Magnia SG20 General Customization Guide

Software Developer's Kit

Technical White Paper

By TAIS Computer System Engineering

Published 9/2002

Table of contents

Magnia SG20 General Customization Guide

Introduction

The Toshiba Magnia SG20 is a multi-function server appliance, designed to provide ease of use and a friendly setup and management environment for the end user. It is based on Red Hat Linux 7.31. Because the Magnia SG20 is an appliance, it contains the Red Hat 7.3 RPMs needed to provide its feature set, as well as a variety of tools that allow direct access and modification of the system for more experienced users. It does not contain the entire suite of Red Hat 7.3 RPMs.

The Magnia SG20 provides a number of features targeted for small workgroups, remote offices and small business. These features are configurable through its administrative web interface, and include:

- Second HDD option (customer installable)
- Built in 7 port local LAN switch
- Built in wireless access point option
- Built in modem for dial-in networking and dial out to Internet
- Ethernet port for broadband Internet connections
- Parallel port for printer sharing
- LCD panel to display status (configurable)
- Soft power shutdown
- Easy to use client setup CD
- System reports
- File sharing (SMB)
- Local email
- Internet email retrieval service
- Internet gateway (sharing of single Internet port, dialup or broadband)
- Firewall
- Internet caching and content filtering
- Backup and restore to clients or Internet
- Primary HDD snapshot to second HDD (optional)
- HDD used for additional space (optional)
- User accounts and security
- Disk quotas
- Software upgrade process over the Internet
- Internal hardware health monitoring
- Built in Intranet (customizable)
- Easy to use VPN for remote access
- FTP, Telnet
- Digital Central web site with picture, music sharing and camera monitoring

- Home automation controller interface

When customizing a Magnia SG20 image to create a new preinstall image for your customers, it is important to assure the preset factory configuration is not modified. Doing so can result in a custom image shipped from the Toshiba factory to your end users with modifications which may result in increased service calls and a reduced initial customer satisfaction with his new system. This white paper outlines some basic areas of which you should be aware, and provides some techniques for customizing the system while maintaining Magnia SG20s customer out of box experience.

The Magnia SG20 uses a proprietary web based interface and a set of middleware tools to allow the end user to manage the system. The internal mechanisms of this middleware use some unique methods of managing system configurations, such as templatized configuration files, and triggers. Direct modification of configuration files is not recommended. This document contains information on the internal management techniques and how you might take advantage of them during your application integration process.

Customization Overview

Overview

Before beginning the process of customizing a Magnia SG20 image, you may wish to capture the clean, unmodified image on another hard disk. This will allow you to re-install a factory fresh image if you need to restart the integration process. See the section on custom image backup for more information on how to do this.

Generally, the process of installing a custom application involves three basic steps:

- Identifying and installing added Linux based software needed to support your integration process, such as libraries, newer versions of standard Linux software, or custom support software.
- Moving your application on to the Magnia SG20 and installing it as necessary.
- Optionally, adding links or web pages to the Magnia SG20 web based administration or preinstalled intranet pages.

Accessing the Magnia SG20

Access to the Magnia SG20 is accomplished through three mechanisms.

First, for most customization, installation and modifications, you should directly access the system using Telnet. This will provide direct access to the system using a shell command prompt. You can accomplish almost any task from this environment.

To access the Magnia SG20 using telnet, set up a client computer connected to a private Ethernet port. Verify you can access the web administration pages using the URL http://192.168.1.1. Once this has been verified, bring up a command prompt and run Telnet:

```
telnet 192.168.1.1
```

When presented by the Linux login prompt, use the account telnetuser with the default password toshiba2. Once at a shell prompt, you can switch to the root account using the su command.

Once at a root command prompt, you can proceed to install, recompile or modify software as required for your application integration.

Copying Software to the Magnia SG20

Once you have access to the Magnia SG20, you will need to copy your application to the system, as well as possible additional Linux support software such as databases or other tools. There are two main ways of copying new software to the Magnia SG20.

- FTP
- SMB file sharing access (Samba)

To copy files using FTP, invoke a command prompt from any client attached to the Magnia SG20 built in Ethernet switch. When prompted, log in to FTP as an anonymous user (the default Magnia SG20 configuration only allows anonymous access). Use normal FTP commands to copy files to the incoming directory. Once these files are transferred, use telnet to browse to the /home/ftp/incoming directory. You can then move and use these files as needed.

To copy files using file sharing, connect a client to the Magnia SG20 local network and browse the network neighborhood. The Magnia SG20 will appear in workgroup saworkgroup and will have the (default) computer name of myserver. Navigate to the share named public. Copy files to this directory, then use telnet to move and use these files within Linux as needed. The public share is mapped to the directory /home/public on the Magnia SG20.

Modifying the Magnia SG20 GUI

Once your application is installed and running on the Magnia SG20, you may wish to provide additional links or methods of accessing your application web interface. There are several ways of accomplishing this:

- Add descriptive text and a hyperlink to your own web pages.
- Add complete web pages to the existing web administration or preinstalled intranet
- Add your own intranet web site.

More detail on this type of modification is included in another document.

Listing Installed Linux Packages

To determine the list of Linux packages installed on your copy of the Magnia SG20 preinstall image, use the RPM command. This will allow you to determine what revision of each package has been installed on your working system. Use the following command to list the installed RPMs:

```
rpm qa
```

User Accounts / Passwords

Default accounts and Passwords

The Magnia SG20 is delivered to the end user with three predefined accounts. These accounts are:

- applianceadmin: This account is the primary administrative account for the Magnia SG20. It has access to the entire Web Administration UI, as well as complete access to all files on the system. It cannot be deleted using the Web Administration.
- telnetuser: This account is used to log in to the system using telnet. While not encouraged for the typical end user, it represents an important method of access to the system.
- root: This is the standard linux root administrative account with complete system privileges.

