Carrier Grade Linux Requirements Definition

The Linux Foundation

1796 18th Street
Suite C
San Francisco
CA 94107, USA
+1 (415) 723-9709

Version 5.0

Prepared by the Carrier Grade Linux Working Group

CONTRIBUTORS TO THE CGL 5.0 REQUIREMENTS DEFINITION INCLUDE (IN ALPHABETICAL ORDER):

Last Name	First Name	Company
Anderson	Matt	HP
Anderson	Tim	MontaVista Software
Awad	Majid	Intel
Aziz	Khalid	HP
Badovinatz	Peter	IBM
Bozarth	Brad	Cisco
Cauchy	Dan	MontaVista Software
Chacron	Eric	Alcatel
Chen	Terence	Intel
Cherry	John	OSDL
Christopher	Johnson	Sun Microsystems
Cihula	Jospeh	Intel
Cress	Andrew	Intel
Dague	Sean	IBM
Dake	Steven	MontaVista Software
Flaxa	Ralf	Novell
Fleischer	Julie	Intel
Fleischer	Julie	OSDL
Fox	Kevin	Sun Microsystems
Gross	Mark	Intel
Haddad	Ibrahim	Ericsson
Heber	Troy	HP
Howell	David P.	Intel
Hu	Michael	Radisys
Ikebe	Takashi	NTT
Ishitsuka	Seiichi	NEC
Jagana	Venkata	IBM
Johnson	Christopher P.	Sun Microsystems
Kevin	Fox	Sun Microsystems
Kimura	Masato	NTT Comware
Krauska	Joel	Cisco
Kukkonen	Mika	Nokia
La Monte.H.P	Yarrol	Timesys
Lavonius	Ville	Nokia
Liu	Bing Wei	Intel
Lynch	Rusty	Intel
* MacDonald	Joe	Wind River Systems
Manas	Saksena	Timesys
Nakayama	Mitsuo	NEC
Peter-Gonzalez	Inaky	Intel
Pourzandi	Makan	Ericsson
Rossi	Frederic	Eicsson

Saksena	Manas	Timesys
Sakuma	Junichi	OSDL
Saskena	Manas	Timesys
Seiler	Glenn	Wind River Systems
Smarduch	Mario	Motorola
Takamiya	Noriaki	NTT Software
Weijers	Gé	
Witham	Timothy D.	OSDL
Wright	Chris	OSDL
Yarroll	La Monte H.P.	Tomesys
Zou	Yixiong	Intel

* Specification editor

Comments on the contents of this document should be sent to lf_carrier@linuxfoundation.org.

TABLE OF CONTENTS

1 OVERVIEW OF CARRIER GRADE LINUX

In the time since the fourth major version of the Carrier Grade Linux Specification has been published there has been a great shift in both the telecommunication industry and the open source community. Most consumers of mobile communications devices see them as conduits for communication, be that voice, text, locations services, and general internet browsing. Providers need to ensure that voice and data traffic shares the network seamlessly with the same correctness and performance regardless of the packet. This pushes the need for carrier-grade reliability to nearly every application server and it must be available to the very edges of the network. This makes “old” ideas about scalability, handling hundreds of thousands of calls with predictable performance, seem almost quaint when carriers are now expecting to be able to handle that as well as stream video, audio and packet traffic all with varying, but immutable, service requirements. At the same time this level of reliability is seen as being needed beyond the “carriers” because almost every server is connected to an ever-on world-wide network with users awake every hour of the day. This has helped many of the features published in earlier versions of the CGL specification to become accepted parts of the Linux mainstream.

While the usage models and goals described above evolve, this is accompanied by a simultaneous shift away from proprietary platform architectures to commercial off-the-shelf (COTS) platforms and open software environments. This continues to pick up pace but now there is also increasing demand for integration with acceleration technologies and performance tuning options rarely seen in the past. Open software and COTS hardware were once seen as a means for rapid deployment of voice and data services; now they are considered essential in many areas and without continued advancement and adoption in both areas the competitive nature of the market risks fracturing the community that has formed.

Carrier Grade Linux (CGL) still stands at the centre of all of this. More than seven years ago a group of industry representatives from platform vendors, Linux distribution suppliers and network equipment providers set out to define how “Carrier Grade Linux” could enable environments with higher availability, serviceability, and scalability requirements and formed the Carrier Grade Linux Working Group. The working group has produced four major versions of a specification to define the required capabilities. The result is that Linux distribution suppliers have been able to demonstrate that they meet the needs of

telecommunications by disclosing how their products address the requirements in this document.

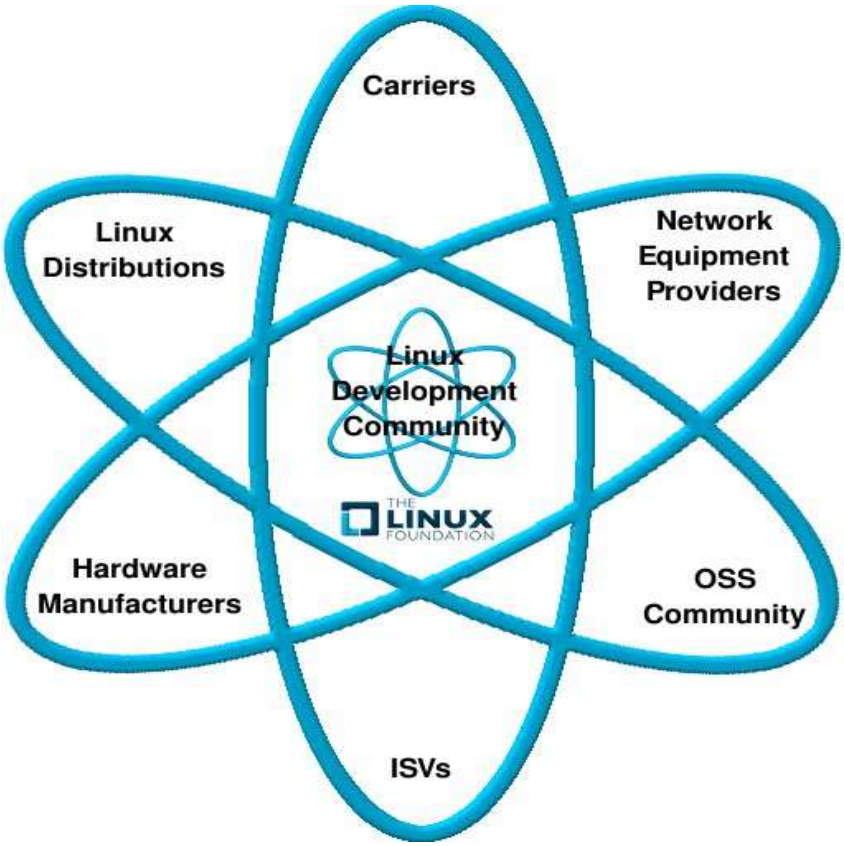

Illustration 1: The Linux Ecosystem

Today the CGL working group represents interests from Linux distribution suppliers as well as telecommunications industry equipment manufacturers, service providers and end users. The CGL working group continues to strive to bring these various groups together and to foster open communication and collaboration, always with the goal of championing these requirements to the community and bringing carrier-grade improvements to everyone.

High availability middleware components and service availability middleware that run on CGL systems are addressed by organizations such as the Distributed Management Task Force (DMTF), the Object Management Group (OMG), and the Service Availability Forum (SAF). High availability hardware platforms underlying CGL are addressed by organizations such as the PCI Industrial

Computer Manufacturers Group (PICMG) and the Intelligent Platform Management Interface (IPMI). In addition, organizations like the SCOPE Alliance address several layers applicable to carrier grade environments. The SCOPE Alliance defines profiles for hardware, OS, and middleware; its purpose is to help, enable, and promote the availability of open carrier grade platforms based on commercial-off-the-shelf (COTS) hardware and software.

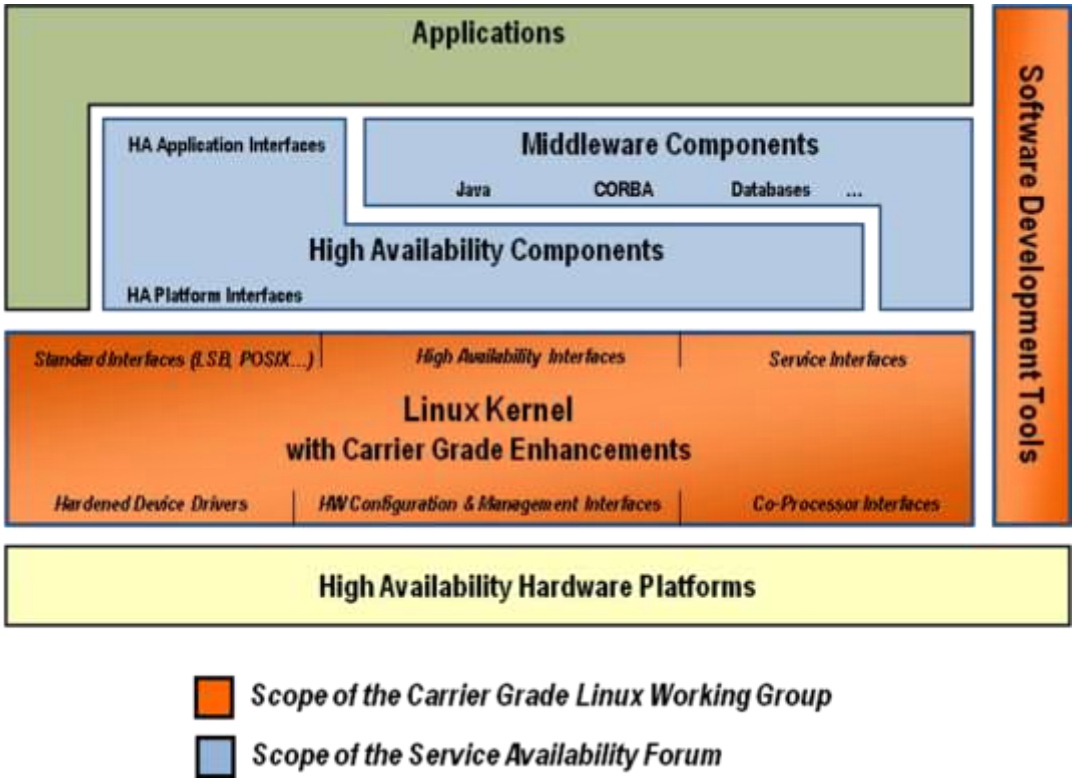

Illustration 2: Full Carrier Grade Application Stack

The CGL 5.0 specification is an upwardly compatible superset of the CGL 4.0 specification. As with the 4.0 specification, many requirements have been deprecated, since at the time of publication they have been deemed to be ubiquitous and therefore no longer relevant for the purposes of meeting carrier requirements. These deprecated requirements represent the broad adoption described earlier of carrier-grade objectives by the community and can be viewed as validation of the objectives of the group as a whole.

In 2003 and 2004, member companies were producing communications products based on the CGL 1.1 specifications. In the latter half of 2004 and 2005, Linux distributors began to announce Linux offerings based on the CGL 2.0.2 specification. In 2006 several vendors registered for CGL 3.2. In 2007 CGL 4.0 introduced a new registration process and within weeks of the process being available the first of the 4.0 distributions appeared. The CGL 5.0 registration process will be very similar to the process used for the 4.0 specification and therefore a very smooth transition is expected for carriers and equipment providers as Linux distribution suppliers incorporate CGL 5.0 capabilities in 2011 and beyond.

As always, development is underway on many of the CGL capabilities that do not appear in mainline distributions. While the CGL requirements are specified for Linux-based platforms in the communications industry, a high availability, high performance, scalable system is viewed as beneficial to the entire Linux user community. These developments are both in areas identified here as requirements and as gaps and while this version of the specification is expected to be the definitive version for some time to come, the CGL working group anticipates that many of the gap items today will become not only requirements in the future but features so basic as to be expected of all Linux distributions. Discussions of these developments are encouraged and can be directed to the Carrier Grade community at lf_carrier@linuxfoundation.org.

2 DOCUMENT ORGANIZATION

For clarity and ease of use, the specification has been split by topic into the following sections:

1. **Requirements Overview**

   Describes the requirements and gap formatting, terminology used throughout the remainder of the document and the registration implications of requirements and gaps.

2. **Availability Requirements Definition**

   Describes useful and necessary functionality for single node availability and recovery.

3. **Clustering Requirements Definition**

   Describes useful and necessary components to build a clustered set of individual systems. The key target is clustering for high availability, although load balancing and performance are secondary aims. It is

recognized that “one size fits all” is not achievable, so not all features will always be used together.

4. **Serviceability Requirements Definition**

   Describes useful and necessary features for servicing and maintaining a system and coverage of tools that support serviceability.

5. **Performance Requirements Definition**

   Describes useful and necessary features that contribute to adequate performance of a system, such as real-time requirements. Also describes base operating system components for supporting performance tools (requirements for the tools themselves are not addressed).

6. **Standards Requirements Definition**

   Provides references to useful and necessary APIs, specifications, and standards, such as POSIX, IETF, and SA Forum standards.

7. **Hardware Requirements Definition**

   Describes useful and necessary hardware-specific support that relates to a carrier operating environment. This section is much reduced in size and scope since the CGL 4.0 specification in recognition that support for hardware is largely coming from hardware vendors and therefore is not normally a requirement on the distribution supplier any longer.

8. **Security Requirements Definition**

   Describes useful and necessary features for building secure systems. It is recognized that “one size fits all” is not achievable, so not all features will always be used together.

3 REQUIREMENTS OVERVIEW

Throughout the remainder of this document the terms **requirement** and **gap** will be used extensively. The definitions of these terms as adopted by the Carrier Grade Linux working group are as follows.

A **requirement** is an aspect, feature or application that is viewed as essential to achieving and/or implementing one of the above carrier grade objectives (that is, availability, clustering, serviceability, performance, hardware support, security or standards implementation) that has at least one **active**, **open source** implementation available. Depending on the priority of the requirement the open source implementation may or may not be available on multiple architectures.

An application or implementation is considered **open source** so long as the code has been provided under an OSI-approved license.

An application or implementation is considered **active** so long as it has not obviously been abandoned by the developers and / or the community at large. Signs of abandonment may be an official announcement by the developer with no other developers adopting the project; it may be a lack of updates to support new functionality or in response to new developments in the community or simply to support new versions of the underlying software (for example a lack of updates to support newer kernel versions). There is no strict definition of a reasonable amount of time to expect updates in a project since mature projects move at a pace quite different from emerging ones; however as a general guideline the CGL working group has adopted a window of two (2) years as a good indication of whether a project is still active.

A **gap** is an aspect, feature or application that is viewed as very important to achieving and/or implementing one of the above carrier grade objectives that does not currently have an **open source** (see above) implementation available.

The motivation behind the above definitions for requirements and gaps is to ensure that there is no barrier to entry to the carrier grade distribution space and to encourage developers to contribute their code back to the community under a free and open source license. The Carrier Grade Linux working group believes that this is the best way to both recognize carrier requirements and encourage healthy collaboration and competition in the community.

