



Urchin[®] 3.4

**Installation and
Administration Guide**
for Urchin Datacenter on UNIX[®]
Platforms

Industry-leading speed,
scalability, and efficiency

Advanced e-commerce
reporting provides true
ROI analysis

Intuitive, browser-based reports

Copyright 2002 Quantified Systems Inc., 2165 India Street, San Diego, CA 92101 USA. All rights reserved.

This product and document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Quantified and its licensors, if any. Quantified, Quantified Systems, Inc., Quantified Systems, Urchin, Urchin Pro, Urchin Dedicated, Urchin Enterprise, the Quantified logo, and the Urchin logo are trademarks, registered trademarks, or service marks of Quantified Systems, Inc. Sun, Sun Microsystems, Solaris, and the Sun logo are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. The Netscape Communications Corporation logo, Netscape, and Netscape Navigator are trademarks of AOL Time Warner Corporation. All other trademarks are property of their respective holders.

DOCUMENTATION AND SOFTWARE ARE PROVIDED "AS IS." QUANTIFIED'S LICENSOR, QUANTIFIED SYSTEMS, INC. ("QUANTIFIED"), MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE. QUANTIFIED DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. IN NO EVENT WILL QUANTIFIED, AND ITS DIRECTORS, OFFICERS, EMPLOYEES, OR AGENTS, COLLECTIVELY QUANTIFIED, BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL, OR INDIRECT DAMAGES INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE, ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF QUANTIFIED HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. PLEASE DON'T SUE US, IT ISN'T NICE

Document Version: sysadmin.unix.dc.3.400.rel.us.1.1.rel

Table of Contents

SUPER QUICKSTART GUIDE.....	5
1. INTRODUCTION.....	6
1.1 WHAT'S NEW WITH URCHIN 3?.....	6
1.2 SYSTEM REQUIREMENTS	7
1.3 PRIVACY AND SECURITY	8
1.4 UPGRADING FROM 2.X TO 3.X	9
2. SYSTEM READINESS.....	10
2.1 LOG FORMATS.....	10
2.2 LOG ROTATION	11
2.3 REVERSE DNS	12
2.4 URCHIN REPORTING REQUIREMENTS	14
3. INSTALLATION	16
3.1 DOWNLOAD AND EXPAND ARCHIVE	16
3.2 SELECT LOG ROTATION MECHANISM.....	16
3.3 CONFIGURE REVERSE DNS	17
3.4 CONFIGURE SYSTEM REPORT.....	17
3.5 CONFIGURE WEBSITE REPORTS.....	18
3.6 TEST RUN URCHIN	18
3.7 AUTOMATE URCHIN.....	19
3.8 LICENSING.....	19
4. CONFIGURATIONS.....	20
4.1 ONE LOG FILE PER SITE MODE	20
4.2 ONE BIG LOG FILE FOR ALL SITES MODE.....	22
4.3 SCRIPTING MANY SITES USING COMMAND LINE	23
4.4 NORMAL DECENTRALIZED REPORTING	24
5. FILTERING AND USING DYNAMIC URLS.....	25
5.1 SUBREPORT FILTERING	26
5.2 FILTER IN AND FILTER OUT.....	27
5.3 DYNAMIC URL FILTERING	28
6. MAINTENANCE	30
6.1 HOW TO ADD A NEW SITE REPORT	30
6.2 HOW TO EDIT OR DELETE SITES	30
6.3 USING THE SYSTEM REPORT	31
7. CREATING GROUPS OF USERS	32
7.1 CREATING A NEW GROUP	33
7.2 SPECIFYING GROUPS IN THE CONFIGURATION.....	33
7.3 SPECIFYING GROUPS ON THE COMMAND LINE	34

7.3 USING THE CGI SPECIFICATION	34
7.4 SPECIFY GROUPS MANUALLY	35
8. CUSTOMIZING AND CO-BRANDING URCHIN 3	36
8.1 CO-BRANDING THE TOP FRAME OF URCHIN	36
8.2 MODIFYING THE NAVIGATION.....	37
8.3 MODIFYING TEXT AND HELP TEXT	39
8.4 MODIFYING COLORS AND FONTS	39
8.5 MODIFYING ICONS	42
9. CONFIGURATION DIRECTIVES	45
9.1 GLOBAL DIRECTIVES.....	45
9.2 SITE SPECIFIC DIRECTIVES	48
9.3 COMMAND LINE OPTIONS	52
10. THE E-URCHIN MODULE (E-COMMERCE REPORTING).....	55
10.1 ARCHITECTURE	55
10.2 CONFIGURATION	56
10.3 ELF, IN DETAIL.....	56
APPENDIX A. SAMPLE INSTALLATIONS.....	58

Super QuickStart Guide

This compact installation guide is designed for experienced system administrators who are familiar with Urchin and their web servers. For complete installation details, refer to Chapters 2-4. Also, there are some samples in Appendix A. If you are calling Urchin from a script, refer to section 4.3.

1. Download: The latest version of Urchin can be found at <http://www.urchin.com/>. Expand the archive using `gunzip` and `tar -xvf`.

2. Log Rotation: It is important to process each hit only once. If you want Urchin to do log rotation, uncomment and edit the “RestartCommand” and “LogDestiny” directives in the “config” file.

3. Reverse DNS: Urchin has an extremely fast DNS module. You should turn off “HostnameLookups” in the webserver configuration and uncomment the “ProcessDNS” and “ResolverIP” directives in the Urchin “config” file. Be sure to edit the “ResolverIP” to point to a valid local DNS server.

4. System Report: If you have multiple sites, pick a unique web-accessible directory for the System Report and uncomment and edit the “SystemReport” and “SystemDirectory” directives in the “config” file (previous versions of Urchin called this the Webmaster Report).

5. Site Configurations: For each site in the server create a `<Report>...</Report>` entry. Uncomment, copy, and edit the example in the “config” file for each site. Each report should have a unique report directory. Set the “ReportName” to the domain of the site, without the “www.”. Error log processing is optional and should only be used if you have a unique log for each site. Currently, Urchin can only process Apache error logs.

6. Run Urchin: Run Urchin with the command, “`./urchin`” Urchin will process each site and create the reports in each directory. Fix any errors that occur such as permissions or missing directories.

7. View Reports: Point your browser at one of the report directories. Each report directory needs CGI capability. If you do not see the Urchin report, check the webserver configuration for the following lines. See Chapter 2 for more information. In Apache, you will need:

```
AddHandler cgi-script .cgi
DirectoryIndex index.html index.cgi
<Directory ...>
Options FollowSymLinks ExecCGI
</Directory>
```

8. Automate Urchin: Add Urchin to crontab. For midnight operation, the following entry does the trick:

```
0 0 * * * cd /usr/local/urchin3xxx;./urchin > ./log 2>&1
```

1. Introduction

Urchin Datacenter 3.4 is a software product targeted to hosting providers and businesses that serve multiple sites one or more servers. Urchin 3 is designed to run directly on the webserver, though this is not a requirement (a dedicated "log server" may be employed). Urchin processes log files generated from the webserver and creates easy-to-read, web-based reports that your company and/or user community will find immensely valuable and useful.

Urchin uses the Common Log Format, Extended Combined Log Format, or the W3C-extended formats as described in Chapter 2, and is designed to work with most web servers including Apache, Netscape, and IIS. By running each night (user specified), Urchin keeps log files in check, reports current, and minimizes system loading. Designed to be completely automated, once configured, Urchin requires virtually no maintenance.

The Urchin 3 Installation and Administration Guide contains instructions and information to install, upgrade, and configure Urchin 3 software on various UNIX platforms. This guide is intended for experienced webmasters and system administrators who:

- Are familiar with the operating system in question.
- Understand the webserver configuration on the system.
- Have superuser (root)/administrator access on the system.

Chapters 2 through 5 and Appendix A cover the installation and configuration of the Urchin software and its interaction with the webserver. Chapter 6 covers Urchin Maintenance. Chapters 7 and 8 give clear instructions on using the co-branding and customizable features of Urchin. Chapter 9 provides a complete list of all configuration directives. Chapter 10 covers the E-commerce module.

This document will use the `Courier` font to indicate commands, files, directories and any on-screen input or output. If part of the command or file is in italics, then you should replace that part with actual information. For example,

```
cd /distributionpath/ugroups/default/
```

is a command where you would replace the "distributionpath" with the actual location before executing.

1.1 What's New with Urchin 3?

Urchin 3 includes these new features:

- Superfast reverse DNS module.
- Optional e-commerce reporting module (e-Urchin)

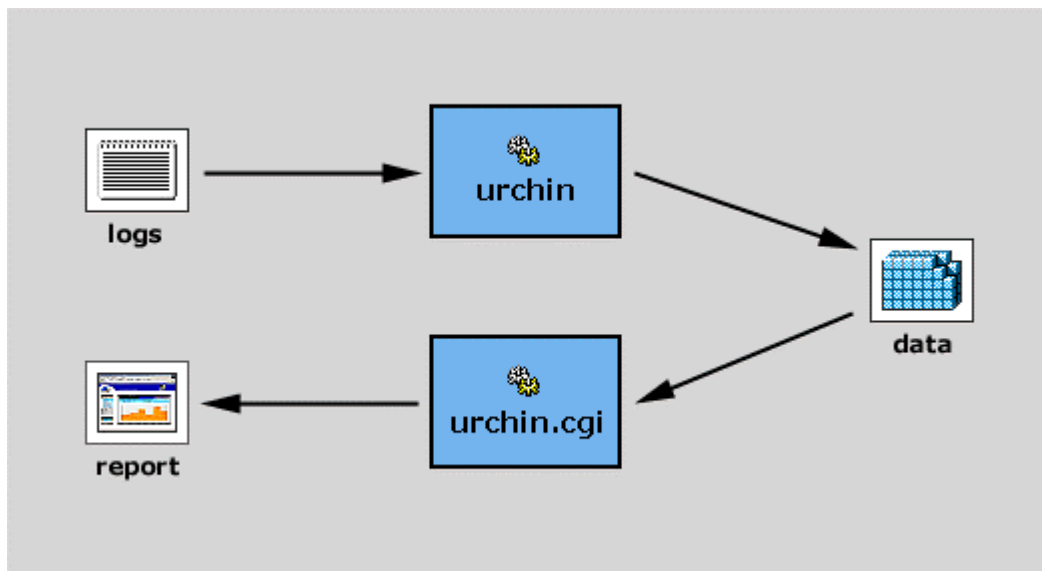
- Massively improved look and functionality
- Vastly expanded reporting capabilities
- DynamicURL parsing capabilities
- Search engine reporting
- Co-brandable and customizable on a group basis

These new features combined with a speed-record shattering stat processing engine make Urchin 3 the optimal package for hosting providers, busy websites, or anyone that wants to get great stats from their site(s).

Since Urchin's reports are web based, users can view their reports via the web. This benefits the ISP in many ways. Clients see the results of their work, and are inspired to maintain their sites, and, of course, pay their hosting fees. ISPs that bill according to bandwidth usage have a simple way to check usage via the System Report. Simply install, configure and enjoy the benefits.

1.2 System Requirements

Once installed, Urchin runs as specified in `crontab` (usually once per day), processing log files created by your webserver. The data from the log file is extracted and entered into the detailed reporting data. After processing, log files are automatically deleted (unless the `LogDestiny` directive is set to `archive` or `don't touch`), keeping hard drive usage in check. Urchin can process multiple sites either one at a time or all at once depending the configuration.



For each new site in the configuration, Urchin will automatically create the links to the icons and CGI engine it needs to deliver the reports. Only one copy of icons and engines are needed. When the users view the reports, they access the CGI engine through their

own unique report directory. This engine queries the data storage and delivers reports dynamically.

Urchin 3 is a very efficient program and requires minimal disk space and memory. The distribution will use ~4 MB of disk space. Each site's Urchin report directory will use anywhere from 200-500k of disk space depending on how busy a site is. While running, Urchin will use approximately 4 MB RAM plus ~250k per site reported on. Extremely busy servers may use extra memory to handle the large amount of data generated, but typically disk space and memory are not issues with most modern web servers.

Urchin uses certain built-in functions of the UNIX operating systems. It is important to be aware of which functions these are in order to ensure consistent operation. The following is a list of functions that Urchin will use:

- cron daemon
- webserver

The webserver interaction is covered in the Chapter 2, System Readiness. Urchin will require certain CGI capabilities in order to deliver reports. The cron daemon is typically used to automate the processing each night.

1.3 Privacy and Security

Urchin has a very clear privacy policy. Urchin does not use cookies; it does not aggregate data or make any connections outside your network; it does not profile users or record any personal or private information. Urchin simply reports on webserver activity. Urchin does count visitors and keeps track of which networks and browsers are being used, as well as referring sites. But all of these parameters are part of the main HTTP protocol and are passed to the server by all standard browsers.

Urchin's web-based reporting easily integrates into existing access security mechanisms. Because most hosting companies already have login and security systems in place, Urchin does not come with its own security system, but relies on existing systems. It is recommended that all Urchin reporting directories be protected from unauthorized access using standard user verification means. Urchin uses standard CGI methods for delivering content and with decentralized reporting, there are no mechanisms for indirectly accessing other reports or data.

If you are using the reverse-DNS module, during log processing, Urchin will use UDP network sockets to connect to the DNS name server specified by the ResolverIP directive. If you are operating behind a firewall, you may experience difficulty connecting. Please check with your administrator for details.

1.4 Upgrading From 2.x to 3.x

Out of necessity, Urchin 3 is not backward compatible with Urchin 2. We will be providing a data conversion program to upgrade the urchin2.x data files to the new format. Please check with our website, <http://www.urchin.com/> for the latest information on upgrading.