All these accounts are assigned a default password of toshiba when delivered from the factory. Changing this password is not recommended, as it can make it difficult or impossible for the end user to log in and administer the system.

Recommendation: Do not change the passwords of the system from the default.

Initial Account Creation

In order to assure that default password for critical privileged accounts such as applianceadmin are changed from the default, the Magnia SG20 uses a specific mechanism to update the password on server initialization. When the user creates the first user account, the password to all three predefined accounts are changed to the password of this first account.

An example scenario is for the user to open his new Magnia SG20, turn it on, and connect a PC to the built in network. He would then run the Setup CD to configure his client PC. During the Setup CD setup process, the Setup CD software creates a new level 3 account for him on the server (for example, jsmith). The user also selects a password at this time (for example maxmoney1). This process changes the predefined accounts passwords to be maxmoney1.

This process of assigning the initial user account password to the three predefined accounts takes place only on the first user account creation. Subsequent user account creation will not affect the predefined privileged account passwords.
This process will take place when creating an account using either the Magnia SG20 Setup CD, or using the Administrative Web interface.

Recommendation: Do not create a new user on your Magnia SG20 using the Setup CD or the Administrative Web interface. Use the existing applianceadmin account for access to the Web Administration or for file system access (network neighborhood). Use the existing telnet and root accounts for access to the system using telnet. Direct modifications to the system internals can be accomplished using these accounts.

Server Setup Wizard

When the Magnia SG20 is delivered from the factory, it is internally flagged as an unconfigured, or uninitialized system. When the Setup CD is run on a client connected to the Magnia SG20, it will check to see if the server is in its unconfigured, uninitialized state. If it is connected to an uninitialized system, the Setup CD will (upon completion of the client setup) invoke the Server Setup Wizard.

Server Setup Wizard

Once the user has completed the Server Setup Wizard, the system is flagged as no longer uninitialized, and subsequently the Server Setup Wizard will no longer be presented to the end user.

Recommendation: Do not run the Client Setup CD. This will cause the Server Setup Wizard to be invoked, and the system will no longer be flagged as uninitialized.

Network Configurations

The Magnia SG20 is delivered with the Internet (public) connection set to none. This allows the user to configure the system to their personal ISP requirements (Cable, DSL, modem).

When customizing the Magnia SG20, it may be desirable to configure the system to allow Internet or corporate LAN access. This can be done, but you should be aware that after changing networking configurations, they should be returned to their original state prior to shipping an image back to Toshiba. Failure to do so may result in unintended preset networking configurations being delivered to your end users.

If it is necessary to configure the public network for Internet or LAN access, the best method is to use the Corporate LAN or Cable Modem option. This option can be reset to none with little impact on the system.

Configuring public network access using the phone modem or DSL is less desirable. Some settings, such as phone number or user account / password, can be cached and may become part of your final customized image (this is an ease of use feature of the Magnia SG20 UI). This can result in the end user seeing your account (the password will be masked) if they configure modem access.

Recommendation: Avoid changing the networking configuration of the Magnia SG20 system while you are creating the custom image. If you require Internet or LAN access, use the Corporate LAN or Cable Modem option. Reset the configuration to None before shipping the customized image.

There are numerous other networking configuration options that may be changed while implementing a custom image. These can include:

- Internet Caching and Filtering
- Firewall settings
- High security mode
- Dial in access
- VPN settings
- Wireless settings

These settings can be safely changed, and then reset back to their factory defaults if required. You can even change these settings to your own preferred setting for shipment if desired.

Custom Image Backup

When implementing the custom image, it is easy to make a mistake and need to restore the system to a factory original state, or to recover from an earlier version of your custom image. The easiest and best way to accomplish this is to take the original Magnia SG20 SDK preinstall image, and create a copy of it prior to doing any work using the disk snapshot feature.

Second Disk Snapshot

The disk snapshot feature will take a complete snapshot of the entire disk and place it on the secondary disk in the second slot of the Magnia SG20. Work can then proceed; if you need to go back to the original image for any reason, simply switch the primary and secondary disks and boot from the snapshot backup.

With an additional hard disk, you can also take periodic snapshots to preserve your image in case you need to return to an earlier version during your customization process.

Recommendation: Take a snapshot of the original factory image before beginning work on a second disk. If needed, take periodic snapshots of your work on a third disk.

Software Upgrades Support

The standard retail version of the Magnia SG20 is supported with a Software Upgrades site that allows end users to view, download and apply new features and fixes. This site services only customers using the standard Toshiba preinstall image.

Because Toshiba does not control the changes or configuration of your preinstall image, each system integrator will have their own Software Upgrades site. Toshiba may occasionally inform you of a software upgrade available on the standard site which you might wish to make available to end users with your custom image. It is then up to you to decide whether Toshiba should deploy the upgrade for your customers.

In order to provide an Upgrade Site specific to your preinstall images, Toshiba will edit a file on your image to point to this custom site before release to the factory. If you wish to retest your image after this change or if you do not want Toshiba to make this change, please inform your Toshiba sales person.

General System Architecture

The Magnia SG20 is partitioned in the following manner:

Partition	Description	Size
/	Root (system) partition, non-volatile	1 GB
/var	Scratch file partition, email, log files and other temporary data	1.2 GB
/home	User data	Expands to fill remaining amount of disk
Swap Partition	Swap area for memory pages	120 MB

The /home partition is where all user data is placed. It contains some infrastructure for the support of the Magnia SG20 system, but very little. User private data and the public share, as well as user posted Intranet information are stored here. The /home directory contains the following subdirectories:

Directory	Description
/home/backups	This directory is used as a temporary location for data backup processing.
/home/docs	Documents posted for viewing using the Magnia SG20 preinstalled intranet are placed here.
/home/ftp	This directory contains the FTP applications sub directories, such as incoming and public.
/home/intranet	This directory is a pre-mapped location for a customized intranet. You can reach it using the URL http://myserver/intranet. You can easily create your own intranet site by placing the web pages here.
/home/public	This directory is the open, public directory in which users can share files.
/home/users	This directory contains the individual private subdirectories for each user with an account on the system.

