GROUNDWORK
OPEN SOURCE

GroundWork Distributed Monitoring Agent Build and Installation Instructions

Revision 0.4.1

January 6, 2008

Prepared by: **GroundWork Open Source, Inc.**
139 Townsend Street, Suite 100
San Francisco, CA 94107

Contents

List of Tables

1 Overview

The GroundWork Distributed Monitoring Agent (GDMA) provides prepackaged software to probe status and metric data on a variety of hosts and feed the data back to a central monitoring server. An important goal is to make this software very easy to deploy on all the supported platforms, using each platform's native packaging system. That allows the local administrator to install and maintain the agents using the site's standard software distribution mechanisms, with very little manual intervention.

This manual does not currently delve into the general capabilities of monitoring with GDMA, nor of how its configuration files are distributed out to monitored machines. Rather, we just cover the basics of getting the software deployed in production at a customer site.

1.1 On-Site Deployment Objects

The following objects get installed on the distributed machines to be monitored:

- a `gdma`:`gdma` user/group, used to own the distribution files and to run the daemon and plugins
- a dedicated GDMA file tree, which includes:
  - the home directory for the `gdma` user
  - a daemon process (`gdma_check.pl`) to run the configured plugins periodically
  - a script to fetch updated configuration files from the monitoring server (`gdma_getconfig.pl`)
  - a script to send status/metric results to the server (`send_nsca.pl`)
  - a set of plugins for each supported platform, to probe status and metric info on the machine
  - whatever libraries are needed to support the plugins
  - secure-access key files, for sending results back to the central monitoring server
  - config files
  - log files

  Under Linux, this file tree is installed as `/usr/local/groundwork/` . Under Solaris, this file tree is installed as `/opt/groundwork/` .
- the `/etc/init.d/gdma` script, used to start and stop the daemon process

These distributed-machine objects are all deployed on a given machine using two packages in that platform's native format, one for the base product and one to supply the secure-access key files which have been generated for that customer's monitoring server.

The following objects get deployed on the central monitoring server:

- a `gdma`:`gdma` user/group
- the home directory for the `gdma` user
- login authorization keys (populated from the public-key files installed on all the distributed machines), used to control which machines are allowed to send results to the monitoring server
- configuration files to be distributed out to the monitored machines

(FIX MAJOR: briefly describe how the server-side objects are installed and managed)

1.2 Software-Build Objects

The following objects are involved in building the software for distribution:

- the Subversion repository for the software, in both pre-built and ready-to-build forms
- makefiles and other scripts used to build the various components
- a release of the standard Nagios plugins distribution, drawn from `http://nagiosplugins.org/`
- ancillary files maintained in Subversion for this product

The following chapters describe how to install the software once you have a complete build in hand, and how to build both the base-product packages and the ancillary key packages.

2 GDMA Delivered Software Components

This chapter presents details of what is delivered with the GroundWork Distributed Monitoring Agent, and how to install and configure it.

(FIX MAJOR: still to describe in this chapter:

- installing server-side GDMA software
- locally modifying the server-side copies of GDMA config files, to reflect the desired monitoring profiles for different classes of machines
- how to determine the content of the GDMA config file that gets distributed to each GDMA-monitored machine, to allow different monitoring profiles to be operating on different classes of customer machines

and anything else the administrator must do on the server side)

2.1 Supported Platforms

GDMA is supported on a large number of platforms, for both Linux/x86 and Solaris/SPARC. Supported Linux releases currently include:

- RHEL 4 32-bit
- RHEL 4 64-bit
- RHEL 5 32-bit
- RHEL 5 64-bit
- SLES 9 32-bit
- SLES 10 32-bit
- SLES 10 64-bit

RPMs are supplied for the above Linux platforms.