The following table shows an example of a **requirement**:

ID	Name	Category	Priority
STD.1.0	Linux Standard Base Compliance http://www.linuxbase.org	Standards	P1

CGL specifies that carrier grade Linux shall be compliant with at least the Linux Standard Base (LSB) 3.0 The LSB 3.0 specifications has been split into a generic LSB core, a generic module for C++, and a set of architecture specific modules. Required LSB 3.0 modules for CGL are:

- Generic LSB-Core
- Generic LSB-CXX
- For each supported architecture, one LSB-Core module and one LSB-CXX module

The developer may choose to implement more than one architecture platform. In this case, each supported architecture platform shall contain an implementation of at least one architecture specific LSB-Core module and one architecture specific LSB-CXX module.

NOTE: LSB 3.0 Certification program requires all 3 parts (core, C++, and graphics) to be certificated. The graphics part will be a stretch for CGL to require as it is not essential for carrier grade server type of applications. CGL WG to work with FSG/LSB to initiate the subprofile certification program to allow CGL distribution to be certified.

Each **requirement** contains the following fields:

ID A unique identification number including:

- An acronym identifying a category for the requirement (first field)
- An ID number for the requirement (second field)
- An ID number for a sub-requirement (third field). A “0”in this field indicates the requirement is a stand-

alone requirement. A number in this field indicates this requirement is a sequentially numbered sub-requirement

Name Short description of the requirement

Category The category to which the requirement is assigned. This example contains Standards (STD) requirements.

Priority P1 – Required: Must be implemented and the implementation must be disclosed as part of the CGL 5.0 registration process.

P2 – Disclosure: Does not have to be implemented but the CGL 5.0 registration must include a statement whether the requirement has been implemented and, if it has been implemented, how the requirement is met in the distribution.

Description Detailed description of the requirement.

A **gap** is follows a similar formatting:

ID	**PID**	**Name**
GAP.1.0	AVL.3.2	Forced Un-mount
CGL specifies that carrier grade Linux shall provide support for forced unmounting of a file system. The un-mount shall work even if there are open files in the file system. Pending requests shall be ended with the return of an error value when the file system is unmounted.		

Each gap contains the following fields:

ID A unique identification number including:

- The **GAP** identifier (first field)
- A unique ID number for the gap (second field)

- An ID number for a sub-requirement (third field). A “0”in this field indicates the requirement is a stand-alone requirement. A number in this field indicates this requirement is a sequentially numbered sub-requirement

PID	Is the gap had previously been assigned an ID by an earlier version of the CGL specification, it will be identified here.
Name	Short description of the gap.
Description	Detailed description of the gap.

4 AVAILABILITY REQUIREMENTS DEFINITION

Telecommunication customers expect their voice and data services to always be available. System availability is dependent on the availability of individual components in the system. To help ensure 24/7 service, it must be possible to perform system maintenance and system expansion on running telecommunication networks and servers without disrupting the services they implement. Systems must be able to withstand component failures, making redundancy of components such as power supplies, fans, network adapters, storage, and storage paths essential. Software failures can also significantly impact the availability of a compute node, so robust application software, middleware, and operating system software is required for single node availability.

This section is a collection of requirements that address the robustness of a single computing node. Availability is further enhanced by clustering individual computing nodes so that a node cannot represent a single point of failure. The single node requirements in the Availability section can be categorized as:

- On-line operations
- Redundancy
- Monitoring
- Robustness

ON-LINE OPERATIONS

On-line operations enable the system to continue to provide a service while the software or the hardware is replaced or upgraded on the system. For instance, when a file system needs repair, repair procedures may require rebooting the system. However, CGL requires that it be possible to forcibly un-mount a file system, allowing repair and remounting without rebooting. The ability to replace or upgrade hardware such as disks, processors, memory, or even entire processor/memory blades without bringing down that node or the network contributes significantly to continuous service availability.

REDUNDANCY

A highly available system must be composed of redundant components and must be able to take advantage of redundant hardware such that the system continues to function when a component fails. Ideally, designs can eliminate all single points of failure from a system. Using redundant communication paths, such as redundant network ports and host adapters, together with network fail-over software capabilities, such as Ethernet bonding, improve network availability. Redundant storage paths, such as redundant fiber channel ports and host adapters used with multipath I/O, improve storage availability. Redundancy of memory components may not be possible, but error detection and correction can be used to mask memory cell failures; CGL requires software Error Correction Code (ECC) support. Single bit errors are reported when they are detected in the hardware and logged by the kernel. The kernel invokes a panic routine whenever uncorrectable multi-bit errors are detected.

MONITORING

Rapid detection of hardware or software failures requires health monitoring. Health monitoring is also needed to check for hardware or software that is beginning to fail, such as ECC memory checking, predictive analysis for disks, and processes that do not respond in a predicted way. Examples of CGL monitoring requirements include Non-Intrusive Monitoring of Processes and Memory Over-commit Actions. The Non-Intrusive Monitoring of Processes requirement detects abnormal behavior by a process, such as process death, and initiates an action, such as the creation of a new process. The Memory Over-commit Actions requirement monitors system memory usage and controls process activity when memory usage exceeds specified thresholds.

ROBUSTNESS

A highly available system must be composed of redundant components and must be able to take advantage of redundant hardware such that the system continues to function when a component fails. Ideally, designs can eliminate all single points of failure from a system. Using redundant communication paths, such as redundant network ports and host adapters, together with network fail-over software capabilities, such as Ethernet bonding, improve network availability. Redundant storage paths, such as redundant fiber channel ports and host adapters used with multipath I/O, improve storage availability. Redundancy of memory components may not be possible, but error detection and correction can be used to mask memory cell failures; CGL requires software Error Correction Code (ECC) support. Single bit errors are reported when they are detected in the hardware and logged by the kernel. The kernel invokes a panic routine whenever uncorrectable multi-bit errors are detected.

AVAILABILITY REQUIREMENTS

AVL.2.0 SINGLE-BIT ECC HANDLING

ID	Name	Category	Priority
AVL.2.0	Single-bit ECC handling	Availability	P2
CGL specifies that carrier grade Linux shall provide a mechanism for reporting when hardware error checking and correcting (ECC) detects and/or recovers from a single-bit ECC error.			

AVL.2.1 MULTI-BIT ECC HANDLING

ID	Name	Category	Priority
AVL.2.1	Multi-bit ECC handling	Availability	P2
CGL specifies that carrier grade Linux shall provide a panic trigger mechanism when hardware error checking and correcting (ECC) detects multi-bit ECC errors.			

AVL.4.1 VM STRICT OVER-COMMIT

ID	Name	Category	Priority
AVL.4.1	VM Strict Over-Commit	Availability	P1

CGL specifies that carrier grade Linux shall provide the ability to control kernel virtual memory allocation adjustments based on the specific needs of the system. Control of virtual memory shall include but not be limited to the following:

- Heuristic over-commit handling. Obvious over-commits of address space are refused. Used for a typical system. It ensures a seriously wild allocation fails while allowing over-commit to reduce swap usage. root is allowed to allocate slightly more memory in this mode. This is the default.

- Always over-commit. Appropriate for some scientific applications.

- Don't over-commit. The total address space commit for the system is not permitted to exceed swap + a configurable percentage (default is 50) of physical RAM. Depending on the percentage you use, in most situations this means a process will not be killed while accessing pages but will receive errors on memory allocation as appropriate.

AVL.5.3 PROCESS-LEVEL NON-INTRUSIVE APPLICATION MONITOR

ID	Name	Category	Priority
AVL.5.3	Process-Level Non-Intrusive Application Monitor	Availability	P1

CGL specifies that carrier grade Linux shall provide control and management capabilities for processes that cannot be altered to incorporate a monitoring API. Such capabilities are known as non-intrusive monitoring. These capabilities must be implemented programmatically using commands or scripts.

Another issue for many such processes is that the start script itself may spawn an application process that is not under the control of the management process. This sub-requirement assumes that this does not happen, and the child process remains under the control of the management entity.

Capabilities required:

- The following capabilities must be enabled for controlling processes:

  The ability to start a process (or a list of processes)

  The ability to stop a process (or a list of processes)

- The following capabilities must be enabled for monitoring processes:
  - The ability to detect the unexpected exit of a process
  - The ability to configure a set of actions in response to an unexpected exit of a process
- The following services must be provided beyond those currently provided by inittab:
  - The ability to configure whether to restart the application if the process dies
  - A configurable amount of time to wait before restarting the application
  - A limit on the number of times to restart the application

AVL.6.0 DISK PREDICTIVE ANALYSIS

ID	Name	Category	Priority
AVL.6.0	Disk Predictive Analysis	Availability	P1
CGL specifies that carrier grade Linux shall provide capabilities to assist in monitoring storage systems. The aim of this support is to assist in predicting situations likely to lead to failure of disks. This allows preventive action to be taken to avoid the failure and resulting disruption of service.			

AVL.7.1.1 MULTI-PATH ACCESS TO STORAGE: MULTI-PATH DETECTION

ID	Name	Category	Priority
AVL.7.1.1	Multi-Path Access to Storage: Multi-Path Detection	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. The software shall determine if multiple paths exist to the same port of the I/O device.			

AVL.7.1.2 MULTI-PATH ACCESS TO STORAGE: I/O BALANCING

ID	Name	Category	Priority
AVL.7.1.2	Multi-Path Access to Storage: I/O Balancing	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. The software shall determine if multiple paths exist to the same port of the I/O device, and, with configurable controls, balance I/O requests across multiple host bus adapters. If multiple paths exist to the same device over two separate device ports on the same host bus adapter, those I/Os will not be balanced.			

AVL.7.1.3 MULTI-PATH ACCESS TO STORAGE: AUTOMATIC PATH FAILOVER

ID	Name	Category	Priority
AVL.7.1.3	Multi-Path Access to Storage: Automatic Path Failover	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. Handling a path failure must be automatic.			

AVL.7.1.4 MULTI-PATH ACCESS TO STORAGE: FAILED PATH REACTIVATION

ID	Name	Category	Priority
AVL.7.1.4	Multi-Path Access to Storage: Failed Path Reactivation	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. A mechanism must be provided for the reactivation of failed paths, allowing them to be placed back in service.			

AVL.7.1.5 MULTI-PATH ACCESS TO STORAGE: AUTOMATIC PATH CONFIGURATION

ID	Name	Category	Priority
AVL.7.1.5	Multi-Path Access to Storage: Automatic Path Configuration	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. It must be possible to automatically determine and configure multiple paths.			

AVL.7.1.6 MULTI-PATH ACCESS TO STORAGE: AUTOMATIC VOLUME CONFIGURATION

ID	Name	Category	Priority
AVL.7.1.6	Multi-Path Access to Storage: Automatic Volume Configuration	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. Automatic configuration shall allow automatic multi-path configuration of complete disks and partitions located on those disks.			

AVL.7.1.7 MULTI-PATH ACCESS TO STORAGE: ROOT FILE SYSTEM HOSTING

ID	Name	Category	Priority
AVL.7.1.7	Multi-Path Access to Storage: Root File System Hosting	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. A multipath device feature that allows multipath detection and mapping early in the boot process must be provided so that the root file system can exist on a multipath device.			

AVL.7.1.8 MULTI-PATH ACCESS TO STORAGE: LINK FAILURE REPORTING

ID	Name	Category	Priority
AVL.7.1.8	Multi-Path Access to Storage: Link Failure Reporting	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable multiple access paths from a node to storage devices. The mechanism should implement error logging functions that clearly identify the failing device path.			

AVL.8.1 FAST LINUX RESTART BYPASSING SYSTEM FIRMWARE

ID	Name	Category	Priority
AVL.8.1	Fast Linux Restart Bypassing System Firmware	Availability	P1
CGL specifies that carrier grade Linux shall provide a mechanism to speed up operating system initialization by bypassing the system firmware when one instance of Linux reboots to another instance of Linux.			

AVL.9.0 BOOT IMAGE FALLBACK MECHANISM

ID	Name	Category	Priority
AVL.9.0	Boot Image Fallback Mechanism	Availability	P2
CGL specifies that carrier grade Linux shall provide a mechanism that enables a system to fallback to a previous "known good" boot image in the event of a catastrophic boot failure (i.e. failure to boot, panic on boot, failure to initialize HW/SW). System images are captured from the "known good" system and the system reboots to the latest good image. This mechanism would allow an automatic fallback mechanism to protect against problems resulting from system changes, such as program updates, installations, kernel changes, and configuration changes."			

AVL.10.0 APPLICATION LIVE PATCHING

ID	Name	Category	Priority
AVL.10.0	Application Live Patching	Availability	P2
CGL specifies that carrier grade Linux shall provide a mechanism and framework by which a custom application can be built so that it can be upgraded by replacing symbols in its live process. Dynamic replacement of symbols allows a process to access upgraded functions or values without requiring a process restart and in many circumstances can lead to improved process availability and uptime. The mechanism should be applied only to user applications. Patch to underlying distribution software component may lose distribution support.			

AVL.12.0 NFS CLIENT PROTECTION ACROSS SERVER FAILURES

ID	Name	Category	Priority
AVL.12.0	NFS Client Protection Across Server Failures	Availability	P2
CGL specifies that carrier grade Linux shall provide mechanisms that allow an NFS server to have failover capability to provide service continuity upon a node failure. The NFS service has to be resumed on another node without any impact on NFS clients other than the retransmission of pending requests (open files must remain open). Clients authenticated on the old server must remain authenticated on the new server.			

AVL.13.1 PARALLEL USER INITIALIZATION DURING STARTUP

ID	Name	Category	Priority
AVL.13.1	Parallel User Initialization During Startup	Availability	P2
CGL specifies that the user initialization procedure executed by the program /sbin/init shall provide a mechanism to allow multiple init scripts to run in parallel. CGL further specifies that a service is only started once its dependent services have started.			

AVL.15.0 FAST APPLICATION RESTART MECHANISM

ID	Name	Category	Priority
AVL.15.0	Fast Application Restart Mechanism	Availability	P2

CGL specifies that carrier grade Linux shall provide a mechanism that enables a quick application restart. Typical applications in a carrier environment use multiple processes with inter-process communications. As applications become more complex, application initialization times become longer.

To speed up application initialization, the mechanism shall provide the functionality to simultaneously save memory images of multiple processes (including the kernel resources used by each process) and to restore the images.

When the application completes initialization, including making connections between processes and setting up kernel resources for inter-process communication, the application invokes a save function that makes a copy of the memory images of the process and kernel resources. If the application hangs, the mechanism restores the memory images and kernel resources and restarts the application.

AVL.17.0 MULTIPLE FIB SUPPORT

ID	Name	Category	Priority
AVL.17.0	Multiple FIB Support	Availability	P2

CGL specifies that Linux shall support multiple Forwarding Information Base (FIB) quick look-up tables with forwarding addresses to allow better server virtualization of overlapping addresses. An FIB is a table that contains a copy of the forwarding information in the IP routing table. All hooks/changes required to support multiple FIBs shall be added.

AVL.21.0 ETHERNET LINK BONDING USING IPV4

ID	Name	Category	Priority
AVL.21.0	Ethernet link bonding using IPV4	Availability	P1

CGL specifies that carrier grade Linux shall support bonding of multiple Ethernet NICs within a single node using IPV4. The bonding supports the following functions:

- Ethernet link aggregation: Supports multiple Ethernet cards to be bonded for bandwidth aggregation.
- Ethernet link failover: Supports automatic failover of an IP address from one Ethernet NIC to another within a single node using the Ethernet bonding. Some mode of bonding requires IEEE 802.3ad support on switches; however, other modes do not require special protocol support.