The Magnia SG20 system user interface, middleware and configuration are placed almost entirely in the /sa2 directory. The user interface and management general architecture is displayed below:

- The Perl based CGI uses the HTML templates, along with information from system configuration files, to create the final web based output for the user interface. In addition, it takes configuration changes from the user, applies them

to the templatized configuration files, and invokes triggers to regenerate the system configuration files and restart any services as necessary.

- The HTML page templates contain the basic structure and graphics for the web based user interface. These files are combined with actual data to create the final web pages seen by the user.
- Triggers are Perl based scripts that are executed when certain types of events occur. Triggers are called by the user interface CGI, as well as being invoked by some system callback routines.
- Templatized configuration files contain the basic information about many of the systems actual configuration files. These templates are read by triggers, and used to regenerate the system configuration files when certain events occur.

The file system reflects these elements in its directory organization:

Directory	Description
/sa2/templates	This directory contains the configuration file templates that are used to regenerate the system configuration files.
/sa2/lang	The HTML and other files used to generate the actual Magnia SG20 user interface are contained in this directory. Because the Magnia SG20 can support multiple languages, this area of the system is separated in to various language directory trees. The /sa2/lang/en tree is used to support English.
/sa2/triggers	This directory contains the trigger scripts invoked to perform system reconfiguration and other maintenance.
/sa2/firewall	This directory contains definition files for special firewall rules. By adding new directories and files here, you can add your own optional firewall rules.
/sa2/conf	All the Magnia SG20 specific configuration files are contained here. These configuration files are specific to the Magnia SG20 implementation. These configuration files do not replace the standard Linux configuration files. The primary configuration file of interest is main.conf, which is the central place where most configuration items from the user interface are placed.
/sa2/lib	This directory contains Perl and other modules used as general routines to support the CGI code and other Perl programs.
/sa2/web	The Perl, Java script, graphics and other web-based CGI code driving the Magnia SG20 are contained here. Web sites represented here are: • System Administration (/sa2/web/admin) • Preinstall Intranet (/sa2/web/intranet)
/sa2/firewall	This directory contains subdirectories for each custom

	firewall rule. Each subdirectory corresponds to firewall rule the user can enable or disable through the advanced firewall settings option in the web administration UI.
/sa2/log	This directory contains logs specific to features implemented in the Magnia SG20 user interface. The logs in this directory do not replace the standard Linux application logs, typically contained in /var.
/sa2/bin	This directory contains general utility scripts and programs used by the Magnia SG20 to interface with the Linux system and the system hardware.
/sa2/upgrades	The upgrades directory contains information about the current state of the systems software upgrades. As this area is downloaded from, and communicates with the Toshiba Software Upgrades site, it is a good idea not to modify it. If your preinstall image is sent to the Toshiba factory, Toshiba will change a file in this directory to point to a custom software upgrade web site for your company.

Customizing the Firewall

The Magnia SG20 is a Linux IP Tables based firewall. The default configuration for the firewall is established in the firewall template in /sa2/templates/etc/rc.d/init.d/iptables. In addition, there are custom firewall rules defined which the user can select or deselect as desired. These rules are accessible through the web based remote administration screens, in the advanced firewall page. These rules can be checked on or off (the default is off). These custom rules open holes in the firewall for items such as FTP access, Cisco VPN pass through, and Internet gaming. Adding custom firewall rules in this area is the preferred method for customization of the Magnia SG20 firewall.

Modifying the firewall rules is not recommended unless you are experienced with IP Tables and firewall definitions. Inappropriate rules added to the firewall can result in your system becoming completely inaccessible, requiring the disk be returned to the factory for re-imaging. Use care when establishing new firewall rules.

Recommendation: take a disk snapshot before changing or adding custom firewall rules, so that you can restore the system if needed.

Adding a custom firewall rule that can be selected and deselected using the web administration is simple. By simply creating a new directory under the /sa2/firewall directory, and placing several simple files in this directory, a new user selectable firewall rule is added to the user interface.

To add a new custom firewall rule, connect to the SG20 as root using telnet, and perform the following steps.

1. Create a new directory under /sa2/firewall, using a name relating to the new rule you are adding. For example, a new directory may be added:

   /sa2/firewall/vpnport500

2. In this new directory, create a new file named rule which contains the IP Tables command for the rule you wish to add. You can use other rule files in /sa2/firewall as examples. For example, you might add the following contents to a rule file

```
[% IF firewall.enabled -%]

# allow Sonic VPN clients through

# Note that port 500 is the source port, not destination port.

$IPTABLES -A FORWARD -p udp -s ANY/0 --sport 500 -i $INTIF -o
$EXTIF -j

ACCEPT

[% END -%]
```

3. Next, create a new file description in the directory, containing a short (less than one line) description of the rule to be added. This description will be displayed on the user interface next to the check box. An example description might be:

   en=Custom VPN Client

   The en= indicates this text description is used for any English based display. For other languages, you can specify sp, de, it, fr, and du for Spanish, German, Italian, French and Spanish. If a language is invoked on the user interface, and there is no corresponding language text in the description file, English will be used as the default.

4. Next, create a new file called index which specifies the location of the firewall within the firewall table. Location can be important, so be sure to review the other custom rules as well as the default rules to understand placement of the rule. The index file should simply contain a number, such as:

   ```
   70
   ```

5. Finally, create a new file called type in the directory, containing the word client, server or both which is used to describe who the firewall rule applies to. An example type file might be:

   client

Once these steps have been performed, a new selectable firewall rule will be added to the user interface.

