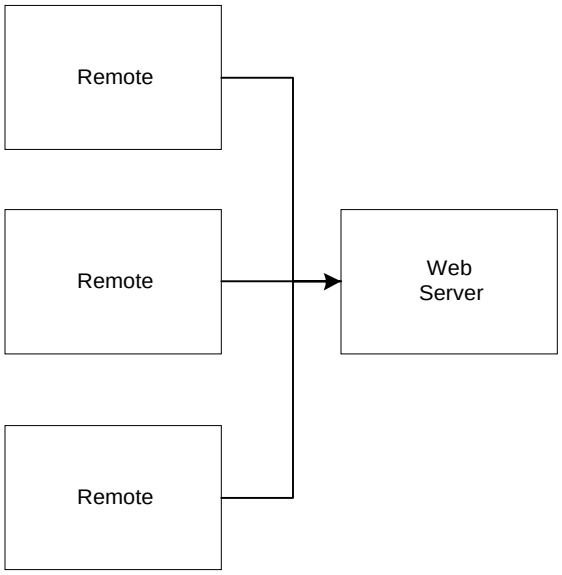
Document History

Author: Bruce W. Mohler
Date Last Updated: March 20, 2001

Please address all comments and defect reports about this document to Bruce Mohler at brucemohler@users.sourceforge.net.

Installation Overview

There are two "pieces" of software in syssumm: the **remote** software that generates the summary information and the **web server** software displays the summaries.



The "**remote**" part of syssumm is installed on each system that you wish to profile and is the Perl code that actually generates the summary of that system and e-mails it to your web server.

The "**web server**" part of syssumm is installed on your web-server; it handles incoming summaries and automatically installs new summaries or merges them into existing summaries and makes them available to users with a web browser via a CGI script. We'll actually install this part *first*.

Next, we make sure that the "web server" and "remote" pieces of software are **working together properly**.

Then, we find out how to **customize the appearance** of the web pages generated by the Perl script that generates the HTML.

Next, once summaries begin to be sent to the web server, there are steps that you can do to **tailor the summaries** for an individual system by adding additional information that could only be known by the system administrator (not a mere Perl script).

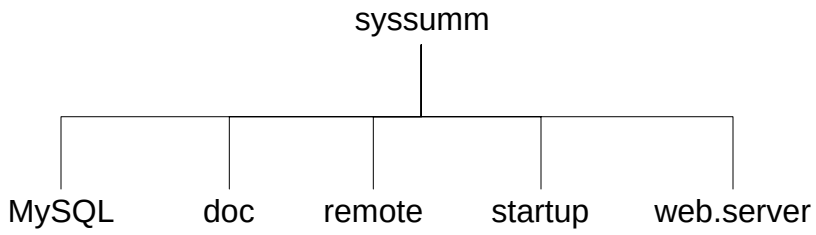
Then, we will customize the output from the “remote” part of syssumm by using **override files**.

Finally, we’ll take some steps to make your web server **more secure**.

The web server software is installed **only once** on your web server while the remote software is installed on each system to be summarized. Since the remote software is identical on each system, once it is installed on one system, you can use `rdist` to move it to each of the other systems you want to run it on.

What You Find in a Syssumm Tarball

When you un-tar the syssumm tarball, you’ll end up with a set of directories like:



The top level directory contains some text files like `CHANGES` (high level description of changes in each release), `CHECKSUMS`, `COPYING` (GNU General Public License, Version 2), `INSTALL` (pointers to this document), `MANIFEST` (a list of all of the files in the tarball), and `README` (a brief overview of syssumm).

Of the remaining 5 sub-directories, `remote` and `web.server` are the most important. The `remote` sub-directory contains all of the files needed on each system that you wish to profile. The `web.server` sub-directory contains all of the files needed on your web server system to process incoming summaries and make them available through to browsers.

