Home

Author: Ryan Morgan
Date: Feb 2, 2017 11:20 AM
URL: https://support.hyperic.com/display/SIGAR/Home

Table of Contents

1 SIGAR - System Information Gatherer And Reporter

Table of Contents

2 Overview

The Sigar API provides a portable interface for gathering system information such as:

- System memory, swap, cpu, load average, uptime, logins
- Per-process memory, cpu, credential info, state, arguments, environment, open files
- File system detection and metrics
- Network interface detection, configuration info and metrics
- TCP and UDP connection tables
- Network route table

This information is available in most operating systems, but each OS has their own way(s) providing it. SIGAR provides developers with **one** API to access this information regardless of the underlying platform. The core API is implemented in pure C with bindings currently implemented for Java, Perl, Ruby, Python, Erlang, PHP and C#.

The following platforms are currently supported:

Operating System	Architecture	Versions	Binary Included
Linux	x86	2.2, 2.4, 2.6 kernels	✓
Linux	amd64	2.6 kernel	✓
Linux	ppc	2.6 kernel	✓
Linux	ppc64	2.6 kernel	✓
Linux	ia64	2.6 kernel	✓
Linux	s390	2.6 kernel	✗
Linux	s390x	2.6 kernel	✓
Windows	x86	NT 4.0, 2000 Pro/Server, 2003 Server, XP, Vista, 2008 Server, 7	✓
Windows	x64	2003 Server, Vista, 2008 Server, 7	✓
Solaris	sparc-32	2.6, 7, 8, 9, 10	✓
Solaris	sparc-64	"	✓
Solaris	x86	8, 9, 10	✓
Solaris	x64	"	✓
AIX	ppc	4.3, 5.1, 5.2, 5.3, 6.1	✓

AIX	ppc64	5.2,5.3,6.1	✓
HP-UX	PA-RISC	11	✓
HP-UX	ia64	11	✓
FreeBSD	x86	4.x	✗
FreeBSD	x86	5.x, 6.x	✓
FreeBSD	x64	6.x	✓
FreeBSD	x86,x64	7.x,8.x	✗
OpenBSD	x86	4.x,5.x	✗
NetBSD	x86	3.1	✗
Mac OS X	PowerPC	10.3, 10.4	✓
Mac OS X	x86	10.4, 10.5, 10.6	✓
Mac OS X	x64	10.5, 10.6	✓

While SIGAR only depends on the Linux kernel version, the following distributions have been certified:

Distribution	**Versions**
Red Hat	6.2, 7.3, 8.0, 9.0
RHEL	3, 4, 5, 6
CentOS	3, 4, 5
Fedora	2, 3, 4, 5, 6, 7, 8, 9, 10
SuSE	8, 9, 10, 11
Ubuntu	6.06, 8.04, 8.10, 9.04
Debian	2.6, 3.0, 3.1, 3.2, 4.0, 5.0
VMware ESX	2.x, 3.0
XenServer	3.1, 3.2, 4.0, 4.1, 5.0
Slackware	10, 11
Mandrake	10
Scientific Linux	5
Gentoo	

To test drive run the following command:

```
% java -jar sigar-bin/lib/sigar.jar
sigar> help
```

The shell and commands are implemented in Java, the source code is located in bindings/java/src/org/hyperic/sigar/cmd/.
Including implementations of well-known commands such as:

- **df**
- **du**
- **free**
- **ifconfig**
- **iostat**
- **netstat**
- **ps**
- **route**
- **top**
- **ulimit**
- **uptime**
- **who**

Perl, Ruby, Python, Erlang, PHP and C# interfaces are still a work in progress.