A very easy way to get started with a new firewall rule is to use on of the existing firewall rule directories. Simply copy it and its contents to a new directory name, and then begin modification of the individual files to change the rules display name and IP Tables rule.

Templatized Configuration Files

Many of the configuration files in the Magnia SG20 are managed using templates. With this methodology, a set of templates for each configuration file is maintained. Then, when a change is made to the configuration file by the Magnia SG20 user interface or other mechanism, the configuration file is regenerated using these templates.

The Perl based middleware and CGI code interfaces may need to modify the system configuration files(as when a user is added or a configuration changed). When a change has been made, a trigger is invoked, which will regenerate the corresponding system configuration file using the configuration file templates.

Because these templatized configuration files are regenerated periodically, changes to the files must be made in the templates, not in the file itself. Changing the configuration file directly will not work.

Templates are stored in /sa2/templates. Each files templates are located under this directory based on the files location in the system. For example, the /etc/passwd template files are located in /sa2/templates/etc/passwd. The contents of this directory are shown below.

05system	15ftp	15postgress	footer
10static	15mysql	20dynamic	header

As you can see, the configuration file may be broken up in to several templates. Each template represents one part of the configuration file, and the order in which they are used is determined by the alphabetical sorting order of the template file name. This is why a two-digit number, which determines its placement within the final configuration file when it is regenerated, precedes each template file name.

To place added elements in the configuration file, such as a new user in the password file, you can add a new template file in the configuration files template directory. Add the text desired for placement in the configuration file in a new template file. The file name should contain two leading digits that will determine your additions placement within the regenerated configuration file.

Note that adding a template file will not necessarily cause the configuration file to be regenerated. Regeneration of a configuration file from its templates is only performed when the system determines there has been a change requested by the user in the user interface (in some cases system events may cause a file to be regenerated, but this is rare). For example, the password files are not regenerated unless the user adds, deletes or modifies a user account using the web administrative user interface.

Configuration changes can be announced to the system using triggers. Triggers are the Magnia SG20s method of assuring whatever actions are appropriate for a configuration change are taken. Execution of a trigger will cause the appropriate configuration files to be regenerated. Please see the separate section of this document describing triggers for more detail.

The following configuration files are templatized, and should not be modified directly. This list is current as of the time of writing. Please check your system for the actual list.

Directory	File
/etc	aliases
	crontab
	dhcpd.conf
	ftpaccess
	group
	htgroup
	htpasswd
	inittab
	named.conf
	passwd
	pptpd.conf
	printcap
	shadow
	sysctl.conf
	userdb
	wvdial
/etc/atalk	atalkd.conf
	netatalk.conf
/etc/fetchmail	fetchmail
/etc/httpd/conf	httpd.admin
	httpd.intranet
/etc/isdn	ibod.cf
/etc/mail	access
	genericstable
	local-host-names
	sendmail.mc
	virtusertable
/etc/mgetty+sendfax	login.config
	mgetty.config
/etc/ppp	chap-secrets
	ioptions
	ioptions.dialin
	pap-secrets
	pppoe.conf

/etc/ppp/peers	dialin
	pptp
/etc/rc.d/init.d	dhcpd
	iptables
	named
	network
	pcmcia
/etc/samba	smb.conf
	smbpasswd
/etc/squid	squid.conf
/etc/sysconfig	clock
	ipchains
	iptables.sh
	network
/etc/sysconfig/network-scripts	ifcfg-ethn
	ifdown-ippp
	ifdown-ppp
	ifup-ippp
	ifup-post
	ifup-ppp
/etc/sysconfig/tsb/tsb_ap_proc	channel
	distance_between_aps
	dtim_period
	enable_encryption
	enable_macfilter
	key1
	key2
	key3
	key4
	keys
	mac_accept
	multicast_rate
	network_name
	reject_any
	station_name
	transmit_key_id
/etc/xinet.d	ipop3
	telnet
	wu-ftpd
/sa2/conf	lcdkbd.conf
/sa2/web/admin	htaccess
/sa2/sbin	ifdown-local
	ifup

	ifup-local
/var/named	domain
	domain.rev
/var/spool/cron	root

Template Perl Programming Interface

The template processor has both a Perl object-oriented interface and a command line wrapper. Each of these take two parameters: filename and mode.

The usage statement of the proctmpl command-line wrapper is:

```
usage: /sa2/bin/template -f <filename> -m <octal mode> -o <owner>
```

Output from the command-line wrapper is generated to STDOUT so a redirect is required to overwrite the current configuration file.

Where filename is the name of the actual file you want to generate. Call it like this:

```
[/sa2/bin]#  ./template -f /etc/sysconfig/ipchains -m 700 o root >
/etc/sysconfig/ipchains
```

The perl object interface works like this:

```
use SA::Template;

my $tmpl = new SA::Template;

$tmpl->process('/etc/sysconfig/ipchains', '0700') or

    die "Couldn't process firewall";

$tmpl->process('/etc/sysconfig/network', '0700') or

    die "Couldn't process network globals";
```

This will take the template files contained in the relative path under /sa2/templates/, concatenate them, and process the concatenated value using the Template Toolkit and the values contained in the /sa2/conf/main.conf.

System Startup Scripts

There are a couple of different methods of having services or scripts run at boot time. The Magnia SG20 supports the standard RC boot scripts common on several Linux and Unix distributions. Depending on when your service or script needs to be run during the boot sequence, you can use two methods of execution:

1. Create a standard RC script and create the proper links under /etc/rc.d/rc3.d to enable your script to be executed at a certain point in the boot.
2. Add a Perl script to the /sa2/triggers/started directory to be executed during the started trigger which is executed near the end of ever boot. This method is not recommended for starting daemon tasks.