The `MySQL` sub-directory contains some initial work at storing the system summaries in an SQL database. The `doc` sub-directory contains some stale files with lists of the developers and those on the original general mailing list. The `startup` sub-directory contains a sample “rc” file for a Linux box that can be adapted to other versions of UNIX.

Installing the Web Server Software

All of these steps take place on the system where the web server is. [Duh. Sorry.]

1. Copy all of the files in the `syssumm/web.server` sub-directory (except for the RCS sub-directory) to some neutral directory on your web server computer. An easy way to do this is to type "`make web.server.tar.gz`" in the syssumm top level directory and it will create a tarball for you of the files you need from the web server sub-directory.
2. "`su`" to root.
3. Create an account for syssumm. Note the home directory for this account. If the path to this directory is anything *other* than `/home/syssumm`, you'll need to edit the `incoming.pl` and `sysquery.pl` scripts. Search for the following lines:

   ```
   # CUSTOMIZE
   use lib "/home/syssumm";
   ...
   ```

   Change the path in the "`use lib ...`" statement to be the path to the home directory for the syssumm account. This will help these scripts to be able to locate the `common.pm` Perl module.
4. syssumm assumes that the Perl interpreter resides in `/usr/bin/perl` or that there is a link there pointing to the actual location of the Perl interpreter. The `install` script will create that symbolic link automatically if it doesn't exist.
5. Run the install script. See below for an excessively verbose description of what this script does (or just "use the source").

   ```
   ./install
   ```

   Note that there are **two** "install" scripts. One is a shell script merely called `install` and the other is a Perl script called `install.pl`. You want to run the **shell** script. Once it gets done, it will go ahead and run the Perl script itself. There is code in the scripts that helps you run them in the correct order.

   This script has 3 command line arguments. "-d" turns on some additional debugging output. "-f" allows you to run the script not as root ("force" it to run); not everything may work the way you want unless you're su'd to the "syssumm" account. "-h" displays some help on the install script then stops.
6. Note that the install script may stop at certain points to prompt you to do certain things (like creating the `syssumm` account). Just follow the instructions and restart the `install` script.
7. If you are doing this for the first time and you've just created the `syssumm` account, one thing that you can do to quickly make testing easier is to go into the `/etc/passwd` file and temporarily change the uid for the `syssumm` account to *your* uid. This way you own the `syssumm` home directory and all of the files. Once you're done, set the uid back to what you created the account with.
8. Once the `syssumm` account exists, the first time you run the `install` script it will prompt you for information that it will save and re-use the next time that you run it. The following is an example of what this dialog might look with your responses **bolded**.

```
Enter e-mail address for your local System Administrator:
-> root@pachycephalosaurus.saic.com
Enter type of web server (e.g., apache, netscape)
-> apache
Enter path to your CGI directory:
-> /home/httpd/cgi-bin
Enter path to your web server document root directory:
-> /home/httpd/html
Enter the path to your local mail command:
-> /bin/mail
Enter the URL for the web server:
-> http://ornithomimus.saic.com/
Enter "name" for webmaster on bottom of web page:
-> Webmistress
Enter e-mail address for Webmaster:
-> root@ornithomimus.saic.com
```

The type of web server should be a single word, lower case that indicates the type of web server (*e.g.*, "apache", "netscape", *etc*.). At the present time, the value of this field is not validated. The only *significant* value is "netscape" which causes the debug output to be written to a log file in `/tmp` rather than to STDERR. The Netscape web server doesn't seem to be able to distinguish between STDOUT and STDERR.

Note that the answer to the "local mail command" prompt is *usually* `/bin/mailx` except for Linux which retains the older-style name `/bin/mail`.

The file where the information is stored is called `web.server.info` and is located in the `syssumm` account home directory. It is an ASCII file that can be edited with your editor of choice.