Go through the installation process as described in Chapter 3. Be aware that there are subtle changes in the `config` file. It is recommended to create a new `config` file and not reuse one from a previous version. Be sure to remove the previous Urchin from crontab when you enter the new one.

2. System Readiness

While Urchin is designed to be easy to install, there is some system integration necessary. This chapter describes those interactions, and explains how to set things up for proper running of Urchin. The sections include information on log formats, rotating logs, using Urchin's DNS resolver, and what webserver features Urchin will need for reporting. Be sure to read through these sections in order for proper installation.

2.1 Log Formats

Urchin can process many different log formats including e-commerce logs, error logs, and more. But for basic operation, Urchin needs the standard "access-log" produced by your webserver. In order to capture the maximum data for your reports, it is desirable to configure your server to use the Extended Combined Log Format for UNIX servers and the WC3 Extended Format for IIS servers. For error logs only processing of Apache error logs is currently supported.

For Apache servers, the combined format should be the default setting. The following sections outline how to configure various popular servers to use the longer format. If your server is not listed below, you may find that you can apply one of these procedures to your server.

Note: It is possible to run Urchin on log files that contain either too few or too many fields of information. Urchin will ignore missing or extra data. To change the log format to Extended Combined, follow these instructions.



Configure Apache for Extended Combined Logging

To change the log format to the Extended Combined Log Format, you will need to edit the `httpd.conf` file in the Apache configuration. This is located in the `conf` directory within the Apache distribution which is typically installed in `/usr/local/`, `/usr/local/etc/`, or `/opt`. Add the following line exactly as it appears.

```
LogFormat "%h %v %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
```

Once you have located the file, you will need to add the following line to the `httpd.conf` file (if it's not already there). Add the line *before* any `<VirtualHost>` entries. This will set the format globally for all sites. Add the line exactly as it appears below including quotations.

If you were using the `AgentLog` and `RefererLog` directives in the configuration, you may remove these as they are no longer necessary. After making this change, you will need to restart the server.



To change the log format to the Extended Combined Log Format, use the Administration Server via a browser to access the configuration. For each server on the machine which is listed below the "Servers Supporting General Administration" banner, follow these steps:

- Click on the button for a particular server.
- Click on the "Server Status" button at the top.
- Click on the Log Preferences link on the left.
- Scroll in the middle until you find the Format: options.
- Select the Only Log option. Note: you may need to stop the server or change to the name of the Log File for this step.
- Check the first 8 boxes directly below the Only Log option up to and including HTTP header, user-agent.
- Click the "OK" button at the bottom.
- Click the "Save and Apply" button at the bottom.

Repeat the above steps for each server.

Webservers such as Apache are generally configured to produce access logs in one of two ways: one log file per site, or one big log file for the whole server. Urchin can work with either one if configured properly.

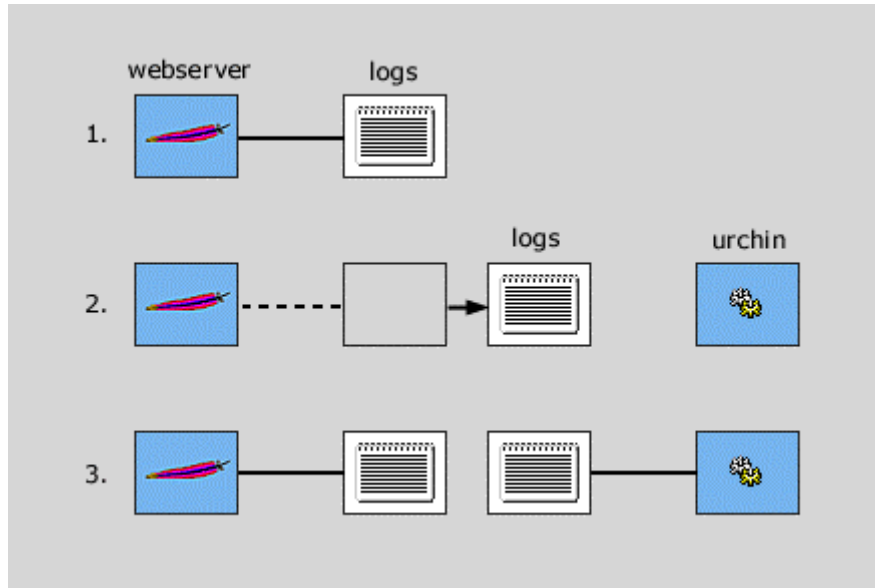
Why use one or the other? The one-logfile-per-site method is simpler to set up, but requires the administrator to add `<Report>...</Report>` directives for every site added to the server. The one centralized log file setup has the advantage of automatically accommodating new sites and reporting on them as traffic occurs. See chapter 4 for more details.

2.2 Log Rotation

In order to fully automate Urchin, it is important to take a look at log rotation. You may already have a log rotation script or mechanism in place that will take care of everything. If not, Urchin has some built in capabilities that can be used to automate the process.

It is important to remember that Urchin does not remember which files have already been processed. Because you may have multiple servers or scripts running around Urchin, it does not assume that all hits processed are sequential. This means that if you process the same hits twice, Urchin will count them twice in the reporting. For this reason, it is important to have precise log rotation so that no data overlap occurs.

If you already have log rotation in place, you can skip to the next section. Just be sure that Urchin runs after the log rotation takes place. If you need Urchin to do log rotation, read on. The following diagram illustrates the sequence of events that Urchin will do automatically, once configured.



Step 1 shows the webservice with an open log represented with the solid line between the two boxes. On UNIX platforms, the webservice will have a partial lock on the file, which permits it to be moved or copied, but not actually deleted. If you delete a log file while the server is still running, the file is not actually deleted until the server is restarted.

In Step 2, Urchin will automatically move (rename) the log files. This is accomplished by using the “RestartCommand” directive. If Urchin sees this directive in the configuration, it assumes that all logs need to be moved before calling this restart command. (This behavior can be overruled by using the LogDestiny: Don’t Touch directive.)

Immediately after moving the logs, Urchin will call the restart command which will cause the webservice to reopen a new empty log where the old one use to be. Urchin will then continue processing the moved log files as shown in Step 3. To use this built in log rotation mechanism, use the following lines in your config file:

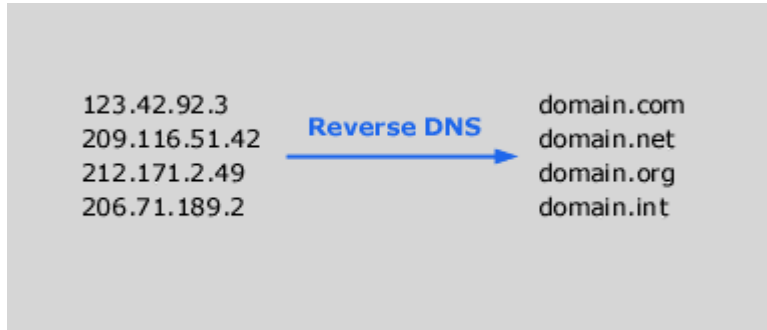
```
RestartCommand: /usr/local/apache/bin/apachectl restart
LogDestiny: delete
```

If you don’t want Urchin to totally delete the logs, you can specify the LogDestiny as “archive,” which will save the files with a date stamp appended to the name. You’ll need to change the italicized text to an actual restart command. It should come with your webservice; check your webservice documentation. If you need assistance with this, contact our technical support for more information.

2.3 Reverse DNS

Reverse DNS allows you to take the raw numeric addresses of the visitors to your site and convert them into meaningful domain and network names such as aol.com or

xerox.co.jp. This information can be useful in determining what percentages of your visitors are from which networks.



Unfortunately, these “lookups” can take a long time and add significant load to the webserver. Even worse, if you are having the webserver do the lookups, it will most likely wait until the lookup is complete before delivering any content to the visitor. This can appear as a 1-3 second delay on getting your home page to the visitor.

Fortunately, Urchin 3 has a super fast multi-threading reverse-DNS module that can do the work of reverse lookups relieving the webserver of this costly responsibility. Pages are delivered immediately without waiting for lookups to occur. Urchin then processes the information later at off-peak hours and does not require visitor to wait.

In order to use Urchin’s reverse-DNS module, first turn off hostname lookups in your webserver configuration.



Turn off reverse-DNS in Apache

In Apache’s httpd.conf, find and edit this line to look like:

```
HostnameLookups Off
```

This is actually the default setting for apache. You’ll need to restart to make this change take affect.



Turn off reverse-DNS in Netscape

For Netscape, you’ll need to fire up the admin server and for each server listed below the “Servers Supporting General Administration” follow these steps:

- Click on the button for a particular server.
- Click on the "Server Status" button at the top.
- Click on the Log Preferences link on the left.
- Look for the “Record” option.
- Select “IP Address”.
- Click the "OK" button at the bottom.

- Click the "Save and Apply" button at the bottom.

Now that the webserver is logging IP addresses and not domain names, you can use the "ProcessDNS" directive in Urchin. In the config file you will have two lines that look like:

```
ProcessDNS:    on
ResolverIP:   127.0.0.1
```

The "ResolverIP" address should point to a caching capable fully operating name server. This can either be the local machine or one setup to do name service. It should definitely be on the local network and not across to many hops. If you have large log files, Urchin will make many requests to this machine simultaneously. The rate of the requests is choked to a level that most modern servers can tolerate; and the overall bandwidth is not that high. Nevertheless, you should notify the network administrator of your intention to use a particular name server.

2.4 Urchin Reporting Requirements

The current version of Urchin uses a slick web interface to produce all reports. These reports are designed to be lightweight on network resources while providing maximum information. The reports are produced dynamically by a CGI application, and thus require some capabilities of the webserver to be activated. Specifically, the Urchin reporting will use CGI capabilities and Symbolic Links.

While you may already have these features turned on, it is a good idea to check to make sure that these features are on.



Configure Apache for CGI operation

For Apache servers, you will need to check for three lines. These edits are made in the httpd.conf file located in the Apache distribution:

```
Options FollowSymLinks ExecCGI
AddHandler cgi-script .cgi
DirectoryIndex index.html index.cgi
```

The first line enables the options for following symbolic links and executing CGI scripts. This line will go between a pair of "<Directory>...</Directory>" tags. If you have multiple "Directory" entries, be sure to enable these options for the one that controls the directory where you want to put the Urchin reporting.

The second line turns on recognition of ".cgi" files as CGI scripts. This can be done globally for the whole server. And, the last entry needs the "index.cgi" option added to the end of the DirectoryIndex line. Urchin will use a linked script called "index.cgi" as the main page in the reporting directory. Be sure to restart

after making any changes. If you need assistance with these options, you can contact our technical support or refer to the Apache documentation.



Configure Netscape for CGI operation

For Netscape, you'll need to fire up the admin server and for each server listed below the "Servers Supporting General Administration" follow these steps:

- Click on the button for a particular server.
- Click on the "System Settings" button at the top.
- Click on the "Symbolic Links" link on the left.
- If necessary change the settings to "Always" and "Yes".
- Click on the "Programs" button at the top.
- Click the "CGI File Type" link on the left.
- If necessary click the "Yes" button next to activate.
- Save Changes and restart server.

That's it for System Readiness. You are ready to install Urchin 3.

3. Installation

To ensure a successful installation, be sure to read the previous chapter on System Readiness. In particular, the last section on Urchin Reporting Requirements explains which capabilities for CGI operation are needed in order for the web-based Urchin reporting to function properly. This installation chapter is designed to be step-by-step and should be followed in order. The installation process can be performed from a client or directly on the webserver. Use a command tool or telnet to access the server.

Please note that Urchin 3 is not backward compatible with Urchin 2 or before. If you are upgrading from a previous release, please see the “Upgrading from 2.x to 3.x” in section (iii) at the beginning of this document.

You may need to install and run Urchin as `root` in order to have permissions on deleting logs, restarting servers, creating links, etc. If you are not installing Urchin as `root`, you will need to check permissions on executables and log files. The following example assumes that Urchin is installed as `root` in the `/usr/local` directory.

3.1 Download and Expand Archive

If you haven't already downloaded the Urchin archive, you should do so at this time from either a web browser or ftp at the following addresses:

```
http://www.urchin.com/download/  
ftp://ftp.urchin.com/pub/
```

Save the archive in the directory you have chosen for installation (e.g., `/usr/local`). You may need to be `root` to do this. Once you have downloaded the archive, follow the instructions below to expand the archive. The archive will automatically create an `urchin3xxx` directory.

```
cd /usr/local/  
gunzip urchin3xxx_solaris26.tar.gz  
tar -xvf urchin3xxx_solaris26.tar
```

3.2 Select Log Rotation Mechanism

After expanding the archive, you will find a “`config.sample`” file within the distribution. If you don't have an existing Urchin config file, then copy this sample file and rename the copy “`config`” Then edit config using `VI` or some other text editor. The first part of the configuration describes some of the global settings for Urchin to use for all sites. Take a look at the “Log Rotation” entry which includes the following commands:

```
RestartCommand: /usr/local/apache/bin/apachectl restart
LogDestiny:      delete
```

If you need Urchin to do log rotation as discussed in the previous chapter, you will need to uncomment (by removing the “#” in front of the lines) and modify these lines to match your system. If you are handling log rotation on your own, leave these lines commented out.

If you are using Apache, most current versions come with a control script located in either bin or sbin of the distribution. This can be called with the “restart” flag to signal webserver to restart. Netscape servers should have a restart script as well, located in the configuration area for that server. Check your webserver’s documentation for the best way to restart.

The above lines will configure Urchin to delete all logs after processing. Urchin has its own internal data structure, so keeping log files around is not necessary. But, if you don’t want Urchin to delete the logs, you can change the “delete” word to “archive” which will attach a date stamp to the file.