To try the Perl examples:

```
% cd bindings/perl
% perl Makefile.PL && make
% perl -Mblib examples/ifconfig.pl
```

3 License

SIGAR is licensed under the Apache License, Version 2.0

4 Binaries

The SIGAR binary distribution contains the following files in **sigar-bin/lib**:

File	Language	Description	Required
sigar.jar	Java	Java API	Yes (for Java only)
log4j.jar	Java	Java logging API	No
libsigar-x86-linux.so	C	Linux AMD/Intel 32-bit	*
libsigar-amd64-linux.so	C	Linux AMD/Intel 64-bit	*
libsigar-ppc-linux.so	C	Linux PowerPC 32-bit	*
libsigar-ppc64-linux.so	C	Linux PowerPC 64-bit	*
libsigar-ia64-linux.so	C	Linux Itanium 64-bit	*
libsigar-s390x-linux.so	C	Linux zSeries 64-bit	*
sigar-x86-winnt.dll	C	Windows AMD/Intel 32-bit	*
sigar-amd64-winnt.dll	C	Windows AMD/Intel 64-bit	*
libsigar-ppc-aix-5.so	C	AIX PowerPC 32-bit	*
libsigar-ppc64-aix-5.so	C	AIX PowerPC 64-bit	*
libsigar-pa-hpux-11.sl	C	HP-UX PA-RISC 32-bit	*
libsigar-ia64-hpux-11.sl	C	HP-UX Itanium 64-bt	*
libsigar-sparc-solaris.so	C	Solaris Sparc 32-bit	*
libsigar-sparc64-solaris.so	C	Solaris Sparc 64-bit	*
libsigar-x86-solaris.so	C	Solaris AMD/Intel 32-bit	*
libsigar-amd64-solaris.so	C	Solaris AMD/Intel 64-bit	*
libsigar-universal-macosx.dylib	C	Mac OS X PowerPC/Intel 32-bit	*
libsigar-universal64-macosx.dylib	C	Mac OS X PowerPC/Intel 64-bit	*
libsigar-x86-freebsd-5.so	C	FreeBSD 5.x AMD/Intel 32-bit	*
libsigar-x86-freebsd-6.so	C	FreeBSD 6.x AMD/Intel 64-bit	*
libsigar-amd64-freebsd-6.so	C	FreeBSD 6.x AMD/Intel 64-bit	*

*** == Required to run on listed OS + Architecture combo**

For example, minimal requirements to use the SIGAR Java API on Windows would be **sigar.jar** and **sigar-x86-winnt.dll**

5 Versions

SIGAR uses a common versioning scheme: *major.minor[.revision[.build]]*
The *minor* number is incremented when binary compatibility is broken between the .jar and the native libraries. This can happen if a new API function is added or a field is added to an existing one. The *revision* number is incremented for bug fixes or enhancements specific to the .jar or a specific native library. For example, an issue where memory metrics were incorrect on Solaris machines with 8GB of RAM. Such as change does not impact the interaction between **sigar.jar** and the native library. The *build* number is generated by the Hudson CI system and is unlikely to be the same across all binaries.

6 Download

SIGAR binary and source release packages are available from sourceforge.

The current stable release is version 1.6.4.

Note that the 1.6.x releases are focused on the C and Java APIs. The 1.7 release of SIGAR improves on other language bindings: Ruby, Python and Perl. We recommend using the sigar.git master branch for these language bindings until 1.7 is released.

Binary snapshot builds are available from svn.hyperic.org and hudson.hyperic.com.

7 Source

The SIGAR master Git repository is hosted on github and can be checked out using:

```
% git clone git://github.com/hyperic/sigar.git sigar.git
```

To use the stable branch:

```
% git checkout --track -b sigar-1.6 origin/sigar-1.6
```

Subversion users can checkout a mirror of the stable branch via:

```
% svn co http://svn.hyperic.org/projects/sigar_mirror/branches/sigar-1.6
```

The development/unstable mirror is available in the trunk:

```
% svn co http://svn.hyperic.org/projects/sigar_mirror/trunk sigar
```

8 Building SIGAR

The native library, Java JNI bindings and Java classes are built using the same build system which has the following requirements:

- JDK 1.4 or higher
- Ant 1.6.5 or higher
- Perl 5.6.1 or higher
- C compiler

Perl is required to generate much of the JNI code as well as many of the Java classes. The ant build system uses a modified version of
cpptasks which works on all platforms supported by SIGAR. The native library can be used by the following languages:

- C/C++
- Java (sigar.jar auto-loads the native library)
- Perl (requires bindings/perl build)
- .NET C# (requires bindings/csharp build)
- Ruby (requires bindings/ruby build)
- Python (requires bindings/python build)
- PHP (requires bindings/php build)
- Erlang (requires bindings/erl build)

Note

The native library includes **Java_org_hyperic_sigar_*** functions, however there are no JRE dependencies for pure C/C++ applications using SIGAR. The Java JNI interface uses runtime linking and a function-pointer interface which allows us to combine both the **sigar**_ C API and JNI implementations into a single library. If this is not desirable, it would be trivial to have your own project build system simply compile **src/*.c** and **src/os/$osname/*.c**, then link the object files directly into your library or application.