If you are starting daemon or application processes at boot time, you need to make sure that they are shutdown properly when an orderly shutdown is started on the SG20. You should create an RC script for starting and stopping your daemon processes. There are several examples in the /etc/rc.d/init.d directory on the SG20 of scripts that stop and start daemon tasks.

A simple and straightforward example of an application start and stop script is the /etc/rc.d/init.d/snmpd startup script. This script starts and stops the SNMPD system daemon on the SG20. As in this example, if your startup script creates an empty file in /var/lock/subsys/ then your startup script will also get called with a stop argument when the system is shutting down. It is recommended that you implement code in your start/stop script to handle the stop parameter and shut down your application tasks in an orderly manner. This will help avoid possible killing of the application tasks and loss of data when the system is shutdown.

You need to make sure that all of your running tasks and daemons are stopped when your startup script is called with a stop command. Your start / stop script should send signals to your application and daemon tasks, arranging for an orderly shutdown and cleanup of temporary files. Tasks that are not shutdown in an orderly manner at this time will be terminated by the operating system during the last stage of system shutdown.

Triggers

The internal middleware supporting the web administration and internal operations of the Magnia SG20 are based on a trigger mechanism. Triggers are scripts (or sets of scripts) that are executed when certain events occur on the system. Though triggers may perform other tasks, the two primary tasks performed are typically the regeneration of templatized configuration files, and the stopping and starting of various related system services. Magnia SG20 trigger scripts are written in Perl.

Events that cause triggers to be executed usually involve any detected change in the state or configuration of a specific system area or service. For example, the networking triggers may be executed when there is a change in the IP address of the public port (such as when a DHCP address is acquired). Triggers are also executed when a change is made through the web administration user interface. For example, adding or deleting a user could execute the mod_users trigger to regenerate various password and user configuration files.

Triggers can be useful when integrating your application in the Magnia SG20 system.

- If you change configuration file templates, you can regenerate all the effected configurations and have the change become effective by launching the associated trigger.
- You can add your own triggers scripts to existing triggers.

The core of the trigger mechanism is a series of directories, one per trigger (named after the trigger), that are organized under /sa2/triggers. These directories may be nested as deeply as desired for organization purposes, but the trigger name will be the portion of the directory path relative to /sa2/triggers. Each directory contains scripts that execute actions that are tied to the trigger. The actions will most likely include, but are not limited to, regenerating the template files for each dependent application, then restarting the dependent service. The scripts in each trigger directory should be indexed by zero-padded numeric string in a manner similar to the templates. The scripts will be executed in the order that ls l [0-9]* produces, so the filename of each script must start with a number.

Trigger Interface

The most common use of triggers when creating custom applications is to assure that templatized configuration changes are applied to the system after modification. In this case, executing the proper trigger from the command line is all that is needed. The usage statement of the trigger command is:

```
usage: trigger <trigger name> <args>
```

Use it like this:

```
[/sa2/bin]# ./trigger network/mod_config
```

If you are writing custom Perl code and need to execute a trigger, you can do so using the trigger object. The Perl object oriented interface works like this:

```
use SA::Trigger;

my $trigger = new SA::Trigger;

$trigger->(network/mod_config) or

die Couldnt trigger mod_config\n;
```

The code above will sort the scripts in the /sa2/triggers/network/mod_config directory, execute them one-by-one, wait for exit status, and return an exit status to the caller. It is important to note that trigger scripts should always return 0 unless there is a serious failure. If a trigger script returns anything other than 0, the trigger script execution chain is stopped and the return code is passed back to the caller.

The trigger parameter correlates to the pre-defined trigger directory under the /sa2/triggers directory. args are passed directly on the argv stack to the action scripts beneath that directory which might need to do their jobs in a more granular way, i.e. you might design a trigger called network/mod_config and pass it the name of the interface on argv so that it only has to do the work correlating to the change of that specific interface. Or you might design a trigger called add_user and pass it the username so that the relevant configuration and password files would not have to be regenerated in their entirety; only the information relating to that user would need to be added.

However, every action script should assume it could be fired by a trigger without any data and should be designed in such a way that in this instance, it regerates all of its configuration files and restarts all services that use those files. The developer must be very careful not to fire recursive events! Also, the developer must know (most likely through consulting the centralized configuration file through SA::Config) whether or not his service is enabled and whether or not he needs to fire given the change that was made, i.e. if the vpn only needs to reload if the external interface was changed and doesnt need to fire if the internal interface is changed, the developer is responsible for acting accordingly to keep the execution time of an trigger to a minimum.

Manually executing a trigger

A trigger can be executed directly by executing the trigger command from a shell command line. This can be useful when you have modified a configuration file template, and want to execute the trigger that will process the template, regenerate the configuration file, and restart any associated services. Invoke a trigger using the syntax trigger <trigger name>. For example,

```
trigger mod_samba
```

Or

```
trigger network/mod_dialin
```

Remember that some triggers will restart networking services, which may disconnect your telnet session.

Adding trigger scripts

You can add your own trigger, which can be useful if you wish to perform an application specific action as a result of a triggering condition. For example, if your application needs to know when the public port IP address has changed, you could add a script in the network/mod_ip trigger directory. Your script will be executed when the system public IP address is changed, along with any other triggers in this directory.

Triggers Implemented

The following list of triggers is accurate as of the time of writing, and can help provide understanding of how this mechanism is used. Verify the triggers on your system for any recent changes.