Native SVR4 packages are supplied for the various releases of Solaris. As of this writing, packages are available for the following Solaris versions:

- Solaris 2.6 SPARC
- Solaris 8 SPARC
- Solaris 9 SPARC
- Solaris 10 SPARC

Packages for Solaris 7 have not yet been produced, though there should be little technical difficulty in doing so. (The main issue would be finding and setting up a build machine.) In a pinch, the packages for Solaris 2.6 might work on Solaris 7, although there are package format changes somewhere between Solaris 2.6 and Solaris 9 that might preclude this. Support for other distributions of Linux and for Solaris/x86 will be considered as market needs dictate.

2.2 Delivered Linux Packages

The Linux software is delivered as a single RPM for each supported platform, for the base release:

```
gdma-2.0.1.rhel4.i386.rpm
gdma-2.0.1.rhel4_64.x86_64.rpm
gdma-2.0.1.rhel5.i386.rpm
gdma-2.0.1.rhel5_64.x86_64.rpm
gdma-2.0.1.sles9.i586.rpm
gdma-2.0.1.sles10.i586.rpm
gdma-2.0.1.sles10_64.x86_64.rpm
```

plus a separate RPM applicable to any of these platforms, containing the remote-access keys for the customer's particular GroundWork Monitor server:

`gdmakey-`*mygroundworkserver*`-2.0.1.noarch.rpm`

A copy of the latter file is created for each GroundWork Monitor server at the individual customer's site that will receive results from GDMA sources. See the following chapter for details on building it.

2.3 Linux Platform Installation

Installing the Linux software for GDMA occurs on both distributed and central machines.

2.3.1 Distributed Machine Installation

(FIX MAJOR: first install other required packages that the plugins depend on, if any)

Currently, the RPM installation will create a `gdma` user and `gdma` group in the local `/etc/passwd` and `/etc/group` files, with the `gdma` home directory being `/usr/local/groundwork/gdma/`. The password on this account will be invalid, to prevent its use via login. Since this act of creation is not synchronized across machines, it will likely result in different numeric UID and GID values being used on different machines. Fixing the inconsistent UID/GID across machines will take some RPM scripting changes, to allow the installation process to accept an already-existing user/group (say, one established in NIS or LDAP instead of local files) and to verify that the home directory for that user is correct.

On each Linux machine to be monitored, you must install the package for the corresponding Linux distribution, along with the access-key package, in that order. Add them to the system while logged in as `root`:

`rpm -Uvh` *gdma_package_file gdmakey_package_file*

Afterward, start the GDMA daemon on the same machine:

```
/etc/init.d/gdma start
```

2.3.2 GroundWork Monitor Server Installation

The `id_dsa.pub` file from the each of the gdmakey RPMs you have installed on the customer's distributed machines contains a DSA public key for the SSHv2 protocol. This file is found in `/usr/local/groundwork/gdma/.ssh/id_dsa.pub` on a representative machine where each distinct gdmakey platform RPM is installed. The contents of each such distinct file must be added to `~gdma/.ssh/authorized_keys` on the GroundWork Monitor server to which those machines will report results, to allow the respective clients to log in using DSA authentication. There is no need to keep the contents of this file secret.

2.4 Delivered Solaris Packages

For simplicity in building the packages, Solaris 2.6 is identified in our distributed filenames as Solaris 6, even though that was never the marketing name for this release.

The Solaris software for each OS release is delivered as three files:

- A package containing all the generic software. There is just one copy of this package for the platform, shared across all customers using GDMA. The delivered package filename will have a form like:

  ```
  GWOSgdma-2.0.4-solaris6-sparc.pkg.gz
  GWOSgdma-2.0.4-solaris10-sparc.pkg.gz
  ```

- A package containing the key-identification data for a particular customer's GroundWork Monitor server. The delivered package filename will have a form like:

  `GWOSgdmak-`*mygroundworkserver*`-2.0.4-solaris6-sparc.pkg.gz`
  `GWOSgdmakey-`*mygroundworkserver*`-2.0.4-solaris10-sparc.pkg.gz`

  where the particular customer's server name is included in the package name for easy identification. For Solaris 2.6 through early releases of Solaris 9, the package name is limited to 9 characters, so it is simply `GWOSgdmak` in spite of the longer identifying filename. Starting with Solaris 9 12/03, longer package names are available, but to build a package that will work across all versions of Solaris 9, we retain the shorter package names regardless. For Solaris 10, the package name can be up to 32 characters, so we take advantage of that and both extend the base package name ("key" instead of "k") and include the server name in the package name: `GWOSgdmakey-`*mygroundworkserver*.