Example build:

```
% tar -zxf hyperic-sigar-1.6.4-src.tar.gz

% cd hyperic-sigar-1.6.4-src/bindings/java
% ant
```

Results in:

```
% ls -1 sigar-bin/lib
libsigar-universal-macosx.dylib
sigar.jar
```

Test using:

```
% java -jar sigar-bin/lib/sigar.jar test
```

9 Projects using SIGAR

Cohesion Platform ->
GridGain ->
Hyperic HQ
Hypertable ->
JBoss Operations Network
MySQL Enterprise Monitor ->
MySQL I_S
P.Bio ->
RHQ
Rio ->
SemanticLIFE ->
Terracotta ->

10 Bugs

SIGAR bugs are tracked using JIRA and issue numbers are included with git commits and the **ChangeLog**. Please submit bugs and/or patches to the SIGAR Developers Forum.

11 Forums and Mail Lists

- SIGAR Users Forum | sigar-users-subscribe@hyperic.org
- SIGAR Developers Forum | sigar-dev-subscribe@hyperic.org

12 History

SIGAR was designed and implemented by Doug MacEachern at Covalent Technologies starting in September of 2002 and has continued with Hyperic as a core component of the HQ product.
SIGAR is not the first attempt to provide a cross-platform API to collect system information. In fact, SIGAR was inspired by libgtop which has been around since 1998 or so and was at version 1.90 when the SIGAR project was started. Doug released a Perl interface to libgtop in December of 1999 upon which Stas Bekman implemented Apache::VMonitor.
Based on that experience, Doug had become a big fan of the libgtop concept, but in practice only had success using it on Linux and with some struggles on Solaris. At the time the SIGAR implementation decision was made, libgtop had become a GNOME component for which there only appeared to be ongoing support for Linux and did not have implementations for other required platforms such as Windows, HP-UX, AIX and Mac OS X. Another requirement was a thread-safe Java interface and the ability to ship a single package containing binaries for all supported platforms. Long story short, libgtop was not the right fit for the requirements but the concept was: An abstract interface defined by C structures and function prototypes with an underlying implementation for each platform to gather the data. So that concept was borrowed, but the implementation was done from scratch and has continued to evolve over the years with broader platform support, more features and language bindings.

13 PTQL

13.1 PTQL (Process Table Query Language)

Hyperic SIGAR provides a mechanism to identify processes called Process Table Query Language. All operating systems assign a unique id (PID) to each running process. However, the PID is a random number that may also change at any point in time when a process is restarted. PTQL uses process attributes that will persist over time to identify a process.

13.2 PTQL Syntax

PTQL Queries must be in the following format:

```
Class.Attribute.operator=value
```

Enclose the query in quotes if it contains any spaces. For example:

```
sigar> ps "Exe.Name.ct=Program Files"
```

Where:

- **Class** is the name of the Sigar class minus the Proc prefix.
- **Attribute** is an attribute of the given Class, index into an array or key in a Map class.

- **operator** is one of the following for String values:
  - **eq** - Equal to value
  - **ne** - Not Equal to value
  - **ew** - Ends with value
  - **sw** - Starts with value
  - **ct** - Contains value (substring)
  - **re** - Regular expression value matches

    operator is one of the following for numeric values:
  - **eq** - Equal to value
  - **ne** - Not Equal to value
  - **gt** - Greater than value
  - **ge** - Greater than or equal value
  - **lt** - Less than value
  - **le** - Less than or equal value

Multiple queries must delimited by a comma.

13.3 PTQL Attributes

The attributes used in PTQL are directly from the sigar.Proc* classes. This document will outline the attributes most commonly used for identifying processes, the complete set of Proc* classes and attributes can be found in the SIGAR javadocs.