AVL.21.1 ETHERNET LINK BONDING USING IPV6

ID	Name	Category	Priority
AVL.21.1	Ethernet link bonding using IPV6	Availability	P1

CGL specifies that carrier grade Linux shall support bonding of multiple Ethernet NICs within a single node using IPV6. The bonding supports the following functions:

- Ethernet link aggregation: Supports multiple Ethernet cards to be bonded for bandwidth aggregation.
- Ethernet link failover: Supports automatic failover of an IP address from one Ethernet NIC to another within a single node using the Ethernet bonding. Some modes of bonding require IEEE 802.3ad support on switches; however, other modes do not require special protocol support.

AVL.22.0 SOFTWARE RAID 1 SUPPORT

ID	Name	Category	Priority
AVL.22.0	Software RAID 1 support	Availability	P1
CGL specifies that carrier grade Linux shall provide RAID 1(Mirroring) support so that the OS maintains duplicate sets of all data on separate disk drives. RAID 1 support shall allow booting off of selected mirror disk drive even if the other drive is failed. RAID 1 implementation shall provide a user-controllable parameter to throttle the syncing operation. Support can be configured out if desired.			

AVL.23.0 WATCHDOG TIMER PRE-TIMEOUT INTERRUPT

ID	Name	Category	Priority
AVL.23.0	Watchdog Timer Pre-Timeout Interrupt	Availability	P1
CGL specifies that carrier grade Linux shall provide support for a watchdog timer pre-timeout interrupt. Where the hardware supports such a capability an interrupt handler routine will be called before the real timeout occurs.			

AVL.24.0 WATCHDOG TIMER INTERFACE REQUIREMENTS

ID	Name	Category	Priority
AVL.24.0	Watchdog Timer Interface Requirements	Availability	P1
CGL specifies that carrier grade Linux shall provide the ability to use an interface to reset the hardware watchdog timer, where the hardware supports such a capability. This timeout value shall be a configurable item. A configurable action can be performed when a timeout occurs.			

AVL.25.0 APPLICATION HEARTBEAT MONITOR

ID	Name	Category	Priority
AVL.25.0	Application Heartbeat Monitor	Availability	P1
CGL specifies that carrier grade Linux shall provide an application heartbeat service that allows applications to register to be monitored via specified APIs. The mechanism shall use periodic synchronized events (heartbeats) between an application and the monitor. If a registered application fails to provide a heartbeat, the monitor shall report the events. The application heartbeat service shall be available to any process or sub-process (thread) entity on the system. A process or thread may register for multiple heartbeats.			

AVL.26.0 RESILIENT FILE SYSTEM SUPPORT

ID	Name	Category	Priority
AVL.26.0	Resilient File System Support	Availability	P1
CGL specifies that carrier grade Linux shall provide support for the installation of a file system that is resilient against system failures in terms of recovering rapidly upon reboot without requiring a full, traditional fsck. This is normally achieved using logging or journaling techniques.			

AVL.27.0 KERNEL LIVE PATCHING

ID	Name	Category	Priority
AVL.27.0	Kernel Live Patching	Availability	P2
CGL specifies that carrier grade Linux shall provide a mechanism for symbols, functions, or variables within a running kernel to be replaced with new symbols, functions, or variables. CGL further specifies this operation be completed without a system shutdown or restart			

AVL.28.1 FILE SYSTEM DE-FRAGMENTATION

ID	Name	Category	Priority
AVL.28.1	File System De-fragmentation	Availability	P1
CGL specifies that carrier grade Linux shall provide support for a file system that allows for de-fragmentation of on-disk data. It is expected that the file system will not be mounted or otherwise in use at the time.			

AVL.28.2 MULTI-ARCHITECTURE FILE SYSTEM SUPPORT

ID	Name	Category	Priority
AVL.28.2	Multi-Architecture File System Support	Availability	P1
CGL specifies that carrier grade Linux shall provide support for a file system where the metadata and data are stored independent of host CPU word length and endianness.			

AVL.28.3 FILE SYSTEM METADATA INTEGRITY CHECKSUM

ID	Name	Category	Priority
AVL.28.3	File System Metadata Integrity Checksum	Availability	P1
CGL specifies that carrier grade Linux shall provide support for a file system that guarantees file system metadata and data consistency and fast recovery in the event of interrupted updates with checksums on all metadata.			

AVL.28.4 FILE SYSTEM BLOCK CHECKSUMMING

ID	Name	Category	Priority
AVL.28.4	File System Block Checksumming	Availability	P2
CGL specifies that carrier grade Linux shall provide support for a file system that provides end-to-end checksums of all blocks currently in use on the file system.			

AVL.28.5 FILE SYSTEM MULTIPLE ACCESS PROTECTION

ID	Name	Category	Priority
AVL.28.5	File System Multiple Access Protection	Availability	P2
CGL specifies that carrier grade Linux shall provide support for shared, simultaneous read and write access to file system data that is assured protection against accidental corruption of the data and/or metadata.			

AVL.28.6 FILE SYSTEM SNAPSHOTS

ID	Name	Category	Priority
AVL.28.6	File System Snapshots	Availability	P2
CGL specifies that carrier grade Linux shall provide support for a file system that allows the creation of atomic snapshots of volumes while mounted. These snapshots must be valid filesystem images that can be mounted as if they were the original volume at the time of the snapshot.			

AVL.28.7 FILE SYSTEM CLONES

ID	Name	Category	Priority
AVL.28.7	File System Clones	Availability	P2
CGL specifies that carrier grade Linux shall provide support for a file system that allows atomic backups while the volume is mounted and in use. These backups should be writable where subsequent updates to the file system will not be reflected in the original and therefore each can be considered a fork of a single, live file system image.			

AVAILABILITY REFERENCES

POSIX:

- Open Group References:

http://www.opengroup.org/

http://www.unix.org/online.html

http://www.opengroup.org/onlinepubs/007908799/

- POSIX conformance data on Linux:

  http://posixtest.sf.net

- POSIX Technical Corrigendum 1 text:
  http://www.opengroup.org/pubs/catalog/u057.htm

- POSIX Specification with current Technical Corrigendum:
  http://www.unix.org/version3/

Linux Standard Base (LSB)	http://www.linuxbase.org/
Free Standards Group	http://www.freestandards.org/
Service Availability Forum (SAF)	http://www.saforum.org/
Internet Engineering Task Force (IETF)	http://www.ietf.org/rfc.html

5 CLUSTERING REQUIREMENTS DEFINITION

The CGL working group conducted a clusters usage model study from which they learned that no single clustering model meets the needs of all carrier applications. So CGL takes a more general approach to defining clustering requirements. CGL defines the functional components of a carrier grade High Availability Cluster (HAC). The requirements for other cluster models, such as a scalability cluster, a server consolidation cluster, and a High Performance Computing (HPC) cluster, have been treated as secondary to requirements for the HAC cluster model. See Illustration 3.

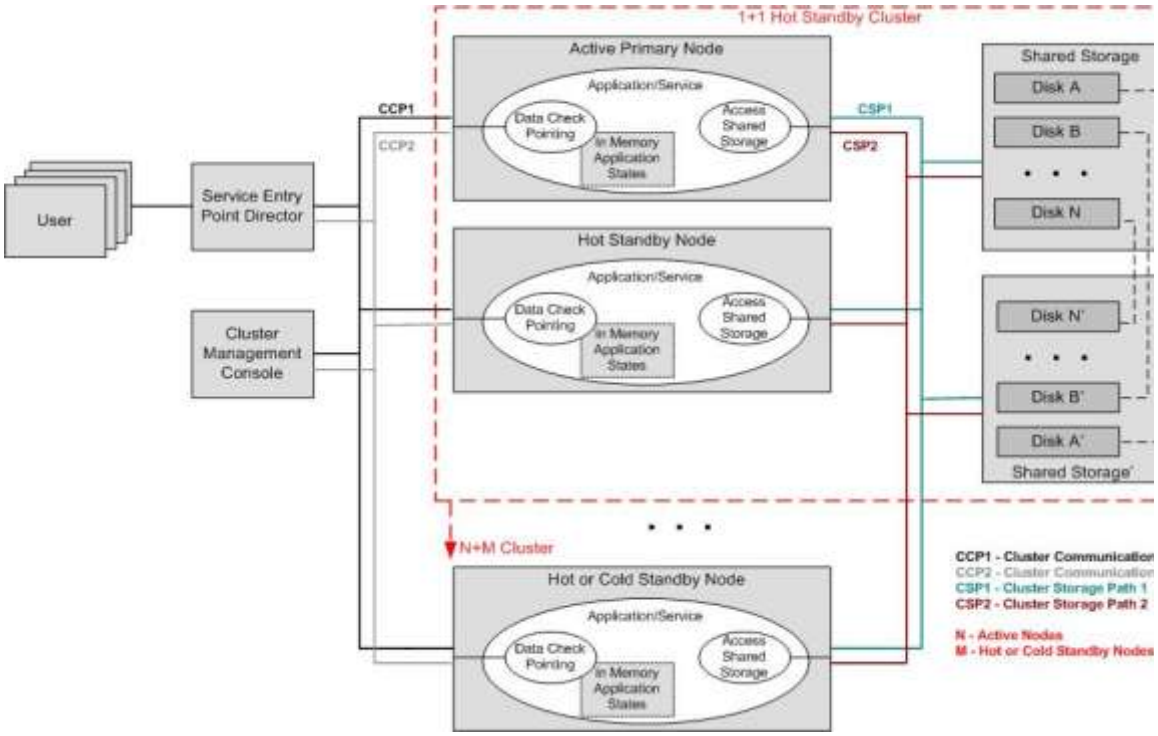

Illustration 3: HAC Cluster View

A CGL high availability cluster is characterized by a set of two or more computing nodes between which an application or workload can migrate depending on a policy-based failover mechanism. Essentially, the cluster nodes can “cover” for each other. Carrier grade services must maintain an uptime of 5 nines (99.999%) or better and, quite often, a failing service must restart in sub-second time frames to maintain continuous operation.

A loosely coupled cluster model with no shared storage is a basic clustering technique that is suitable for many types of telecommunications applications servers. This model eliminates the possibility of a failed shared component affecting the availability of the service or the availability of system.

Whether shared storage is implied or not, a cluster provides the following advantages:

- Prevents a node from being a single point of failure. With hardware faults, the failing node can be replaced or repaired without affecting the service uptime (no unscheduled downtime)
- Allows a software or kernel upgrade to be completed on each node separately without affecting the availability of the service
- Isolates failing nodes from the cluster and enables service to continue using the remaining healthy nodes
- Allows hardware upgrades on each node separately without affecting service availability
- Enables increased capacity to meet load/traffic increases

CGL clustering functional requirements include support for redundancy (no single point of failure), not only at the cluster node level, but at the hardware level as well, including fans, power supplies, memory ECC, communication paths, and storage paths. To support continuous operation of carrier grade services, requirements are defined for node failure detection and various forms of service failover, such as application, node address, and connections failovers.

The CGL clustering requirements are framed around industry standard programming interfaces. The Service Availability Forum (SA Forum) has developed an Application Interface Specification (AIS) that defines service interfaces for clustered applications. The specification is OS-independent and is being used in both proprietary and open source cluster developments. The SA Forum AIS specifies a membership service API, a checkpoint service API, an event service API, a message service API, and a lock service API. AIS also specify an availability management framework (AMF) that provides resource management and application failover policy in the cluster.

CGL CLUSTERING ENVIRONMENT

As stated previously, we learned from our usage model study that no one clustering model fits and meets the needs of all carrier applications. We are not going to create such model. Instead, a more generalized CGL clustering model is presented in this document that serves to identify the functional need of each component of a High Availability Cluster environment. This general model is illustrated in the diagram below, which shows the need for redundancy, stateful failover, and shared storage in a cluster application. This diagram is *not* a

topology of any specific cluster deployment. It is up to application developers and system administrators to determine the usage and configuration of their cluster systems.

The functions shown in Illustration 3 are described below:

- **1+1 Hot Standby Cluster** is composed of one active primary node and one hot standby node and possibly a set of shared storage. It includes redundant paths between cluster nodes and to the storage.
- **Shared Storage** provides a set of mirrored disks (for redundant data) and can be achieved with software or hardware.
- **Redundant Paths** include the multiple communication paths between cluster nodes (CCPs) and the multiple paths from a node to access the storage (CSPs).
- **N+M Cluster** is the extension of a 1+1 hot standby cluster. In this model, the cluster can be configured with additional hot or cold standby nodes as needed by the application. Functional needs of the data check pointing capability and the access to the shared storage remain the same.
- **Data Check Pointing** is part of the cluster services. It constantly synchronizes the in-memory states and data of an application allowing the cluster to provide stateful failover of the application from one node to another node.
- **Access Shared Storage** – A cluster application stores and retrieves application data to and from the redundant shared storage. These data are persistent on the mirrored disks.
- **Service Entry Point Director** routes and directs which cluster node shall provide the service to the service requester.
- **Cluster Management Console** is a node in the system that manages all cluster nodes, but is not part of the cluster membership. It provides a view of the cluster to an operator. It monitors the hardware status of the cluster nodes and monitors cluster events such as cluster node failure. The operator can use it to perform some cluster node failure recovery functions, such as the re-boot of a cluster node allowing the node to re-join the cluster membership.
- **Users** are the service requesters. A user can be a human being, an external device, or another computer system .

End users of carrier grade equipment have prioritized the need for HAC cluster configurations as:

- 2-node (active/hot standby) cluster that supports:
  - Checkpointing of in-memory application states for rapid application failover
  - Shared storage access from a single node at a time.
  - Redundant access to shared storage from a single node
  - Redundant inter-node communication paths
- 2-*node (active/active) cluster* that supports:
  - Concurrent access to shared storage.
- *N node (active/active) cluster* that supports:
  - Storage “scalability”
  - Improved service performance in accessing shared storage.
- *N+M node (active/hot or cold standby) cluster* that supports:
  - Extension of active/standby pair.

RATIONALE FOR CGL CLUSTERING REQUIREMENTS

The requirements described in this section are intended to be independent of specific projects, products, or implementations.

The cluster requirements are framed around industry standard application programming interfaces. For these clustering requirements, the SA Forum Application Interface Specification will be used. The SA Forum AIS services that apply to this specification are:

- SA Cluster Membership Service API (Chapter 6)
- SA Checkpoint Service API (Chapter 7)
- SA Event Service API (Chapter 8)
- SA Message Service API (Chapter 9)
- SA Lock Service API (Chapter 10)

The Availability Management Framework API (Chapter 5) provides the following services to SA-aware applications:

- Registration and un-registration
- Health monitoring
- Availability management
- Protection group management
- Error reporting

Other requirements are described in this document are not related to cluster application APIs, but define requirements that are needed in a cluster. These include items such as shared storage support, synchronized time, and cluster management functions such as monitoring, control, and diagnostics. Items such as a clustered file system and clustered volume manager are also included in this document as they are essential building blocks for HA clustering, although they have no established APIs.