3.3 Configure Reverse DNS

The next section in the “config” file covers reverse DNS. As discussed in the previous chapter, Urchin has a really fast reverse DNS module that allows you to turn off costly hostname lookups on the webserver. If you do want to use Urchin for DNS, uncomment the following lines in the configuration.

```
ProcessDNS:      on
ResolverIP:     127.0.0.1
```

You will need to edit the “ResolverIP” directive to point to your local DNS caching server. This can either be the local machine that Urchin is running on or a dedicated name server within your network. See the previous chapter for more information. If you do not want Urchin to do DNS, then leave these lines commented out.

3.4 Configure System Report

In addition to the reports created for each site in the system, Urchin has the ability to create a system-wide report that gives usage statistics for the entire server and ranks sites according to usage. This can be very useful for monitoring overall performance as well as billing clients for bandwidth usage.

If you do want to use the System Report, uncomment the following lines in the “config” file:

```
SystemReport:    on
SystemDirectory: /www/webmaster/urchin/
```

The System Report (previously known as the “Webmaster Report”) must have a unique reporting directory. Select a web-accessible directory and edit the “/www/...” path accordingly. If you only have one site or do not need the System Report, leave the entry commented.

3.5 Configure Website Reports

There are three basic ways to process logs with Urchin. Urchin can process unique logs for each site. Urchin can process one big centralized log and produce multiple reports. And, Urchin can be called from a script passing command line parameters to process one site at a time. These three methods are covered in detail in the next chapter. Also, there are some examples in Appendix A. But for general operation where you have unique logs for each site, uncomment the following lines in the “config” file and duplicate them for each site in the system:

```
<Report>
ReportName:      yoursite.com
ReportDirectory: /www/yoursite/urchin/
TransferLog:     /www/yoursite/logs/access-log
ErrorLog:        /www/yoursite/logs/error-log
</Report>
```

Each site should have its own <Report>...</Report> block of lines. For each site, change the “ReportName” to match the domain name of the site (without the “www.” is preferable for proper referral filtering). Each site should have a unique web-accessible report directory. Change the “ReportDirectory” to this location. The report directories will need CGI capabilities as described in the previous chapter.

Next, edit the “TransferLog” line to point to the access-log of the site. Error log processing is optional. If you only have one error log for the whole system, it’s best to leave this line commented out. But if you have unique error logs for each site, you can process them as well.

3.6 Test Run Urchin

The next step is to go ahead and run Urchin and take a look at the reports. If you are handling your own log rotation, be sure to rotate logs before running Urchin. You can also provide historical logs for testing as well. Run Urchin from the command line.

```
./urchin
```

If you are using the script – command line option mode, you will need to provide the command line options. But under normal operation, the above command is good. Urchin will process each site and produce reports. While processing logs, Urchin will print out a “.” for each 10,000 hits. While processing DNS, Urchin will print out a “.” for each 1,000 visitors.

If any errors occur during processing, the error message should help specify what the problem was. If there was a permissions problem or a missing directory, go ahead and fix those problems now. If you have errors that don't make any sense, please contact our technical support team for clarification.

At this point go ahead and view one of the reports with your browser. Each report directory should be web accessible and have CGI capabilities. If there is problem viewing the reports, check out the previous chapter on setting up Urchin's reporting requirements.

3.7 Automate Urchin

Once Urchin is running properly, you can automate Urchin by entering it into the `cron` process of the server. Check with your administrator to make sure the "cron daemon" is running. To edit crontab using VI, use the following commands:

```
setenv EDITOR vi
crontab -e
```

Add the following line to the end of your `crontab` replacing `/usr/local` with the actual path. (Those are *zeroes* at the beginning of the line.) This particular entry will tell Urchin to run at midnight.

```
0 0 * * * cd /usr/local/urchin3xxx; ./urchin > ./log 2>&1
```

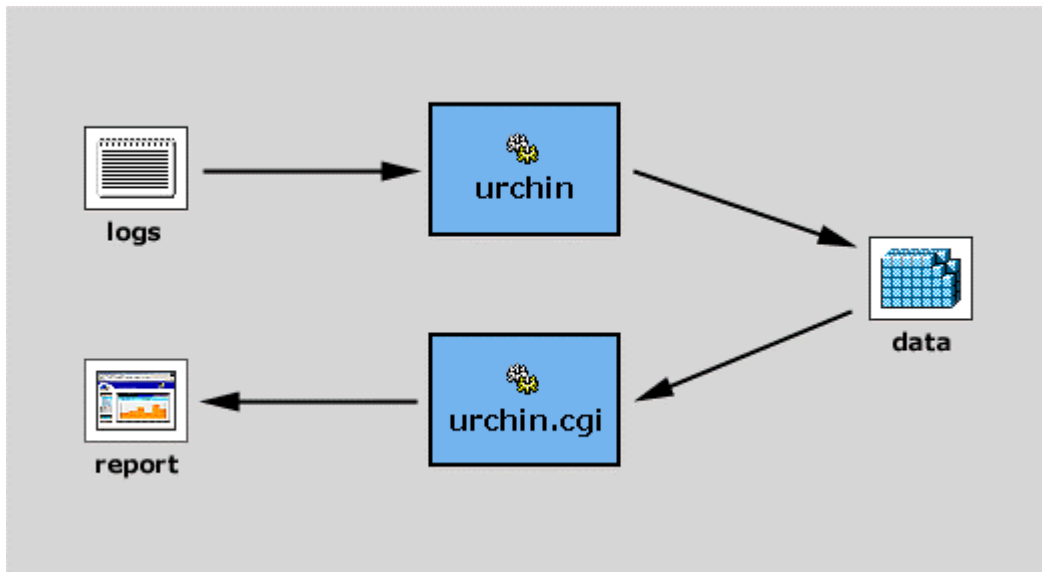
If you are upgrading be sure to remove previous Urchin entries. Write and quit the file. For more information on using cron, refer to the man pages of the system.

3.8 Licensing

The downloaded copy of Urchin most likely contained a demo license so you could test out the software. After this license expires, you will need to acquire a license to continue using the software. Please contact your sales representative for pricing details. Information about us can be found at <http://www.urchin.com>.

4. Configurations

Normal Urchin operation involves crunching log files into data sets and using those data sets to produce reports. The following diagram shows a general flow chart of how Urchin works. Log files are crunched by the “urchin” executable and reports are generated via the “urchin.cgi” application.

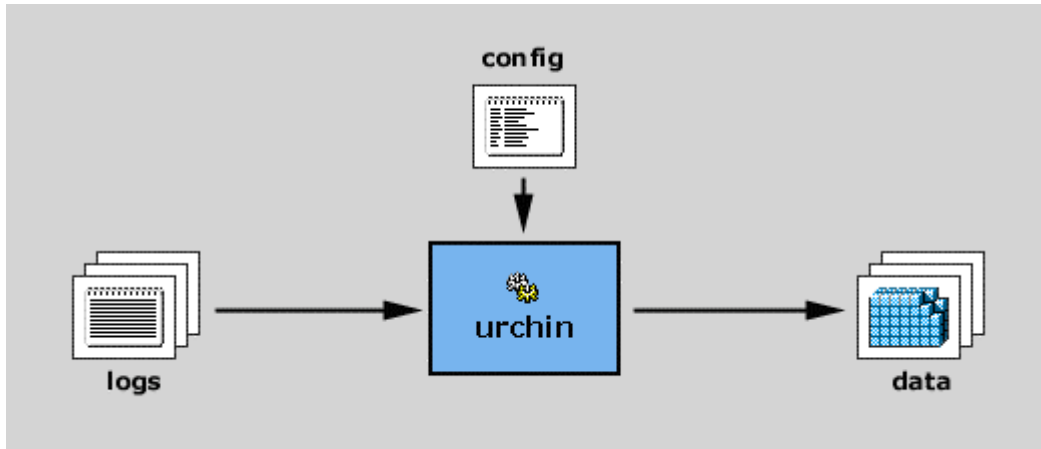


Urchin has its own database structure that compactly stores the information from the log files. This means that you must only process log files once; otherwise, Urchin will count the data twice. Urchin does not keep track of what’s been processed, because you might have multiple log files or want to process historical data. Be sure to follow the installation instruction in the previous chapter on rotating logs.

The following sections describe three ways of processing log files and two ways of producing reports. Under normal configuration, each site will have its own log files and report directory. But, if you use one log file for multiple sites or want to script the Urchin processing, these sections should get you going.

4.1 One Log file per Site Mode

This is probably the most common mode of operation, and it's the most straightforward to set up. In the diagram below, each site has one or more log files and a unique reporting directory where the data is stored. A single central configuration will instruct Urchin to process the logs sequentially, one at time.



This type of configuration is very similar to the Virtual Host concept used in Apache webserver configuration. In fact, for each “<VirtualHost>” entry in Apache, you should have one “<Report>” entry in the Urchin configuration. This configuration can be modified by hand or some clients have written Perl scripts to create the Urchin configuration automatically. Here is an example of an Apache configuration and the corresponding Urchin configuration:

Apache Configuration (httpd.conf)

```

LogFormat "%h %v %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""

<VirtualHost 208.111.57.100>
ServerName www.acmehosting.com
DocumentRoot /www/acmehosting
TransferLog /www/acmehosting/logs/access-log
</VirtualHost>

<VirtualHost 208.111.57.100>
ServerName www.acme-design.com
DocumentRoot /www/acme-design
TransferLog /www/acme-design/logs/access-log
</VirtualHost>

```

Urchin Configuration (config)

```

<Report>
ReportName: acmehosting.com
ReportDirectory: /www/acmehosting/urchin/
TransferLog: /www/acmehosting/logs/access-log
</Report>

<Report>
ReportName: acme-design.com
ReportDirectory: /www/acme-design/urchin/
TransferLog: /www/acme-design/logs/access-log
</Report>

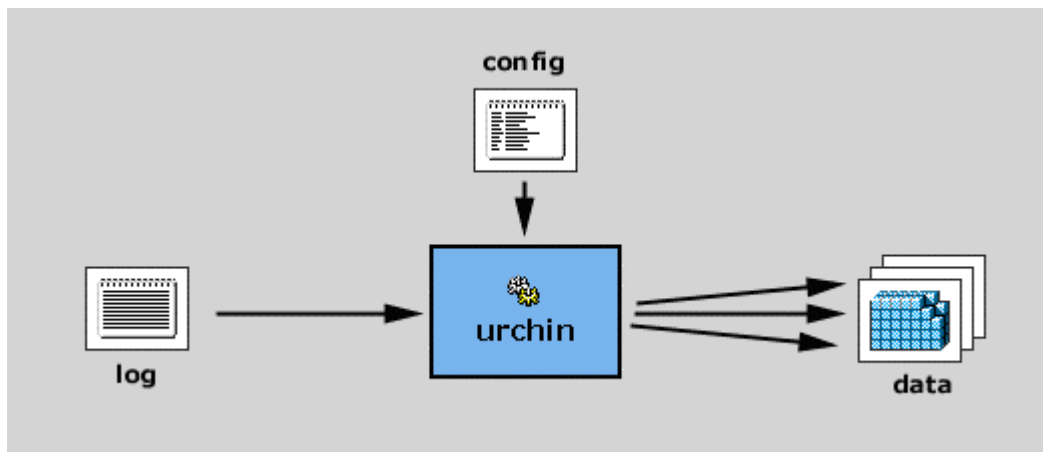
```

Just as in the Apache configuration, the Urchin configuration contains one entry for each site. Notice that each site has a unique report directory in the Urchin configuration. In fact, this directory is located within the document root of the site. This allows each user to go to their site + “/urchin” to view their Urchin report. You could put the Urchin

report in the same place as the logs, or create a new directory as in this example. When Urchin runs, it will automatically create links and data in this reporting directory. See section 4.4 below on exactly what is done inside a reporting directory. But under normal operation, the user simply points their browser to this web accessible directory, and that's it.

4.2 One Big Log file for All Sites Mode

In addition to Urchin's standard one log file per site mode, it can be run in "one single log file for all sites" mode. Some hosting companies prefer this type of setup and will configure their servers to put all logging for all sites into the same files. If you are using this type of setup, Urchin can automatically filter the logs into their individual reports.



In above diagram, we still have multiple data locations, one per site, but only one log file. Using a central configuration, Urchin will process this log file and produce multiple reports filtering the hits into their respective site directories.

The advantages to this type of operation is that the logging is a little simpler and the Urchin configuration only needs one "<Report>" entry for all sites instead of one for each site. As hits for new sites are detected, Urchin will automatically setup those reporting directories. There are two disadvantages to this type of setup. First, Urchin will use extra RAM during processing as it has to swallow larger log files at once. And ultimately, this type operation is not as scalable as it cannot be parallelized as described in the next section. If you are hosting over 50,000 sites, please consult with us on which configuration is best for your situation.