Trigger	Associated configuration files	Triggering Events
disktwo/autodetect		System boot
disktwo/crontab		UI change
disktwo/detect		System boot
disktwo/mirror		UI execution or scheduled mirror
disktwo/setconfig		UI change
disktwo/status		UI refresh
firstboot	All templatized files	Installation firstboot only
fullrestore		UI execution of full restore
mod_apache	/etc/httpd/conf/httpd.admin /etc/httpd/conf/httpd.intranet	UI change
mod_country		UI change
mod_crontab	/var/spool/cron/root	UI change
mod_email	/etc/mail/sendmail.mc /etc/fetchmail/fetchmail /etc/mail/access /etc/aliases /etc/xinet.d/ipop3 /etc/userdb	UI change

	/etc/mail/genericstable /etc/mail/virtualusertable	
mod_firewall	/etc/sysconfig/iptables.sh /etc/samba/smb.conf	UI change, Network change
mod_health_contactinfo		UI change
mod_lcd	/sa2/conf/lcdkbd.conf	UI change
mod_lcd/locale		UI change
mod_lcd/restart		UI change
mod_samba	/etc/samba/smb.conf	Samba Workgroup change, firewall change, public IP address change if firewall is down
mod_security	/sa2/web/admin/htaccess	UI change
mod_syslang		UI change
mod_time	/etc/sysconfig/clock /var/spool/cron/root	UI change
mod_users	/etc/passwd /etc/shadow /etc/group /etc/htpasswd /etc/htgroup /etc/samba/smbpasswd /etc/ppp/pap-secrets /etc/ppp/chap-secrets	Add/Remove system user
mod_wireless	/etc/sysconfig/tsb/tsb_app_proc/channel /etc/sysconfig/tsb/tsb_app_proc/distance_between_aps /etc/sysconfig/tsb/tsb_app_proc/dtim_period /etc/sysconfig/tsb/tsb_app_proc/enable_macfilter /etc/sysconfig/tsb/tsb_app_proc/keys /etc/sysconfig/tsb/tsb_app_proc/mac_accept /etc/sysconfig/tsb/tsb_app_proc/multicast_rate /etc/sysconfig/tsb/tsb_app_proc/network_name /etc/sysconfig/tsb/tsb_app_proc/reject_any /etc/sysconfig/tsb/tsb_app_proc/station_name	UI change to wireless card settings, fullrestore trigger
modem_event		Modem status change
mondisk		Harddrive status change
monhw/hw_err		Hardware error detected
monhw/hw_limit		Hardware limit exceeded
monhw/hw_ok		Hardware status returns to OK state

monhw/hw_remote		Remote Health monitoring report status
monhw/remote_notification		Remote Health monitoring error notification
network/bring_down		Stopping of any network interface
network/enable_isdn		UI change
network/mod_dialin	/etc/ppp/ioptions.dialin /etc/ppp/pap-secrets /etc/inittab /etc/mgetty+sendfax/login.config /etc/mgetty+sendfax/mgetty.config /etc/ppp/peers/dialin	UI change
network/mod_ipsec	/etc/ipsec.conf /etc/ipsec.secrets	UI change
network/mod_proxy	/etc/squid/squid.conf	UI change
network/mod_staticip	/etc/named.conf /var/named/domain var/named/domain.rev	UI change of private IP address
network/bringup		Starting of any network interface
network/mod_config	/etc/sysctl.conf /etc/sysconfig/network /etc/sysconfig/network-scripts/ifcfg-ethn /sbin/ifup-local /sbin/ifdown-local /etc/ppp/chap-secrets /etc/ppp/pap-secrets /etc/ppp/pppoe.conf /etc/wvdial.conf /etc/ppp/ioptions /etc/sysconfig/network-scripts/ifup-ippp /etc/named.conf /var/named/domain /var/named/domain.rev /etc/rc.d/init.d/dhcpd /etc/dhcpd.conf /etc/ppp/peers/dialin /etc/atalk/config /etc/atalk/atalkd.conf /etc/ppp/ioptions.dialin /etc/inittab /etc/mgetty+sendfax/login.config /etc/mgetty+sendfax/mgetty.config	Any network setting modification
network/mod_ip	/etc/ipsec.conf /etc/ipsec.secrets /etc/named.conf /var/named/domain /var/named/domain.ver	IP address change for public port

network/mod_pptp	/etc/inittab /etc/pptp.conf /etc/ppp/peers/pptp /etc/ppp/chap-secrets	UI change to pptp VPN settings
network/mod_snmp	/etc/snmp/snmp.conf /etc/snmp/snmpd.conf	UI change to SNMP settings
remote_health		System Health Report
shutdown		Shutdown of system
started		Boot time startup scripts
started/lcdmsg		Boot time LCD initialization
starting		Boot time startup scripts
status_backup		Backup completion status
status_restore		Restore completion status
upgrades/autocheck		Auto check cron job for server upgrades
upgrades/config		UI change of upgrade autocheck settings
upgrades/current_rpms		UI execution of Upgrade check
upgrades/finish		Completion of Upgrade installation
upgrades/init		System installation run only once.
upgrades/Newcat		New upgrade catalog received
upgrades/Start		Start of a system upgrade
upgrades/Status		Status query of system upgrade progress
users/admin_reset		Reset of admin user password

Useful Files

There are several useful files in the Magnia SG20. While not all of these files are ones that will help your application integration process, they contain useful information.

File	Contents
/sa2/conf/BUILDINFO	This file contains the current Magnia SG20 release number. This number is combined with the build version number and displayed on the LCD panel.
/sa2/BUILDVERSION	This file contains the build number of the current release.
/sa2/conf/main.conf	While this file contains too many fields to describe here, it is a good place to go if you want to retrieve some information about the current system configuration. Its contents are reasonably self-explanatory.
/sa2/conf/system.conf	This file contains several items of primary interest. The system serial number is extracted and placed in this file each time the system boots. Thus, the serial number in this file should be a reliable way of uniquely identifying the specific unit. This can be useful when implementing licensing.
/sa2/RPMVERSIONS	This file contains a list of all software packages installed on the system.

Recommendation: Place your applications initials (and possibly version number) in the BUILDVERSION file. This will help users and Toshiba support to identify systems running your special version of the Magnia SG20 software. Remember that this text must fit in the 16 character LCD window (along with the other build information).