- **Pid.Pid** - The process ID
- **Pid.PidFile** - File containing the process ID
- **Pid.Service** - Windows Service name used to pid from the service manager
- **State.Name** - Base name of the process executable
- **CredName.User** - User Name of the process owner
- **CredName.Group** - Group Name of the process owner
- **Cred.Uid** - User ID of the process owner
- **Cred.Gid** - Group ID of the process owner
- **Cred.Euid** - Effective User ID of the process owner
- **Cred.Egid** - Effective Group ID of the process owner
- **Exe.Name** - Full path name of the process executable
- **Exe.Cwd** - Current Working Directory of the process
- **Args.*** - Command line argument passed to the process
- **Env.*** - Environment variable within the process
- **Modules.*** - Shared library loaded within the process

13.4 PTQL Building

The process of building a process query will vary depending on the application and the need to identify a unique process or group of processes. For these examples, we will use the sigar shell. The sigar shell is started using the following command:

```
% java -jar sigar.jar
```

The sigar.jar file is located in the agent/pdk/lib directory within HQ and sigar-bin/lib within the standalone SIGAR distribution. When the shell is started, you'll be given a prompt:

```
sigar>
```

The help command will show the complete list of top-level commands. We will focus on the handful that are useful for building PTQL queries:

- **ps** - Process Status
- **pargs** - Process Arguments
- **penv** - Process Environment
- **pfile** - Process File Information
- **pinfo** - Other Process Info

Each of the commands listed above require an argument of either a process ID or PTQL query. For certain commands like ps you can use tab completion in the shell to see the possible values.

13.5 Simple Process Identification

The simplest of queries can use 'State.Name', the basename of the process executable, to identify a process. For example, the cron daemon on a Linux system:

```
sigar> ps "State.Name.eq=crond"
560     root    13:03   536K    536K    456K    S       0:0     syslogd
```

This approach works to uniquely identify other daemons, such as 'syslogd', 'dhclient' and others where there should only be 1 process with the given name. However, in the case of a daemon such as sshd, there will likely be multiple instances:

```
sigar> ps "State.Name.eq=sshd"
729     root    13:05   1.4M    1.4M    1.3M    S       0:0     /usr/sbin/sshd
1124    root    13:53   2.0M    2.0M    1.8M    S       0:0     /usr/sbin/sshd
1126    dougm   13:53   2.2M    2.2M    2.0M    R       0:2     /usr/sbin/sshd
```

The easiest way to find the listening sshd server is to use the pid file:

```
sigar> ps "Pid.PidFile.eq=/var/run/sshd.pid"
729     root    13:05   1.4M    1.4M    1.3M    S       0:0     /usr/sbin/sshd
```

While this will also work on Windows platforms, it is less common to find a pid files, especially for Windows specific products. It is very common however, for a server process to be registered as Windows Service. Example for the Windows Event Log service:

```
sigar> ps "Pid.Service.eq=Eventlog"
1308    SYSTEM  16:02   5.0M    2.1M    -       R       0:39    C:\WINDOWS\system32\services.exe
```

If you happen to be running Cygwin sshd:

```
sigar> ps "Pid.Service.eq=sshd"
4408    SYSTEM  15:58   2.1M    1.2M    -       R       0:0     C:\cygwin\bin\cygrunsrv.exe
```

13.6 Identifying a server that uses different names

Certain server applications, such as Apache, may have a different 'State.Name' depending on platform, vendor or configuration.

- *httpd* - The standard name on unix platforms
- *Apache* - The standard name on windows platforms
- *httpsd* - Apache-SSL
- *httpsd.prefork, httpsd.worker* - Covalent's Apache ERS product
- *apache2* - gentoo

A regular expression can be used to match any of these flavors. Example on a Linux system:

```
sigar> ps "State.Name.re=^(https?d.*|[Aa]pache2?)$"
6807    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6808    dougm   15:10   3.0M    3.0M    1.6M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6809    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6810    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6811    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6812    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
6813    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
```

Example on a Windows system:

```
sigar> ps "State.Name.re=^(https?d.*|[Aa]pache2?)$"
5124    SYSTEM  15:11   5.7M    2.6M    -       R       0:0     c:\Program Files\Apache
Group\Apache2\bin\Apache.exe
6016    SYSTEM  15:12    10M    8.9M    -       R       0:0     C:\Program Files\Apache
Group\Apache2\bin\Apache.exe
```