- A public-key file for the particular customer's GroundWork Monitor server. The delivered filename will have a form like:

  `GWOSgdmak-`*mygroundworkserver*`-2.0.4-solaris6-sparc.id_dsa.pub`
  `GWOSgdmakey-`*mygroundworkserver*`-2.0.4-solaris10-sparc.id_dsa.pub`

  This file will be used on the GroundWork Monitor server to allow controlled access from the distributed machines being monitored. It is simply a copy of the `/opt/groundwork/home/gdma/.ssh/id_dsa.pub` file in the corresponding `GWOSgdmak-` or `GWOSgdmakey-`*mygroundworkserver*`-2.0.4-solaris`*release*`-sparc.pkg.gz` package, delivered separately for ease of installation on the central server.

The latter two files are created for the individual customer. See the following chapter for details on building them.

2.5 Solaris Platform Installation

Installing the Solaris software for GDMA occurs on both distributed and central machines.

Note: Installation of Solaris packages can occur in several different contexts: on a standalone machine; into a diskless client's file tree from the server that supports that file tree; into a JumpStart repository, a LiveUpgrade repository, a network-install repository, or a Flash archive; in a non-global zone (under Solaris 10); into a virtual-machine copy of Solaris; and so forth. As of this writing, the GDMA packages have only been written and tested in the standalone-server mode, meaning that they must be installed while you are logged into the individual machine on which the packages are to be installed. No relocation or client support is included at this time. On Solaris 10, only global-zone installation has been tested so far.

2.5.1 Distributed Machine Installation

Solaris 8 and earlier releases do not ship with a copy of SSH, either in the base OS or in the Software Companion packages. SSH is important to operation of GDMA because it allows the distributed agents to have their configuration files automatically updated. Specifically, the `scp` program is used for that purpose. If you install a copy in `/usr/local/bin/scp` (say, from www.sunfreeware.com packages, starting with the openssh distribution), you will need to create a symbolic link so this copy can be found by the GDMA scripts:

```
su root
cd /usr/bin
ln -s /usr/local/bin/scp
exit
```

On a Solaris 2.6 machine, some special setup must occur because the base operating system does not ship with a version of Perl already integrated. Perl is used by the GDMA software. The easiest way to install an up-to-date Perl is to visit www.sunfreeware.com and pick up the already-packaged copy of Perl for this version of Solaris. That copy of Perl installs the main binary as `/usr/local/bin/perl` whereas the GDMA scripts look for it as `/usr/bin/perl`, so a symbolic link must be created:

```
su root
cd /usr/bin
ln -s /usr/local/bin/perl
exit
```

On newer releases such as Solaris 9 and Solaris 10, Perl comes as an integrated part of the OS, already available under `/usr/bin/perl`, so no special action in this regard is needed on those platforms unless the standard Perl packages were not originally installed along with the operating system.

The GDMA software runs as a fixed `gdma` user. If you wish to have this account utilize consistent numeric UID and GID values across your monitored machines, you must set that up before installing the GDMA software. This can be done through LDAP, NIS, or the local password, shadow, and group files, as desired at the customer site. The password should be locked because no logins are expected on this account.

If the customer will create this account on the monitored machines, the relevant information is:

- User ID: `gdma`
- Group ID: `gdma`
- Home Directory: `/opt/groundwork/home/gdma`
- Password: locked (i.e., no direct logins are allowed)

If this is not set up beforehand, installation of the Solaris packages will automatically create a new local `gdma` user with the settings above, via the standard `/usr/sbin/useradd`, `/usr/sbin/groupadd`, and `/usr/bin/passwd` programs.