9. At this point, you have a choice to make. When summaries are generated and arrive at the web server, they are stored in their own sub-directory. The name of the sub-directory is either the FQDN (Fully Qualified Domain Name) or the *shorter* hostname of the remote system being profiled. The **default** behavior is to name the sub-directory using the FQDN. If all of your systems are in the same domain and you'd just prefer to use the hostname, then edit the `common.pm` file that is in the `syssumm` account home directory. Find the following lines:

   ```
   # CUSTOMIZE - only comment one of the choices...
   $HostNameType = "FQDN";
   # $HostNameType = "hostname";
   ```

   Comment in the first `$HostNameType` line and *un*-comment the second line so that they look like:

   ```
   # CUSTOMIZE - only comment one of the choices...
   # $HostNameType = "FQDN";
   $HostNameType = "hostname";
   ```

10. If you changed the `syssumm` account uid to your own to assist your testing, change it back now.

The following information is taken from the comments from the beginning of the `install.pl` Perl script and describe what the script will do.

```
#
# Makes sure we're running as root (you can force *not*
# running as root by using the "-f" command line option
# - YMMV).
#
# Makes sure all required files are in current working
# directory. (It checks the MANIFEST.)
#
# Makes sure syssumm account is setup properly.
#
# Copies incoming.pl to syssumm home directory and makes
# sure .forward file exists.  Sets ownership and
# permissions.
#
# Checks for web server configuration file in syssumm
# home directory and creates it if it doesn't exist.
# This file contains path to CGI directory and base of
# Root of directory tree for HTML documents.  Sets
# ownership and permissions.
#
# It copies sysquery.pl into the CGI directory.
#
# It make sure that the syssumm sub-directory exists
# under HTML root directory.  If syssumm.format doesn't
# exist, it copies that file into the syssumm
# sub-directory otherwise it prompts the user to let
# him/her know that the file exists but may have local
# modifications.
#
```

All of the incoming summaries arriving as e-mail get sent to the "syssumm" account on the web server machine. The directories under the web server document root are all owned by the syssumm account so that it can update them as new summaries arrive.

One problem that we had early on in the development process were that different people had their web servers under different directories and it was a "pain" to re-code the directory paths every time that we sent out a release. This install script fixes this.

11. If you are re-installing (*i.e.*, you already have summaries that been installed on the system), the `install` script will have told you that a format file already exists in the syssumm directory under the document root of the web server. You'll need to compare the RCS log comments at the top of the new format file to your format file to see if you need to update anything manually.

12. If you are re-installing (*i.e.*, you already have summaries that been installed on the system), run the updatelists.pl script.

    This script will create 4 files in the same directory as the format file which are lists of the Operating Systems, Vendors, Locations, and Organizations in your existing summaries. These will be available to users to search over from the web browser interface. As new summaries arrive on the system, the lists will be automatically updated by the incoming.pl script.

If you are running your web server on a system using an `smrsh`-enabled sendmail, then you will need to perform the additional step: you will need to create a symbolic link in your `smrsh` control directory pointing to the `incoming.pl` script. This will allow sendmail to execute this script when incoming e-mail messages arrive for the syssumm account.

For example, assume that on your system the `smrsh` control directory is in `/etc/smrsh` (Red Hat Linux 6.0) and that the syssumm home directory is in `/home/syssumm` and that that is where the `incoming.pl` script resides.

```
ln -s /home/syssumm/incoming.pl /etc/smrsh/incoming.pl
```

Disclaimers:

- This script will probably not work really well (at all) under a non-UNIX operating system.
- If you rename your CGI scripts to something like *.cgi, the current install script cannot handle this. The CGI script as packaged in syssumm is called `sysquery.pl` and that is what is copied into the CGI directory.

  The advantage of this is that if you maintain your "diverse" suffixes on your CGI files, you can tell whether it's a shell script, C program, Perl script or Python script by looking at the suffix. It sometimes makes it easier for the administrator. You may have local policies that make it easier for someone else...