13.7 Identifying the parent process of a forked daemon

In the apache examples above, we were able to use a regular expression to find Apache server processes with different names. However, the examples returned a process listing for the parent process as well as its children. PTQL operators support the notion of a parent flag, 'P', which converts the given query branch to get the attribute of the parent process. For example:

```
sigar> ps "State.Name.eq=httpd,State.Name.Pne=httpd"
6807    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
```

In this example, the first branch of the query, 'State.Name.eq=httpd' will match several processes. The second branch, 'State.Name.Pne=httpd', only matches if the State.Name of the parent process is NOT equal to httpd.

The hardcoded string 'httpd' in the second branch can be replaced with the special variable $1, which is the return value of the attribute (State.Name) in the first branch of the query:

```
sigar> ps "State.Name.eq=httpd,State.Name.Pne=$1"
6807    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
```

Let's say we change the query to where the first branch matches a certain username (CredName.User), with State.Name moving to the second branch, we then need to use '$2' to get the return value of State.Name:

```
sigar> ps "CredName.User.eq=dougm,State.Name.eq=httpd,State.Name.Pne=$2"
6807    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
```

Use of these variables is particularly useful when combined with our regex to find the parent process of any Apache flavor:

```
sigar> ps "State.Name.re=^(https?d.*|[Aa]pache2?)$,State.Name.Pne=$1"
6807    dougm   15:10   2.6M    2.6M    1.5M    S       0:0
/local0/dougm/apps/httpd-2.0.54/bin/httpd
```

13.8 Identifying a Unique Java Process

'State.Name' may be enough to identify certain processes, but this is almost never the case with java applications, where the executable basename is 'java' for all applications:

```
sigar> ps "State.Name.eq=java"
3872    dougm   16:12   241M    330M    -       R       6:8     java:org.jboss.Main
3888    dougm   16:15   211M    208M    -       R       7:33
java:com.ibm.ws.bootstrap.WSLauncher
6060    dougm   11:24    12M     12M    -       R       0:0
java:net.hyperic.sigar.cmd.Runner
```

The results are 3 processes: a JBoss server, a WebSphere server and the sigar shell itself.

Hey, why didn't eclipse show up in the listing? If you are on windows, certain java applications will use 'javaw' rather than 'java', simply adjust the query to use the 'sw' operator to match both:

```
sigar> ps "State.Name.sw=java"
3872    dougm   16:12   241M    330M    -       R       8:28    java:org.jboss.Main
3888    dougm   16:15   211M    208M    -       R       2:51
java:com.ibm.ws.bootstrap.WSLauncher
4232    dougm   09:26   154M    150M    -       R       3:13
javaw:org.eclipse.core.launcher.Main
3772    dougm   13:38    12M     12M    -       R       0:0
java:net.hyperic.sigar.cmd.Runner
```

To view the command line arguments for a specific process:

```
sigar> pargs 3872
pid=3872
exe=C:\j2sdk1.4.2_04\bin\java.exe
cwd=D:\jboss\bin
   0=>C:\j2sdk1.4.2_04\bin\java<=
   1=>-Dprogram.name=run.bat<=
   2=>-Xms128m<=
   3=>-Xmx512m<=
   4=>-Djava.endorsed.dirs=d:\jboss\bin\..\lib\endorsed<=
   5=>-classpath<=
   6=>C:\j2sdk1.4.2_04\lib\tools.jar;d:\jboss\bin\run.jar<=
   7=>org.jboss.Main<=
```

For most java applications, the main class name can be used to uniquely identify the process, in this case argument 7 is the JBoss main class name:

```
sigar> ps "State.Name.eq=java,Args.7.eq=org.jboss.Main"
3872    dougm   16:12   241M    330M    -       R       6:27    java:org.jboss.Main
```

Using the exact argument may not work depending on how the server is configured. Another alternative is to use -1, which means the last argument:

```
sigar> ps "State.Name.eq=java,Args.-1.eq=org.jboss.Main"
...
```

Again, this approach can also fall apart if there are arguments after the main class, using * will match any of the command line arguments:

```
sigar> ps "State.Name.eq=java,Args.*.eq=org.jboss.Main"
...
```