager creates separate group log files for each group defined within the `sedc_srv.ini` file (using the group_names directive) and saves them in location specified in configuration file (using the `file_data_dir` directive). The default location for SEDC group log files is `/tmp/SEDC_FILES`.

The SEDC log file names describe the location and the type of sensor readings that are contained within the files. For example, on Cray XC series systems, cabinet controller level log file names begin with `CC_`, such as the `CC_HSS_VOLTS_log` file, which contains data collected from voltage sensors on the I/O and compute blades; blade controller level log file names begin with `BC_`, such as the `BC_VOLTS_log` file. The first line in the log file describes the data record fields.

For each SEDC collection group, the number of files to save and the maximum file size is also defined in the configuration file.

For more information about the related sedc_srv.ini file options, see *Directives That Apply to All Configurations*.

To parse through the SEDC log files and display specific records, execute the getSedcLogValues script from the SMW. See the getSedcLogValues(8) man page for additional information.

For more information, see *Configure SEDC*.

If a node is not powered on, node voltages and node temperatures cannot be obtained. For this reason, the SEDC log files will contain the value NA for these sensor readings if SEDC data collection is performed on nodes that have not been powered on. The following example shows node temperature readings for which node 0 on the blade was powered off:

```
c0-0c0s5,2012-09-16 13:54:36,,,,,,,,,,,,,,21,23,30,20,20,20,20, \
23,29,20,19,19,19,23,24,30,21,20,20,20,24,30,20,20,20,20,22,24,32,20,20,21,21,22,
30,21, \
20,20,20,26,27,27,28,20,22,24,25,46,,,,
```

SEDC logs the values of cabinet and blade health status bit-field scan IDs as hexadecimal numbers. The status scan IDs that are not bit fields are logged as decimal numbers.

To parse through the SEDC log files and display specific records, execute the getSedcLogValues script from the SMW. See the getSedcLogValues(8) man page for additional information.

Connection Between Log Files and Group Definitions

The sedc_manager creates separate group log files for each group defined within the sedc_srv.ini file and saves them in the directory defined by the STR:file_data_dir directive in the sedc_srv.ini file.

The SEDC log file names describe the type of sensor readings that are contained within the files. For example, on Cray XC series systems, cabinet controller level log file names begin with CC_, such as the CC_HSS_VOLTS_log file, which contains data collected from voltage sensors on the I/O and compute blades; blade controller level log file names begin with BC_, such as the BC_VOLTS_log file.

For each SEDC collection group, the number of files to save and the maximum file size is also defined in the configuration file. For information about the related sedc_srv.ini file options, see *Directives That Apply to All Configurations* on page 12.

Automatic Rotation of SEDC Log Files

SEDC automatically rotates log files if num_files_to_rotate is set to a value greater than 0. The naming convention acts like the Linux logrotate command; the file numbers when sorted from lowest to highest represent the newer to oldest data.

For example, if a group is defined as CC_STATUS and num_files_to_rotate is set to 3, the sedc_manager saves SEDC records in files named CC_STATUS_log, CC_STATUS_log.1, CC_STATUS_log.2, and CC_STATUS_log.3.

Notes About Collected Data

SEDC creates a log file for each group defined in sedc_srv.ini configuration file. However, SEDC collects and reports only the data relevant to the hardware configuration. Depending on the system hardware configuration, some of the group log files may be empty or partially populated.

Cray XC30-AC (air cooled) systems have the following architecture differences, compared to other Cray XC series systems:

- Cray XC30-AC cabinets have one chassis while a Cray XC30 may have up to three.
- There is also difference in number of rectifiers per cabinet. A Cray XC30 AC cabinet with fully populated shelves will have 12 rectifiers (three shelves with four rectifiers per shelf. A Cray XC30 cabinet with fully populated shelves will have 36 rectifiers (six shelves with six rectifiers per shelf)
- Cray XC30-AC systems do not have blower cabinets or pre-conditioner cabinets. The blower of a Cray XC30-AC system is controlled by a variable frequency drive (VFD). Thus, the CC_VFD_ENV group is specific to Cray XC30-AC systems.
- The temperature strip sensors (CC_INLET_TEMPS group) are also specific to Cray XC30-AC systems. This group will be empty on XC-30 (liquid cooled) cabinet

The SEDC collection on Cascade blade controllers from node-level sensors can be obtained only for nodes that are powered up. The following example shows node temperature readings for which node 0 on the blade was powered off:

```
 c0-0c0s5,2014-09-16 13:54:36,,,,,,,,,,,,,21,23,30,20,20,20,20, \
23,29,20,19,19,19,23,24,30,21,20,20,20,\
24,30,20,20,20,20,22,24,32,20,20,21,21,22,30,21, \
20,20,20,26,27,27,28,20,22,24,25,46,,,,
```

SEDC logs the values of cabinet and blade health status bit-field scan IDs as hexadecimal numbers. The status scan IDs that are not bit fields are logged as decimal numbers.

Cray XE System and Cray XK Systems: SEDC Log Examples

Display all SEDC log files.