You'll need to perform the following procedure whether you installed the web server scripts using the `install` script of manually.

The following paragraph describes how to set up pointers to the CGI script in a web page:

On the web page where you wish to place a link to the system\summary query page, you should put something like:

```
<a href=/cgi-bin/sysquery.pl?target=form>System Summaries</a>
```

This will generate the HTML web page that a user can use to either request a specific system or display all of the systems for which system summaries exist.

Note that the HTML web page that is generated is very primitive and will not follow your local stylistic standards. It is meant to be a template that you can modify to get working.

In order to generate the web page which summarizes all existing system summaries, use:

```
<a href=/cgi-bin/sysquery.pl?target=showall>System Summaries</a>
```

Any other value for the target argument is assumed to be either hostname or the fully qualified domain name of an existing system summary.

The sysquery.pl Perl script is capable of generating frame-based HTML. It is possible to specify a frame in which to display the target's summary information. You do this by specifying the name of the frame as a cgi-argument:

```
sysquery.pl?target=form&frame=<frame_name>
sysquery.pl?target=showall&frame=<frame_name>
sysquery.pl?target=<fqdn>&frame=<frame_name>
```

For more information, see the file README.sysquery.and.frames in the web.server sub-directory.

There are some conditions where you may not want to create a "syssumm" account for the processing of incoming e-mail and ownership of directories and files. While this is *not* recommended, it is possible. You would need to change the value of the variable "$SyssumUserid" for each of the scripts in the web.server sub-directory to the name of the account you wish to use. This mode of using syssumm is *deprecated*.

Installing the Remote syssumm Software

These steps take place on each system that you wish to profile. [Duh. Sorry, again.]

1. Copy all of the files in the `syssumm/remote` sub-directory to some neutral directory on the computer system you wish to profile. An easy way to do this is to type "`make remote.tar.gz`" in the syssumm top level directory and it will create a tarball for you of the files you need from the remote sub-directory.

   As you are choosing "some neutral directory" to put the scripts in you might be tempted to just put it under your home directory on that system. Resist that urge.

   Put the scripts in a common location across all of your servers. Choose a directory like `/usr/local/syssumm` or `/opt/syssumm` so that as your system administrators change server responsibilities, they will always know where the scripts on each server are.

2. syssumm assumes that the Perl interpreter resides in `/usr/bin/perl` or that there is a *symbolic* link there pointing to the actual location of the Perl interpreter. Please make sure that things work this way on your web server.

3. On the remote system, gunzip and un-tar the tarball and it will place the files in a sub-directory called `remote` just under the current working directory.

4. Change to that directory.

5. There is a shell script called "`x`". This file is heavily comments for your benefit.

6. First, edit this file. Find the lines near the top that look like:

   ```
   WebServer="webserver.xxx.com"
   ```

   The "webserver" field should be the **nodename** of your web server (not the fully qualified domain name). The "xxx.com" should be your local DNS domain. For example, for testing here at SAIC, those field look like:

   ```
   WebServer="allosaur.saic.com"
   ```

   Our *production* web server is called `syssumm.saic.com` so that the line usually looks like

   ```
   WebServer="syssumm.saic.com"
   ```

   You might want to edit this file *earlier* (before you tar up the remote directory) since you'd have to make the same changes to every system that you'll be running the remote script on.

7. Note that there are 3 separate tests in the `x` script:

   - The first generates the summary of the local system but doesn't e-mail it anywhere. This allows you to make sure that the `syssumm.pl` script is working.
   - The second mails a pre-existing summary file (output from the previous test) to the web server. This allows you to make sure that the `incoming.pl` script is working okay.

- The third runs the `syssumm.pl` script and e-mails the output to the web server. You can plagiarize from this when you get ready to generate system summaries on a regular basis in Production.

Each of these tests, by default, is **commented out** in the script. You need to uncomment the test you need to run.