The easiest way to setup this type of configuration is to modify the logging format of the webserver to include the name of the site. For example, in Apache, if you change the logging format to:

```
LogFormat "%h %v %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
```

The second field, “%v”, will contain the name of the site (i.e. www.mysite.com). If there is only one log file for all sites, Urchin can key on this field to determine which site it’s for. Here is a sample Urchin configuration:

Urchin Configuration (config)

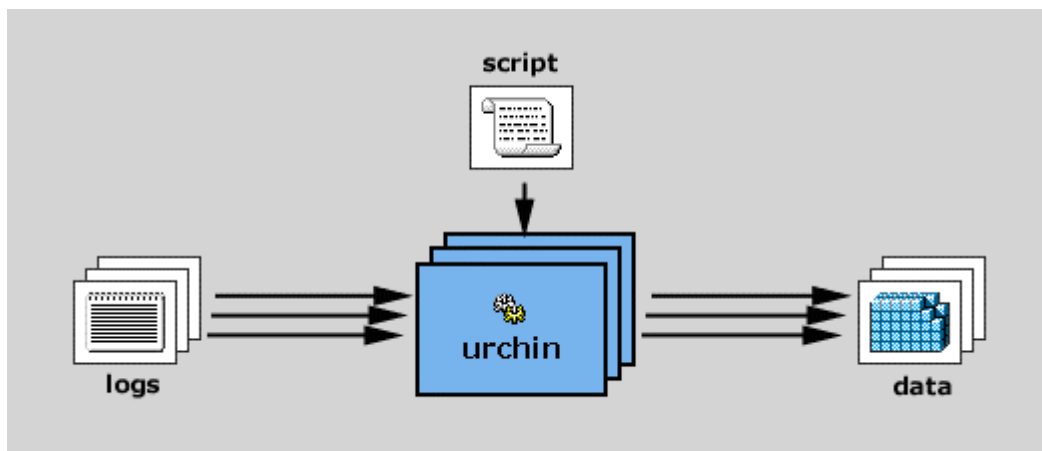
```
<Report>
SubreportMode: on
SubreportField: 1
SubreportFilter: (.*)
ReportName: $1
ReportDirectory: /www/$1/urchin/
TransferLog: /www/logs/access-log
</Report>
```

In this example, Urchin will use the “SubreportMode” directive to filter the “%v” field (field #1, the first field is #0). Using regular expressions, Urchin captures the contents of this field and uses it as “\$1” to determine the name of the site and the report directory. For each site encountered, Urchin will automatically setup the report directory. Each user can point their browser to their own report directories.

For more detail on using the “SubreportMode” directive to filter the logs into multiple reports, please see the next chapter on Filtering.

4.3 Scripting Many Sites Using Command Line

In addition to using a config file for controlling operation, Urchin comes equipped with a set of command line options that can be used to script the operation. For configurations with lots of sites, this can be handy for controlling the timing of operation and allowing for parallelization.



In the diagram above, we have multiple sites each with their own logs being processed by multiple calls to Urchin producing unique data sets and report directories. Since the logs and report directories are unique for each site, multiple copies of Urchin could be run in parallel for even better performance. Also, this type of operation is easy to use with

database driven information systems as the names and locations of sites and log files is passed on the command line.

While a complete description of all command line options is given in Chapter 9, the following example covers the basic operation. Keep in mind that Urchin can use both the config file and command line options together. That is, global information such as the location of the WebmasterDirectory, or the address of the DNS Resolver can be specified once in the config file, while specific site related information is passed via the command line.

```
./urchin -R acme.com -r /www/acme.com/urchin -l /www/acme.com/logs/*
```

In this example, Urchin is called on to process the “acme.com” site. The “-R” option specifies the ReportName, the “-r” option specifies the ReportDirectory, and the “-l” option specifies the log files. Notice that wildcards can be used to grab multiple log files all at once.

If you have a lot of sites, this can be a really nice way to integrate Urchin into your existing site maintenance scripts. Your System Administrators will be able to easily use this method within existing log rotation or Perl maintenance scripts.

For installation, follow these additional steps:

- 1 Edit the “config” file, entering global options for DNS and the System Report. Leave all the <Report> blocks commented out. Also, you will need to do your own Log Rotation.
- 2 In your script, loop over the sites and call urchin with the following command line options. Run urchin -help for details.

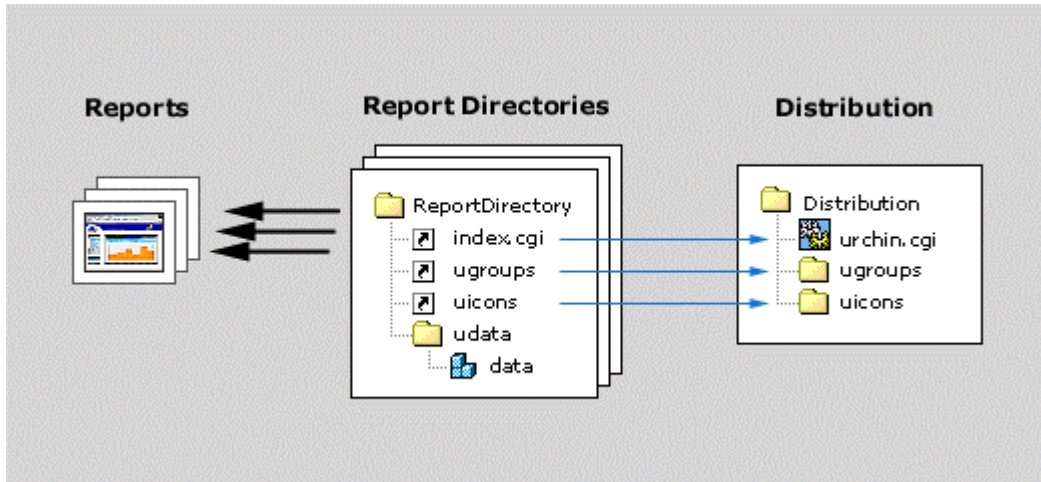
```
./urchin -r ReportDirectory -R ReportName -l LogFile(s)
```

You can use a wildcard with -l option to specify more than one file, or simply list them separated by spaces.

- 3 In your script, after the loop, run “./urchin -cow” which cleans up the System Report.
- 4 Run your script and follow the rest of the installation instructions regarding viewing reports and automating Urchin.

4.4 Normal Decentralized Reporting

How exactly does the reporting work? It’s simple and automatic. Under normal operation, each site will have a ReportDirectory specified in the configuration or on the command line. Urchin will use this directory to store the compact Urchin data, and provide access to the reports.



As shown in the diagram, within the Distribution there are three things needed for reporting. Instead of copying these to each Report Directory, symbolic links are created from the Reporting Directories back to the distribution. This saves disk space as only one copy of icons, group information, and the Urchin Report Engine (`urchin.cgi`) are needed. These links are then used during the reporting to produce the actual reports.

In addition to the three links, each Report Directory will contain a “`udata`” directory where all Urchin data for this particular site is stored. When the user accesses their Report Directory, they access the “`index.cgi`” link, which points back to the “`urchin.cgi`” application in the distribution. All of this is created automatically, but you will need to be sure that Report Directories have proper web server configuration so that the “`index.cgi`” link works properly. Check out Chapter 3 on installation for details on setting this up.

Whether you are using the simple “one log file per site mode”, centralized logging with subreport mode, or command line scripting, Urchin will automatically setup new report directories as they are encountered. Simply make entries in the configuration (or don’t with subreport mode), and Urchin will detect new sites and setup the report directories.

5. Filtering and Using Dynamic URLs

Urchin 3 has a couple of different filtering capabilities. It can filter centralized logs creating multiple reports all at once (subreport filter). It can filter out stuff from a particular log file that you don’t want to get into a report (filter out). It can require that all log lines contain some pattern in order to make it into the reports (filter in). And, Urchin 3 can filter Dynamic URLs, such as those produced by CGI scripts, in order to determine a better *pagename* for reporting.

All of the filtering mechanisms covered in this chapter use *POSIX extended regular expressions*. Regular expressions allow you to use wildcards in order to perform pattern matches used in the filtering. If you have never used regular expressions before, you

might need to consult with another administrator or with our technical support group if you have trouble setting it up. It's not really that complicated. Here is a list of some of the basic components of the expressions that will help you get started.

Regular Expressions	
wildcard	meaning
.	match any single character
*	match zero or more of the previous item
+	match one or more of the previous item
?	match zero or one of the previous item
()	remember contents of parenthesis as item
[]	match one item in this list
-	create a range in a list
	or
^	match to the beginning of the line
\$	match to the end of the line
\	escapes any of the above

The regular expression is compiled and applied according to each purpose outlined below. Multiple filters can be applied simultaneously, although each one will incur a performance hit. For each of the types of filters below, examples are provided to help understand the basic usage.

5.1 Subreport Filtering

This type of filtering is used with centralized logging as described in section 4.2. If you have one set of logs that contain multiple sites, then you will need to use the subreport filtering. This filter allows you to specify a field number in the log line and a regular expression to filter this field in order to determine the name of the site. This information is then used to produce the report. For example, if you have the second field set to the domain name in Apache logging and you want to capture the entire field as the name of the site, you would have in your Urchin config:

```
SubreportMode:    on
SubreportField:   1
SubreportFilter:  (.*)
```

The first field is number zero, so the "1" points to the second field. The above filter matches any string of characters and the parenthesis capture the entire thing. With subreport filtering, you must use one set of parenthesis in order to capture the name of the site.

The information captured from the parenthesis can then be used in the ReportName and ReportDirectory directives as the “\$1” variable. For example:

```
ReportName:      $1
ReportDirectory: /www/$1/urchin/
```

In this example, Urchin will replace the “\$1” with whatever is found by the subreport filter. Following this example, two log lines that start with:

```
121.2.32.13 www.site1.com - [14/Apr...
208.32.6.36 www.site2.com - [14/Apr...
```

will be filtered and entered into two reports. The first report would be named “www.site1.com” with a report directory located in /www/www.site1.com/urchin/ and the second would be “www.site2.com” with a report directory located in /www/www.site2.com/urchin/.

If you wanted to only capture the domain not including the “www”, you could change the SubreportFilter entry above to:

```
SubreportFilter:  www\.(.*)
```

This filter requires that the field contain the string “www.”. Everything after the “.” is then captured by the parenthesis and used in \$1.

5.2 Filter In and Filter Out

The FilterIn and FilterOut directives allow you to control which hits in a log file actually make it into a report. This can be used to filter unwanted hits or only include hits that match a certain format. It is perfectly acceptable to have different <Report> entries use the same log files. Urchin will recognize this fact and reuse the log files for each occurrence. This allows you to create multiple reports from the same log file. Using filters, these reports can be tailored to a specific need. Please note that for each report you will need a separate <Report> entry and a separate ReportDirectory. Also, each filter directive applies only to the next “TransferLog” entry. If you have multiple logs for one site, you will need to have a Filter command *before* each one.

The FilterIn directive requires that each hit matches the specified filter. If a hit does not match the requirement, then it is ignored. If you wanted to create an entire report based on a subdirectory within the website, you could use this directive to require that each matches that subdirectory name.

On the opposite side of the fence, the FilterOut directive can be used to block certain hits from getting into the report. Anything that matches the specified pattern will be ignored and not counted in the reporting.

Here is an example that will help illustrate the usage of the two filters. The main website in question is `http://www.serious.com` and is located in the directory `/www/serious/`. Within this directory we have the main corporate website, and a fun subsite located in a subdirectory called `smurfs`. Since both sites are logged together, we may want to use the filtering capabilities in order to create separate reports for the serious site and the smurfs site. To do this, you would make entries in the Urchin `config` file similar to:

```
#specify which field the filter applies to
# 0 is the first field, 4 is the request.
FilterField:      4

#main serious report
<Report>
ReportName:      serious.com
ReportDirectory: /www/serious/urchin/
FilterOut:       smurfs
Transferlog:     /www/serious/logs/access-log
</Report>

#smurf report
<Report>
ReportName:      serious.com/smurfs
ReportDirectory: /www/serious/smurfs/urchin/
FilterIn:        smurfs
Transferlog:     /www/serious/logs/access-log
</Report>
```

Since each hit for the fun site will contain "smurfs" in the path, we can use the `FilterOut` directive to *exclude all entries that contain the name*; and use the `FilterIn` directive to *only include entries that contain the name*. We used the `FilterOut` directive for the serious corporate site, and the `FilterIn` directive for the fun site.

5.3 Dynamic URL Filtering

Many sites today will use a CGI, ASP or other scripting mechanism to provide dynamic content. Often, a single script is used to deliver multiple pages of information. While this can be a handy way to track users sessions or provide "live" content, it poses an additional challenge for meaningful reporting.

Under normal operation the "Pages" reporting section reports on the actual object, page or script requested and discards any parameters that may appear in the request line as arguments to the script or page. The `DynamicURL` directive allows you to use regular expressions to capture this potentially valuable information. Let's use an example to illustrate the concept.

In this example, a CGI script is used to deliver information about all products in a catalog. The script draws from database, and uses parameters passed through the request

(known as the GET method), to determine which product to display. A particular hit or request may look like:

```
/cgi-bin/showProduct.cgi?sessionId=123456789&productId=knobs  
|_____||_____|
```

Under normal operation, Urchin reports on the first part of the request only. That is everything before the question mark in the above example. And while it is nice to know people are looking at products, it would be real nice if we know which ones. By using a DynamicURL filter we can grab this valuable information embedded in the second part of the request.

Now in this example, we don't necessarily want to capture the entire second part of the request because of the "sessionId." Let's assume that this parameter changes for each visit and we get 30,000 visits per day. Including this piece of information would render the reports useless as they would be much too granular. Instead we just want to capture the "productId" and report on that information.

```
/cgi-bin/showProduct.cgi?sessionId=123456789&productId=knobs
```

We may still want to know which script was used as well as which product was implicated in the request. By using a DynamicURL filter, we can capture multiple parts of the request and recombine them into a new, formatted request ready for reporting. Here is an example of a filter that could be used in this example:

```
DynamicURL: (/cgi-bin/showProduct.cgi\?).*productId=(.*)
```

This regular expression will match the above request no matter what the value of the sessionId or productId was. And the parenthesis capture the parts of the request that we want to keep for reporting. The effective request of the above example would look like:

```
/cgi-bin/showProduct.cgi/knobs
```

Up to 5 sets of parenthesis can be used. And, multiple filters can be applied. If a request does not match the DynamicURL filter, it is left unmodified, but still included in the reporting. This allows you to use multiple DynamicURL filters for each area of a site. Keep in mind there is a slight performance hit for each filter used. If you need assistance getting this to work correctly, please contact our technical support. Be sure to send them an example from a log and what pieces you want to capture.