To list the existing SEDC log files, execute the following command:

```
crayadm@smw:/tmp/SEDC_FILES> ls *_log
L0_BAX_STATUS_log   L0_SIO_STATUS_log  L0_XT5_STATUS_log      L1_XT4_STATUS_log
L0_BAX_TEMPS_log    L0_SIO_TEMPS_log   L0_XT5_TEMPS_log       L1_XT4_TEMPS_log
L0_BAX_VOLTS_log    L0_SIO_VOLTS_log   L0_XT5_VOLTS_log       L1_XT4_VOLTS_log
L0_FSIO_STATUS_log  L0_XT3_STATUS_log  L1_SLOTTEMP_log        L1_XT5_STATUS_log
L0_FSIO_TEMPS_log   L0_XT3_TEMPS_log   L1_SLOTTEMP_SS_log     L1_XT5_TEMPS_log
L0_FSIO_VOLTS_log   L0_XT3_VOLTS_log   L1_XT3_COLUMNTEMP_log  L1_XT5_VOLTS_log
L0_G34_STATUS_log   L0_XT4_STATUS_log  L1_XT3_STATUS_log
L0_G34_TEMPS_log    L0_XT4_TEMPS_log   L1_XT3_TEMPS_log
L0_G34_VOLTS_log    L0_XT4_VOLTS_log   L1_XT3_VOLTS_log
```

Display sensor readings for a specific scan ID from a specified log file.

The output of this command displays the sensor readings for the scan ID `L1_T_XT5_VALERE_FET_SH0_SL1` in log file `L1_XT5_TEMPS_log`.

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues L1_T_XT5_VALERE_FET_SH0_SL1
L1_XT5_TEMPS_log | more
c0-0   2015-09-03 17:39:13   48
c0-0   2015-09-03 17:40:16   48
c0-0   2015-09-03 17:41:17   49
c0-0   2015-09-03 17:42:18   45
c0-0   2015-09-03 17:43:20   44
c0-0   2015-09-03 17:44:22   44
c0-0   2015-09-03 17:45:24   45
c0-0   2015-09-03 17:46:25   45
c0-0   2015-09-03 17:47:26   46
c0-0   2015-09-03 17:48:28   46
c0-0   2015-09-03 17:49:29   46
.
.
.
```

Display scan IDs from a specific SEDC log file.

The following command provides a list of the different scan IDs from the `L1_XT5_STATUS_log` file. The Cray XT5 L1 scan item names apply to Cray XE systems.

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues -s L1_XT5_STATUS_log
L1_S_XT5_FWLEVEL
L1_H_XT5_PWRSTATUS
L1_H_XT5_CABHEALTH
L1_S_XT5_FANSPEED
L1_S_XT5_FANMODE
L1_S_XT5_VFD_REG
L1_S_XT5_DOORSTAT
L1_H_XT5_CAGE0VRMSTAT
L1_H_XT5_CAGE1VRMSTAT
L1_H_XT5_CAGE2VRMSTAT
L1_H_XT5_VALERE_SH0_SL0
L1_H_XT5_VALERE_SH0_SL1
L1_H_XT5_VALERE_SH0_SL2
L1_H_XT5_VALERE_SH1_SL0
L1_H_XT5_VALERE_SH1_SL1
L1_H_XT5_VALERE_SH1_SL2
L1_H_XT5_VALERE_SH2_SL0
L1_H_XT5_VALERE_SH2_SL1
L1_H_XT5_VALERE_SH2_SL2
L1_S_XT5_VALERE_SHAREFAULTS
L1_H_XT5_XDPALARM
```

Display sensor readings for a specific scan ID for a component from a specified log file.

The output of this command displays the sensor readings for the scan ID `L1_T_XT5_VALERE_FET_SH0_SL1` for component `c0-0` from log file `L1_XT5_TEMPS_log`.

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues -c c100 L1_T_XT5_VALERE_FET_SH0_SL1
L1_XT5_TEMPS_logc0-0   2015-09-03 17:39:13   48
c0-0   2015-09-03 17:40:16   48
c0-0   2015-09-03 17:41:17   49
c0-0   2015-09-03 17:42:18   45
c0-0   2015-09-03 17:43:20   44
c0-0   2015-09-03 17:44:22   44
c0-0   2015-09-03 17:45:24   45
c0-0   2015-09-03 17:46:25   45
c0-0   2015-09-03 17:47:26   46
.
.
.
```

Cray XC Series Systems: SEDC Log Examples

Display all SEDC log files

To list the existing SEDC log files, execute the following command.

```
crayadm@smw:/tmp/SEDC_FILES> ls /tmp/SEDC_FILES/*_log
BC_AOC_RX_ENV_log            BC_DIMM_TEMPS_log      BC_SOCKET_VRM_log
CC_HSS_VOLTS_log
BC_AOC_TX_ENV_log            BC_GPU_POWER_log       BC_SOCKET_VRM_TEMPS_log
CC_INLET_TEMPS_log
BC_ARIES_ENV_log             BC_IBB_SOCKET_VRM_log  BC_TEMPS_log
CC_RECTIFIERS_log
BC_CPU_ACCUM_ENERGY_log      BC_IVOC_ECB_ENV_log    BC_VOLTS_log             CC_TEMPS_log
BC_CPU_TEMPS_log             BC_KNC_POWER_log       CC_AIR_TEMPS_log
CC_VFD_ENV_log
BC_CPU_THERM_ACTIVATION_log  BC_KNC_STATUS_log      CC_AIR_VELOCITY_log
BC_CPU_THERM_STATUS_log      BC_KNC_TEMPS_log       CC_BLOWER_FANSPEED_log
BC_CPU_THROTTLE_log          BC_KNC_VOLTS_log       CC_BLOWER_TEMPS_log
BC_CUPS_log                  BC_MEM_THROTTLE_log    CC_CHASSIS_ENV_log
BC_DIMM_DRAM_ENERGY_log      BC_PCH_THERMAL_log     CC_ENV_INFO_log
```

SEDC creates a log file for each defined group. Depending on the system hardware, some of the group log files may be empty.

Display scan IDs from a specific SEDC log file

The following command provides a list of the different scan IDs from the `CC_HSS_VOLTS_log` file.

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues -s CC_HSS_VOLTS_log
CC_V_VCC_5_0V
CC_V_VCC_5_0V_FAN1
CC_V_VCC_5_0V_SPI
CC_V_VDD_0_9V
CC_V_VDD_1_0V_OR_1_3V
CC_V_VDD_1_2V
CC_V_VDD_1_2V_GTP
CC_V_VDD_1_8V
CC_V_VDD_2_5V
CC_V_VDD_3_3V
CC_V_VDD_3_3V_MICROA
CC_V_VDD_3_3V_MICROB
CC_V_VDD_5_0V
```