LCD Panel Customization

The Toshiba Magnia SG20 contains a built-in, easy to read and use LCD panel. This display is normally used to present information to the user concerning the system status and operations. Messages include items such as:

- Boot and shutdown progress
- Backup / restore progress
- Networking operations
- Health alerts

The LCD panel and the items it displays are controlled by the internal operating system of the SG20. Because the internal software controls the LCD panel, it is easy to add and modify what it displays. No modifications to the system firmware are required.

This section covers a general description of the LCD panel operation, its interface, the software controlling message display, and how to display information on the panel directly without using the Magnia SG20 provided software. There are two methods to interact with the Magnia SG20 LCD panel. When using the Magnia SG20 preinstalled software, interaction with the LCD panel is direct, through the LCD daemon. If using your own software preinstall image that does not include the Magnia SG20 software for LCD management, you can create your own programs and procedures to interact with the LCD panel and the system buttons.

LCD Interface Technical Overview

The Magnia SG20 contains two main boards. The PC motherboard contains most of the controllers and interfaces normally associated with a PC, including the CPU, IDE controllers, serial and parallel port controllers, etc. A second board, the mezzanine board, contains some additional components such as the internal built in Ethernet switch and the system LCD firmware.

Communications to the LCD display are transmitted from the operating system software running on the main system board through the COM1 serial port. The LCD controller firmware listens to the serial port and displays most characters sent through this serial port.

The LCD panel display is a two-line display. Each line can display up to 16 characters.

In addition to display and control of the LCD panel, the LCD firmware monitors and reports the button presses of the LCD select button and the soft power button, both located on the front of the Magnia SG20.

LCD Panel Direct Access

It is possible to interact directly with the Magnia SG20 LCD panel without using the supplied LCD control software. This can be useful when a different operating system has been loaded on the SG20 (such as a generic Linux system, or DOS).

Please note that the techniques described in this section are for use on operating systems other than the one normally supplied with the Magnia SG20 by Toshiba. The standard Magnia SG20 software contains a Linux daemon that controls and monitors the LCD and system buttons. Therefore, attempts to control the LCD panel or receive button press information on the standard Magnia SG20 operating system installation will conflict with the LCD control daemon.

Direct LCD message display

Because characters sent to the LCD COM1 port are automatically displayed on the LCD screen, it is very easy to display your own messages. For example, on DOS, the command:

```
echo Hello World > COM1
```

Will cause the characters Hello World to appear on the LCD panel. The equivalen Linux command is:

```
echo hello world > /dev/ttyS0
```

More complicated strings, controlling wrapping on the two LCD lines and spacing are available. The following table shows the characters that are accepted for printing and formatting as part of the message text on the LCD panel.

Received Character	Name	Operation	
A-Z, a-z, 0-9	ASCII printable code.	Display character at the current cursor position and advance cursor to the next location. Scroll line as needed.	
SP =	Space char	Advance the cursor to the next cursor position and scroll as needed.	
NL := \n	Newline	Advance the cursor to the next line, scrolling as needed.	
CR := \r	Carriage return	Return the cursor to the first column on the current row.	
ESC	Escape	Prepare for the next received char as a command code.	

LCD Control Codes

In addition to direct transmission of characters to the LCD, there are several special command sequences that can be sent to the LCD controller. These sequences are designed to ease formatting and manipulation of the LCD display. All command sequences begin with the escape character, and are following by a character indicating a specific action.

The table below shows the available escape command sequences and their use.

Received Character	Name	Operation
H		Home the cursor.
X		Clear the display.
B		Turn on backlight
b		Turn off backlight
<sec>q		Power down the system in <sec> seconds. Should only be sent at end of shutdown sequence.

Examples of the use of these commands appear below:

To send the LCD cursor to the top row, far left character position:

```
echo n e \033H > /dev/ttyS0
```

To clear the LCD display:

```
echo n e \033X > /dev/ttyS0
```

To turn off the backlight of the LCD display:

```
echo n e \033b > /dev/ttyS0
```

To shut the power of the system off in 10 seconds:

```
echo n e \03310q > /dev/ttyS0
```

Please note that issuance of the power off command illustrated above should not be issued unless the operating system is being shut down in an orderly manner. Sufficient time should be given for the OS to complete its shutdown procedures.

Button Status Codes

The LCD controller also monitors and reports whenever one of the control buttons on the Magnia SG20 is pressed. These buttons include the LCD scroll button located directly

adjacent to the LCD panel and the power down button located on the front of the Magnia SG20 and to the right of the LCD panel.

Button status information is sent in on the internal COM1 port.

Status values sent across the COM1 port are listed below. No preceding escape or other characters are sent from the LCD controller to the motherboard.

Transmitted Character	Name	Operation
P	Power	Power button pressed.
S	Select	Select button pressed.

Because these characters are sent across the COM1 port without a CR/LF, it may be necessary to write specific code to monitor the incoming characters. This code should place the device in a raw mode in order to receive characters as soon as they are sent, so that they are not buffered. While this can be accomplished using command line level instructions, it is significantly better to write a program to monitor and process these commands.

While command / shell scripts can be used to send information to the LCD for display with ease, monitoring and responding to button commands is best handled in a program specifically written for the purpose.

Magnia SG20 Queued LCD Message Interface

If you wish to add custom LCD messages when using a Toshiba Magnia SG20 preinstall, you need to integrate your messaging in to the internal messaging management system. The internal queuing system manages the display of LCD messages, their positioning, translation and cycling through the multiple messages available for display using the LCD scroll button.

There is an LCD/KBD Daemon (LCDKBD) used to coordinate client application access to the LCD/KB micro-controller sub-system. This is a Linux daemon that runs at system startup and is used by client applications to queue messages to display on the LCD panel. Client applications do not directly communicate to the device.