CLUSTERING REQUIREMENT SUB-CATEGORIES

Requirement Sub-Category	Sub-Category Description
CMS	Membership Service
CES	Event Service
CCS	Checkpoint Service
CCM	Communication and Messaging
CLS	Lock Service
CAF	Availability Framework
CMON	Monitoring
CCON	Control
DIAG	Diagnostics
CSM	Shared Storage Management
CFH	Fault Handling

CLUSTERING REQUIREMENTS

CFH.1.0 CLUSTER NODE FAILURE DETECTION

ID	Name	Category	Priority
CFH.1.0	Cluster Node Failure Detection	Cluster	P2

CGL specifies that carrier grade Linux shall provide a fast, communication based cluster node failure mechanism that is reflected in a cluster membership service. At a minimum, the cluster node failure mechanism maintains a list of the nodes that are currently active in the cluster. Changes in cluster membership must result in a membership event that can be monitored by cluster services, applications, and middleware that register to be notified of membership events. Fast node failure detection must not depend on a failing node reporting that the node is failing. However, self-diagnosis may be leveraged to speed up failure detection in the cluster. This requirement does not address the issue of how to prevent failing nodes from accessing shared resources (see CFH.3.0 Application Fail-Over Enabling).

Fast node failure detection shall include the following capabilities:

- Ability to provide cluster membership health monitoring through cluster communication mechanisms.
- Support for multiple, redundant communication paths to check the health of cluster nodes.
- Support for fast failure detection. The guideline is a maximum of 250ms for failure detection. Since there is tradeoff between fast failure detection and potentially false failures, the health-monitoring interval must be tunable.
- Ability to provide a cluster-membership change event to middleware and applications.

Cluster node failure detection must use only a small percentage of the total cluster communication bandwidth for membership health monitoring. The guideline is that the bandwidth used by the health monitoring mechanism shall be linear with respect to the number of bytes per second per node.

CFH.2.0 PREVENT FAILED NODE FROM CORRUPTING SHARED RESOURCES

ID	Name	Category	Priority
CFH.2.0	Prevent Failed Node From Corrupting Shared Resources	Cluster	P1

CGL specifies that carrier grade Linux shall provide a way to fence a failed or errant node from shared resources, such as SAN storage, to prevent the failed node from causing damage to shared resources. Since the surviving nodes in the cluster will want to failover resources, applications, and/or middleware to other surviving nodes in the cluster, the cluster must make sure it is safe to do the failover. Killing the failed node is the easiest and safest way to protect shared resources from a failing node. If a failing node can detect that it is failing, the failing node could kill itself (suicide) or disable its ability to access shared resources to augment the node isolation process. However, the cluster cannot depend on the failing node to alter the cluster when it is failing, so the cluster must be proactive in protecting shared resources.

External Specification Dependencies: This requirement is dependent on hardware to provide a mechanism to reset or isolate a failed or failing node.

CFH.3.0 APPLICATION FAIL-OVER ENABLING

ID	Name	Category	Priority
CFH.3.0	Application Fail-Over Enabling	Cluster	P2

CGL specifies that carrier grade Linux shall provide mechanisms for failing over applications in a cluster from one node to another. Applications and nodes are monitored and a failover mechanism is invoked when a failure is detected. Once a failure is detected, the application failover mechanism must determine which policies apply to this failover scenario and then begin the process to start a standby application or initiate the re-spawn of an application within 1 second.

NOTE: The full application failover time is dependent upon application and node failure detection, the time to apply the failover policies, and the time it takes to start or restart the application. The aggregate failover time for an application must allow the cluster to maintain carrier grade application availability.

CSM.1.0 STORAGE NETWORK REPLICATION

ID	Name	Category	Priority
CSM.1.0	Storage Network Replication	Cluster	P1

CGL specifies that carrier grade Linux shall provide a mechanism for storage network replication. The storage network replication shall provide the following:

- A network replication layer that enables RAID-1-like disk mirroring, using a cluster-local network for data.
- Resynchronization of replicated data after node failure and recovery such that replicated data remains available during resynchronization.

CSM.2.0 CLUSTER-AWARE VOLUME MANAGEMENT FOR SHARED STORAGE

ID	Name	Category	Priority
CSM.2.0	Cluster-aware Volume Management for Shared Storage	Cluster	P2

CGL specifies that carrier grade Linux shall provide management of logical volumes on shared storage from different cluster nodes. Volumes in such an environment are usually on physical disks accessible to multiple nodes. Volume management shall include the following:

- Enabling remote nodes to be informed of volume definition changes.
- Providing consistent and persistent cluster-wide volume names.
- Managing volumes from different cluster nodes consistently.
- Providing support for the striping and concatenation of storage. Clustered mirroring of shared storage is not included in this requirement (see CSM.3.0 Shared Storage Mirroring).

CSM.4.0 REDUNDANT CLUSTER STORAGE PATH

ID	Name	Category	Priority
CSM.4.0	Redundant Cluster Storage Path	Cluster	P1
CGL specifies that Linux shall provide each cluster node with the ability to have redundant access paths to shared storage. CGL Availability Requirement: AVL.7.1.x Multi-Path Access To Storage			

CSM.6.0 CLUSTER FILE SYSTEM

ID	Name	Category	Priority
CSM.6.0	Cluster File System	Cluster	P1
CGL specifies that carrier grade Linux shall provide a cluster-wide file system. A clustered file system must allow simultaneous access to shared files by multiple computers. Node failure must be transparent to file system users on all surviving nodes. A clustered file system must provide the same user API and semantics as a file system associated with private, single-node storage.			

CSM.7.0 SHARED STORAGE CONSISTENT ACCESS

ID	Name	Category	Priority
CSM.7.0	Shared Storage Consistent Access	Cluster	P1
CGL specifies that carrier grade Linux shall provide a consistent method to access shared storage from different nodes to ensure partition information isn't changed on one node while a partition is in use on another node that would prevent the change.			

CCM.2.2 CLUSTER COMMUNICATION SERVICE: FAULT HANDLING

ID	Name	Category	Priority
CCM.2.2	Cluster Communication Service: Fault Handling	Cluster	P1
CGL specifies that carrier grade Linux shall provide a reliable communication service that detects a connection failure, aborts the connection, and reports the connection failure. An established connection must react to and report a problem to the application within 100 ms upon any kind of service failure, such as a process or node crash. The connection failure detection requirement must offer controls that allow it to be tailored to specific conditions in different clusters. An example is to allow the specification of the duration of timeouts or the number of lost packets before declaring a connection failed.			

CAF.2.1 ETHERNET MAC ADDRESS TAKEOVER

ID	Name	Category	Priority
CAF.2.1	Ethernet MAC Address Takeover	Cluster	P1
CGL specifies a mechanism to program and announce MAC addresses on Ethernet interfaces so that when a SW Failure event occurs, redundant nodes may begin receiving traffic for failed nodes.			

CAF.2.2 IP TAKEOVER

ID	Name	Category	Priority
CAF.2.2	IP Takeover	Cluster	P1
CGL specifies a mechanism to program and announce IP addresses (using gratuitous ARP) so that when a SW Failure event occurs, redundant nodes may begin receiving traffic for failed nodes.			

CDIAG.2.1 CLUSTER-WIDE IDENTIFIED APPLICATION CORE DUMP

ID	Name	Category	Priority
CDIAG.2.1	Cluster-Wide Identified Application Core Dump	Cluster	P1
CGL specifies that carrier grade Linux shall provide a cluster-aware application core dump that uniquely identifies which node produced the core dump. For instance, if a diskless node dumps core files to network storage, the core dump will be uniquely identified as originating from that node.			

CDIAG.2.2 CLUSTER-WIDE KERNEL CRASH DUMP

ID	Name	Category	Priority
CDIAG.2.2	Cluster-Wide Kernel Crash Dump	Cluster	P1
CGL specifies that carrier grade Linux shall provide a cluster-aware kernel crash dump that uniquely identifies which node produced the crash dump. For instance, if a diskless node dumps crash data to network storage, the data will be uniquely identified as originating from that node.			

CDIAG.2.3 CLUSTER WIDE LOG COLLECTION

ID	Name	Category	Priority
CDIAG.2.3	Cluster Wide Log Collection	Cluster	P1
CGL specifies that carrier grade Linux shall provide a cluster-wide logging mechanism. A cluster-wide log shall contain node identification, message type, and cluster time identification. This cluster-wide log may be implemented as a central log or as the collection of specific node logs.			

CDIAG.2.4 SYNCHRONIZED/ATOMIC TIME ACROSS CLUSTER

ID	Name	Category	Priority
CDIAG.2.4	Synchronized/Atomic Time Across Cluster	Cluster	P1
CGL specifies that carrier grade Linux shall provide cluster wide time synchronization within 500mS, and must synchronize within 10 seconds once the time synchronization service is initiated. In a cluster, each node must have be synchronized to the same wall-clock time to provide consistency in access times to shared resources (i.e. clustered file system modification and access times) as well as time stamps in cluster-wide logs.			

CLUSTERING REFERENCES

- Birman, Kenneth P. 1997. *Building Secure and Reliable Network Applications*. Manning Publishing Company and Prentice Hall.
- Birman, Ken, et al (circa 2000). “The Horus and Ensemble Projects: Accomplishments and Limitations.”
- Chandra, Tushar, Vassos Hadzilacos, Sam Toueg. June 1996. “The Weakest Failure Detector for Solving Consensus”.
- Davis, Roy G. 1993. *VAX Cluster Principles*. Digital Press.
- Dolev, Danny, and Dalia Malki. 1996. “The Transis Approach to High Availability Cluster Communication.” *Comm. of the ACM* 39 (April): 64-70.
- Pfister, Greg. 1998. “In Search of Clusters”, Second Edition, Prentice Hall PTR.
- Simmons, Chuck, and Patty Greenwald. 1994. “Oracle Lock Manager Requirements,” Oracle Corporation.
- Thomas, Kristin. 2001. “Programming Locking Applications,” IBM Corporation.
- van Renesse, Robbert, Kenneth P. Birman, and Silvano Maffeis. 1996. “HORUS: A flexible Group Communication System.” *Comm. of the ACM* 39 (April): 76-83.
- Service Availability Forum http://www.saforum.org/
- Open Cluster Framework http://www.opencf.org

The following references discuss virtual synchrony:

- Birman, Kenneth.1987. "Exploiting virtual synchrony in distributed systems"
- Extended Virtual Synchrony: http://www.cs.jhu.edu/~yairamir/dcs-94.ps

The following cluster-related whitepapers can be found at http://developer.osdl.org/cherry/cluster-whitepapers/.

- OSDL Cluster Architecture (OSDL-cluster.html)
- Carrier Grade Linux Clustering Model (cluster_alcatel.doc)
- Ericsson Clustering Model Proposal (cluster_ericsson.pdf)
- The Telecom System View (cluster_intel.pdf)
- Foundational Components of Service Availability (cluster_mv.pdf)
- NTT Clustering Model (cluster_ntt.pdf)

DEFINITION OF CLUSTER TERMS

[ ] indicates a term that is defined elsewhere in the definitions of terms.

APPLICATION

A set of [processes], running on a computer [system], that provides a service to the [users] of this [system]. An application is usually referred to as the non operating system portion of the software in a [system].

AVAILABILITY

Availability is the amount of time that a [system] [service] is provided in relation to the amount of time the [system] [service] is not provided. [System] [service] downtime could be the result of [system] [failures] (unscheduled downtime) or for things like upgrades, system relocation, or backups (scheduled downtime). A [system] [service] is provided if the [service] is functioning at an acceptable level of [performance] or [scalability]. Availability is commonly expressed as a percentage (see [five-nines] or [six-nines]).

Percent Availability = (time service is provided / total time) X 100

CLUSTER

Two or more computer [nodes] in a [system] used as a single computing entity to provide a [service] or run an [application] for the purpose of [high availability], [scalability], and distribution of tasks.

COMMUNICATION

The exchange of information between [processes]. These [processes] can be running on the same [node] (intra-node) or on different [nodes] (inter-nodes). The information includes [events] and [messages].

DATA

Numerical or other information represented in a form suitable for processing by a [process].

DATA CHECKPOINTING

The mechanism by which [application] state is transmitted from an active [service unit] to one or more standby [service units].

EVENT

A [communication] with or without data which notifies a set of zero or more [processes] that something took place. This communication can be either within a [node] and/or between [nodes].

EVENT SERVICE

A publish/subscribe event service that manages [events]. [Events] may be grouped into named channels and handle attributes such as priority, ordering, retention times, and persistence. A [subscriber] informs the event mechanism that it wishes to receive a certain event. A [publisher] posts an event to the event mechanism to be delivered to all [subscribers] of that event. This way the [publisher] and [subscriber] are decoupled, they do not have to directly know about each other, just about the event. Events may be asynchronous or synchronous. A [publisher] posting a synchronous event will block or be informed when all [subscribers] have received the event. The [publisher] of an asynchronous event will not block waiting for delivery or be informed when the event is delivered to any [process].

FAILBACK

The process to migrate back to a [node] after it has been [repaired]. It can be controlled or automatic.

FAILOVER

The ability to automatically switch a [service] or capability to a [redundant] [node], [system], or [network] upon the [failure] or abnormal termination of the currently-active [node], [system], or [network].

FAILURE

The inability of a [system] or [system] component to perform a required function within specified limits. A failure may be produced when a [fault] is encountered. Examples of failures include invalid data being provided, slow response time, and the inability for a [service] to take a request. Causes of failure can be hardware, firmware, software, network, or anything else that interrupts the [service].

FAILURE DETECTION

A failure is ultimately caused by an unmasked [fault] in the [system]. Failure detection is the process, usually from external view, to detect a [failure] of the [service] the [system] is providing.

FAULT

An error in a computer [system] or the [service] it provides. A fault may be masked and not impact the [application] or the [service] it provides. A fault can also be classified as transient or permanent. A fault is often associated with a [system] defect in the software or hardware. A fault can be caused by external stimulus to the [system].

FAULT CONFINEMENT

Equivalent to [fault isolation].

FAULT DETECTION

Ability to detect an abnormal condition (device failure, temperature error, etc.) in the [system].

FAULT DIAGNOSIS

The localization of a [fault] to its repair unit.

FAULT ISOLATION

Ability to protect the rest of the [system] from the effects of a [fault].

FAULT PREDICTION

Detecting or forecasting [faults].

FAULT TOLERANCE

Ability for a [system] to mask a set of [failures] from impacting the [service] it provides.

FIVE-NINES

Five-nines is measured as 99.999% [service] [availability]. It is equivalent to 5 minutes a year of total planned and unplanned downtime of the [service] provided by the [system].

GROUP MULTICAST

The sending of a single [message] to a set of destination [processes].

HAND-OVER

Equivalent to [switch-over]

LOCK SERVICE

The lock [service] is a distributed lock [service], suitable for use in a [cluster], where [processes] in different [nodes] might compete with each other for access to shared resources. A lock [service] may provide the following capabilities: exclusive and shared access, synchronous and asynchronous calls, lock timeout, trylock, deadlock detection, orphan locks, and notification of waiters.

MESSAGE

A [communication] with [data] in a form suitable for transmission. A message may contain attributes of the [communication] such as source, destination, time stamps, and authorization information, etc. It may also contain [application] specific information.

MTTF

Mean Time To [Failure]. The interval in time which the [system] can provide [service] without [failure].