Display sensor readings for scan ID `CC_V_VCC_5_0V` fin log file `CC_HSS_VOLTS_log`

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues CC_V_VCC_5_0V CC_HSS_VOLTS_log
c2-0   2012-10-11 08:31:14   5.277
c2-0   2012-10-11 08:32:14   5.304
c2-0   2012-10-11 08:33:14   5.304
c2-0   2012-10-11 08:34:14   5.307
c2-0   2012-10-11 08:35:14   5.304
c2-0   2012-10-11 08:36:14   5.304
c2-0   2012-10-11 08:37:14   5.304
c2-0   2012-10-11 08:38:14   5.289
c2-0   2012-10-11 08:39:15   5.304
c2-0   2012-10-11 08:40:15   5.304
c2-0   2012-10-11 08:41:15   5.301
c2-0   2012-10-11 08:42:15   5.304
c2-0   2012-10-11 08:43:15   5.289
.
```

```
.
.
```

Display sensor readings for scan ID `CC_V_VCC_5_0V` for component `c1-0` from log file `CC_HSS_VOLTS_log.13`

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues -c c1-0 CC_V_VCC_5_0V CC_HSS_VOLTS_log.13
c1-0   2012-10-11 12:24:02   5.319
c1-0   2012-10-11 12:25:02   5.307
c1-0   2012-10-11 12:26:02   5.286
c1-0   2012-10-11 12:27:02   5.283
c1-0   2012-10-11 12:28:02   5.304
c1-0   2012-10-11 12:29:02   5.298
c1-0   2012-10-11 12:30:02   5.286
c1-0   2012-10-11 12:31:02   5.265
c1-0   2012-10-11 12:32:02   5.289
c1-0   2012-10-11 12:33:02   5.286
c1-0   2012-10-11 12:34:02   5.286
c1-0   2012-10-11 12:35:03   5.265
c1-0   2012-10-11 12:36:03   5.289
c1-0   2012-10-11 12:37:03   5.286
c1-0   2012-10-11 12:38:03   5.286
c1-0   2012-10-11 12:39:03   5.289
c1-0   2012-10-11 12:40:03   5.277
.
.
.
```

Display sensor readings for scan ID `CC_V_VCC_5_0V` for component `c0-0` from log file `CC_HSS_VOLTS_log.13`

```
crayadm@smw:/tmp/SEDC_FILES> getSedcLogValues -c c0-0 CC_V_VCC_5_0V CC_HSS_VOLTS_log.13
c0-0   2012-10-11 09:56:43   5.280
c0-0   2012-10-11 09:57:43   5.295
c0-0   2012-10-11 09:58:43   5.295
c0-0   2012-10-11 10:00:18   5.280
c0-0   2012-10-11 10:01:18   5.298
c0-0   2012-10-11 10:02:18   5.295
c0-0   2012-10-11 10:03:18   5.298
c0-0   2012-10-11 10:04:18   5.298
c0-0   2012-10-11 10:05:18   5.274
c0-0   2012-10-11 10:06:18   5.295
c0-0   2012-10-11 10:07:18   5.298
c0-0   2012-10-11 10:08:18   5.301
c0-0   2012-10-11 10:09:18   5.277
.
.
.
```

Display the cabinet controller rectifiers log file, `CC_RECTIFIERS_log`. Because Cray XC30-AC systems have 12 rectifiers, the `CC_RECTIFIERS_log` for a CRay XC30-AC will look like this:

```
crayadm@smw:/tmp/SEDC_FILES> cat /tmp/SEDC_FILES/CC_RECTIFIERS_logc0-0,2013-03-20
07:07:45,51.970,51.910,51.960,51.950,,,51.970,51.990,51.960, \
51.940,,,51.940,51.930,51.950,52.000,,,,,,,,,,,,,,,,,,,,,8.800,8.400,10.200, \
8.600,,,9.500,9.400,9.800,8.500,,,10.300,9.100,9.500,10.700,,,,,,,,,,,,,,,,,,,,, \
112.800,5861.000
c0-0,2013-03-20 07:08:45,51.980,51.900,51.940,51.970,,,51.950,51.990,51.960, \
51.980,,,51.940,51.930,51.930,52.020,,,,,,,,,,,,,,,,,,,,,9.500,8.600,8.600, \
9.300,,,9.300,10.400,8.700,10.700,,,9.400,10.700,9.300,9.100,,,,,,,,,,,,,,,,,,,,, \
113.600,5902.000
```