Note that DynamicURL filters can only be applied to the base URL and query string that form the page request. They cannot be used to filter referrals or any other fields in the log file. Also, when DynamicURLs and FilterIn/FilterOut are used together the DynamicURL will be applied after the other filters. So consideration must be given to how one set of filters affects the others when choosing what to filter.

6. Maintenance

This section covers the basic things you may need to do during administration of Urchin. After making changes to the Urchin configuration, you do not need to restart Urchin. Urchin runs automatically from the `cron` daemon and will read in the configuration each time.

If you are using centralized reporting or scripting, you may not need to modify the Urchin configuration when new sites are added. Urchin can auto-detect new sites and setup their directories automatically. But, for normal setups, each site will have an entry in the Urchin configuration. As you add or remove sites from your system, you will need to add or remove them from the Urchin configuration.

6.1 How to Add a New Site Report

Adding sites to the Urchin configuration is easy and straightforward. The first step is to add the site to your webserver operation. Make a note of the location of the new log files, decide where you want to put the new reports, and then add the following lines to the Urchin config file replacing the italicized parts with your specific information. For a complete description of each directive, please refer to Chapter 9.

```
<Report>
ReportName:      domain.com
ReportDirectory: /www/domain.com/urchin/
Transferlog:     /www/domain.com/logs/access-log
</Report>
```

Note: While the `ReportName` entry can be arbitrary, it is best to set it to the primary domain of the site. For example, for the `www.quantified.com` site, we would set the `ReportName` to “`quantified.com`”. This allows Urchin to properly filter the referrals reports which is normally reserved for external referrals.

6.2 How to Edit or Delete Sites

You can make changes to the Urchin configuration at any time. Simply edit the `config` file and your changes will be loaded the next time Urchin runs.

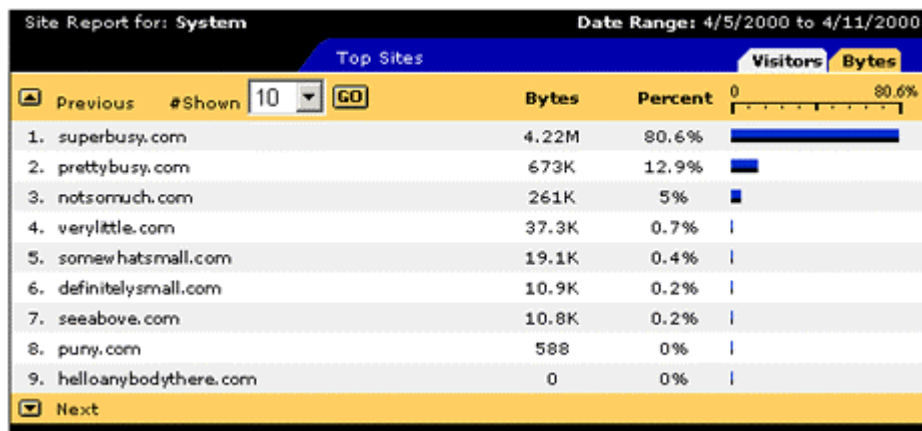
To delete a site from the Urchin configuration, simply remove the `Report` entry for that site starting with `<Report>` and ending with `</Report>`. Your changes will be recognized the next time Urchin runs.

WARNING: Make sure that your webserver configuration is accurate. If you are deleting a site from your webserver, be sure to delete the site from the *webserver configuration* and not just the content and log files. If a logging directory has been

removed, but the site is still in the configuration, Apache will not restart. It will issue an error and exit. And, if you are having Urchin restart the webserver in order to rotate logs, this could happen in the middle of the night when you are not there. So, when deleting a site, be sure to remove the site from the *webserver configuration*.

6.3 Using the System Report

The System Report (formerly the Webmaster Report) is a very useful internal reporting tool provided by Urchin under normal operations. If you have more than one site in your system, this report can tell you which sites are doing the most traffic. All sites are ranked by amount of traffic in either Visitors or Bytes Transferred. The Bytes Transferred report for all sites can help in the billing process for exact bandwidth usage billing of customers.



The System Report also gives totals for the entire system. Graphs can show which hours are the peak and off-peak, which days are the busiest, and total traffic and data transferred during any time period. For load-balancing clients, there is a Top Servers report that ranks the amount of traffic handled by each server in the cluster. This reporting can detect malfunctioning load-balancing software or tell you when it's time to add another box.

To use the System Report, simply uncomment or enter the following lines in the Urchin configuration:

```
SystemReport:      on
SystemDirectory:  /mypath/urchin/report
```

Change the path to a location to put the System Report. This should be different than any of the normal report locations for Urchin. The next time Urchin runs, it will read the configuration and produce the System Report in this location. All links are automatically setup and the same CGI requirements apply to this directory as for the other reports.

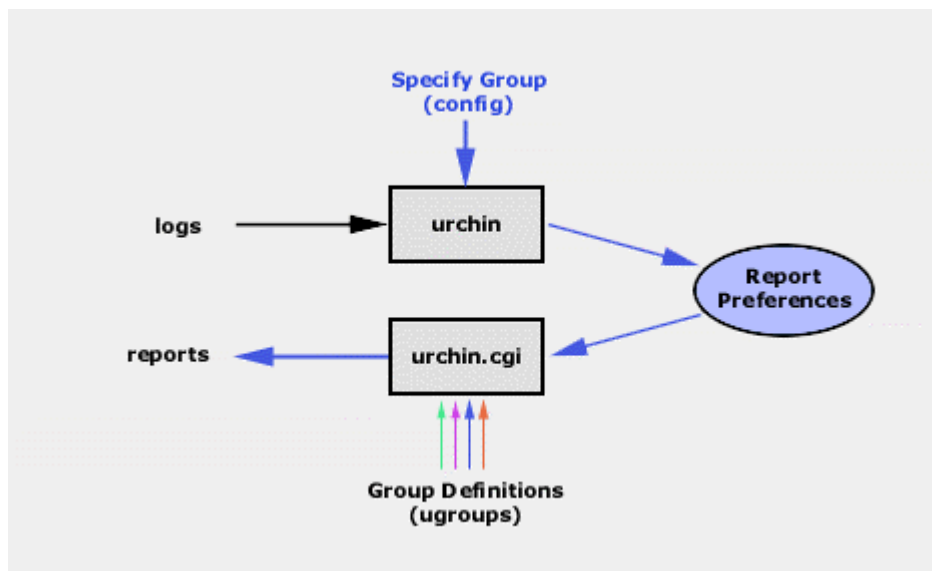
7. Creating Groups of Users

Whether you have multiple brands in your hosting service or different levels of hosting, this chapter describes how to create groups of users. Each defined group can have a different set of navigation, icons, branding, etc. The next chapter describes how to modify these features, but first, we need to know how to create the groups.

There are basically four ways to specify a group for a particular site. Each way has its advantages depending on your architecture:

- Specifying Groups in the Configuration
- Specifying Groups on the Command Line
- Using the CGI Specification
- Specify Groups Manually

Each way is described in detail below starting with section 7.2. Before specifying a group for a particular site, let's see how the group information gets into the reporting. The diagram below shows a flow chart clockwise starting at the top left.



Log files are processed by the Urchin log engine (urchin). During this process Urchin will read the configuration either from the config file or from command line options. The configuration for a particular site can specify the name of the group. If no group is specified, then the “default” group is used. This information is then written into preferences for that site located in its reporting directory.

When it is time create the reports, the Report Engine (urchin.cgi) reads the group information from the preferences and selects the appropriate group definitions from the

ugroups directory. These definitions, which are described in detail in the next chapter, allow you to control the branding, navigation, and other parameters of the report.

In order to use `groups` capability, you need to setup your group in the `ugroups` area of the distribution as described in the next section, “Creating a New Group.” Once the group is in the `ugroups` definitions, you can specify that a particular site belongs to that group using one of the techniques starting in 7.2.

7.1 Creating a New Group

Before specifying a group for a particular site, we need to setup the group parameters, so Urchin will now how to behave for that group. Follow these steps for setting up a group:

Step 1: Choose a name for the group. The name must not contain white space or any special characters and should be less than 30 characters long.

Step 2: Go into the “ugroups” directory within the distribution.

```
cd /distributionpath/ugroups
```

Step 3: Copy the default group directory and everything in it to a new directory named after your new group name.

```
cp -r default newgroup
```

That’s it. You now can use the `newgroup` name in a specification below. Choose the specification method that matches your setup. Most sites can use 7.2, specifying the group in the config file. The next time Urchin runs, it will put the new group name into the preferences file for that site. Then, when you access that site’s report, Urchin will read the preferences and use the parameters setup for that group. To customize Urchin for a particular group, see the next chapter.

If the group is not specified, then the “default” group and subdirectory is used for determining branding, navigations, languages, etc. The name of the directory is the same as the name of the group.

7.2 Specifying Groups in the Configuration

Most architectures will be able to use the Urchin configuration to determine the Group for a particular report. There is a “ReportGroup” directive that allows you to specify a group for any <Report> entry in the config file. Groups should not contain white space and are case sensitive. Here is an example of a config file that uses two different groups:

```
<Report>
ReportName:    site1.com
ReportGroup:   group-a
...
```

```

</Report>

<Report>
ReportName:    site2.com
ReportGroup:   group-b
...
</Report>

```

In this example, the first report for “site1.com” is assigned to “group-a” while the second is assigned to “group-b.” These group settings get written into the preferences for each site, which is then accessed when the user views the reports. The report engine reads the preferences and determines which group the site belongs to. This group information can be used to deliver custom branding and features.

Before running Urchin with the above configuration, you should setup these group definitions in the `ugroups` directory according to section 7.1.

7.3 Specifying Groups on the Command Line

In addition to the configuration directives, Urchin can be used from a script using command line options. The “-G” flag can be used to determine the report group. For example:

```
./urchin -r /www/mysite/urchin -R mysite.com -G mygroup -l logs
```

If you are using command line scripting to run Urchin for each site, one at a time, then the “-G” flag can be used to specify the group. Without the “-G”, Urchin will use the “default” group. In the above example, the group is specified as “mygroup”.

Follow the instructions in 7.1 for setting up the group directories.

7.3 Using the CGI Specification

This option is usually reserved for use with centralized reporting. Under centralized reporting, all data for all sites is stored centrally, and each user accesses their reports from the central location. In order to figure out which report to deliver, the `urchin.cgi` application needs to know the *username* of the authenticated user, and optionally, the *groupname*. This group name is then used to deliver custom branding, etc. in the normal fashion.

After setting up groups as per section 7.1, we need to pass the “GROUP” environmental variable to Urchin via the API. Please contact our technical support if you are using centralized reporting for more information.

7.4 Specify Groups Manually

Since the preferences file is human-readable, it is fairly easy to edit this file either manually or with a script. The file is located in each site's report directory in the "udata" subdirectory:

```
cd /reportdirectory/udata/  
vi prefs.udb
```

The preferences file contains the group information in the following format:

```
ReportName:   site1.com  
ReportGroup:  group-a  
SerialNumber:  
LicenseCode:  
LastHit:     955325875  
Language:    english  
UrchinLogo:  0
```

Each line contains one name-value pair separated by a colon with potential white space. Using Perl or some other scripting language, administrators should easily be able to edit this file and change the ReportGroup entry. As long as the ReportGroup entry is not specified in the configuration, Urchin will not overwrite.

The preferences file is then used by the reporting engine to determine which group the site belongs to, and which set of navigation, branding, icons, etc. to use. Follow the instructions in 7.1 for setting up the group directories and the instructions in the next chapter for customizing Urchin.

8. Customizing and Co-branding Urchin 3

This chapter describes the different methods for customizing the look and feel of all reports produced by the Urchin 3 interface. If it is desired that the customizations apply to a group of users, please see the chapter on “Setting up Groups of Users in Urchin 3” for more information on that subject. This section covers:

- Co-branding the Top Frame of Urchin.
- Modifying the Navigation.
- Modifying Text and Help Text.
- Modifying Colors and Fonts.
- Modifying Icons.

Most of these changes are made in the distribution and require some knowledge of javascript and how web pages are put together. Since these changes are made in either the “ugroups” or “uicons” directory in the distribution, one should make a backup copy of any files before modification.

Please note that while it is possible and documented how to modify many features of Urchin, it is recommended that only the Top Frame be modified as described in section 1. Modifications of other components may have adverse effects on the form and function of the application.

8.1 Co-branding the Top Frame of Urchin



The top frame of Urchin can be easily modified to incorporate your company’s color, logo and, optionally, banner advertising. The top frame is actually two frames: a left frame containing the Urchin logo, and the right frame containing the company logo and banner space. The top right frame is easily modified by editing the HTML in the “topframe.html” file within the distribution. For the default group of users, this file is located in:

```
/distributionpath/ugroups/default/topframe.html
```

This file can be completely changed or simply modified to create the colors and effect you are looking for. After changing the file, simply reload the Urchin interface to see your changes. To simply modify the background color and company logo, follow these steps:

1. Edit the line `<BODY bgcolor=0000AA ...` changing the `0000AA` to the hex color of your choice. This will change the dark blue to your color.
2. Next, edit the company logo by changing the line `<IMG SRC="uicons/default/logo.gif" ...` to your company logo image. You can either replace the `logo.gif` in the `uicons` directory or change the `SRC` path to wherever it is located. You may need to change the height and width tags to reflect the new image.
3. Finally, you may need to modify or replace the `visreport.gif` file in the `uicons` directory so the background matches your new background color.



After reloading, you will see your new top frame branding. If you have chosen a different background color than the default blue, you may wish to change the Urchin background color so that it matches the entire top banner. To do this, edit the `topcolor.html` file located in the same place as the `topframe.html`. This file contains a single hex code for the color of the top left frame. For the default group this file is located in:

```
/distributionpath/ugroups/default/topcolor.html
```

Simply change the RGB hex value to the color of your choice. After writing the file, reload the Urchin interface to see the final changes. Your top frame branding should now be complete and consistent.