8. Run the each of the tests in the `x` script in order.
9. You can then plagiarize from the `x` script if you wish to further automate this process at your site.

   For example, if you'd like to run the script by hand, you can just type in

   ```
   ./syssumm.pl -m syssumm@your.web.server -v 2>&1
   ```

   where *`your.web.server`* should be the fully qualified domain name of your web server.

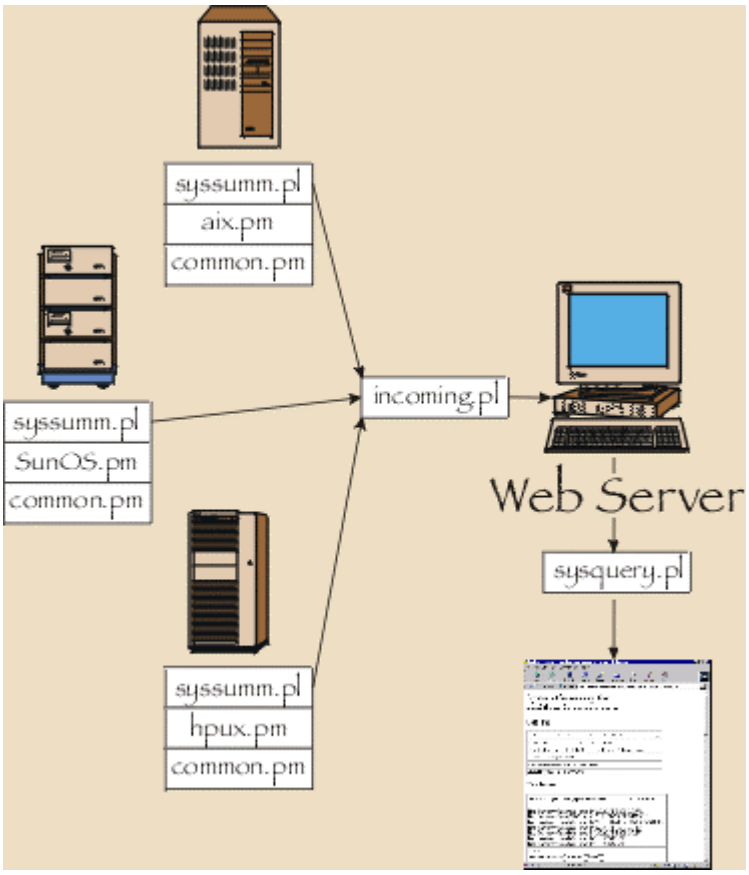
   If you wish to run the syssumm script each time you reboot your system, there is an example `rc` file in the `startup` sub-directory under the main syssumm directory.

That's it!

If you have any problems with the installation process, please mail a description of them to the `syssumm-bugs@lists.sourceforge.net` mailing list.

Getting the Pieces to Work Together

One thing that will help here is to imagine how the various pieces work together. A picture is worth a thousand words:



1. This step occurs on the system where your web server is.

   Start by renaming the `.forward` file in the `syssumm` home directory. This will allow you to get the remote piece of the software working so that it will build the summary and e-mail it to the `syssumm` account on your web server.

   You can then use the e-mail tool of your choice to verify that the summaries are arriving properly in the mailbox of the `syssumm` account.

2. The next few steps occur on a system that wish to summarize.

   Get the remote stuff running first on at least one system. I usually tar up the `remote` sub-directory and just un-tar it in my home directory on the system I want to summarize.

   By the way, there's an `distfile.eg` (rdist) file in the top level source directory. The first thing to do is to rename it to just `distfile`. I have a workstation that I primarily work from. Each time I install the remote software on a system, I add that system name to the list of systems in the `distfile`. Then, when I "receive" an update of the remote software, I just `rdist` the updated source code to all of the systems where I have the remote software installed in one easy step.