Whereas a Cray XC30 system will show entries for 36 rectifiers, such as:

```
c1-0,2013-03-22 07:49:50,52.070,52.020,52.040,52.020,52.010,52.000,52.070, \
52.080,52.040,52.010,52.030,52.040,52.100,52.060,52.020,52.100,52.060,52.040, \
52.030,52.040,52.030,52.030,52.030,52.010,52.050,51.990,52.060,52.080,52.000, \
51.990,52.050,52.050,52.010,52.040,52.040,52.030,13.600,13.500,12.300,12.200, \
12.900,11.900,12.600,13.900,12.600,12.800,12.300,12.000,12.600,11.900,11.600, \
14.200,12.600,13.300,11.100,11.400,12.100,11.900,12.900,12.600,12.500,11.400, \
12.400,13.300,11.200,12.700,14.000,13.400,11.600,13.200,12.300,12.600,451.400,23490.000
c2-0,2013-03-22 07:49:50,52.040,52.030,52.090,52.030,52.110,52.040,52.050, \
52.030,52.060,52.060,52.070,52.070,52.050,52.070,52.060,52.020,52.090, \
52.100,52.070,52.040,52.020,52.030,52.050,52.060,52.070,52.100,52.040, \
52.000,52.020,52.050,52.050,52.070,52.050,52.070,52.070,52.070,13.700, \
12.000,12.000,13.200,11.200,12.200,12.900,13.200,12.800,12.300,13.100, \
12.000,11.800,12.500,13.400,13.700,11.800,13.400,12.300,12.900,12.800, \
13.500,13.700,11.400,12.000,12.400,13.200,11.300,13.600,12.500,12.800, \
12.900,12.900,12.300,13.300,12.900,454.600,23666.000
c0-0,2013-03-22 07:49:50,52.100,52.070,52.030,52.040,52.030,52.090,52.090, \
52.060,52.050,52.070,52.000,52.020,52.020,51.990,51.990,52.030,52.010,52.040, \
52.120,52.010,51.940,52.040,52.050,52.050,52.070,52.040,52.030,52.030,52.050, \
52.060,52.080,52.060,51.950,52.010,52.010,52.050,13.500,12.400,14.300, \
13.200,12.600,14.200,13.400,12.300,13.500,13.400,12.000,12.800,14.400,11.800, \
12.500,12.600,12.500,13.000,14.100,12.700,0.000,13.300,12.300,1.400,14.000, \
14.100,13.300,12.900,13.700,13.300,12.500,13.400,14.100,12.800,12.600,12.200, \
445.700,23196.000
c3-0,2013-03-22 07:49:51,52.010,52.030,52.030,52.020,52.040,52.060,51.980, \
52.040,52.010,52.040,52.030,52.060,52.000,52.040,51.980,52.000,52.010,51.990, \
52.040,52.050,51.960,51.980,52.030,51.990,52.020,52.010,52.050,52.070,52.050, \
52.030,51.950,52.040,51.980,51.950,51.990,52.030,12.800,11.600,12.300,12.000, \
12.700,12.200,12.700,11.300,12.300,11.800,12.400,11.200,10.500,11.900,11.500, \
10.900,11.500,12.100,11.800,12.600,11.100,12.400,11.900,10.700,11.700,11.300, \
11.000,11.800,12.700,12.400,12.000,12.200,10.900,10.500,12.600,11.000, \
424.300,22071.000
```

Display inlet sensor entries of the `CC_INLET_TEMPS_log` file

```
crayadm@smw:/tmp/SEDC_FILES> cat /tmp/SEDC_FILES/CC_INLET_TEMPS_log
service id,time,CC_T_AVRG_AIR_INLET_TEMP,CC_T_INLET_TEMP0,CC_T_INLET_TEMP1, \
CC_T_INLET_TEMP2,CC_T_INLET_TEMP3,CC_T_INLET_TEMP5,CC_T_INLET_TEMP6,CC_T_INLET_TEMP7CC_T_
INLET_TEMP2,CC_T_INLET_TEMP3,CC_T_INLET_TEMP5,CC_T_INLET_TEMP6,CC_T_INLET_TEMP7
c0-0,2013-03-20 07:07:45,11.070,10.500,10.500,11.500,11.000,11.500,11.500,11.000
c0-0,2013-03-20 07:08:45,11.070,10.500,11.000,11.500,10.500,11.500,11.500,11.000
c0-0,2013-03-20 07:09:45,10.920,10.500,10.500,11.500,10.500,11.000,11.500,11.000
c0-0,2013-03-20 07:10:45,11.070,10.500,10.500,11.500,11.000,11.500,11.500,11.000
c0-0,2013-03-20 07:11:46,11.000,10.500,10.500,11.500,11.000,11.000,11.500,11.000
c0-0,2013-03-20 07:12:46,10.920,10.500,10.500,11.000,11.000,11.000,11.500,11.000
c0-0,2013-03-20 07:13:46,11.000,10.500,10.500,11.500,10.500,11.500,11.500,11.000
c0-0,2013-03-20 07:14:46,10.850,10.500,10.500,11.000,10.500,11.000,11.500,11.000
c0-0,2013-03-20 07:15:46,10.920,10.000,10.500,11.500,10.500,11.500,11.500,11.000
c0-0,2013-03-20 07:16:47,10.850,10.500,10.500,11.500,10.500,11.000,11.000,11.000
c0-0,2013-03-20 07:17:48,10.780,10.500,10.500,11.000,10.500,11.000,11.000,11.000
c0-0,2013-03-20 07:18:48,10.780,10.500,10.500,11.000,10.500,11.000,11.000,11.000
c0-0,2013-03-20 07:19:49,10.710,10.000,10.500,11.000,10.500,11.000,11.000,11.000
c0-0,2013-03-20 07:20:49,10.850,10.500,10.500,11.000,10.500,11.000,11.500,11.000
c0-0,2013-03-20 07:21:49,10.780,10.500,10.500,11.000,10.500,11.000,11.000,11.000
c0-0,2013-03-20 07:22:49,10.710,10.000,10.500,11.000,10.500,11.000,11.500,10.500
c0-0,2013-03-20 07:23:49,10.710,10.000,10.500,11.000,10.500,11.000,11.000,11.000
.
.
.
```

SEDC Configuration

The `sedc_manager` is the central point of control for SEDC data collection. It is started with rest of CRMS daemons via the `/etc/init.d/rsms` script. The SEDC configuration file, `opt/cray/hss/default/etc/sedc_srv.ini`, contains parameters that configure the `sedc_manager` and data collections. The parameters in the SEDC configuration file are preceded by data type indicators. The recognized data type indicators are: `STR`, `INT`, and `DBL`.