MTTR

Mean Time To [Repair]. The interval in time it takes to resume [service] after a [failure] has been experienced.

NETWORK

A connection of [nodes] which facilitates [communication] among them. Usually, the connected nodes in a network use a well defined [network protocol] to communicate with each other.

NETWORK PROTOCOLS

Rules for determining the format and transmission of data. Examples of network protocols include TCP/IP, UDP, etc.

HIGH AVAILABILITY

The state of a [system] having a very high ratio of [service] uptime compared to [service] downtime. Highly available systems are typically rated in terms of number of nines such as [five-nines] or [six-nines].

NODE

A single computer unit, in a [network], that runs with one instance of a real or virtual operating system.

NODE MEMBERSHIP

The mechanism by which computer [nodes] join and leave a cluster as well as the mechanism to detect [node] [failure]. A [node] is deemed to be a member if it has joined the [cluster] successfully. A [node] is deemed to be a non-member if it has not joined the cluster or if it has left the cluster. A detected [failure] may result in the [node] leaving the cluster or being isolated from the cluster, depending on node membership policy.

PERFORMANCE

The efficiency of a [system] while performing tasks. Performance characteristics include total throughput of an operation and its impact to a [system]. The combination of these characteristics determines the total number of activities that can be accomplished over a given amount of time.

PROCESS

A single instance of a software program running on a single [node].

PROCESS GROUP

A collection of processes registered within [cluster] software.

PROCESS GROUP MEMBERSHIP

The mechanism by which [process] registration, un-registration, and [failure detection] is managed. A [process] is deemed to be a member if it has registered with the [process group] successfully. A [process] is deemed to be a non-member if it has not registered with the process group. A [detected] failure may cause the [process] to become a non-member, depending on the process group membership policy. A [process] can gracefully un-register to depart from the process group. The process group membership also handles authorization to join the membership. Process group membership depends upon [node membership] if process group membership is available on multiple [nodes]. Process group membership is used to execute application [failover] policy.

PUBLISHER

A [process] that sends [events].

RAS

[Reliability], [availability], and [serviceability]

RECOVERY

To return a failing component, [node] or [system] to a working state. A failing component can be a hardware or a software component of a [node] or [network]. Recovery can also be initiated to work around a [fault] that has been detected; ultimately restoring the [service].

REDUNDANCY

Duplication of hardware, software, or network components in a [system] to avoid [Single Points of Failure].

RELIABILITY

The continuation of [service] in the absence of [failure]. Reliability is commonly measured as the [MTTF] of a [system].

REPAIR

The process to remove a [fault].

REPLICATION

A component, [node], or [system] which is configured identically to a base component, [node] or [system] for the purpose of [fault tolerance], [performance], or ease of [service].

SCALABILITY

How well a solution to some problem will work when the size of the problem increases? In the CGL context, the scalability is defined as the ability of a [system] to provide the same level of [high availability] performance when the work load of the [service] increases. The solution to increase the [system] or [service] scalability can be software or hardware oriented.

SERVICE

A set of functions provided by a computer [system]. Examples of communications services include media gateway, signal, or soft switch types of applications. Some general examples of services include web based or database transaction types of applications.

SERVICE UNIT

A collection of one or more software [processes] that provide [service] to a [user].

SERVICEABILITY

The capability for a [system] to be maintained and updated. Often, serviceability is measured by how easy a maintenance task can be performed or how quickly a [system] [fault] can be tracked down and repaired so that the [system] can resume the [service].

SINGLE POINT OF FAILURE

Any component or [communication] path within a computer [system] that would result in an interruption of the [service] if it failed.

SIX-NINES

Six-nines is measured as 99.9999% [service] [availability]. It is equivalent to 30 seconds a year of total planned and unplanned downtime of the [service] provided by the [system].

SUBSCRIBER

A [process] that receives [events]. A [subscriber] may subscribe to one or many [events]. A subscriber may join and leave an event subscription at any time without involving the publishers.

SWITCH-OVER

Ability to switch to a [redundant] [node], [system], or [network] upon a normal termination of the currently-active [node], [system], or [network]. Switch-over can happen with or without human intervention.

SYSTEM

A computer system that consists of one computer [node] or many nodes connected via a computer network mechanism.

USER

An external entity that acquires [service] from a computer [system]. It can be a human being, an external device, or another computer [system].

6. SERVICEABILITY REQUIREMENTS DEFINITION

This section specifies a set of useful and necessary features for servicing and maintaining a system. Telecommunication systems such as management servers, signaling servers, and gateways must have the capability to be managed and monitored remotely, have robust software package management for installations and upgrades, and have mechanisms for capturing and analyzing failure information. A single point of control is required for applications, software, hardware, and data for functions such as data movement, security, backup, and recovery.

CGL systems will support remote management standards such as Simple Network Management Protocol (SNMP), Common Information Model (CIM), and Web-Based Enterprise Management (WBEM). Local management standards

include IPMI and the Service Availability Forum's Hardware Platform Interface (HPI).

Debuggers, application and kernel dumpers, watchdog triggers, and error analysis tools are needed to debug and isolate failures in a system. Diagnostic monitoring of temperature controls, fans, power supplies, storage media, the network, CPUs, and memory are needed for quick failure detection and failure diagnosis.

SERVICEABILITY SUB-CATEGORIES

Requirement Sub-Category	Sub-Category Description
SMM	Management and Monitoring
SPM	Software Package Management
SFA	Failure Analysis

SERVICEABILITY REQUIREMENTS

SMM.3.1 SERIAL CONSOLE OPERATION

ID	Name	Category	Priority
SMM.3.1	Serial Console Operation	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support for a connection to a system console via a serial port on the system where a serial port exists. All output that would appear on a local console must appear on the remote console.			

SMM.3.2 NETWORK CONSOLE OPERATION

ID	Name	Category	Priority
SMM.3.2	Network Console Operation	Serviceability	P1
CGL specifies that Linux shall provide support for a management console connection via a network port in addition to providing the standard support for a management console connection via a serial port.			

SMM.4.0 PERSISTENT DEVICE NAMING

ID	Name	Category	Priority
SMM.4.0	Persistent Device Naming	Serviceability	P1
CGL specifies that carrier grade Linux shall provide consistent device naming functionality. The user-space system name of the device shall be maintained when the device is removed and reinstalled even if the device is plugged into a different bus, slot, or adapter. A device name shall be assigned, based on hardware identification information using policies set by the administrator.			

SMM.5.0 KERNEL PROFILING

ID	Name	Category	Priority
SMM.5.0	Kernel Profiling	Serviceability	P1
CGL specifies that Linux shall support profiling of a running kernel and applications to identify bottlenecks and other kernel and application statistics.			

SMM.5.1 APPLICATION PROFILER (WAS AVL.19.0)

ID	Name	Category	Priority
SMM.5.1	Application Profiler (was AVL.19.0)	Serviceability	P1

CGL specifies that carrier grade Linux shall provide a mechanism to profile critical resources of the kernel and applications. The critical resources that are profiled by this mechanism shall include (but are not limited to):

- Time used
- Memory used
- Number of semaphores, mutexes, sockets, and threads/child processes in use
- Number of open files. Monitoring shall happen at configurable, periodic intervals or as initiated by the user.

SMM.7.1 TEMPERATURE MONITORING

ID	Name	Category	Priority
SMM.7.1	Temperature Monitoring	Serviceability	P1

CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of system temperature settings and conditions.

SMM.7.2 FAN MONITORING

ID	Name	Category	Priority
SMM.7.2	Fan Monitoring	Serviceability	P1

CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of system fan settings and conditions.

SMM.7.3 POWER MONITORING

ID	Name	Category	Priority
SMM.7.3	Power Monitoring	Serviceability	P1
CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of system power settings and conditions.			

SMM.7.4 MEDIA MONITORING

ID	Name	Category	Priority
SMM.7.4	Media Monitoring	Serviceability	P1
CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of media settings and conditions for system media, such as hard disks or hardware specific disk sub-systems.			

SMM.7.5 NETWORK MONITORING

ID	Name	Category	Priority
SMM.7.5	Network Monitoring	Serviceability	P1
CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of system network settings and conditions.			

SMM.7.6 CPU MONITORING

ID	Name	Category	Priority
SMM.7.6	CPU Monitoring	Serviceability	P1
CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of CPU settings and conditions, such as current utilization totals, per process totals and trends, and current speed settings.			

SMM.7.7 MEMORY MONITORING

ID	Name	Category	Priority
SMM.7.7	Memory Monitoring	Serviceability	P2
CGL specifies that carrier grade Linux shall provide a capability that supports the monitoring of memory conditions, such as current utilization totals, and per process totals and trends.			

SMM.8.1 KERNEL MESSAGE STRUCTURING

ID	Name	Category	Priority
SMM.8.1	Kernel Message Structuring	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support that allows the structuring of kernel messages using an event log format to provide more information to identify the problem and its severity, and to allow client applications registered for the fault event to take policy-based corrective action.			

SMM.8.2 PLATFORM SIGNAL HANDLER

ID	Name	Category	Priority
SMM.8.2	Platform Signal Handler	Serviceability	P1
CGL specifies that carrier grade Linux shall provide an infrastructure to allow "hardware errors" to be logged using the event logging mechanism. A default handler shall be provided.			

SMM.8.3 REMOTE ACCESS TO EVENT LOG

ID	Name	Category	Priority
SMM.8.3	Remote Access to Event Log	Serviceability	P2
CGL specifies that carrier grade Linux shall provide support for a remote access capability that allows a centralized system to access the Linux OS event log information of a remote system.			

SMM.9.0 DISK AND VOLUME MANAGEMENT

ID	Name	Category	Priority
SMM.9.0	Disk and Volume Management	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support for the installation of a subsystem that supports hard disks to be managed without incurring downtime:

- Physical disks can be grouped into volumes and the volume definitions can be modified without downtime.
- Filesystems that are defined within volumes can be enlarged without requiring unmounting.
- Support can be configured out if desired.

SMM.12.0 REMOTE BOOT SUPPORT (WAS PMT.2.0)

ID	Name	Category	Priority
SMM.12.0	Remote Boot Support (was PMT.2.0)	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support for remote booting across common LAN and WAN communication media to support diskless systems.			

SMM.13.0 DISKLESS SYSTEMS (WAS PMS.4.0)

ID	Name	Category	Priority
SMM.13.0	Diskless Systems (was PMS.4.0)	Serviceability	P1
CGL specifies that carrier grade Linux shall provide for Linux on diskless systems.			

SMM.15 THREAD NAMING

ID	Name	Category	Priority
SMM.15	Thread Naming	Serviceability	P2
Linux Foundation CGL specifies that carrier grade Linux shall provide the ability to uniquely identify threads with a symbolic name in addition to the existing process and thread ID mechanism. These symbolic names can be assigned via an API exposed to applications and can be assigned either at process / thread creation time or at any time after the process / thread has been started.			

SMM.16 SYSTEM BLACK BOX

ID	Name	Category	Priority
SMM.16	System Black Box	Serviceability	P2

CGL specifies that carrier grade Linux shall provide a system-wide monitoring and logging facility, a system black box, with at least the following attributes:

- Kernel and operating system events must be logged to the black box.
- An API must be provided for applications to log events to the black box.
- An API must be provided that allows controlling which events are logged and from what facilities.
- All logged events must be stored in a way that will available after a system crash / reboot.
- Tools must be provided to analyze events following a system crash /

ID	Name	Category	Priority
reboot.			

SMM.17 DISCOVERY OF PLATFORM CPU ARCHITECTURE

ID	Name	Category	Priority
SMM.17	Discovery of Platform CPU Architecture	Serviceability	P1
CGL specifies that carrier grade Linux shall provide a mechanism for applications to discover at runtime the number of caches and the sizes of each. This mechanism must present such architectural information in a format that is uniform across platforms.			

SMM.18 API FOR NON-UNIFORM MEMORY ARCHITECTURES

ID	Name	Category	Priority
SMM.18	API for Non-Uniform Memory Architectures	Serviceability	P1

CGL specifies that carrier grade Linux shall implement the notion of a latency domain, defined as a set of CPUs with directly attached, local memory. All systems shall have at least one latency domain, representing uniform memory architecture. Additional latency domains can exist for non-uniform memory architectures, in which case carrier grade Linux will provide an API that allows a process to:

- **Identify the NUMA topology of the system including:**
  - The latency of each latency domain
  - The number of CPUs
  - The amount of memory in the latency domain
- **Specify the desired memory allocation policy including:**
  - Local: Memory allocations will first occur from the local latency domain.
  - Specific: Memory allocations will first occur from the specified latency domains.
  - Interleaved: Memory allocations will be spread across all latency domains.

SPM.1.0 REMOTE PACKAGE UPDATE AND INSTALLATION

ID	Name	Category	Priority
SPM.1.0	Remote Package Update and Installation	Serviceability	P1

CGL specifies that carrier grade Linux shall provide a remote software package update feature. The package shall include functions that allow kernel modules and application software to be installed or upgraded remotely, while minimizing downtime of the system. The use of the term "remotely" does not imply a central package management platform, nor does it preclude such a system. This requirement only necessitates that a single device may be upgraded without requiring the administrator to be physically at the device. Note: Due to the wide range of platforms and applications in use, CGL does not specify a specific downtime limit metric. Downtime targets will vary based on the system application.

SPM.2.0 NO SYSTEM REBOOT FOR UPGRADE OF KERNEL MODULES

ID	Name	Category	Priority
SPM.2.0	No System Reboot for Upgrade of Kernel Modules	Serviceability	P2

CGL specifies that carrier grade Linux shall provide remote software installation and upgrade mechanisms that requiring no system reboots:

- No reboot shall be required to upgrade kernel modules.
- Remote software installation and upgrade mechanisms will not require more reboots than the same upgrade done using the console.

SPM.2.1 NO SYSTEM REBOOT FOR APPLICATION PACKAGE UPDATE

ID	Name	Category	Priority
SPM.2.1	No System Reboot for Application Package Update	Serviceability	P1
CGL specifies that carrier grade Linux shall provide remote software installation and upgrade mechanisms that require no system reboots: • No reboot shall be required to upgrade user-space applications provided by CGL system software.			

SPM.3.0 VERSION AND DEPENDENCY CHECKING VIA PACKAGE MANAGEMENT

ID	Name	Category	Priority
SPM.3.0	Version and Dependency Checking via Package Management	Serviceability	P1
CGL specifies that carrier grade Linux shall provide remote software installation and upgrade capabilities that include provisions for version compatibility and dependency checking at the package level.			

SPM.4.0 UPGRADE LOG

ID	Name	Category	Priority
SPM.4.0	Upgrade Log	Serviceability	P2
CGL specifies that carrier grade Linux shall provide remote software installation and upgrade mechanisms that perform transaction logging of dates, times, changes, and the identity of the user performing a change.			

SFA.1.0 KERNEL PANIC HANDLER ENHANCEMENTS

ID	Name	Category	Priority
SFA.1.0	Kernel Panic Handler Enhancements	Serviceability	P1

CGL specifies that carrier grade Linux shall provide enriched capabilities in response to a system panic. Currently the default system panic behavior is to print a short message to the console and halt the system. CGL systems shall provide a set of configurable functions, including:

- Logging the panic event to the system event log
- Cycling power (rebooting) or powering off
- Forcing a crash dump

CGL shall support enhanced kernel panic reporting, at a minimum supporting proper resolution of in-kernel symbols. This will make kernel panic reports useful to administrators that do not have access to the kernel for which the report was generated.