8.2 Modifying the Navigation

The menus and report features of Urchin can be optionally turned off or disabled. If, for example, you are not processing the referral information for a particular site, you may want to disable the referrals section of the navigation so there are no blank reports provided to your user group. Or, if there is no e-commerce associated with you may want to disable the e-commerce component of Urchin. There also might be a time when you want to show samples of a particular section and use the samples to upsell the end-user from one hosting service to another. Both of these options are available and are provided by using the following two javascript functions:

```
parent.disable(reportnumber);  
parent.upsell(reportnumber);
```

These entries are made in the `navigation.js` file located in the `ugroups` directory. If you are editing the default group, then the file is located at:

```
/distributionpath/ugroups/default/navigation.js
```

Simply edit the file adding the above commands, one per line. Some examples are provided in the file. The *reportnumber* is replaced with a code for the specific report or menu you wish to modify. For example, if you wish to turn off the “Directory Tree” report, you would disable report number 1202. Here is a complete list of report numbers:

Report number	Report Title
1100	Traffic*
1101	Snapshot*
1102	Summary
1103	Hourly Graph
1104	Daily Graph
1105	Monthly Graph
1106	Top Servers
1200	Pages
1201	Top Pages
1202	Directory Tree
1203	File Types
1204	Status/Errors
1205	Posted Forms
1300	Referrals
1301	Top Referrals
1302	Top Keywords
1303	Referral Tree
1304	Search Engines
1400	Domains
1401	Top Domains
1402	Domain Tree
1403	Top Countries
1500	Browsers
1501	Browser Tree
1502	Platform Tree
1503	Top Combos
1600	Tracking
1601	Top Entrances
1602	Top Exits
1603	Click Through
1604	Depth of Visit
1605	Usernames
1608	Length of Visit
1700	E-Commerce
1701	Totals
1702	Top Products
1703	Product Tree
1704	Regions
1705	Top Stores

The first two entries indicated with an (*) cannot be disabled. After making changes to the file, be sure to reload the Urchin interface in your browser and check your results.

8.3 Modifying Text and Help Text

All the text displayed in the navigation and reports can be modified. While a deviation from the default text may cause confusion in the extended help documentation or multi-lingual capabilities, it is possible to make some changes. Specifically, if you want the extended help links (provided at the bottom of each report) to link to your own help/support area instead of the www.urchin.com, follow these instructions.

All of the text information is provided in a language javascript file located in the distribution. For the default group, the directory:

```
/distributionpath/ugroups/default/languages/
```

contains all of the language files. It is important when editing these files to make sure the integrity of the javascript is maintained. Each word, phrase or paragraph used in Urchin is found in these language dictionaries. For example, if you wanted to change the word “Snapshot” to “SuperGraph” in the English version, you would edit the `english.js` file in the above directory, changing the line:

```
parent.ssLoad( 2, "Snapshot" );  
  
to  
  
parent.ssLoad( 2, "SuperGraph" );
```

After reloading the Urchin interface in your browser, you should see the modification. Be sure to hit the “reload” button. It is important that each entry remains on one and only one line and that all quotes, parenthesis, and other marks are intact for the javascript.

Help Text Links

In another example, if you wish to change the links in help information to point to your local support area, you could modify all the lines that contain:

```
href=\"http://www.urchin.com/help/en30/...html\"
```

to your own local links. Be sure to keep the backslash in front of each quote. This is critical for the javascript to pass the link correctly to the browser.

Upsell Information

If you plan on using the upsell feature of Urchin, you will want to edit entry number 67 in the language files. This is the text that is displayed below a sample report that what set as an upsell feature. The text can contain HTML images and links as long as all quotes are preceded with a backslash.

8.4 Modifying Colors and Fonts

This section describes how to change the colors and fonts used in the reporting and navigation areas. While it is straightforward to modify these features, it should be done with caution and only when necessary. Changes to the colors and fonts could have an

adverse impact on the functionality of the interface. Our graphics team spent many moons designing the thing for maximum functionality with all browsers and platforms. Certain fonts and colors work better across different systems, and changing these could cause unpredictable results.

All modifications for colors and fonts are accomplished by modifying the `graphics.js` file located in the appropriate group's directory. For the default group, this file is located in the distribution:

```
/distributionpath/ugroups/default/graphics.js
```

Upon inspection, this file contains some javascript that sets the icons directory, fonts, and colors used in the reporting and navigation. Modifying icons is described in the next section, but here we cover the fonts and colors. The file should look like:

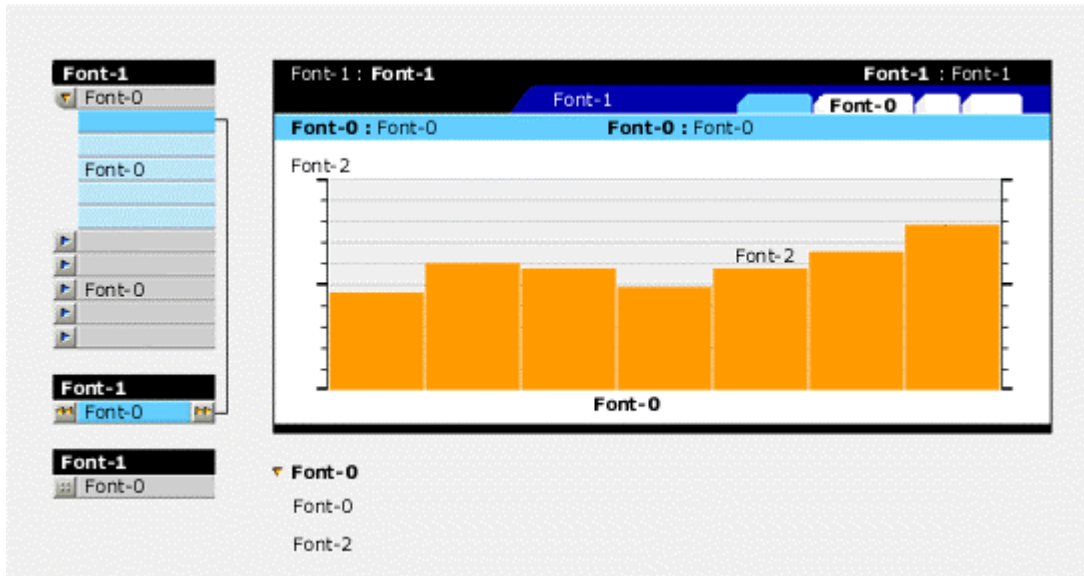
```
parent.icons = "default";

parent.fonts[0] = '<FONT FACE="Verdana,Arial,Helvetica" size=-2 color=black>';
parent.fonts[1] = '<FONT FACE="Verdana,Arial,Helvetica" size=-2 color=white>';
parent.fonts[2] = '<FONT FACE="Times Roman" size=-7 color=black>';

parent.colors[0] = "FFFFFF";
parent.colors[1] = "000000";
parent.colors[2] = "0000AA";
parent.colors[3] = "CCCCCC";
parent.colors[4] = "EEEEEE";
parent.colors[5] = "FFFFFF";
parent.colors[6] = "66CCFF";
parent.colors[7] = "B9E7FF";
parent.colors[8] = "FFCC66";
parent.colors[9] = "FFE699";
parent.colors[10] = "8FDD8F";
parent.colors[11] = "CCE9CC";
parent.colors[12] = "66CCCC";
parent.colors[13] = "AFEBEB";
parent.colors[14] = "CCCC66";
parent.colors[15] = "E3E3AA";
parent.colors[16] = "CCCCFF";
parent.colors[17] = "E3E9FF";
parent.colors[18] = "FF9900";
parent.colors[19] = "339933";
```

While easy to modify this file, it is critical that the javascript integrity is maintained. Only edit the blue text above and be sure to keep the rest of the text the same including all quotes, equal signs, spaces, etc.

Towards the top of the file, there are three lines that describe the fonts used in the Urchin reporting and navigation. These lines begin with `parent.fonts` and are subscripted zero through two. By modifying the blue text in each line, you can change the fonts. Since each font is subscripted with a number, we will refer to the first font as *font-0*, the next as *font-1*, and so forth. The following diagram shows the location of each font in the Urchin interface.



As indicated in the above diagram, the three fonts are used in both regular and boldface formats. In general, Font-0 is used as the standard black font; Font-1 is used as the white font; and Font-2 is used as a really small black font. After changing fonts, be sure to check all of the navigation, as the new fonts may have increased the width of the navigation words beyond their storage area.

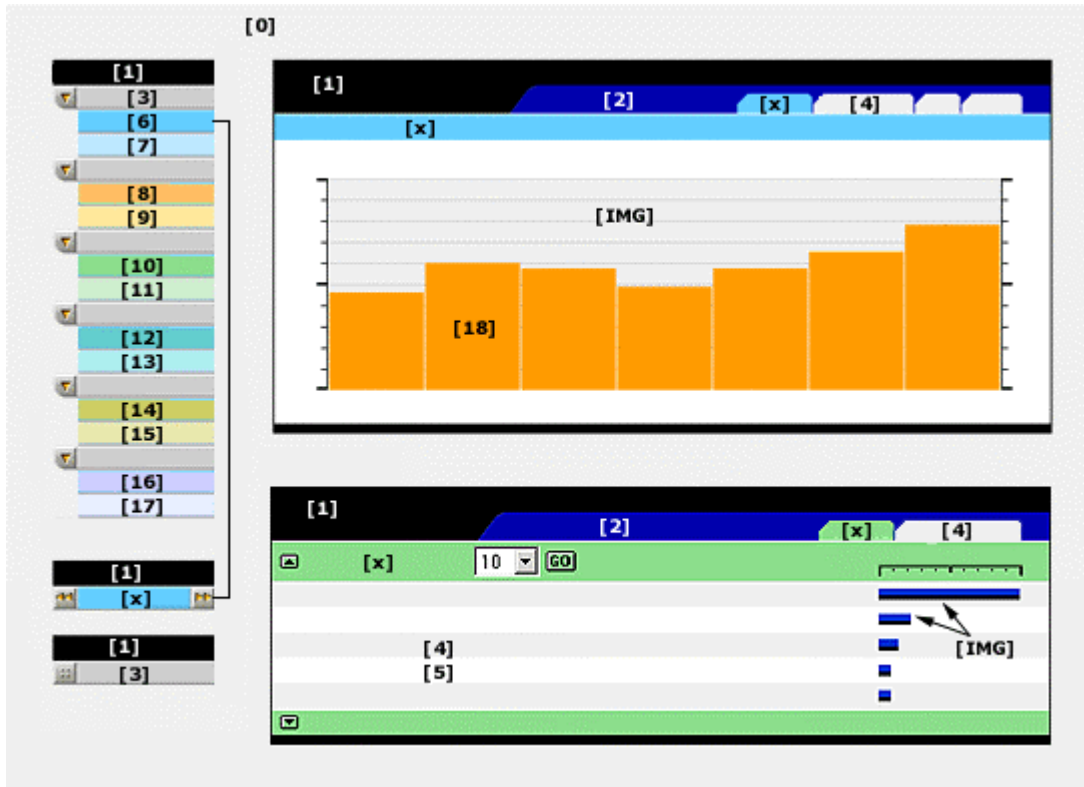
The colors used in the reports and navigation are changed in the same way. By modifying the blue text in the `graphics.js` file as described above, one can change just about every color in the reports. All of the color lines begin with `parent.colors` and are indicated by the subscript between the brackets. We will refer to the first color by its index, *Color-0*, the next color as *Color-1*, and so forth. To change one of the colors, simply edit the RGB hex value between the quotes. For example, to change the *Color-0* from white to yellow, you could change the line:

```
parent.colors[0] = "FFFFFF";
```

to

```
parent.colors[0] = "FFFF00";
```

Be sure to leave the quotes and equal signs intact. Also, it is probably a good idea to make a backup copy of this file before editing, so that you can always go back to the default colors. To see which colors affect which areas, refer to the diagram below.



The diagram shows an expanded view of the navigation and two reports so that all colors can be viewed. Colors are indicated in brackets. An [x] indicates that the color is controlled by the currently selected navigation item. An [IMG] indicates that the region or item is not controlled by a color, but is in fact an image in the icons directory. Be sure to reload the Urchin interface after changing colors. One extra thing to check is the visibility of the fonts on the colors behind them. Using various monitors, one should verify that the new colors do not affect text legibility.

8.5 Modifying Icons

This section describes the procedure for modifying icons in the distribution. In addition to the colors and fonts, there are a limited number of icons that are used to draw the navigation and reports. These icons (gifs) reside in the `uicons` directory within the distribution.

It is possible to have multiple sets of icons which can be used for different groups of users – see the chapter on setting up groups. The control for which set of icons is used in the reports is located in the `graphics.js` file, described above. Located in the `ugroups` directory, this file contains the line:

```
parent.icons = "default";
```

which specifies the icon subdirectory in the `uicons` directory to use. For example, the above entry means that the icons will be used from the directory:













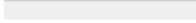






```
/distributionpath/uicons/default/
```

It is a good idea to create a working directory for icons and leave the default ones alone. This can be accomplished by running:

```
cd /distributionpath/uicons/  
cp -r default myicons
```

Edit ../ugroups/default/graphics.js **changing default to myicons**

This way you can mess with the icons in the `myicons` directory and still have the default ones for later use or reference. All icons are gifs and can be edited using standard tools such as Adobe Photoshop®. Be sure to keep the width and height of the images in pixels the same in order for the HTML tables to display properly. The following table of icons and filenames will help you figure out which icons to modify.