3. First, I run the script that generates the summary with the following command line:

   ```
   ./syssumm.pl -d -v 2&gt;&amp;1 | more
   ```

This will run the `syssumm.pl` script with debugging and verbose output turned out, piping the output of STDOUT and STDERR to the `more` command so that you can look at it at your leisure. Examine the output to make sure that it seems reasonable.

For operating system versions that are already *supported*, this should be fairly routine, however, Paul Farrell has pointed out that he and I have 2 HP-UX 10.20 systems that produce different output from the `ioscan` command for the hardware addresses of SCSI controllers. This makes coding patterns for filtering output alittle interesting.

4. Look at the `syssumm.out` file. The script creates this file. It is what will be e-mailed to the web server in the next step of testing.

5. If you are having a problem in a certain category of software, in order to speed up the "turn around time" during test, you can test *just* that category.

   The -C and -X command line options are a contribution from Jeff Putsch and allow testing of one or more categories without having to change the source code. Here is his description:

   ```
   I changed syssumm.pl so it doesn't need to be edited to turn
   on and off different categories while debugging.  I added a -
   C switch to turn on categories and a -X switch to turn off
   categories. I made them mutually exclusive.

   For example:

   # run all BUT Hardware and Network categories
   syssumm.pl -X Hardware,Network

   # run only Software categories
   syssumm.pl -C Software

   The default is to run all categories. Again this is a hack, I
   should be setting the default behavior for -C based upon the
   Category hash, but I hard-coded it for now.
   ```

   Fortunately, work since his contribution has fixed the problem of hard-coded categories.

6. Once things look clean, you're ready to invoke `syssumm.pl` script "for real". Use something like the following command line:

   ```
   ./syssumm.pl -m syssumm@your.web.server -v 2>&1
   ```

   The "`-m email`" command line option tells the `syssumm.pl` script where to e-mail the `syssumm.out` file containing the summary information.

   Substitute the fully qualified domain name of *your* web server for the *`your.web.server`* in the command line example above.

   In case you have problems mailing your summary to the web server system, here is some information from Frank Crawford on using the e-mail command line options (-m, -s, -u) to the `syssumm.pl` script:

`syssumm.pl` uses a Perl module called `Sendmail.pm` to deliver the message to the web server. Normally, the only command line option you would ever need to use to deliver the message to the web server is the "-m" command line option:

```
syssumm ... -m "syssumm@your.web.server" ...
```

where the address is to the syssumm account (usually "`syssumm`") on the web server.

However, there may be some networks where the webserver e-mail is delivered via an `MX` record. The Sendmail.pm module does *NOT* support `MX` processing, so it would blindly try to deliver the message to the server portion of the e-mail address specified by the "-m" command line option. You can over-ride this and "emulate" `MX` processing with the "-s" command line option:

```
syssumm ... -m "syssumm@your.web.server" -s other.server ...
```

In this example, the message for "`syssumm@your.web.server`" would actually be delivered to the "other.server" server for actual delivery.

Note that the code that drives the `Sendmail.pm` module will try to deliver the message to the following servers (in this order):

- SMTP host (if specified via "-s" option)
- localhost (if email available)
- host portion of the e-mail address (if SMTP host *NOT* specified)

The reason that the summary is mailed to the localhost is that if the web server is not available, the SMTP process on the local host will queue the mail message until the web server is available. This means that you won't have summaries "dropping on the floor" if, for any reason, you need to take your web server down.

7. There are times when there are certain checks that you **never** want performed at your site. For example, there was a site that had several thousand printers and running the section which summarized the printers configured on the local system took an hour or more and basically added noise to the resulting profile rather than useable information.

   If you are in a situation like this, you can create a file called `local.exclusions` which should be in the same directory as `syssumm.pl` and the OS modules. This file should contain a list of categories/sub-categories that you do **not** wish to run. New source releases will never touch this file.