The `GWOSgdma` and `GWOSgdmak-` or `GWOSgdmakey-`*mygroundworkserver* packages for the appropriate release of Solaris must be installed in that order on each Solaris machine to be monitored. Unzip the delivered files, then add them to the system while logged in as `root`:

```
pkgadd -d GWOSgdma_package_file
pkgadd -d GWOSgdmakey-mygroundworkserver_package_file
```

Finally, start the GDMA daemon on the same machine:[1]

```
/etc/init.d/gdma start
```

If SSH is not available on the distributed machine being monitored, the GDMA software will not be able to initially fetch or later update its configuration file from the central GroundWork Monitor server. Failure to update is benign, in that the attempt to do so will be made periodically but it will fail without serious consequence. However, failure to fetch the initial configuration file is fatal to the monitoring, so it must be manually set in place. This is particularly an issue with Solaris 2.6 through Solaris 8, for which SSH does not come standard with either the base OS or the Software Companion packages. The customer can either compile it themselves or load prepackaged software from a trusted source such as the www.sunfreeware.com repository.

If manual installation of the config file is used, it must be placed here:

```
/opt/groundwork/home/gdma/config/gwmon_hostname.cfg
```

where *hostname* is the unqualified name of the machine. This is the node name (found by `uname -n`), not necessarily a network-interface-specific name. When `scp` is used to fetch the configuration file from the GroundWork server, the software assumes that is the name by which the GroundWork server knows the distributed machine being monitored.

2.5.2 GroundWork Monitor Server Installation

The `GWOSgdmak-`*mygroundworkserver*-*version*`-solaris`*release*-*architecture*`.id_dsa.pub` or `GWOSgdmakey-`*mygroundworkserver*-*version*`-solaris`*release*-*architecture*`.id_dsa.pub` file from each of the Solaris distributions of GDMA you have installed on the customer's distributed machines contains a DSA public key for the SSHv2 protocol. The contents of each of these files must be added to `~gdma/.ssh/authorized_keys` on the GroundWork Monitor server to which those machines will report results, to allow the respective clients to log in using DSA authentication. There is no need to keep the contents of this file secret.

1. We don't start the daemon automatically in a post-install script from within the package because we don't necessarily know the context in which the installation is being performed. For instance, if we were installing the software on the server's copy of a diskless client's filesystem, it would be inappropriate to start the agent on the server. Similar situations arise for constructing JumpStart images, and so forth. So until we have all such expected environments worked out, it's safest to simply insist on a manual startup (or a reboot, which would accomplish the same thing).

2.6 Plugins on Each Supported Platform

The currently available GDMA packages and their respective data-probing utilities are listed in the following table.

Table 1: Available Plugins

Plugin or Supporting File	Platform				
	Linux, all x86 varieties	Solaris 2.6 SPARC	Solaris 8 SPARC	Solaris 9 SPARC	Solaris 10 SPARC
`check_adptraid.sh`	●				
`check_apc_ups.pl`	●				
`check_appletalk.pl`	●				
`check_apt`	●	●	●	●	●
`check_asterisk.pl`	●				
`check_axis.sh`	●				
`check_backup.pl`	●				
`check_bandwidth.sh`	●				
`check_breeze`	●	●	●	●	●
`check_breeze.pl`	●				
`check_by_ssh`	●	●	●	●	●
`check_clamd`	●	●	●	●	●
`check_cluster`		●	●	●	●
`check_connections.sh`	●				
`check_cpu.pl`	●				
`check_dell_hw.pl`	●				
`check_dhcp`	●	●	●	●	●
`check_dig`	●			●	●
`check_digitemp.pl`	●				
`check_disk`	●	●	●	●	●
`check_disk.pl`	●				
`check_disk_remote.pl`	●				
`check_disk_smb`	●	●	●	●	●
`check_disk_smb.pl`	●				
`check_dl_size.pl`	●				
`check_dns`	●	●	●	●	●
`check_dns_random.pl`	●				
`check_dummy`	●	●	●	●	●
`check_email_loop.pl`	●				
`check_file_age`	●	●	●	●	●
`check_file_age.pl`	●				