Filename	Image	Description
highlight.gif		Top of navigation buttons.
shadow.gif		Bottom of navigation buttons.
ind1.gif		Closed menu button.
ind2.gif		Open menu button.
ind12.gif		Button for date range.
ind13.gif		Button for date range.
ind14.gif		Button for unselected date range.
ind4.gif		Tools button.
ind5.gif		Tools button.
ind6.gif		Tools button.
bigtab.gif		Curve before report title.
tabside1.gif		Left side of tabs in report.
tabside2.gif		Right side of tabs in report.
hbar.gif		Horizontal graph bars
grid2.gif		Snapshot graph background.
green.gif		Green bar graph color.
up.gif		Previous button.
down.gif		Next button
go.gif		Go Button
ind9.gif		Closed tree element.
ind10.gif		Open tree element.
harrow.gif		Help icon.

9. Configuration Directives

The bulk of Urchin configurations are handled by the `config` file located in the Urchin distribution. This file contains configuration directives that tell Urchin how to run and what to process. There are two types of configuration directives: global and `<Report>` directives. The global directives guide the entire operation of Urchin including the System Report, whereas the `<Report>` directives are site specific. They define the location, URL, and other information associated with a specific site.

In addition to the `config` file, there are command line options that can be used from a script to specify single site reporting options.

9.1 Global Directives

`DistributionPath:` /usr/local/... (default = ./)

This directive sets the location of the Urchin distribution, which is needed for languages and icons.

`DNS_Nloops:` integer (default = 5)

This directive specifies the maximum number of DNS processing loops Urchin will use when resolving IP addresses in the webserver log files. The default is 5 loops, which is generally adequate. Urchin will finish DNS resolution processing after processing this number of loops, or if the resolution specified by the `DNS_Target` directive is met, whichever comes first. If you want greater resolution (at a performance cost), increase this number to a maximum of 8. Decrease the number if you want DNS resolution to run faster. Be sure to see the `DNS_Target` directive as well.

`DNS_Target:` 0-100 (default = 80)

This directive specifies the DNS resolution percentage that Urchin will strive for when resolving IP addresses in webserver log files. The default is 80%. If you want greater resolution (at a performance cost), increase this number up to a maximum of 100. Decrease the number if you want DNS resolution to run faster. Increased resolution may require scaling the `DNS_Nloops` directive up from the default of 5 as well.

`KeepVisitorData:` on|off (default = off)

This directive allows visitor data to be merge between Urchin processing. Normally, you will not need this directive, but if you have multiple logs for the same site, and process them at different times, this directive can be enabled. It will increase the size of data storage.

`License:` ABCD-ABC... (default = 0)

This directive will store the license string provided by our licensing center upon registration. Please see our website for information on obtaining a license.

Limit_DBTables: integer (default = 10000)

This directive controls the number of records that will be stored in the database files. This works similarly to how MAX_PAGES works on page entries, except that Limit_DBTables affects the domains, referrals, and browser reports. This value can be set arbitrarily low to conserve resources. To prevent any decrease in processing performance it is recommended that this value not exceed 30,000.

Max_Pages: integer (default = 998)

This directive specifies the maximum number of unique pages that Urchin will keep track of for the Pages report. If Urchin finds more unique pages than Max_Pages, it will lump all remaining pages into the page report as “other.html”. If you are seeing a lot of page counts for “other.html”, increase this number.

NOTE: runtime memory requirements increase significantly for values greater than 1600. It is recommended that you increase this value in small increments (e.g. 100 at a time) to decrease “other.html” pages.

ProcessBrowsers: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with browsers.

ProcessDNS: on|off (default = off)

If you wish to use Urchin’s fast reverse-DNS module, then you will want to turn on this directive. Be sure to specify the “ResolverIP” as well.

ProcessDomains: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with domains.

ProcessEcommerce: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with e-commerce. Urchin will automatically detect the presence of e-commerce data.

ProcessIP: on|off (default = off)

Changes behavior of the Domains report so that IP addresses are shown instead of resolved hostnames. In order for the ProcessIP directive to work, the ResolverIP directive must not be set. The Top Countries report will not produce a country list when only IPs are used for Domains reporting.

ProcessPages: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with pages.

ProcessReferrals: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with referrals.

ProcessTracking: on|off (default = on)

Normally on, this directive allows you to explicitly turn off the processing and data storage associated with tracking.

Quiet: on|off (default = off)

If set to “on”, this directive will turn off all runtime output from Urchin. This can be useful if you have a large number of sites and don’t want to see the output for each one.

ReprocessDay: YYYYMMDD (default = 0)

If this directive is specified, Urchin will only extract hits for the specified date when processing the webserver logs. Urchin will first zero out the statistics for that day, then reprocess that day’s entries. This is useful to redo a particular day’s log file(s) if the logs were incomplete, corrupt or missing when Urchin was originally run.

ResolverIP: 1.1.1.1 (default = 0)

If you are using the Urchin DNS module, you will need to specify the IP of a local caching DNS machine. It can be the local machine or one on the local network.

RestartCommand: kill -HUP... (default = _none)

This directive allows you to call a restart command in order to rotate logs. If the LogDestiny directive is set to archive, Urchin will rename all log files by appending a date stamp. It then calls this command which to restart the webserver and thus reopen new logs, and then continue processing the renamed logs.

RotateCommand: rotatelogs.pl (default = _none)

This script/command specified by this directive is called before any webserver logs are touched. Typically, this command would be used to call a site-specific logrotation script.

Serial: 1234-123... (default = 0)

The licensing of Urchin requires both a serial number and a license code. Check with our website to purchase a serial number.

ServerName: name (default = default)

This directive can be used to specify a server name which is then reported on in the System Report.

ShiftUTC: on|off (default = off)

This directive will use the offset in Apache logs to calculate the time of each entry in UTC time, which will change the way the report data is displayed.

Note: IIS logs already enter all log data in UTC.

Symlinks: linkall|linknone|linkdir (default = linkall)

Controls the way Urchin uses symbolic links in the ReportDirectory directory. By default Urchin will create symbolic links for ugroups, uicons, and index.cgi in ReportDirectory that point back to the real location of the files in the Urchin installation directory. This saves disk space by not copying files unnecessarily. The default can be changed to address special web reporting configurations (e.g. ones where links to executable programs outside the web documents area are not allowed).

The options for Symlinks are:

Linkall: Create ugroups, uicons, index.cgi symlinks (default behavior)

Linknone: Don't create any of the standard symbolic links

Linkdirs: Create symbolic links for ugroups and uicons directories, but copy urchin.cgi as index.cgi into ReportDirectory.

SystemDirectory: _____ /usr/local/... (default = ./)

Use this directive to specify the location of the System Report. The System Report needs its own directory for reporting.

SystemReport: _____ on|off (default = off)

Enable this directive to turn on the System Report function of Urchin. Be sure to specify a valid System Directory as well.

Testonly: _____ on|off (default = off)

Usually used as the -t flag on the command line, this directive will not actually process log files. Urchin will just print out the current configuration.

TimeOffset: _____ integer|LocalTime (default = 0)

This directive will shift data back from UTC time into any time zone chosen. Can be specified in seconds or as LocalTime. Seconds can be specified as positive or negative to go in either direction from UTC. If the string LocalTime is specified the data will be shifted back to the local time for the server on which Urchin is running. When this directive is used you must also set ShiftUTC to "on", otherwise TimeOffset is ignored.

Urchin_Priority: _____ High|Normal|Low (default = _Normal)

This sets the value of Urchin's runtime process priority. Setting this value allows tuning of Urchin's CPU usage.

VarCode: _____ HostingCompany (default = off)

This directive specifies the Reseller code which is used when end-user customers do licensing so that the licensing transaction is flagged with the Hosting Company's identification.

Visitor_Timeout: _____ integer (default = 1800)

This directive specifies the number of seconds between hits from the same IP address that define a Visitor (a.k.a user session). By default, if no hits are seen from a visitor IP address in 30 minutes (1800 seconds), Urchin will consider the next IP address from this IP address to be a new visitor.

9.2 Site Specific Directives

<Report>...</Report>

Each site report needs to be surrounded by one of these pairs. Much like an HTML tag, they signify the beginning and ending of a report entry. Each directive should be on its own line in the config file.

CentralizedReporting: on|off (default = off)

This directive specifies that all sites should use the same reporting directory and data structure. This is a special option designed for large on-line communities and is not included in the normal distribution.

DecodeURL: on|off (default = off)

This directive controls whether Urchin will translate HTML encoded characters (e.g. %A5) it parses in the URL request, the URL query string, the referral, or the referral keywords so that the real character is displayed in the reports. When set to off Urchin will simply map the percent sign to an underscore and display the rest of the HTML encoding as it appears (e.g %A5 will be reported as _A5). With DecodeURL set to on, %A5 would appear as the character it represents, the yen currency symbol ¥.

DefaultPage: pagename.suffix (default = _none)

Used to specify the name of the default page Urchin will use to report page views for URL requests such as “http://www.site.domain/”. Some common choices for DefaultPage may include index.html or default.asp.

DynamicURL: /index.cgi?(.*) (default = _none)

This filter applies to only the next log entry. Using regular expressions, it can be used to reproduce a URL from a dynamic one that would ordinarily not report correctly.

EcommerceFormat: elf (default = elf)

This directive specifies the log format of all e-commerce logs that appear after this directive. While it’s not normally needed, it is used on special installations. Currently only ELF (E-Urchin Log Format) is supported.

EcommerceLog: /logs/ecommerce_log (default = _none)

Specifies the location of an e-commerce log file. Be sure to use the full path to the file. See the [TransferLog](#) directive for a description of how to specify a log file name that includes a timestamp.

ErrorFormat: apache (default = error)

This directive specifies the log format of all error logs that appear after this directive. While it’s not normally needed, it is used on special installations.

ErrorLog: /logs/error_log (default = _none)

Specifies the location of an error log file. Be sure to use the full path to the file. See the [TransferLog](#) directive for a description of how to specify a log file name that includes a timestamp.

FilterField: integer (default = 4)

This directive specifies which field in the log line to filter using the FilterIn, FilterOut, and DynamicURL filters. The default is the fifth field which is the URL request. The first field is number zero.

FilterIn: good\.html (default = _none)

This directive specifies a regular expression filter that is applied to only the next log entry. Used in conjunction with the FilterField directive, this directive requires that all hits match the filter. Multiple filters can be used.

FilterOut: bad\.html (default = _none)

This directive specifies a regular expression filter that is applied to only the next log entry. Used in conjunction with the FilterField directive, this directive eliminates all hits that match the filter. Multiple filters can be used.

LogDestiny: delete|archive|dont touch (default = dont touch)

This directive specifies what to do with logs after processing. The default is to not do anything which assumes that you have control over your own log rotation. The delete or archive options should be used with Urchin log rotation.

LogEncoding: en|jp (default = en)

This directive toggles Urchin's ability to process log files that contain Japanese double byte character. For a given log file, choosing en or jp will produce somewhat different numbers in the Keyword and Search Engine reports. This is because the character encoding in the keywords, which affects both these reports, will be interpreted differently based on your LogEncoding setting. There is some potential additional overhead in processing time when using the jp setting.

LogFormat: combined|iis (default = combined)

This directive specifies the format of the log entries that follow it. The default is the extended combined format (Apache, Netscape, most Unix web servers). Microsoft IIS 4.0 and 5.0 WC3 log formats are also supported.

QueryToken: character (default = ?)

This directive allows the user to select what character will be used to separate the base URL in a GET request from the query string portion of the request (e.g. such as a parameter list passed to a CGI script). Typically the separator is a question mark, but in some cases a semi-colon is used.

ReferralLevel: integer (default = 2|3)

This directive sets the granularity with which Urchin will report on domains in referral URLs. The default behavior is that if Urchin processes a referral with a standard US domain (e.g. .com, .gov, .mil) it will report 2 levels of the domain for the referral, so ftp.quantified.com would be reported as quantified.com in the Top Referrals report. If Urchin processes a referral for a foreign domain (e.g. .fr,, .nl) it will report 3 levels, so ftp.quantified.co.uk would be reported as quantified.co.uk. By setting ReferralLevel to a specific integer you can force all referral domains to be reported with the same number of levels.

ReportDirectory: /www/logs/... (default = ./)

This directive specifies the report directory for the current report. A unique complete path should be specified for each site in the configuration.

ReportDomains: widget.com,widgets.com (default = "")

This directive specifies the list of comma-separated domains that should be considered synonyms for the site that you are doing the reporting on, up to a maximum of 499 characters. This directive should be specified when you have multiple domains that point to the same website (e.g. urchin.com and quantified.com) so that Urchin will properly filter internal referrals from these domain names. It should also be used if you want to use an arbitrary name in the ReportName directive. If ReportDomains is not specified, Urchin will use the domain name given in the ReportName directive.

ReportGroup: brandx (default = default)

This directive specifies the report group for the current report. This report group name can be used to determine branding and other options at report time.

ReportLanguage: english (default = english)

While the user can change the display language on their own, this directive allows you to specify which language is used by default.

ReportLicense: ABCD-ABC... (default = 0)

For domain based licenses, this directive is used to store the license code for this report. Normally, this is entered through the reporting interface, itself.

ReportName: mysite.com (default = default)

The name of each report should be specified using this directive. For domain based sites, one should use the domain as the name without the www. Urchin will recognize this as the primary domain of the site and properly filter external referrals. If you wish to use an arbitrary report name, or you have multiple domains that point to the same site (e.g. urchin.com and quantified.com), you should specify both the primary domain name and any secondary domain names in the ReportDomains directive.

ReportSerial: 1234-123... (default = 0)

For domain based licensing, this directive specifies the serial number of the license. This is normally entered directly through the reporting interface.

SubreportField: integer (default = 4)

Used in conjunction with SubreportMode, this directive specifies which field to use for filtering the name of the site from. The first field is zero, and the default is the URL request field.