Direct Command Line Interface

The command line interface to the LCD/KB port is used for direct display of messages on the LCD panel. The following describes the command line interface for script access.

setlcd [<cmd> [<data>]] <text>

Where -<text> is an ASCII text string with NL (\n) & CR (\r) support.

Where -<cmd> is described by the following:

Cmd	Description	Comment
-clear	Clear the display	Clear display and homes the cursor.
-home	Home the cursor.	Move to the row=1, col=1 position.
-home2	Move cursor to second row.	Move to the row=2, col=1 position.
-col <col>	Move the cursor to the column position	Move the col to <col>.
-row <row>	Move the row to the row position	Move the row to < row >.

Note that because this command bypasses the message queuing system, it is reserved for use by the LCDKBD and for debugging.

Queued LCD Messages

Under normal circumstances, all LCD messages are displayed and managed by the lcdkbd daemon. This daemon takes messages that should be displayed on the LCD panel and places them in a queue. Because there are multiple messages available for display on

the LCD panel, the LCD scroll button on the front of the machine is used to communicate with the lcdkbd daemon.

When the lcdkbd daemon detects a button press, it will cycle through the various messages in the message queue. Some messages can be forced to the top of the queue, placing the message on the LCD without requiring the user press the LCD button. This technique is used for urgent messages such as alerts. Messages can also be displayed for one time only. This type of message appears on the LCD until the user presses the LCD scroll button, and is then removed from the message queue so that it does not appear again.

Use the Perl object ShowMsg to tell the lcdkbd daemon to display specific messages on the LCD panel, as described below.

Adding LCD Messages

Because the contents of the LCD display are managed by the lcdkbd daemon, display of new messages involves creating a new message definition in an LCD message definition file. Once defined properly, the LCD message can be displayed on the LCD panel by sending the lcdkbd daemon an instruction. Use the following procedure to add your own LCD message to the display message queue.

1. Go to the directory /sa2/lang/en/lcdmsg. This directory contains all LCD message definition files for the English language (which is the default). When customizing messages to display in different languages, go to the corresponding language directory, such as /sa2/lang/es/lcdmsg for Spanish, or /sa2/lang/de/lcdmsg for German.

2. Create a new definition file for your LCD message. An example of a new message definition file called newmsg appears below:

   newmsg.desc=Acme Application Messages

   newmsg.name=ACME

   newmsg.readonly=0

   newmsg.hidden=0

   newmsg.msgup=-center Acme App center Running

   newmsg.msgdown=-center Acme App center Not Running

   newmsg.msgalert=-ontop -center Acme App center %s Error

   newmsg.msgmail=-ontop -onetime center Acme App center Check Mail

See the section below for more information on these fields and their operation.

3. Add a Perl script to your application that invokes the messages defined in your LCD message definition file. This script should incorporate the appropriate ShowMsg call to display a message. An example program is shown below:

```
use SA;

use SA::LCD::Disp;

my $objDisp = SA::LCD::Disp->new();

$objDisp->ShowMsg(newmsg.msgmail);

exit(0);
```

   This script would place the message Acme App, and Check Mail on the first and second lines of the LCD panel.

   Here is an example of how to display an LCD message that incorporates dynamic text, as in the newmsg.msgalert LCD message above:

```
$objDisp->ShowMsg(newmsg.msgalert, Download);
```

4. To complete the configuration process for the new message, execute the mod_lcd trigger. This trigger will regenerate the LCD configuration files and inform the lcdkbd daemon it should refresh its message list. Note that execution of this trigger is only necessary when you have made a change to the LCD message list.

   trigger mod_lcd

LCD Message Display Options

The LCD message definition files contained in the /sa2/lang/en/lcdmsg directory contain a variety of options for the messages displayed. The message definition file example given above outlines these options. The fields and options contained in this file are described here by using this example. Fields are defined by the name of the message with a specific configuration parameter suffix. In the example above, the message name is newmsg, so the description field in this file is newmsg.desc.

```
newmsg.desc=Acme Application Messages
```

The desc portion is the configuration field parameter indicating the text description of the LCD message. This text description will appear on the LCD configuration page in the web administration user interface that allows the user to select which LCD messages they wish to appear.

LCD Configuration Screen

The name suffix provides and internal name used by the Magnia SG20 middleware to help identify the message being displayed or configured.

```
newmsg.name=ACME
```

In this case, the name given to the message is ACME.

```
newmsg.readonly=0
```

The readonly field specifies whether the user will be allowed to change the LCD setting. In the default LCD configuration screen, the IP address display, and time of day are always checked and cannot be unchecked by the user. To allow the user to deselect this LCD message from displaying on the LCD panel, set the readonly field to 0. To prevent the user from deselecting this message and force the message to always display, set the readonly field to 0.

```
newmsg.hidden=0
```

The hidden field specifies whether the LCD message appears on the configuration screen. Some messages, such as health monitoring, do not even appear on the user LCD configuration screen, and are always displayed. To prevent a message from appearing on the configuration screen, set the hidden field to 1. To present the message on the user configuration screen, set the hidden field to 0.

```
newmsg.msgup=-center Acme App center Running

newmsg.msgdown=-center Acme App center Not Running

newmsg.msgalert=-ontop -center Acme App center %s Error

newmsg.msgmail=-ontop -onetime center Acme App center Check Mail
```

A single LCD message can have several states, or sub-messages. For example, the backup message may present the backup status as Not Performed, Started or Completed. Only one of these messages can be displayed on the LCD panel at a time. If the Started message is displayed, and then later the Completed message is displayed, the Completed message will replace the Started message.

In this example, the Acme App message will display Acme App on the first LCD line. The second line will be Running, Not Running, Error or Check Mail.

The following formatting options are available:

- -center: center the text in the middle of the LCD line
- -ontop: immediately display this message by placing it on the top of the message queue
- -onetime: clear this message from the LCD message queue after it has been viewed by the user once