Plugin or Supporting File	Platform				
	Linux, all x86 varieties	Solaris 2.6 SPARC	Solaris 8 SPARC	Solaris 9 SPARC	Solaris 10 SPARC
check_flexlm	●	●	●	●	●
check_flexlm.pl	●				
check_fping	●				
check_ftp	●	●	●	●	●
check_ftpget.pl	●				
check_game	●				
check_hpjd	●				●
check_hprsc.pl	●				
check_http	●	●	●	●	●
check_http_not.pl	●				
check_hw.sh	●				
check_ica_master_browser.pl	●				
check_ica_metaframe_pub_apps.pl	●				
check_icmp	●	●	●	●	●
check_if.sh	●				
check_ifconfig.pl	●				
check_ifoperstatus		●	●	●	●
check_ifstatus		●	●	●	●
check_imap	●	●	●	●	●
check_inodes-freebsd.pl	●				
check_inodes.pl	●				
check_ircd	●	●	●	●	●
check_ircd.pl	●				
check_jabber	●	●	●	●	●
check_javaproc.pl	●				
check_joy.sh	●				
check_ldap				●	●
check_ldap_conf.pl	●				
check_ldaps				●	●
check_linux_raid.pl	●				
check_lmmon.pl	●				
check_load	●	●	●	●	●
check_load_remote.pl	●				
check_log	●	●	●	●	●

Plugin or Supporting File	Platform				
	Linux, all x86 varieties	Solaris 2.6 SPARC	Solaris 8 SPARC	Solaris 9 SPARC	Solaris 10 SPARC
check_log2.pl	●				
check_logs.pl	●				
check_lotus.pl	●				
check_mailq	●	●	●	●	●
check_mailq.pl	●				
check_mem.pl	●				
check_meminfo.pl	●				
check_mrtg	●	●	●	●	●
check_mrtgtraf	●	●	●	●	●
check_ms_spooler.pl	●				
check_mssql.sh	●				
check_mssql2.sh	●				
check_mssql2000.sh	●				
check_mssql_errorlog.sh	●				
check_mssql_log.sh	●				
check_mysql	●				●
check_mysql.pl	●				
check_mysql_query	●				●
check_nagios	●	●	●	●	●
check_nagios_latency.pl	●				
check_nagios_status_log.pl	●				
check_nmap.py	●				
check_nntp	●	●	●	●	●
check_nntps	●	●	●	●	●
check_nrpe	●				
check_nt	●	●	●	●	●
check_ntp	●	●	●	●	●
check_ntp.pl	●				
check_ntp_peer		●	●	●	●
check_ntp_time		●	●	●	●
check_nwstat	●	●	●	●	●
check_nwstat.pl	●				
check_oracle	●	●	●	●	●
check_oracle_autoext.sh	●				

Plugin or Supporting File	Platform				
	Linux, all x86 varieties	Solaris 2.6 SPARC	Solaris 8 SPARC	Solaris 9 SPARC	Solaris 10 SPARC
check_oracle_invobj.sh	●				
check_oracle_logmode_new.sh	●				
check_oracle_maxext.sh	●				
check_oracle_maxprc.sh	●				
check_oracle_maxssn.sh	●				
check_oracle_online.sh	●				
check_oracle_spcrit.sh	●				
check_oracle_stats.sh	●				
check_oracle_status.sh	●				
check_oracle_tspfull.sh	●				
check_overcr	●	●	●	●	●
check_pcpmetric.py	●				
check_pfstate	●				
check_pgsql					●
check_ping	●	●	●	●	●
check_pop	●	●	●	●	●
check_pop3.pl	●				
check_procl.sh	●				
check_procr.sh	●				
check_procs	●	●	●	●	●
check_qmailq.pl	●				
check_real	●	●	●	●	●
check_remote_nagios_status.pl	●				
check_rpc	●	●	●	●	●
check_rpc.pl	●				
check_sap.sh	●				
check_sensors	●	●	●	●	●
check_sensors.sh	●				
check_simap	●	●	●	●	●
check_smart.pl	●				
check_smb.sh	●				
check_smtp	●	●	●	●	●
check_snmp	●				●
check_sockets.pl	●				