   For example, say that you're in the same situation of having thousands of printers and wish to exclude the printer configuration from your summaries. Further assume that for some really obscure reason, you don't care about whether or not your running a font server. Create the `local.exclusions` file containing the following lines:

   ```
   Software:Printers
   Software:FontServer
   ```

   Note that the category and sub-category must be separated by a ":" (colon), must be on their own line, and must correspond to a real category and sub-category. You can check the `%Routines` hash near the beginning of `syssumm.pl` for a complete list of these.

8. The remaining steps will take place on your web server system.

   Next, save one of the messages in the `syssumm` mailbox to a file in the `syssumm` home directory so that you can use it as input to the `incoming.pl` script.

   Type the following lines to feed the "summary" to the `incoming.pl` script while running the script in debug mode.

   ```
   ./incoming.pl -d <input_file
   ```

   All of the debug output will be written to a file called `incoming.log` in the `syssumm` home directory. Any errors will appear during the running of the script or in the debug log file.

9. Once the `incoming.pl` script is working, move the `.forward` file back into place so that `sendmail` will automatically feed incoming messages straight to the `incoming.pl` script.

   You should be all set at this point. You can keep the "-d" command line option in the `.forward` file if you wish to continue monitoring incoming summaries via the `incoming.log` file.

10. You can also turn on debugging in the `sysquery.pl` script unfortunately not from the command line. Within the script itself is the following line:

    ```
    $Debug = 0;          # set this to 1 during debugging
    ```

    By setting the value of the `$Debug` variable to 1, debugging is turned on.

    For web browsers other than Netscape, the debugging output is written to STDERR and is appended to the web server error log. For the STDERR-impaired Netscape, output is written to `/tmp/sysquery.log`.

Customizing the Appearance of the Web Pages

So, you don't like the default gray background of the syssumm web pages? Your site has a standard CCS file that they use for all of their web pages?

This section will describe how to customize syssumm to follow your standards. First, the key area we're talking about on each web page is the end of the <head> section and the beginning of the <body> section. syssumm, out of the box, will generate the following code for these areas:

```
<!-- Built-in header information -->
</head>
<body>
```

Pretty boring. New in Release 15 of the software is a file that lives in the `$DOCROOT/syssumm` directory (along with the format file) called `syssumm.header`. The purpose of the file is to allow you to place additional directives in the “head” section and to modify the opening “body” tag. By default, this file looks like:

```
<!-- External header information -->
</head>
<body>
```

The comment is in both places so that you can tell where the "HTML" code is coming from. This file is installed using the `install` script that is used to install the web server software.

If you'd like to change the appearance of your web pages, do the following:

1. Edit the `syssumm.header` file. For example, to change the background color of your web pages, change the <body> line to look like:

```
<body bgcolor="#5F9F9F">
```

2. Assume that you want to add a reference so that the syssumm output has a background color and includes a local CSS file. Edit the file to look something like:

```
<!-- External header information -->

<link rel=stylesheet type="text/css" href="local.css">
</head>

<body bgcolor="#5F9F9F">
```

These examples should give you an example of what you can begin to do to customize the appearance of your syssumm output.

Tailoring the Summaries

Remember the "PROTECTED" fields? These are fields that you can use to add system-specific information to a summary (*e.g.*, installation procedure, boot-up or shutdown procedures, *etc.*).

All of the summaries are stored under your web server's Document Root directory in a sub-directory called `syssumm`. In this directory there is a sub-directory for each system that you have summarized. This allows you to keep all of the files for a system in that system's sub-directory.

Within a system's own sub-directory, the summary is named `syssumm.info`. If multiple summaries have been generated, the previous summary is saved as `syssumm.info.bak`.

Within a system's summary, the PROTECTED fields say "To be provided.". When you see this, you can replace this phrase with the appropriate information. Each time you re-run the `syssumm.pl` script on the remote system to create an updated summary, the PROTECTED fields will not be over-written, so your customized fields will survive newly generated summaries.