SubreportFilter: /www/(.*)/docs/ (default = _none)

This regular expression filter is applied during SubreportMode operation to determine the name of the site being reported on. The information in parenthesis is kept and used as the "\$1" variable. This variable is then used in the ReportName and ReportDirectory directives.

SubreportMode: on|off (default = off)

When on, this directive specifies that multiple websites are represented in one log file. Using the SubreportFilter and SubreportField, Urchin can determine which site a particular hit belongs too. See the Manual for details.

TransferLog: /logs/access_log (default = _none)

This directive specifies the complete path to a log file for processing. Multiple transferlogs can be used under certain licensing operations. The LogFormat directive specifies the format of this file. If the log file is specified with the pattern YYYYMMDD in it, i.e. /logs/access_log.YYYYMMDD, Urchin will look for a log file with yesterday's timestamp substituted in for the YYYYMMDD section of log specification. For example, if Urchin is run on 02/14/2001, it would substitute the pattern 20010213 in the log file name and actually process /logs/access_log.20010213.

9.3 Command Line Options

Command line options can be used to control or modify the operation of Urchin either in conjunction with or without entries in the config file.

Command line options that correspond to global config file directives should cause Urchin to read the entire config file, then substitute the command line specified global options for the ones that were set in the config file.

Site specific directives (i.e. those within <Report></Report> blocks) must be used in conjunction with the -l LogFile option. If the -l option is used to specify a log file, then Urchin will only use the config file to read in values set by the global directives. It will not read any site specific directives within <Report></Report> blocks. It's important to recognize that this means that any other site specific directives will then also need to be specified explicitly on the command line. Any site specific directives that are not specified on the command line when the -l option is used will assume their default values as described in the section 9.2. Conversely, site specific directives specified on the command line will be ignored if the -l option is not also used to specify the log file.

Command line flags modify Urchin's behavior as follows:

- | | |
|------------------|---|
| -cow | This option should be used by itself after all sites have been processed. This is only needed if you are running sites via the command line options. This option instructs Urchin to finish up the System Report. |
| -C en jp | Toggles LogEncoding behavior to allow processing of logs with Japanese double characters |
| -d YYYYMMDD | Reprocesses only log file data for hits that match the date specified. Any existing statistics for this day are zeroed out before the log(s) are processed. |
| -D DNSresolverIP | Turns on reverse DNS and sets the IP address of the DNS server to be |

used for reverse lookups.

-e ErrorLog [ErrorLog] Specifies the name of the error log(s). Log file names must be separated by spaces. The Unix wildcard asterisk can be used within filenames.

-E EcommerceLog [EcommerceLog] Specifies the name of the e-commerce log(s). Log file names must be separated by spaces. The Unix wildcard asterisk can be used within filenames.

-f ConfigFile Sets the location and name of the configuration file. The default location is the distribution path and the default name is "config".

-F Format Sets the format for all log file(s) used on the command line.

-g integer Set the number of levels in a referral that will be displayed in reports (see ReferralLevel directive)

-G ReportGroup Sets the group name of the report, which can be used for customizing the interface.

-h Displays the help information you are viewing now.

-k Generate key: Urchin creates a licensing key needed for the Datacenter version.

-l LogFile [LogFile] ... Sets the location and name of the log file(s).

-L ReportLanguage Sets the reporting Language if it's not already set in the prefs.udb file.

-N ReportDomain [,ReportDomain] Specifies the alternate names that represent the same website in order to filter out internal referrals. Must be a comma separated list.

-notracking Turns off Tracking reports which tend to use a lot of resources.

-p Urchin_Priority Sets the runtime process priority for Urchin. Excepted values are High, Normal, and Low.

-P DefaultPage Sets the value for the default page.

-q Quiet mode: Suppresses details about logs and reports that are otherwise printed out while Urchin runs.

-Q "char" Sets QueryToken used to separate a URL from its query string. It's advisable to put double quotes around the chosen character to prevent interpretation by the command shell.

-r ReportDirectory Sets the location of the report directory.

-R ReportName Sets the name of the report.

-t Test mode: Directs Urchin to read the configuration, print out the configuration details to stdout, and then immediately exit.

-U Toggles DecodeURL behavior to decode HTML encoded characters

10. The e-Urchin Module (e-Commerce Reporting)

ELF is an e-commerce log format that allows in-depth analysis of shopping behavior on e-commerce enabled websites. Combining the webserver's standard access log with ELF, online merchants gain unprecedented insight into the relationship between all website visitation parameters and actual money spent - Return on Investment Reporting is here!

ELF is produced by several popular e-commerce systems, including Miva Merchant 2.0 and above. If your e-commerce package does not support ELF, please consult the urchin.com site for the latest supported formats, or request that your vendor include ELF support.

10.1 Architecture

Urchin allows multiple log files to be processed and merged before creating reports. Visitors from one log file are compared and correlated with the other log files. With the addition of e-commerce reporting, this feature can be used to correlate data from the normal webserver log with the ELF e-commerce log. In order to take advantage of this feature, there are two preparatory steps:

1. Create the ELF log file with your e-commerce system.
2. Make both the ELF log and the normal webserver access log available on the same system.

The first step is to create the ELF log file from your on-line transaction system (if you use Miva Merchant 2.0+ you have already got the ability). If your e-commerce system is proprietary or uses simple cgi scripts, you should be able to have your site administrator add the ELF log functionality to the system. A complete description of the log file format follows below.

If your system is off the shelf or provided by an ISP or hosting company, please forward them to <http://www.urchin.com/> for more information. We have many resources available and are currently working with many merchant software companies to provide seamless integration.

Once you have the ELF log file format created, you will need to have the file accessible along with the normal webserver access log. This can be trickier than it seems as often the main webserver can be a completely different machine than the secure machine running the e-commerce system. You will need to work with your system administrator to see which solution makes the most sense for your system. If the log files are on different machines, you may need to use a script, NFS, FTP or some other way to transfer the log from one machine to another.

10.2 Configuration

Once both log files are available to the Urchin program, you will need to edit the Urchin config file to activate the e-Urchin module and take advantage of the cross correlation. The following example can be used for most systems:

```
<Report>
ProcessEcommerce:      on
ReportName:            yoursite.com
ReportDirectory:       /www/yoursite
TransferFormat:        combined
TransferLog:           /usr/local/apache/logs/access-log
TransferFormat:        elf
TransferLog:           /usr/local/commerce/logs/elf-log
</Report>
```

Specify the TransferFormat of each log file BEFORE the actual log file entry.

10.3 ELF, in Detail

The E-Urchin Logfile Format (ELF) format is a tab-separated, multiline format. The transaction begins with the "!" exclamation mark character (which is thusly prohibited from the rest of the data). The first line contains the geographic and overall information on the transaction. Subsequent lines contain details on individual products. The basic format of the file will look something like:

```
!transfield1transfield2...
productfield1productfield2...
productfield1productfield2...
.
.
.
!transfieldtransfield2...
etc.
```

blank fields should contain a "-" dash character. The format of the transaction line is as follows:

```
\!% {ORDERID}%h% {STORE}% {SESSIONID}%t% {TOTAL}% {TAX}% {SHIPPING}% {BILL_CITY}
}% {BILL_STATE}% {BILL_ZIP}% {BILL_CNTRY}
```

where

% {ORDERID}	is the order number
%h	is the remote host (see apache.org)
% {STORE}	is the name/id of the storefront
% {SESSIONID}	is the unique session identifier of the customer
%t	is time in the common log format (see apache.org)
% {TOTAL}	is the transaction total including tax and shipping. (decimal only, no "\$" signs)
% {TAX}	is the amount of tax charged to the subtotal

%{SHIPPING}	is the amount of shipping charges
%{BILL_CITY}	is the billing city of the customer
%{BILL_STATE}	is the billing state of the customer
%{BILL_ZIP}	is the billing zip of the customer
%{BILL_CNTRY}	is the billing country of the customer

the product line format is:

```
%{ORDERID}%{PRODUCTCODE}%{PRODUCTNAME}%{VARIATION}%{PRICE}%{QUANTIT
Y}%{UPSOLD}
```

where

%{ORDERID}	is the order number.
%{PRODUCTCODE}	is the identifier of the product.
%{PRODUCTNAME}	is the name of the product.
%{VARIATION}	is an optional variation of the product for colors, sizes, etc.
%{PRICE}	is the unit price of the product (decimal only, no "\$" signs).
%{QUANTITY}	is the quantity ordered of the product.
%{UPSOLD}	is a boolean (1 0) if the product was on sale.

Here are some example entries:

```
!12313 ppp-46.mia-tc-2.netrox.net StuffStore 1102323131
[27/Jul/1999:11:43:02 -0700] 198.12 8.12 10.00 Cedar Rapids Iowa 52403 US
12313 102 T Shirt XL 10.00 10 0
12313 103 Boxers L 9.00 10 0
!12314 213.12.54.123 - 110123413 [27/Jul/1999:11:43:02 -0700] 11.75
0.75 1.00 Santa Ana CA 92705 US
12314 102 T Shirt S 10.00 1 0
```

For more information, please check the urchin.com website or contact tech support: support@urchin.com

Currently Supported Shopping/e-Commerce Systems:

Miva Merchant 2.0 and above
ShoppingQ 2.0 and above

Appendix A. Sample Installations

Example 1

This example is a single site single log file setup that uses Urchin to do DNS and rotate logs. Urchin is being used to process both the access-log and error-log produced by Apache. The System Report is not really necessary since there is only site.

```
#-----  
# Urchin 3 Config file  
#-----  
  
### Section 1: Global Configuration  
  
RestartCommand:    /usr/local/apache/bin/apachectl restart  
LogDestiny:        delete  
  
ProcessDNS:        on  
ResolverIP:        127.0.0.1  
  
### Section 2: Report Configurations  
  
<Report>  
ReportName:        knobs.com  
ReportDirectory:   /www/knobs/urchin/  
TransferLog:       /usr/local/apache/logs/access-log  
ErrorLog:          /usr/local/apache/logs/error-log  
</Report>
```

Example 2

This example is a multi site example. Each site has its own access-log file located in the logs directory of the site's document root. Urchin is set to process DNS and rotate logs. We do have the System Report on so we can total bandwidth usage for the entire server.

```
#-----  
# Urchin 3 Config file  
#-----  
  
### Section 1: Global Configuration  
  
RestartCommand:    /usr/local/apache/bin/apachectl restart  
LogDestiny:        delete  
  
ProcessDNS:        on  
ResolverIP:        127.0.0.1  
  
SystemReport:      on  
SystemDirectory:   /www/webmaster/urchin/  
  
### Section 2: Report Configurations  
  
<Report>  
ReportName:        bobsknobs.com  
ReportDirectory:   /www/bobsknobs.com/urchin/  
TransferLog:       /www/bobsknobs.com/logs/access-log  
</Report>  
  
<Report>  
ReportName:        stevesleaves.com  
ReportDirectory:   /www/stevesleaves.com/urchin/  
TransferLog:       /www/stevesleaves.com/logs/access-log  
</Report>
```

```

<Report>
ReportName:          nicksslicks.com
ReportDirectory:    /www/nicksslicks.com/urchin/
TransferLog:        /www/nicksslicks.com/logs/access-log
</Report>

```

Example 3

This example is a multi site example that uses the one log file for all sites mode. There is only one access-log file for the entire server. But, the log format was modified in the Apache configuration to:

```
LogFormat "%h %v %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
```

The second field of the log format, “%v” captures the Virtual Host name of each hit. For example, log lines may look like:

```

129.22.42.123 www.bobsknobs.com [14/Apr...
129.22.42.123 www.stevesleaves.com [14/Apr...
129.22.42.123 www.bobsknobs.com [14/Apr...
129.22.42.123 www.nicksslicks.com [14/Apr...

```

Urchin is set to process DNS and rotate logs. We do have the System Report on so we can total bandwidth usage for the entire server and rank sites. Assuming that the directories for each site are /www/bobsknobs.com/, /www/stevesleaves.com/, etc., we are going to use Urchin with subreport mode to filter the second field of the log file (field #1).

```

#-----
# Urchin 3 Config file
#-----

### Section 1: Global Configuration

RestartCommand:    /usr/local/apache/bin/apachectl restart
LogDestiny:        delete

ProcessDNS:        on
ResolverIP:        127.0.0.1

SystemReport:      on
SystemDirectory:   /www/webmaster/urchin/

### Section 2: Report Configurations

<Report>
SubreportMode:     on
SubreportField:    1
SubreportFilter:   www\.(.*)
ReportName:        $1
ReportDirectory:   /www/$1/urchin/
TransferLog:       /usr/local/apache/logs/access-log
</Report>

```

Example 4

This example uses a Perl script to process all sites. Each site has an entry in a database and its own log file. The global parameters are specified in the config file, and then the site specific info is passed on the command line. The global info does not contain a restart command as log rotation must be handled outside of Urchin in this case. There are multiple access files for each site, so the wildcard is used on the command line.

```
#-----  
# Urchin 3 Config file  
#-----  
  
### Section 1: Global Configuration  
  
ProcessDNS:          on  
ResolverIP:         127.0.0.1  
  
SystemReport:       on  
SystemDirectory:   /www/webmaster/urchin/
```

The Perl systems calls are equivalent to:

```
cd /usr/local/urchin3088/urchin;  
./urchin -R bob.com -r /www/bob.com/urchin -l /www/bob.com/logs/access*  
./urchin -R carl.com -r /www/carl.com/urchin -l /www/carl.com/logs/access*  
./urchin -R dave.com -r /www/dave.com/urchin -l /www/dave.com/logs/access*  
.  
.  
./urchin -R zoe.com -r /www/zoe.com/urchin -l /www/zoe.com/logs/access*  
./urchin -cow
```

After all sites are processed, the `-cow` option is used to finish up the System Report. This is necessary when running command line options.