Plugin or Supporting File	Platform				
	Linux, all x86 varieties	Solaris 2.6 SPARC	Solaris 8 SPARC	Solaris 9 SPARC	Solaris 10 SPARC
check_spop	●	●	●	●	●
check_ssh	●	●	●	●	●
check_ssmtp	●	●	●	●	●
check_swap	●	●	●	●	●
check_swap.pl	●				
check_swap_remote.pl	●				
check_tcp	●	●	●	●	●
check_time	●	●	●	●	●
check_traceroute.pl	●				
check_udp	●	●	●	●	●
check_ups	●	●	●	●	●
check_users	●	●	●	●	●
check_vcs.pl	●				
check_wave	●	●	●	●	●
check_wave.pl	●				
check_wins.pl	●				
mrtgext.pl	●				
negate	●	●	●	●	●
packet_utils.pm	●				
rblcheck-dns	●				
rblcheck-web	●				
restrict.pl	●				
sched_downtime.pl	●				
urlize	●	●	●	●	●
urlize.pl	●				
utils.pm	●	●	●	●	●
utils.py	●				
utils.sh	●	●	●	●	●

3 Building the GDMA Packages

The GDMA software is found in the `factory` tree of the Operations Subversion, here:

> `http://geneva.groundworkopensource.com/svn/operations/trunk/factory/products/gdma`

For the base-product package, you can either check out the source code and build it yourself, or pick up the pre-built package for your platform, found in Subversion here:

> `.../factory/products/gdma/distribution/`*platform*`/`

In the general case, *platform* may include the OS name, the OS release level, and the CPU architecture. In some situations, it might also refer to the choice of compiler, and perhaps the compiler release level, because of potential incompatibilities in generated library code.

Our current standard list of *platform* abbreviations is as follows:

```
rhel4.i386
rhel4_64.x86_64
rhel5.i386
rhel5_64.x86_64
sles9.i586
sles10.i586
sles10_64.x86_64
solaris6.sparc
solaris8.sparc
solaris9.sparc
solaris10.sparc
```

We will make up a set of additional Solaris x86 platform names when we start getting customers that need them. With the packages we've built so far, we don't need to distinguish compiler versions under Solaris (the main issue to watch out for is the use of Sun compilers vs. GNU compilers, and different versions of the GNU compilers). We can deal with that if and when it arises.

For key packages, which are specific to the individual customer, we have a specific machine for building the key, where the build environment is already set up, so a PE can quickly take care of business. For Linux-based packages on any of the supported Linux platforms, the `furnace` machine will be used. For Solaris packages, our standard build machines for the various Solaris releases will be used.

We intend to keep these environments set up so there is no need to waste time playing around with Subversion checkouts every time a PE wants to create a key, and also so that whenever we make a change to the logic, that it gets updated in one place, and everyone gets the latest version without needing to be so careful about ensuring you have the latest release checked out.

We have a special login account for building official releases of this code, including customer-specific key packages. That account will be named `factory`, so it's very clear from the get-go what it's about.