> **NOTE:** Cray software manages the `sedc_srv.ini` file as a symbolic link to one of the following files:
>
> - `/opt/cray/hss/default/etc/sedc_srv.ini.cascade` (Cray XC series systems only)
> - `/opt/cray/hss/default/etc/sedc_srv.ini.xtek` (Cray XE and Cray XK systems only)

The `sedc_manager` reads the configuration file upon startup and is responsible for sending data collection configuration down to SEDC daemons that reside on the L0 and L1or the CC and BC controllers. SEDC can be configured to run at all times or only when a client is listening. The SEDC configuration file provided by Cray has automatic data collection set as the default action.

When the contents of the configuration file are modified, `sedc_manager` must be instructed to update configurations. This is done by sending a `SIGHUP` to the`sedc_manager` process. This will cause `sedc_manager` to re-read the configuration file, stop all SEDC data collections, re-send the scanning configurations to all cabinet and blade controllers, and then restart the data collection. To change the SEDC configuration file path, use the CRMS_SEDC_CONF environment variable. For example, you can add a line to the /etc/init.d/rsms script prior to where it starts the sedc_manager: export CRMS_SEDC_CONF=/opt/cray/hss/default/etc/filename

To change the SEDC configuration file path, use the `CRMS_SEDC_CONF` environment variable. This can be done, for example, by adding a line to the `/etc/init.d/rsms` script prior to where it starts the `sedc_manager`:

```
export CRMS_SEDC_CONF=/opt/cray/hss/default/etc/filename
```

Directives That Apply to All Configurations

The `/opt/cray/hss/default/etc/sedc_srv.ini` file includes a set of global directives that that control sedc_manager and affect all SEDC groups that are defined. Multiple SEDC groups are possible, as described in *Directives Per Group* on page 13. The `/opt/cray/hss/default/etc/sedc_srv.ini` file provided from Cray has the following modifiable settings:

`INT:startup_action = 1`

Determines whether SEDC runs and collects data constantly (the default) or only when clie

- If the value is set to `0`, SEDC runs only when clients are connected. When clients such collection starts and continues until no further clients are connected.
- If the value is set to `1`, data collection is not affected by client connections, but continue

The `sedc_srv.ini` file provided by Cray has this option set to `1`.

`INT:client = 5`

Indicates the number of seconds between client heartbeat messages. The default is `5`.

`INT:max_noreport = 5`

When the cabinet and blade SEDC daemons scan various sensors, if the reading is the sa reading is not reported. The `max_noreport` variable controls the maximum times that a s

`INT:warning_frequency = 0`

Specifies when to issue a warning; `0` (default) issues a warning on first occurrence of scan values are out of limits. The default is `0`.

`INT:compress = 0`

Specifies compression of rotated log files; `0` (default) indicates no compression of rotated l `gzip` (for example, *XXX*`_log.3.gz`) with compression set to `6` (default for `gzip`). The default i

STR:plugin_path = /opt/cray/hss/default/lib64/libcrms_mon_filewriter.so
Provides the absolute path to the file writer plugin for logging of SEDC scans. The default p
/opt/cray/hss/default/lib64/libcrms_mon_filewriter.so

STR:plugin_func_name = get_writer_inst
Provides the name of the file writer plugin function that controls whether SEDC saves colle
get_writer_inst. This default file writer saves collected data in .CSV format.

STR:file_data_dir = /tmp/SEDC_FILES
Specifies the location of the SEDC log files (also called *group log files*) to be saved. The de

INT:data_file_max_size = 10000000
Specifies the size of each file in bytes. The default file size is 10000000.

INT:num_files_to_rotate = 15
Determines the number of files (per group) to save. The default is 15.

INT:max_no_flush_file = 3
Provides the maximum number of times that a file may be written to before the buffers are

STR:group_names
A comma-separated list of the active data collection groups. This must be modified as need
example, on Cray XC Series systems:

```
STR:group_names = CC_TEMPS,CC_AIR_TEMPS,CC_AIR_VELOCITY,CC_HSS_VOI
CC_BLOWER_TEMPS,CC_BLOWER_FANSPEED,CC_INLET_TEMPS,CC_VFD_ENV,CC_EN
BC_AOC_RX_ENV,BC_ARIES_ENV,BC_IVOC_ECB_ENV
```

Directives Per Group

Creating SEDC groups allows the blade and cabinet SEDC daemons to scan components at different frequencies or as a different combination of scan IDs (for example, a group to monitor temperature only).

To configure each SEDC group, define the following settings in the sedc_srv.ini file to reflect the hardware environment and to specify how the collected data is organized.

Each group has mandatory directives defining configuration specific to the group. These directives are constructed by adding the following directives to the name of the group:

_ids
Lists the components to scan. Specific components listed must be a comma-separated list with no spaces between entries, for example: = ::c0-0,::c1-1,::c0-1. Instead of specifying specific components, one of the following wild cards may be specified: all_blades, all_compute_blades, all_service_blades, and all_cabs.

_target
Lists the scan IDs reflecting the parameters to scan.

_collect_freq
Specifies the frequency with which scans will be performed. The default is 60 seconds.

Rapid scanning uses considerable network bandwidth.

_max_noreport
Specifies maximum number of scans to skip, if the scanned value has not exceeded (+-) range from previous reading.

IMPORTANT: Each time a group is added or deleted, update the STR:group_names directive (see *Cray XE System and Cray XK Systems: Examples of Configuration and Directive Usage* on page 14 or *Cray XC series Systems: Examples of Configuration and Directive Usage* on page 15).