For example, say that you want to specify the location of a particular system. Edit the summary and find the line that looks like:

```
General:Location:To be provided.
```

Change the text "To be provided." to whatever is descriptive of the location at your site. For example:

```
General:Location:In the broom closet, next to the sink.
```

In addition, you can include external files rather than typing in a lot of text. To include an external file use the following syntax:

```
#include path
```

where `path` is the path to a file to be included.

It's customary to keep external files in the same directory with a system's summary. Let's assume that you have typed in a system's installation procedure and you'd like it included automatically in the

summary. We'll further assume that the name of the file containing the installation procedure is `installation.procedure`. Catchy, no?

Edit the `syssumm.info` file and find the line that looks like:

```
Software:InstallationProcedure:To be provided.
```

Change the text "To be provided." to be the "#include" directive followed by the path to the file containing the installation procedure. The line should look like this:

```
Software:InstallationProcedure:#include installation.procedure
```

Note that this is just a text file that can contain HTML tags as well as text. You can use the HTML tags to create an ordered list, each step automatically numbered by the viewer's browser.

```
<h3>Installing Red Hat Linux 6.1</h3>

<p><ol>

<li> Put the Red Hat CD-ROM in the drive and boot up the system.
<li> At the boot: prompt, just press the Enter key.
...

</ol>
```

Note that you can have multiple `#include` directives in a single protected field.

```
Software:InstallProcedure:#include file.1 #include file.2 ...
```

Note that you can also include graphics files of any type supported by typical web browsers. syssumm recognizes files with any of the following suffixes: `.jpg`, `.jpeg`, `.gif`, and `.png`. Rather than being included in-line, the path to the graphic file will be wrapped inside a `<img src=path...>`.

```
Software:InstallProcedure:#include install.part.1 #include illus.1
...
```

Using Override Files

One of our developers at Telcordia (Ed Moroch) had a problem where he had lots of system administrators taking care of lots of servers. The system administrators couldn't update the PROTECTED fields because they were allowed access to the web server.

The solution that Ed came up with was an override file that lived on the same system that the (remote) `syssumm.pl` script is run on. This file, if present, will override any derived values in the output from `syssumm.pl` including PROTECTED fields.

For example:

```
General:Organization:49726
Software:ApplicationsInstalled:Apache, PCM Metrics, SSH,
SSH2, SyMon
Software:BackupProcedure:TSM
General:SystemHandle:sinosov
General:DateSystemInstalled:12/18/1989
General:VendorSerialNumber:835j2397
General:LargerPicture:<A HREF="http://scm.bcr.com/cgi-
bin/query?by=hosts&search=sinosov">Telcordia Host
Database</A>
General:Location:pza-2i89 Grid xyz
General:AssetTag:bcr001287840
```

Every time that `syssumm` processes a field that is in the overrides file, it will use the value from it rather than the derived value (if any).

Okay, then how do you setup an overrides file?

Create a file in the same directory as the `syssumm.pl` script called `syssumm.ovr`. In that file, put your required lines in the format:

```
Category:Subcategory:Value
```

Note that the `syssumm.pl` script does not validate that you have typed everything correctly. If it seems like the values are not being substituted properly, check your entry of the category and subcategory fields.

Securing the Web Server

At the present time, the only way to control the delivery of summaries is either through web server access controls or through protecting who is allowed to invoke the `sysquery.pl` script which generates the HTML from the stored system summaries.

Configuring the web server access controls is beyond scope of these web pages and I refer you to your web server configuration documents.

One of the ways that access to the `sysquery.pl` script was secured was by placing it in a sub-directory `cgi-bin/syssumm` and protecting that directory with an `.htaccess` file.

In the near future, the code the generates the HTML will be moved from `sysquery.pl` to `incoming.pl`. This will allow the HTML to be more static and to sit in its own sub-directory and would allow you to protect individual system summaries via `.htaccess` files.