If you build under the official `factory` account, you **MUST** observe the following rules:

- The `factory` account is only to be used for creating official code releases.
- The only thing you're ever allowed to do under this account is to check out a full release of code for a given module, verify that you've got a clean checkout, build, and clean up.
- No local code modifications or check-ins are EVER allowed from this account, to prevent subverting subversion.
- You're not even allowed to check in final distribution files from the `factory` account. Instead, you must log in as yourself, copy the files into your own checked-out copy of the `distribution` directory, and check them in as yourself. This leaves a trace of who did the work to produce the release.
- Since we have no automatic concurrency control on the use of the `factory` account, you must coordinate manually with whomever else might be wanting to do builds of the same product.

These rules are designed to avoid contaminating the official builds with in-progress or experimental code, to ensure that each build represents a controlled, exact snapshot of what has been checked out from Subversion. Also, the rule against `factory` check-ins ensures that all changes are properly tagged with the name of the user making the changes.

Whatever your platform, it is imperative that you follow the rules outlined below to ensure that the GDMA release number is properly updated whenever changes are made that affect the content of the packages. Under Linux, this means making a new specfile, and updating the associated version numbers in the specfile, in the `makefile`, and in any other place in the code that contains those numbers. Under Solaris, the software release is specified in the `makefile`. See below for details. This vigilance is necessary so the standard package-update mechanisms on each platform can understand what is being requested and how to respond correctly.

3.1 Preparation for Building under Linux

(FIX MAJOR: list our standard Linux build machine for each supported distribution and release.)

(FIX MAJOR: describe how to bump up the release number, in all places that need it: the specfile, the makefile, etc., and how to make sure those changes are checked into Subversion before you go making official builds)

(FIX MAJOR: fill this in with any other important information, such as describing exactly how the collections of pre-compiled Nagios plug-in binaries are built and saved)

3.2 Building under Linux

The Linux software is found under the `gdma/linux/` directory.

(FIX MAJOR: the checked-in Linux code is rather a mess. to begin with, lots of transient directories which are artifacts of the build are checked into Subversion when they should not be. also, the gdma_build.pl script does not have execute permissions when it is checked out.)

(FIX MAJOR: fill this in with other general information about the builds)

3.2.1 Building the Base GDMA Package

There is currently no `makefile` for this build, nor is there currently any platform-specific mechanism for re-creating all the plug-in binaries from scratch. Rather, we have one command that builds all RPMs for all the supported Linux platforms, based on some pre-built collections of binaries. The process for creating those collections is currently unspecified.

```
cd gdma/linux/gdma/bin
chmod +x gdma_build.pl
gdma_build.pl
```

The created RPM files are all left in the `gdma/linux/gdma/rpmbuild/RPMS/` directory.

3.2.2 Building a GDMA Key Package

We have a formal, supported key generation build environment on `furnace`. It is at:

```
/usr/local/factory/products/gdma/linux/
```

To build your key, go to:

```
/usr/local/factory/products/gdma/linux/gdmakey/bin/
```

And run:

`./gdmakey_build.pl` *groundwork_server_name groundwork_server_ip_address*

It will place the resulting RPM in:

```
/usr/local/factory/products/gdma/linux/gdmakey/rpmbuild/RPMS/
```

3.3 Preparation for Building under Solaris

Our standard Solaris build machines are:

`hemera` (Solaris 8)

`hestia` (Solaris 9)

`helios` (Solaris 10)

At the present time, Solaris 2.6 builds are done upon special request using an off-site machine.

Certain adjustments are necessary on particular OS releases to work around installed-software and build deficiencies, as described below.

`gcc` is generally provided in Solaris either as part of the base OS release or as part of the Solaris Companion distribution. For a very old Solaris release where this is not the case, www.sunfreeware.com will have a pre-packaged copy. The main thing to concern yourself with in that case is to check whether the compiled binaries have any dependency on gcc-specific libraries that would also need to be distributed with our packages. So far, that has not been the case.

In general for all Solaris releases (through Solaris 10), Subversion is not provided either in the base OS release or in the Software Companion packages. We therefore generally use the standard precompiled packages from www.sunfreeware.com to provide this critical component.