Directives Per Scan ID

There are specific four scan ID directives. The `range` directive is required; `minlimit`, `maxlimit`, and `unit` are optional. However, if any optional directive is provided for a scan ID, then all optional directives must be provided.

⚠ **CAUTION:** Administrators must consult a Cray service engineer to obtain the appropriate values for their Cray system before changing the Cray-provided scan ID values.

`_range` Specifies the amount of deviation from the previous reading that should be considered a change in value. Type: `DBL`. This is a required directive.

`_minlimit` Specifies the lowest value that will not cause a warning event to be generated. Type: `DBL`. This is an optional directive.

`_maxlimit` Specifies the highest value that will not cause a warning event to be generated. Type: `DBL`. This is an optional directive.

`_unit` Specifies the kind of units in which the scan ID is reported. Type: `STR`. This is an optional directive. Specify any character string, but it cannot exceed 8 characters; for example, `Celsius` or `TempC`.

Cray XE System and Cray XK Systems: Examples of Configuration and Directive Usage

`STR:group_names =`

The `group_names` directive identifies all scanning groups that have been configured at both the cabinet and blade level; for example, `L1_XT5_STATUS_ids`. For the default set, see the `sedc_srv.ini` file provided from Cray.

`STR:L1_XT5_STATUS_ids =`

For each group, define a list of components to scan. In this cabinet controller example, the wild card `all_cabs` is recognized. If this wild card is not used, identify the individual components to scan.

For example, a group may be defined as `L1`, and the IDs for the group may be set to `all_cabs`:

```
STR:L1_ids = all_cabs
```

`STR:L1_XT5_STATUS_target =`

For each group, define a list of scan IDs that represent sensors to collect data from. For example, the group `L1_XT5_STATUS` may have only two scan IDs separated by a comma: `L1_H_XT5_VALERE_SH0_SL0, L1_H_XT5_VALERE_SH0_SL1`

```
STR:L1_XT5_STATUS_target =
L1_H_XT5_VALERE_SH0_SL0,L1_H_XT5_VALERE_SH0_SL1
```

INT:L1_XT5_STATUS_collect_freq = 60

This example defines the collection frequency for the group L1_XT5_STATUS to be 60 seconds. Rapid collection frequencies generate quite a lot of network traffic, so unless there is a need for it, the frequency for collection should be at least 60 seconds. The default is 60.

INT:L1_XT5_STATUS_max_noreport = 3

The maximum number of scans that may be skipped within the group L1_XT5_STATUS. If the global directive is set, the global value will be used.

DBL:L1_H_XT5_VALERE_SH0_SL0_range = 0.9

The default /opt/cray/hss/default/etc/sedc_srv.ini file lists all of the different items that may be scanned by SEDC. For each item to be scanned, define a range. This range equates to how great the deviation (+-) may be from the previous scan reading before a reading is considered a change in value. This example statement indicates the range for shelf 0, slot 0 and health status for Valere rectifiers in the cabinet; deviation of 0.9 indicates change:

```
DBL:L1_H_XT5_VALERE_SH0_SL0_minlimit = 0x807
DBL:L1_H_XT5_VALERE_SH0_SL0_maxlimit = 0x807
STR:L1_H_XT5_VALERE_SH0_SL0_unit = status
```

NOTE: The Cray XT5 L1 scan item names apply for Cray XE systems.

The scan ID directives minlimit, maxlimit, and unit are used with the Cray-provided settings.

Cray XC series Systems: Examples of Configuration and Directive Usage

STR:group_names =

The group_names variable identifies all scanning groups that have been configured at both the cabinet and blade level; for example, CC_TEMPS. For the default set, see the sedc_srv.ini file provided from Cray.

CC_TEMPS_ids =

For each group, define a list of components to scan. For this cabinet controller example, the wild card all_cabs is recognized. If this wild card is not used, identify the individual components to scan.

For example, a group may be defined as CC_TEMPS, and the IDs for the group may be set to all_cabs:

```
STR:CC_TEMPS_ids = all_cabs
```

INT:CC_TEMPS_collect_freq = 60

This example defines the collection frequency for the group CC_TEMPS to be 60 seconds. Rapid collection frequencies generate quite a lot of network traffic, so unless there is a need for it, the frequency for collection should be at least 60 seconds. The default is 60.

`INT:CC_TEMPS_max_noreport = 3`

The maximum number of scans that may be skipped within the group `CC_TEMPS`. If the global directive is set, the global value will be used.

`STR:CC_TEMPS_target =`

For each group, define a list of scan IDs that represent sensors to collect data from. For example, the group `CC_TEMPS` may have only two scan IDs separated by comma: `CC_T_MCU_TEMP,CC_T_PCB_TEMP`

```
STR:CC_TEMPS_target = CC_T_MCU_TEMP,CC_T_PCB_TEMP
```

`DBL:CC_T_MCU_TEMP_range =`

The maximum number of scans that may be skipped within the group CC_TEMPS. If the global directive is set, the global value will be used.

```
DBL:CC_T_MCU_TEMP_range = 1.0
```

`DBL:CC_T_MCU_TEMP_range = 1.0`

The default `/opt/cray/hss/default/etc/sedc_srv.ini` file lists all of the different items that may be scanned by SEDC. For each item to be scanned, define a range. This range equates to how great the deviation (+-) may be from the previous scan reading before a reading is considered a change in value. This example statement indicates the MCU temperature range for the cabinet; deviation of 1.0 indicates change:

```
DBL:CC_T_MCU_TEMP_minlimit = 10
DBL:CC_T_MCU_TEMP_maxlimit = 40
STR:CC_T_MCU_TEMP_unit = Celsius
```

The scan ID directives `minlimit`, `maxlimit`, and `unit` are used with the Cray-provided settings.