SFA.2.1 LIVE KERNEL REMOTE DEBUGGER

ID	Name	Category	Priority
SFA.2.1	Live Kernel Remote Debugger	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support for remote debugging of a live kernel. This shall include support over serial and/or local Ethernet.

SFA.2.2 DYNAMIC PROBE INSERTION

ID	Name	Category	Priority
SFA.2.2	Dynamic Probe Insertion	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support for the ability to dynamically insert software instrumentation into a running system in the kernel or applications.

- The instrumentation must be insertable to any part of the kernel.
- The instrumentation should allow control to be passed to a user-provided module.
- The instrumentation should not require interactive direction, i.e., no user sitting at the kernel debugger.
- The user-provided modules should have access to data the kernel would normally be expected to have access to, e.g., hardware registers, kernel

SFA.2.3 USER SPACE DEBUG SUPPORT FOR THREADS

ID	Name	Category	Priority
SFA.2.3	User Space Debug Support for Threads	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support to fully enable debugging of multi-threaded programs. This support should allow any actions available for debugging a single-threaded (non-threaded) process be extended to be available for every thread in a multi-threaded process. CGL shall provide specific additional debugging capabilities that are unique to multi-threaded applications:

- Automatic notification of a new thread.
- List of threads and the ability to switch among them.
- Apply specific debug commands to a list of threads.

SFA.2.4 MULTITHREADED CORE DUMP SUPPORT FOR THREADED APPLICATIONS

ID	Name	Category	Priority
SFA.2.4	Multithreaded Core Dump Support for Threaded Applications	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support for correctly storing core dumps of multi-threaded user-space applications.			

SFA.3.0 KERNEL DUMP: ANALYSIS

ID	Name	Category	Priority
SFA.3.0	Kernel Dump: Analysis	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support for tools to enable enhanced analysis of kernel dumps. These enhancements must include, but not be limited to, the following capabilities:

- Access to kernel structures
- Virtual-to-physical address translation
- Module access
- Preserve all tools and CPU states

SFA.4.0 KERNEL DUMP: LIMIT SCOPE

ID	Name	Category	Priority
SFA.4.0	Kernel Dump: Limit Scope	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support for configuring the amount of system information that is retained. The minimum type of configuration would be only kernel memory or all system memory. A way must be provided for a system administrator to specify which type of system dump should be performed.			

SFA.8.0 KERNEL FLAT/GRAPH EXECUTION PROFILING

ID	Name	Category	Priority
SFA.8.0	Kernel Flat/Graph Execution Profiling	Serviceability	P1
CGL specifies that carrier grade Linux shall provide support for profiling of the running kernel using a prof or gprof style of recording trace information during system execution.			

SFA.10.0 KERNEL DUMP: CONFIGURABLE DESTINATIONS

ID	Name	Category	Priority
SFA.10.0	Kernel Dump: Configurable Destinations	Serviceability	P1

CGL specifies that carrier grade Linux shall provide support for producing and storing kernel dumps as follows:

- It must be possible to store kernel dumps to disk and across a network.
- Regardless of the specific dump target, dumps must be preserved across the next system boot.

7. PERFORMANCE REQUIREMENTS DEFINITION

This section is a collection of requirements for the Linux operating system that describe the performance and scalability requirements of typical communications systems. Key requirements include a system's ability to meet service deadlines; to scale in order to take advantage of symmetric multiprocessing (SMP), simultaneous multithreading (SMT) technology, and large memory systems; and to provide efficient, low latency communication.

Without predictable execution latencies, it is possible that service deadlines would not be met, resulting in dropped calls, unreasonable call-response characteristics, or even dropping the entire service from active operation. Soft real-time scheduling provides predictable CPU scheduling latencies within defined loads. Latency and scheduling parameters are required to be configurable at runtime, including the scheduling quantum being configurable to 1ms or less. However, the services use many resources other than the CPU; therefore, protection against priority inversion, priority inheritance to system

resources, and appropriate system resource scheduling are also required to maintain predictable scheduling.

To take advantage of scalable hardware architectures, CGL specifies support for SMP and SMT, which includes process affinity, task exclusive binding to logical CPUs and interrupt affinity capabilities. Large memory systems of more than 4GB of physical memory are needed to handle the memory demands of scalable communication applications.

Protocol stacks are required to be prioritized so certain protocols may take scheduling priority over less important network protocols. To improve latency and reduce CPU usage in network communications, zero-copy network protocols may be needed. IPv6 forwarding tables are required to be compact and use a small amount of memory. Support in the Linux Kernel for a 9000 byte Maximum Transfer Unit (MTU) is required.

PERFORMANCE FOCUS AREAS

REAL-TIME PROCESSING

SCOPE

The telecommunications application market faces new technical challenges with the introduction of architectures such as Next Generation Networks and IP multimedia services for mobile networks.

Real-time behavior is a major issue for new applications and protocol classes based on IP services such as VoIP, SIGTRAN, and RTP, where real time behavior drives the quality of service for end-users. Enhancements in real-time behavior would allow Linux to be used for some applications that are currently run on other real-time operating systems.

This document does not make a distinction between hard real-time and soft real-time support in the Linux kernel. Real-time capabilities are defined in terms such as maximum scheduling latency.

HIGH RESOLUTION TIMERS

Incorporating high-resolution timers based on a 1 ms tick, rather than the currently supported 10 ms tick, will enhance the real-time task scheduling capabilities of Linux. If hardware platform support is provided for a 1 ms tick, the

kernel will no longer be required to program a specific timer to elapse after 1 ms, eliminating overhead.

This feature enables:

- A 1 ms quantum to be managed for task scheduling.
- A 1 ms timer to be managed without requiring the kernel to program a specific clock. Configuring the kernel with a 1 ms tick value rather than the current 10 ms tick value allows rescheduling to occur every 1 ms in response to a periodic clock timer interrupt.

POSIX REAL-TIME FEATURES

POSIX real-time and advanced real-time features enable better support for real-time, portable applications at the API level.

PROTECTION AGAINST PRIORITY INVERSION

Priority inversion is an issue for real-time application programming because scheduling priorities defined by design may be inverted causing unexpected latencies. Priority inversion happens when a lower priority thread blocks a higher priority one. The most general case is when a lower priority thread holds a resource needed by the higher priority thread.

Priority inversion protection can be provided in the Linux kernel by dynamically modifying the thread scheduling priority when lower priority threads are holding resources.

Transitive priority inheritance is required to deal with cases where several mutexes are used by several threads.

Scheduling policy can also be dynamically modified by the protection mechanism. For example, time-sharing threads can be promoted to real-time FIFO threads. This can have undesired consequences, however, as timesharing processes are generally not coded with FIFO policy in mind. A means should be provided for the client application to specify priority inheritance or priority protection capabilities for the internal mutexes that they use.

APIs providing this capability should be implemented in such a way so that they will perform correctly if they are promoted to real-time policies.

MESSAGE QUEUES WITH PRIORITY PROMOTION

The priority inheritance protection mechanism can be extended by using a dynamic priority promotion system for message queues. In such a system, the priority of the receiver thread is promoted by the scheduler according to the message priority, enabling processing of urgent messages with high scheduling priority.

HANDLING INTERRUPTS AS KERNEL THREADS

Since interrupt service routines are not allowed to sleep, preemption locks in interrupt handlers normally can't be changed to mutexes. To change preemption locks that are placed in interrupt service routines, interrupt service routines (aside from the timer interrupt routines) could be handled by kernel threads.

Mapping interrupt service routines onto real-time kernel threads enables interrupt handlers to be assigned priorities and soft real-time processes to be given higher priorities than interrupt handlers, allowing better designs. An additional benefit is the reduction of critical sections in interrupt handlers.

SYMMETRIC MULTI-PROCESSING

REDUCING SMP CONTENTION

Improving performance and scalability in an SMP system can be accomplished by reducing resource contention through process affinity interrupt affinity, and Hyper-Threading support.

SMP kernel critical sections can be handled by:

- A spin-lock
- A mutex, if not used in an interrupt handler

Generally, the spin-lock option is the faster in terms of CPU time, but it requires that preemption be disabled and introduces processor-level latency when the resource is already locked. The mutex option adds mutex and context switching costs, but latency remains at the process level.

Using spin-lock with a high number of processors can lead to high latency depending on the critical section length.

Quality of service must be taken into account for following cases:

- When timers are armed in parallel on several processors
- When concurrent file accesses occur
- When shared-memory is accessed by several processors

PROCESS AFFINITY

Process affinity provides for load balancing at the application level. When process affinity is used, it provides more efficient caching. For example, it must be possible to bind real-time processes to specified processors. Other processes in the systems do not need to be assigned to specified processors.

INTERRUPT HANDLER AFFINITY

Assigning the top half of interrupt handlers to a single processor enables load balancing of interrupt handlers. The bottom half and top half of each interrupt handler should be assigned to the same CPU to reduce inter-processor contention.

HYPER-THREADING SUPPORT

Because the logical Hyper-Threaded processors share a cache, the scheduler only needs to keep threads attached to one of the adjacent logical processors. The scheduler can move threads between adjacent logical processors with no performance degradation because the cache is stable between the two logical processors.

MEMORY USAGE

As CPU capabilities increase, memory demands also increase as more communication contexts can be handled per system. Memory related requirements are oriented toward high physical memory (HIGHMEM) and virtual memory.

SUPPORT OF MORE THAN 4G PHYSICAL MEMORY

Support for more than 4G of physical memory is a requirement for 32-bit and 64-bit processor architectures.

COMMUNICATION SERVICE

Communication services have a major impact on performance of telecommunications applications. Performance of Linux stacks should be evaluated as follows:

- Message delivery latency and throughput
- Resource usage including CPU and memory usage
- Load balancing capability on an SMP system

IPV4, IPV6, MIPV6 FORWARDING TABLES FAST ACCESS AND COMPACT MEMORY

The speed at which packets can be routed is limited by the time it takes to perform the forwarding table lookup for each packet.

When a basic lookup method is used, such as the BSD binary trie, the number of nodes equal to the length of the address in bits is potentially traversed in the forwarding table, generating an equivalent number of memory accesses. The current Linux implementation is not highly scalable.

Methods faster than those currently available should be implemented to support 2000 routes updated per second and up to 500,000 routes with low lookup latency. The tradeoff between memory and access latency should also be addressed.

See “Survey and taxonomy of IP address lookup algorithms “ at http://mia.ece.uic.edu/~papers/Surveys/pdf00000.pdf.

CLUSTER COMMUNICATION SERVICE

A cluster benefits from a cluster specific communication service that addresses specific issues such as latency, ordering, and recovery. A cluster communication service can achieve better performance than a general communication service when used in a cluster, because it has knowledge of the local topology, including the cluster membership.

DIFFSERV SUPPORT

Support should be provided for Differentiated Services (RFCs 2474 and 2475) for IPv4 to enable quality of service and traffic control.

PRIORITIZED PROTOCOL PROCESSING

A prioritized protocol processing mechanism enables a high-priority process to quickly obtain data from the network even if massive packets arrive for multiple processes. It is based on a protocol priority assignment mechanism that allows a higher scheduling priority to be given to the protocol with higher priority.

I/O AND FILE SYSTEMS

NETWORK STORAGE REPLICATION

A network storage replication service uses local network and device resources. Performance depends on the local network and storage devices used.

A network storage replication service provides a lower performance level compared to local storage access. The relative difference must be less than 30 % in terms of user throughput in normal conditions when mirrored devices are synchronized.

Upon device resynchronization, the user throughput should not be reduced more than 25% compared to normal conditions.

AVAILABILITY AND INITIALIZATION

APPLICATION PRE-LOADING

The CGL 2.0 requirement for application pre-loading should be extended to enhance dynamic loading performance. Often, several seconds are spent in the dynamic ELF loader for symbol relocation.

PERFORMANCE REQUIREMENTS

PRF.1.4 HIGH-RESOLUTION TIMERS

ID	Name	Category	Priority
PRF.1.4	High-Resolution Timers	Performance	P1
CGL specifies that carrier grade Linux shall provide high-resolution timer support. As specified by POSIX 1003.1b section 14, Clocks and Timers API.			

PRF.1.7 HANDLING INTERRUPTS AS THREADS

ID	Name	Category	Priority
PRF.1.7	Handling Interrupts As Threads	Performance	P2

CGL specifies that carrier grade Linux shall enable handling of interrupt handlers (top half and bottom half) as a task-based process rather than in interrupt processing routine mechanism to allow:

- A mutex-based critical section inside an interrupt handler.
- The ability for an interrupt handler to sleep.
- Prioritization of an interrupt handler based on real-time scheduling priorities.
- Affinity and load-balancing in an SMP. Context switching overhead should be considered case by case in the application design. The interrupts are divided into a critical urgent part that kernel needs to execute quickly, and deferrable part. The thread based interrupt handler should be applied at deferrable part.

PRF.2.1 ENABLING PROCESS AFFINITY

ID	Name	Category	Priority
PRF.2.1	Enabling Process Affinity	Performance	P1

CGL specifies that carrier grade Linux shall enable process affinity. Process affinity enables a process to run on an explicitly designated processor. When process affinity is used, it provides more efficient caching. For example, it must be possible to bind real-time processes to specified processors.

PRF.2.2 ENABLING INTERRUPT CPU AFFINITY

ID	Name	Category	Priority
PRF.2.2	Enabling Interrupt CPU Affinity	Performance	P1
CGL specifies that carrier grade Linux shall enable interrupt CPU affinity. The interrupts are divided into a critical urgent part that the kernel needs to execute quickly and a deferrable part. CGL should enable interrupt CPU affinity on the critical urgent part. Note: The latest stable kernel enables interrupt affinity based on the /proc configuration interface.			

PRF.2.3 (HYPER-THREADING) OPTIMIZED SMT SUPPORT

ID	Name	Category	Priority
PRF.2.3	(Hyper-Threading) Optimized SMT Support	Performance	P1
CGL specifies that carrier grade Linux shall enable optimized symmetric multi-threading (SMT) processors and interrupt migration between logical processors. Note: The latest stable kernel enables this feature.			

PRF.4.2 SUPPORT OF GIGABIT ETHERNET JUMBO MTU

ID	Name	Category	Priority
PRF.4.2	Support of Gigabit Ethernet Jumbo MTU	Performance	P1
CGL specifies that carrier grade Linux shall enable support for a 9000 byte Maximum Transmission Unit (MTU) for the Gigabit Ethernet protocol to enable lower CPU overhead and better throughput. This shall be a configurable option as some applications may prefer low latency to large message sizes. Hardware support is required.			

PRF.5.0 EFFICIENT LOW-LEVEL ASYNCHRONOUS EVENTS

ID	Name	Category	Priority
PRF.5.0	Efficient Low-Level Asynchronous Events	Performance	P1
CGL specifies that carrier grade Linux shall provide an API for applications that allows asynchronous notifications to be delivered based either level or edge triggers.			