3.3.1 Solaris 2.6 Build Preparations

Solaris 2.6 does not come with a native Perl supplied by Sun. Perl is used during the build of the Nagios plugins, although currently none of the Perl-based plugins are included in the package for this platform. The easiest way to install an up-to-date Perl is to visit www.sunfreeware.com and pick up the already-packaged copy of Perl for this version of Solaris.

Solaris 2.6 also does not some with a standard release of SSH (needed for the `ssh-keygen` utility, which is needed to build ou GDMA key packages). We use the standard openssh and supporting packages from www.sunfreeware.com to provide this capability on this release of Solaris.

3.3.2 Solaris 8 Build Preparations

Solaris 8 likewise does not come with native SSH support, either in the base Solaris release or in the Software Companion packages. Therefore, we use the www.sunfreeware.com packages on this platform.

3.3.3 Solaris 9 Build Preparations

On Solaris 9, you will need the `SFWoldap` package installed, which provides:

```
/opt/sfw/lib/liblber.so.2
/opt/sfw/lib/liblber.so.2.0.122
/opt/sfw/lib/libldap.so.2
/opt/sfw/lib/libldap.so.2.0.122
```

You will also need to modify this file:

```
/opt/sfw/lib/libldap.la
```

per the instructions in the `make_nagios_plugins_on_solaris` script, to correct a poorly constructed distribution of this file. It is safe to run the build without checking this first, because if the change is not done, the build will stop with an error message pointing out that this correction must be made.

3.3.4 Solaris 10 Build Preparations

On Solaris 10, you will need the `SMCossl` or `SMCossl98b` package installed, which provides:

```
/usr/local/ssl/lib/libcrypto.so.0.9.8
/usr/local/ssl/lib/libssl.so.0.9.8
```

or the `SUNWopenssl-libraries` package installed, which provides:

```
/usr/sfw/lib/libcrypto.so.0.9.7
/usr/sfw/lib/libssl.so.0.9.7
```

You will also need to modify this file:

```
/opt/sfw/lib/libldap.la
```

per the instructions in the `make_nagios_plugins_on_solaris` script, to correct a poorly constructed distribution of this file. It is safe to run the build without checking this first, because if the change is not done, the build will stop with an error message pointing out that this correction must be made.

3.4 Building under Solaris

If you are creating an official build under the `factory` login account on one of the Solaris build machines, you should find the software checked out under:

```
~factory/svn/factory/products/gdma/
```

Otherwise, you can check it out from Subversion using the URL specified at the beginning of this chapter.

The Solaris version of the software is found under the `gdma/solaris/` directory. Type

```
make
```

to find the standard targets available here.

3.4.1 Building the Base GDMA Package

The command to build the base software package, including all the Nagios plugins, is:

```
make gdma_package
```

This makes the `GWOSgdma` package for the same platform you're building on.

3.4.2 Building a GDMA Key Package

There is apparently some internal difference in the package format between some Solaris versions, but we don't know for sure how many such transitions there are. So at least for the time being, it's best to make a key package for each customer/Solaris-release combination on each of the Solaris releases where it will be needed.

To build a key package which is customized for an individual customer and their particular GroundWork Monitor server, you must first obtain the GW Monitor server name and IP address from the customer. Then use:

```
make gdmakey_package
```

to make the `GWOSgdmak-` or `GWOSgdmakey-`*servername* package for that particular customer's server, along with the companion `GWOSgdmak-` or `GWOSgdmakey-`*servername*`-`*version*`-solaris`*release*`-`*architecture*`.id_dsa.pub` file. You will be prompted for the server name and IP address.

Appendix A: Revision History

The following versions of this document have been issued to date.

Table 2: Document Versions

Version	Date	Description of Changes
0.0.2	Dec 25, 2007	Initial draft.
0.2.0	Dec 26, 2007	Extended text; draft for PE review.
0.3.0	Dec 31, 2007	Updated to reflect changes in the Solaris packaging.
0.4.0	Jan 6, 2008	Updated to fill out the build instructions.