View Configuration Data

All of the environmental scan IDs are referenced by various groups in the default file. To view SEDC data, run the `xtsedcviewer` command-line interface (see *Display SEDC Data* on page 4, and the `xtsedcviewer` man page).

If the `INT:startup_action` value in `sedc_srv.ini` is set to `0` then, when `xtsedcviewer` runs, the command connects to the `sedc_manager` and data collection begins. Data collection continues until the `xtsedcviewer` command exits.

If the `INT:startup_action` value is set to `1`, data collection is not affected by client connections, but continues constantly.

Reinitialize sedc_manager After Changing the Configuration File.

If the SEDC configuration file`sedc_srv.ini` is modified while `sedc_manager` is running, then SEDC *must* be restarted by sending a `SIGHUP` signal to the `sedc_manager` process. This action causes the `sedc_manager` to reread the configuration file `sedc_srv.ini`, update the cabinet and blade SEDC scanning processes, close all log files, and then reopen them using the latest configuration information.

1. Find the process ID (pid) of the `sedc_manager` process.

```
crayadm@smw:~> ps -e | grep sedc_manager
59261 ?        00:00:40 sedc_manager
```

2. Send a `SIGHUP` signal to the `sedc_manager` process. Use the process ID for `sedc_manager` as displayed in the previous step.

```
crayadm@smw:~> /bin/kill -SIGHUP 59261
```

3. Verify process ID (pid) of the `sedc_manager` process.

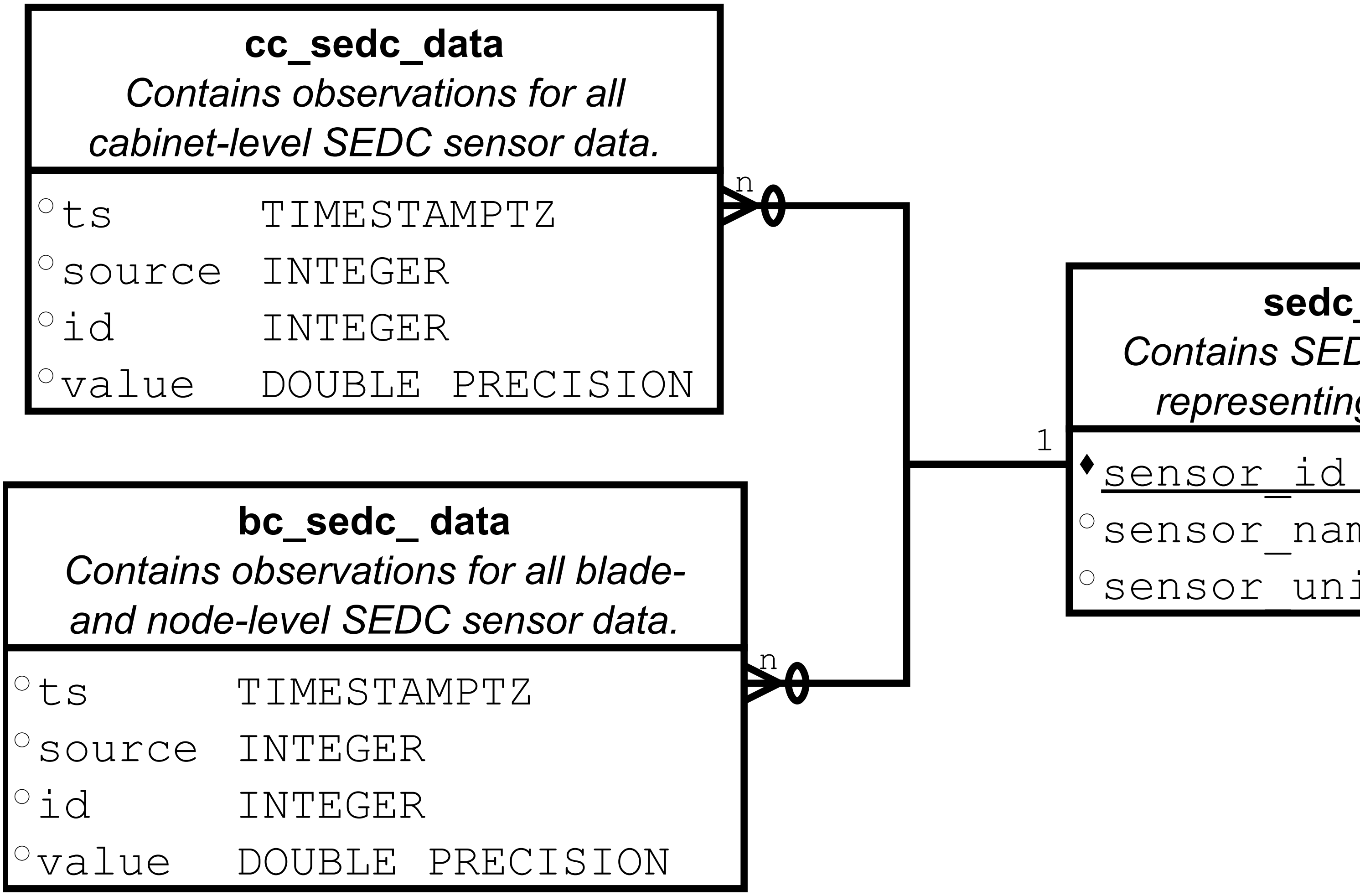
```
crayadm@smw:~> ps -e | grep sedc_manager
```

For additional information about the SEDC manager, see the `sedc_manager(8)` man page.

Use the PMDB for Data Collection

Optionally, administrators of a Cray XC series system can collect and store SEDC data in the Power Management Database (PMDB), which allows for easier searching of the data. For an overview of the PMDB see *Monitoring and Managing Power Consumption on the Cray XC System*. The figure below shows the SEDC schema.