PRF.6.0 MANAGING TRANSIENT DATA

ID	Name	Category	Priority
PRF.6.0	Managing Transient Data	Performance	P1
CGL specifies that carrier grade Linux shall provide support for a self resizing, file system stored in virtual memory for transient data that can be limited to a maximum size.			

PRF.7.0 INTERRUPTLESS ETHERNET DELIVERY

ID	Name	Category	Priority
PRF.7.0	Interruptless Ethernet Delivery	Performance	P1
CGL specifies that carrier grade Linux shall provide for the capability for Ethernet drivers to operate in a pure polling mode in which they do not generate interrupts for arriving frames. This is to prevent interrupt-storms from consuming too many CPU cycles. This is primarily an issue for gigabit Ethernet.			

PRF.8.0 NETWORK STORAGE BLOCK LEVEL REPLICATION PERFORMANCES

ID	Name	Category	Priority
PRF.8.0	Network Storage block level Replication Performances	Performance	P2

CGL specifies that carrier grade Linux shall provide a network storage replication service with the following performance levels:

- Less than 30% decrease in user throughput compared to local storage access using a network interface and with full available network bandwidth.
- Less than 25% decrease in user throughput during resynchronization of redundant devices compared with normal throughput when devices are synchronized.

PRF.14.0 RAID 0 SUPPORT

ID	Name	Category	Priority
PRF.14.0	RAID 0 Support	Performance	P1

CGL specifies that carrier grade Linux shall provide RAID 0 (striping) support that stripes data across multiple disks without any redundant information to enhance performance in either a request-rate-intensive or transfer-rate-intensive environment.

PERFORMANCE REFERENCES

- Linux Scheduler latency, Clark Williams, Red Hat, Inc. March 2002 http://www.linuxdevices.com/files/article027/rh-rtpaper.pdf
- The Linux scalability Project http://www.citi.umich.edu/techreports/reports/citi-tr-99-4.pdf
- Scalable statistic counter project http://lse.sourceforge.net/counters/statctr.html

- Linux 2.5 Timer scalability study from Andy Pfiffer
  http://developer.osdl.org/andyp/timers/
- LK SCTP / TCP performance comparison
  http://datatag.web.cern.ch/datatag/WP3/sctp/tests.htm
- kernel 2.6 includes some scalability enhancements that are referenced in
  http://www.kernelnewbies.org/status/Status-08-Aug-2003.html
- lmbench: Portable Tools for performance analysis:
  http://www.usenix.org/publications/library/proceedings/sd96/full_papers/mcvoy.pdf
- Time-critical tasks in Linux 2.6. Concept to increase the preemptability of the Linux kernel.
  http://inf3-www.informatik.unibw-muenchen.de/research/linux/hannover/automation_conf04.pdf
- CELF-RT working group
  http://tree.celinuxforum.org/pubwiki/moin.cgi/RealTimeWorkingGroup
- Integration New Capabilities into NetPIPE:
  http://www.scl.ameslab.gov/netpipe/np_euro.pdf

8. STANDARDS REQUIREMENTS DEFINITION

One goal of the CGL effort to achieve high reliability, availability, and serviceability (RAS), and application portability is to leverage mature and well-established industry standards that are common and relevant to the carrier-grade environment and include them as part of the CGL requirements.

Open standards are important because they are freely available for anyone or any organization to use and because open standards can evolve with wide community feedback and validation. The CGL WG is actively working with recognized standard bodies, such as the Linux Standard Base (LSB – a workgroup of the Linux Foundation) and the Service Availability Forum (SA Forum). These organizations are producing standards and specifications that address the RAS and application portability gaps between Linux as it exists today and where it needs to be to support highly available communications applications.

The first requirement in this section shows the CGL working group's desire to work alongside recognized standards bodies:

CGL specifies the need for compliance to the Linux Standard Base (LSB) version 3.0 to ensure a CGL 5.0 distribution will have the support for the same level of the application binary compatibility as is required by the LSB standard.

CGL 5.0 requires implementation of the latest interface specifications from the SA Forum to provide a common set of standards and building blocks for high availability architectures and platform management. The SA Forum provides standards specifications that define interfaces for cluster-aware applications (Application Interface Specification - AIS version B.01.01) and for platform management applications (Hardware Platform Interface - HPI version B.01.01). See the SA Forum site (www.saforum.org) for the B.01.01 versions of the AIS and HPI specifications.

Continuing from previous versions of the CGL specifications, the CGL Standards Definition adds more POSIX compliance requirements based on IEEE Std 1003.1-2001. These additional areas of POSIX compliance are intended to bridge the application portability gaps as mainstream communications applications are ported to Linux application environments.

A variety of other standards requirements are included in the CGL Standards Definition to address the networking, communications, and platform needs of carrier environments. Standards requirements such as Stream Control Transfer Protocol (SCTP), Internet Protocols (Ipv4/IPv6), Mobile Internet Protocol (MIPv6), Simple Network Management Protocol (SNMP), Intelligent Platform Management Interface (IPMI), IEEE 801.Q (virtual LAN), Diameter, Common Information Model (CIM), Web-Based Enterprise Management (WBEM), Advanced Configuration and Power Interface (ACPI), and PCI Express, are included.

More open industry standards will become mature and recognized over time. The CGL working group will evaluate them for consideration in future versions of the CG requirements. The CGL working group believes that the adoption of open standards in mainline Linux offerings will benefit application developers and solution providers and will carry Linux to the next level of popularity in the communications industry as well as the general Linux user community.

STANDARDS REQUIREMENTS

STD.1.0 LINUX STANDARD BASE COMPLIANCE

ID	Name	Category	Priority
STD.1.0	Linux Standard Base Compliance	Standards	P1

CGL specifies that carrier grade Linux shall be compliant with the Linux Standard Base (LSB) 3.0 - http://www.linuxbase.org.The LSB 3.0 specification has been split into a generic LSB core, a generic module for C++, and a set of architecture specific modules. Required LSB 3.0 modules for CGL are:

- Generic LSB-Core
- Generic LSB-CXX
- For each supported architecture, one LSB-Core module and one LSB-CXX module

The developer may choose to implement more than one architecture platform. In this case, each supported architecture platform shall contain an implementation of at least one architecture specific LSB-Core module and one architecture specific LSB-CXX module.

STD.3.1 SCTP - BASE FEATURES

ID	Name	Category	Priority
STD.3.1	SCTP: Base Features	Standards	P1

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs below.

- RFC 2960 - The base standard for SCTP.
- RFC 3309 - An RFC that corrects a weakness in the original SCTP for very small packets.

STD.3.2.1 SCTP: ADDITIONAL FEATURES

ID	Name	Category	Priority
STD.3.2.1	SCTP: Additional Features	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs below:

- RFC **4460** - Stream Control Transmission Protocol (SCTP) Specification

STD.3.2.2 EXTENSIONS TO BSD SOCKETS TO SUPPORT SCTP

ID	Name	Category	Priority
STD.3.2.2	Extensions to BSD Sockets to support SCTP	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the Internet draft below:

- draft-ietf-tsvwg-sctpsocket-13.txt

Carrier Grade Linux Standards Requirements Definition Version 4.0

STD.3.2.3 RFC 3873 MIB FOR SCTP

ID	Name	Category	Priority
STD.3.2.3	RFC 3873 MIB for SCTP	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the Internet draft below.

- RFC 3873, MIB for SCTP

STD.3.2.4 EXTENSION FOR ADDING IP ADDRESSES TO SCTP ASSOCIATION

ID	Name	Category	Priority
STD.3.2.4	Extension for adding IP addresses to SCTP association	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the Internet draft below:

- draft-ietf-tsvwg-addip-sctp-15.txt: An extension to SCTP that allows adding and removing IP addresses to an existing SCTP association. This extension is needed to allow for associations that last longer than expiring IPv6 addresses.

STD.3.2.5 RFC 3758 PARTIAL RELIABILITY

ID	Name	Category	Priority
STD.3.2.5	RFC 3758 Partial reliability	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFC below:

- RFC 3758 - An extension to SCTP allowing for partial reliability. Introduces a mechanism for canceling messages no longer worth sending.

STD.3.2.6 SCTP THREATS

ID	Name	Category	Priority
STD.3.2.6	SCTP Threats	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the Internet draft below:

- draft-ietf-tsvwg-sctpthreat-02.txt: Documents additional security issues that implementers need to address.

STD.4.1 IPV6 BASE FEATURES

ID	Name	Category	Priority
STD.4.1	IPv6 Base Features	Standards	P1

CGL specifies that carrier grade Linux shall provide the IPv6 functionality listed in the RFCs below:

- RFC 2460: IPv6 Specification
- RFC 2463: ICMPv6 for IPv6 Specification
- RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)
- RFC 2462: IPv6 Stateless Address Autoconfiguration
- RFC 1981: Path MTU Discovery for IP version 6
- RFC 3493: Basic Socket Interface Extensions for IPv6
- RFC 3542: Advanced Sockets Application Program Interface (API) for IPv6
- RFC 3587: Global Unicast IPv6 Address Format
- RFC 2710: Multicast Listener Discovery for IPv6
- RFC 3810: Multicast Listener Discovery Version 2

STD.4.2.1 IPV6 ADDITIONAL FEATURES: RFC 2451 CIPHERS

ID	Name	Category	Priority
STD.4.2.1	IPv6 Additional Features: RFC 2451 Ciphers	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 2451: The ESP CBC-Mode Cipher Algorithms

STD.4.2.2 IPV6 ADDITIONAL FEATURES: RFC 4213/2893 TUNNELS

ID	Name	Category	Priority
STD.4.2.2	IPv6 Additional Features: RFC 4213/2893 Tunnels	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4213 which replaces
- RFC 2893: Transition Mechanisms for IPv6 Hosts and Routers (IPv6 over IPv4 Tunnel)

STD.4.2.3 IPV6 ADDITIONAL FEATURES: RFC 3484 DEFAULT ADDRESS SELECTION

ID	Name	Category	Priority
STD.4.2.3	IPv6 Additional Features: RFC 3484 Default Address Selection	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 3484: Default Address Selection for Internet Protocol version 6 (IPv6).

STD.4.2.4 IPV6 ADDITIONAL FEATURES: RFC 3315 DYNAMIC HOST CONFIGURATION

ID	Name	Category	Priority
STD.4.2.4	IPv6 Additional Features: RFC 3315 Dynamic Host Configuration	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below: • RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6).			

STD.4.2.5 IPV6 ADDITIONAL FEATURES: RFC 3633 PREFIX OPTIONS FOR DYNAMIC HOST CONFIGURATION PROTOCOL

ID	Name	Category	Priority
STD.4.2.5	IPv6 Additional Features: RFC 3633 Prefix Options for Dynamic Host Configuration Protocol	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below: • RFC 3633: IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6			

STD.4.2.6 IPV6 ADDITIONAL FEATURES: RFC 4191 DEFAULT ROUTER PREFERENCES

ID	Name	Category	Priority
STD.4.2.6	IPv6 Additional Features: RFC 4191 Default Router Preferences	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4191: Default Router Preferences, More-Specific Routes, and Load Sharing

STD.4.2.7 IPV6 ADDITIONAL FEATURES: RFC 2428 FTP EXTENSIONS

ID	Name	Category	Priority
STD.4.2.7	IPv6 Additional Features: RFC 2428 FTP Extensions	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 2428: FTP Extensions for IPv6 and NATs

STD.4.2.8 IPV6 ADDITIONAL FEATURES: RFC 3596 DNS EXTENSIONS

ID	Name	Category	Priority
STD.4.2.8	IPv6 Additional Features: RFC 3596 DNS Extensions	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 1886: DNS Extensions to support IP version 6
- RFC 3596: DNS Extensions to support IP version 6

STD.4.2.9 IPV6 ADDITIONAL FEATURES: RFC 2874 DNS ADDRESS AGGREGATION AND RENUMBERING

ID	Name	Category	Priority
STD.4.2.9	IPv6 Additional Features: RFC 2874 DNS Address Aggregation and Renumbering	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering

STD.4.2.10 IPV6 ADDITIONAL FEATURES: RFC 3646 DNS OPTIONS FOR DHCP

ID	Name	Category	Priority
STD.4.2.10	IPv6 Additional Features: RFC 3646 DNS options for DHCP	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 3646: DNS options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

STD.4.2.13 IPV6 ADDITIONAL FEATURES: NFS

ID	Name	Category	Priority
STD.4.2.13	IPv6 Additional Features: NFS	Standards	P2

CGL specifies that carrier grade Linux shall provide support for IPv6-based NFS.

STD.5.1 IPSEC MAJOR CGL FEATURES

ID	Name	Category	Priority
STD.5.1	IPSec Major CGL Features	Standards	P1

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs below.

- RFC 2367: PF_KEY Key Management API, Version 2
- RFC 2401: Security Architecture for the Internet Protocol
- RFC 2402: IP Authentication Header
- RFC 2406: IP Encapsulating Security Payload (ESP)
- RFC 2403: The Use of HMAC-MD5-96 within ESP and AH
- RFC 2404: The Use of HMAC-SHA -1-96 within ESP and AH
- RFC 2405: The ESP DES-CBC Cipher Algorithm With Explicit IV
- RFC 2409: Support for IKE daemon
- RFC 2410: The NULL Encryption Algorithm and Its Use With Ipsec
- RFC 2451: The ESP CBC-Mode Cipher Algorithms

STD.5.2.1 IPSEC MINOR CGL FEATURES: RFC 4301 SECURITY ARCHITECTURE FOR IP

ID	Name	Category	Priority
STD.5.2.1	IPSec Minor CGL Features: RFC 4301 Security Architecture for IP	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4301: Security Architecture for the Internet Protocol

STD.5.2.2 IPSEC MINOR CGL FEATURES: RFC 4302 IP AUTHENTICATION HEADER

ID	Name	Category	Priority
STD.5.2.2	IPSec Minor CGL Features: RFC 4302 IP Authentication Header	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below: • RFC 4302: IP Authentication Header			

STD.5.2.3 IPSEC MINOR CGL FEATURES: RFC 4303 IP ENCAPSULATING SECURITY PAYLOAD

ID	Name	Category	Priority
STD.5.2.3	IPSec Minor CGL Features: RFC 4303 IP Encapsulating Security Payload	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below: • RFC 4303: IP Encapsulating Security Payload (ESP)			

STD.5.2.4 IPSEC MINOR CGL FEATURES: RFC 4305 CRYPTOGRAPHIC ALGORITHM REQUIREMENTS

ID	Name	Category	Priority
STD.5.2.4	IPSec Minor CGL Features: RFC 4305 Cryptographic Algorithm Requirements	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4305: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)

STD.5.2.5 IPSEC MINOR CGL FEATURES: RFC 4307 CRYPTOGRAPHIC ALGORITHMS FOR USE IN IKE

ID	Name	Category	Priority
STD.5.2.5	IPSec Minor CGL Features: RFC 4307 Cryptographic Algorithms for Use in IKE	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4307: Cryptographic Algorithms for Use in the Internet Key Exchange Version 2

STD.5.2.6 IPSEC MINOR CGL FEATURES: RFC 4322 OPPORTUNISTIC ENCRYPTION USING IKE

ID	Name	Category	Priority
STD.5.2.6	IPSec Minor CGL Features: RFC 4322 Opportunistic Encryption using IKE	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4322: Opportunistic Encryption using the Internet Key Exchange (IKE) -- This document is not part of the basic set of standards required to support IPSec, but is useful if a customer wants to set up IPSec tunnels without coordinating with the administrators at the other end of the tunnels.