Figure 1. PMDB SEDC Tables



The `pmdb.sedc_scanid_info` table contains information about SEDC `scanids`, which represent sensors:

`sensor_id`

Integer field specifying the SEDC `scanid` that represents a sensor. This field corresponds to the `id` field in the `pmdb.cc_sedc_data` and `pmdb.bc_sedc_data` tables. This field cannot be `null`.

`sensor_name` Text field containing the name of the SEDC `scanid`.

`sensor_units` Text field containing the units of measure for the sensor value.

The `cc_sedc_data` and `bc_sedc_data` tables contain data collected from cabinet-level and blade-level sensors, respectively:

`timestamp` Timestamp-with-time-zone field containing timestamp.

`source` Integer field specifying the CC/BC controller that the data is from.

`id` Integer field containing the SEDC `scanid`.

`value` Double precision field containing the sensor value.

IMPORTANT: It is expected that the use of group log files for SEDC data will be deprecated in a future release.

Enable SEDC to Use the PMDB

IMPORTANT: Sites with high-availability (HA) SMW systems should not store SEDC data in the PMDB unless the PMDB resides on a RAID disk shared by both SMWs. Otherwise, when failover occurs, data can be lost, or be difficult to recover. See *Installing, Configuring, and Managing SMW Failover on the Cray XC System* for information on moving the PMDB on an HA system.

To allow sensor data to be stored in the PMDB, call the `sedc_enable_default` command with the `--database` argument. Other arguments to `sedc_enable_default` allow you to provide, either at the blade or cabinet level, a custom JSON file for SEDC configuring data collection and to specify a partition on which to enable the custom configuration. If no options are specified, the command changes the location for storing sensor data to the PMDB, using the default settings on the system.

When SEDC data is stored in the PMDB the default SEDC configuration comes from the `sedc.ini` file, a read-only file that takes its information from the default blade and cabinet level configuration files located at `/opt/cray/hss/default/etc`. Sites can override the default configuration by specifying the path to custom JSON files.

Call `sedc_enable_default` with the `--legacy` option to stop sending data to the PMDB and resume using text files.

For more information, see the `sedc_enable_default(8)` man page.

NOTE: SEDC data can be stored in either the PMDB or in the group log files, but not in both. Also, be aware that existing data is not ported to the new location.

Query PMDB for SEDC scanid Information

SEDC monitors sensors at cabinet level (`CC_` in the scanID name), blade level (`BC_` in the scanID name) and node level (`BC_`*`x`*`_NODE`*`n`*`_` in the scanID name).

The following example query returns a list of every `sensor_id` and the associated `sensor_name` and `sensor_unit`:

```
pmdb=> select * from pmdb.sedc_scanid_info;
sensor_id | sensor_name              | sensor_units
----------+--------------------------+--------------
 991      | CC_T_MCU_TEMP            | degC
 992      | CC_T_PCB_TEMP            | degC
 993      | CC_V_VCC_5_0V            | V
 994      | CC_V_VCC_5_0V_FAN1       | V
 995      | CC_V_VCC_5_0V_SPI        | V
 996      | CC_V_VDD_0_9V            | V
 997      | CC_V_VDD_1_0V_OR_1_3V    | V
 998      | CC_V_VDD_1_2V            | V
 999      | CC_V_VDD_1_2V_GTP        | V
 1000     | CC_V_VDD_1_8V            | V
 1001     | CC_V_VDD_2_5V            | V
 1002     | CC_V_VDD_3_3V            | V
 1003     | CC_V_VDD_3_3V_MICROA     | V
 1004     | CC_V_VDD_3_3V_MICROB     | V
 1005     | CC_V_VDD_5_0V            | V
 1006     | CC_T_COMP_AMBIENT_TEMP0  | degC
 1007     | CC_T_COMP_AMBIENT_TEMP1  | degC
 1008     | CC_T_COMP_WATER_TEMP_IN  | degC
 1009     | CC_T_COMP_WATER_TEMP_OUT| degC
 1010     | CC_T_COMP_CH0_AIR_TEMP0  | degC
. . .
```

Alternatively, this query prints the `sensor_id` information to a CSV file:

```
smw:~> psql pmdb pmdbuser -t -A -F"," -c "select * from pmdb.sedc_scanid_info" \
> ~/tmp/outfile-SEDC-scanids.csvFor an explanation of the options
used in this query, see the psql man page on the SMW.
```

Query PMDB for CPU Temperature Data

The following example query returns the number of cabinets within a specific range of IDs where there were CPUs with a temperature of 50 C or greater:

```
pmdb=> SELECT COUNT(*), source2cname(source) AS cname, id
FROM pmdb.bc_sedc_data WHERE id >= 1300 AND id <= 1307
AND value >= 50, group by source, id;
count | cname | id
-----------------------
 2 | c0-0c0s8 | 1302
 2 | c0-0c0s8 | 1300
```

To determine the specific temperatures and the time of the events:

```
pmdb=> SELECT ts, source2cname(source) AS cname, id, value
 FROM pmdb.bc_sedc_data WHERE id >= 1300 AND ID <= 1307
 AND value >= 50;
 ts | cname | id | value
-------------------------------+----------+------+-------
2014-09-25 09:42:58.822325-05 | c0-0c0s8 | 1300 | 51
2014-09-25 09:43:38.916163-05 | c0-0c0s8 | 1300 | 51
2014-09-25 09:44:19.01072-05 | c0-0c0s8 | 1302 | 50
```

```
2014-09-25 09:44:59.058131-05 | c0-0c0s8 | 1302 | 51
(4 rows)
```