STD.5.2.7 IPSEC MINOR CGL FEATURES: RFC 4434 AES ALGORITHM FOR IKE

ID	Name	Category	Priority
STD.5.2.7	IPSec Minor CGL Features: RFC 4434 AES Algorithm for IKE	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs and internet drafts below:

- RFC 4434: The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)

STD.6.1 MIPV6 CGL MAJOR FEATURES

ID	Name	Category	Priority
STD.6.1	MIPv6 CGL Major Features	Standards	P1

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFC below.

- RFC 3775: Mobility Support in IPv6

STD.6.2 MIPV6 MINOR CGL FEATURES

ID	Name	Category	Priority
STD.6.2	IPv6 Minor CGL Features	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality listed in the RFCs below.

- RFC 3776: Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents.

STD.7.1 SNMP V1, V2, V3

ID	Name	Category	Priority
STD.7.1	SNMP v1, v2, v3	Standards	P1

CGL specifies that carrier grade Linux shall provide SNMPv1, SNMPv2, and SNMPv3 functionality as defined in the RFCs listed below.

- SNMPv1 - RFC 1155 through 1157
- Community-based SNMPv2 - RFCs 1901 through 1908
- SNMPv3 - RFC 2571 through 2575

STD.7.2 SNMP MIBS FOR IPV6/IPV4

ID	Name	Category	Priority
STD.7.2	SNMP MIBs for IPv6/IPv4	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality for the SNMP IPv6/IPv4 MIBs as defined by the RFCs listed below:

- RFC 3411 SNMP-FRAMEWORK-MIB.txt
- RFC 3412 SNMP-MPD-MIB.txt
- RFC 3413 SNMP-TARGET-MIB.txt, SNMP-NOTIFICATION-MIB.txt, SNMP-PROXY-MIB.txt
- RFC 3414 SNMP-USER-BASED-SM- MIB.txt
- RFC 3415 SNMP-VIEW-BASED-ACM- MIB.txt
- RFC 2576 SNMP-COMMUNITY -MIB.txt
- RFC 2578 SNMPv2-SMI.txt
- RFC 2579 SNMPv2-TC.txt
- RFC 2580 SNMPv2-CONF.txt
- RFC 3417 SNMPv2-TM.txt
- RFC 3418 SNMPv2-MIB.txt
- RFC 2742 AGENTX-MIB.txt
- RFC 1227 SMUX-MIB.txt
- RFC 3231 DISMAN-SCHEDULE-MIB.txt
- RFC 3165 DISMAN-SCRIPT-MIB.txt
- RFC 2863 IF-MIB.txt
- RFC 2864 IF-INVERTED-STACK-MIB.txt
- RFC 2856 HCNUM-TC.txt

ID	Name	Category	Priority
• RFC 3291 INET-ADDRESS-MIB.txt • RFC 2665 EtherLike-MIB.txt • RFC 2011 IP-MIB.txt • RFC 2096 IP-FORWARD-MIB.txt • RFC 2012 TCP-MIB.txt • RFC 2013 UDP -MIB.txt • RFC 2465 IPV6-TC.txt IPV6-MIB.txt • RFC 2466 IPV6-ICMP-MIB.txt • RFC 2452 IPV6-TCP-MIB.txt • RFC 2454 IPV6-UDP-MIB.txt • RFC 2790 HOST-RESOURCES-MIB.txt, HOST-RESOURCES-TYPES.txt • RFC 2819 RMON-MIB.txt • RFC 2788 NETWORK -SERVICES- MIB.txt • RFC 2789 MTA -MIB.txt • RFC 1155 -SMI.txt • RFC 1213 -MIB.txt **Note:** There is currently an ongoing effort within IETF to combine IPv4 and IPv6 MIBs into unified MIBs. The developer may choose to implement RFC 2011, RFC 2466.			

STD.8.1 SA FORUM AIS

ID	Name	Category	Priority
STD.8.1	SA Forum AIS http://www.saforum.org	Standards	P2
CGL specifies that carrier grade Linux shall provide the APIs as defined by the SA Forum AIS Release 5 or a subsequent level of the relevant AIS specification			

STD.8.8 SA FORUM HPI

ID	Name	Category	Priority
STD.8.8	SA Forum HPI http://www.saforum.org	Standards	P1
CGL specifies that carrier grade Linux shall provide the functionality defined in the SA Forum HPI B.02.01 specification or a subsequent level of the relevant HPI specification.			

STD.9.0 IPMI

ID	Name	Category	Priority
STD.9.0	IPMI	Standards	P1
CGL specifies that carrier grade Linux shall provide the System Management Software (SMS) functionality to interface with the below-listed levels of the Intelligent Platform Management Interface (IPMI): • IPMI v1.5 specification • IPMI v2.0 specification			

STD.10.0 802.1Q VLAN ENDPOINT

ID	Name	Category	Priority
STD.10.0	802.1Q VLAN Endpoint	Standards	P1

CGL specifies that carrier grade Linux shall provide the functionality defined in the IEEE Std 802.1Q-1998 specification. This standard defines the operation of virtual LAN (VLAN) endpoints that permit the definition, operation and administration of Virtual LAN topologies within a LAN infrastructure.

http://www.ieee802.org/1/pages/802.1Q.html

STD.11.1 DIAMETER PROTOCOL CGL MAJOR FEATURES

ID	Name	Category	Priority
STD.11.1	Diameter Protocol CGL Major Features	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality defined in the following RFCs and Internet drafts.

- RFC 3588 (Diameter Base Protocol)
- draft-ietf-eap-rfc2284bis-07.txt
- draft-ietf-aaa-eap-03.txt

STD.11.2 DIAMETER PROTOCOL MINOR CGL FEATURES

ID	Name	Category	Priority
STD.11.2	Diameter Protocol Minor CGL Features	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality defined in the following Internet drafts.

- RFC 4004 Diameter Mobile IPv4 Application

STD.17.1 ISCSI SUPPORT: RFC 3270 ISCSI

ID	Name	Category	Priority
STD.17.1	iSCSI Support: RFC 3270 iSCSI	Standards	P1

CGL specifies that carrier grade Linux shall provide support for Internet Small Computer Systems Interface (iSCSI) Initiators. The iSCSI Initiators shall support IPv6, SNMP MIBs, error handling, target discovery, and multiple sessions. This functionality is defined in the following RFCs:

- RFC 3720 - Internet Small Computer Systems Interface (iSCSI)reqs, determine which are P1

STD.17.2 ISCSI SUPPORT: RFC 3271 ISCSI NAMING & DISCOVERY

ID	Name	Category	Priority
STD.17.2	iSCSI Support: RFC 3271 iSCSI Naming & Discovery	Standards	P1

CGL specifies that carrier grade Linux shall provide support for Internet Small Computer Systems Interface (iSCSI) Initiators. The iSCSI Initiators shall support IPv6, SNMP MIBs, error handling, target discovery, and multiple sessions. This functionality is defined in the following RFCs:

- RFC 3721 - Internet Small Computer Systems Interface (iSCSI) Naming and Discovery

STD.17.3 iSCSI Support: RFC 3273 iSCSI Securing Block Storage Protocols over IP

ID	Name	Category	Priority
STD.17.3	iSCSI Support: RFC 3273 iSCSI Securing Block Storage Protocols over IP	Standards	P1

CGL specifies that carrier grade Linux shall provide support for Internet Small Computer Systems Interface (iSCSI) Initiators. The iSCSI Initiators shall support IPv6, SNMP MIBs, error handling, target discovery, and multiple sessions. This functionality is defined in the following RFCs:

- RFC 3723 - Securing Block Storage Protocols over IP

STD.18.1 Differentiated Services: RFC 2474 Definition

ID	Name	Category	Priority
STD.18.1	Differentiated Services: RFC 2474 Definition	Standards	P2

CGL specifies that carrier grade Linux shall provide support for differentiated services for IPv4 protocol as defined by the RFCs below. Differentiated services provide network traffic with different levels of service to enable quality of service and traffic control.

- RFC 2474 - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers

STD.18.2 DIFFERENTIATED SERVICES: RFC 2475 DEFINITION

ID	Name	Category	Priority
STD.18.2	Differentiated Services: RFC 2475 Definition	Standards	P2

CGL specifies that carrier grade Linux shall provide support for differentiated services for IPv4 protocol as defined by the RFCs below. Differentiated services provide network traffic with different levels of service to enable quality of service and traffic control.

- RFC 2475 - An Architecture for Differentiated Services

STD.20.1 PKI CA: RFC 2527 X.509 PKI

ID	Name	Category	Priority
STD.20.1	PKI CA: RFC 2527 X.509 PKI	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality for private key infrastructure (PKI) support as defined in the standards:

- RFC 2527 – Internet X.509 Public Key Infrastructure

STD.20.2 PKI CA: RFC 2527 X.509 PKI PROTOCOLS FTP AND HTTP

ID	Name	Category	Priority
STD.20.2	PKI CA: RFC 2527 X.509 PKI	Standards	P2

CGL specifies that carrier grade Linux shall provide the functionality for private key infrastructure (PKI) support as defined in the standards:

- RFC 2585 – Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP

STD.20.3 PKI CA: RFC 3279 ALGORITHMS FOR X.509 PKI

ID	Name	Category	Priority
STD.20.3	PKI CA: RFC 3279 Algorithms for X.509 PKI	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality for private key infrastructure (PKI) support as defined in the standards: RFC 3279 - Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure			

STD.20.4 PKI CA: RFC 3280 X.509 PKI CERTIFICATE STUFF

ID	Name	Category	Priority
STD.20.4	PKI CA: RFC 3280 X.509 PKI Certificate Stuff	Standards	P2
CGL specifies that carrier grade Linux shall provide the functionality for private key infrastructure (PKI) support as defined in the standards: RFC 3280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile			

STD.26.1 LAYER 2 TUNNELING PROTOCOL SUPPORT

ID	Name	Category	Priority
STD.26.1	Layer 2 Tunneling Protocol Support	Standards	P1
CGL specifies that carrier grade Linux shall provide support for Layer 2 Tunneling Protocol (L2TP) as described in RFC 2661: Layer Two Tunneling Protocol "L2TP".			

STD.26.2 LAYER 2 TUNNELING PROTOCOL SUPPORT VERSION 3

ID	Name	Category	Priority
STD.26.2	Layer 2 Tunneling Protocol Support Version 3	Standards	P1
CGL specifies that carrier grade Linux shall provide support for Layer 2 Tunneling Protocol (L2TP) as described in RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3).			

9. HARDWARE REQUIREMENTS DEFINITION

To stay competitive and profitable in the telecommunication industry, standards-based, modular, commercial-off- the-shelf (COTS) hardware components are being used along with open software, including operating systems, middleware, and applications. A goal of the CGL working group is to promote the migration of the telecommunication industry from the proprietary hardware platforms to COTS hardware by insuring that the Linux environment provides adequate support for these COTS platforms. The CGL Hardware Requirements Definition – Version 4.0 identifies a set of widely-used industry hardware platforms and defines the support that is needed in the operating system for these platforms. The scope of these hardware requirements applies to the Linux kernel, kernel interfaces (APIs and libraries), system software, and tools.

This section specifies a set of generic requirements that are common across platform types. It includes support for blade servers, for hardware management interfaces, and for blade hot swap events. To address the need to manage highly available carrier grade systems through hardware out-of-band mechanisms, management capabilities such as those found in the Intelligent Platform Management Interface (IPMI) are also described.

Carrier-grade systems require high performance and high throughput interconnections within a system and between system nodes. Hardware-related requirements, such as PCI Express support, and PCI Express Device Hot Plug, are included. Other hardware related requirements such as a CPU throttle mechanism, iSCSI Initiator Support”, and “iSCSI Target Discovery” are also specified.

Considering the diversity of hardware platforms used in a carrier grade environment, the CGL Hardware Requirements Definition - Version 4.0 does not define requirements for just one type of industry platform. Instead it defines generic platform requirements and then provides an “Industry Platforms” section to provide implementation guidelines for specific architectures. Examples of such industry platforms include AdvancedTCA, BladeCenter, CompactPCI and rack mount types of servers.

HARDWARE SUB-CATEGORIES

Requirement Sub-Category	Sub-Category Description
PLT	General Platform
PIC	Platform Interconnect
PMT	Platform Management
PMS	Platform Miscellaneous

HARDWARE REQUIREMENTS

PMS.1.0 CPU THROTTLE

ID	Name	Category	Priority
PMS.1.0	CPU Throttle	Hardware	P2
CGL specifies that carrier grade Linux shall provide a CPU power consumption management capability that enables adjustment of the CPU frequency. Any power, voltage and frequency settings shall be within the allowed range for the hardware.			

PMS.5.1 ISCSI INITIATOR SUPPORT

ID	Name	Category	Priority
PMS.5.1	iSCSI Initiator Support	Security	P1
CGL specifies that carrier grade Linux shall support the iSCSI protocol to enable block level access to SCSI storage devices using the TCP/IP transport. The support shall be compliant with the RFC 3270 specification and should provide iSCSI initiator support. At a minimum the supported iSCSI initiators should be able to authenticate themselves to potential iSCSI targets using the two-way CHAP authentication algorithm. See STD.17.0 iSCSI.			

PMS.5.3 ISCSI TARGET DISCOVERY

ID	Name	Category	Priority
PMS.5.3	iSCSI Target Discovery	Security	P1
CGL specifies that the iSCSI Initiators implemented by carrier grade Linux shall support the SendTargets Discovery mechanism to discover potential iSCSI targets they can connect. See STD.17.0 iSCSI.			

HARDWARE REFERENCES

This section provides background information for some of the hardware referred to in this specification.

- Intelligent Platform Management Interface (IPMI) Specifications: http://developer.intel.com/design/servers/ipmi
- PCI Express at the PCI-SIG web site: http://www.pcisig.com/
- Intel® Developer Network for PCI Express Architecture: http://www.express-lane.org
- Advanced Switching (ASI-SIG web site): http://www.asi-sig.com/
- Rapid I/O: http://www.rapidio.org

- Advanced Configuration and Power Interface (ACPI): http://www.acpi.info/

10. SECURITY REQUIREMENTS DEFINITION

The telecommunications environment is different from a general-purpose computing environment. The most salient differences to consider in developing a CGL threat model are:

- CGL systems do not have many user accounts.
- User accounts do not reflect individual users.
- CGL systems are configured through custom user interfaces.
- CGL systems are typically configured without shell access.
- Administrators are trusted and competent.

The major threat to the telecommunications environment is, therefore, unauthorized access to management and control interfaces by outsiders. These outsiders can gain access by subverting the operating system or one of the applications it is running.

A severe potential security threat arises when applications need to touch multiple security planes. Many telecommunication services can be provisioned remotely by the end-user.

Many ISPs that offer domain hosting allow customers to create new mailboxes or route incoming calls to 5-digit work extensions to any telephone number in the world with just a few clicks on a web page. Facilities like these create a new set of risks:

- Unauthorized rerouting of email and telephone calls by disgruntled associates or unscrupulous competitors.
- Exploitation of vulnerabilities in software to “jump” from one security plane to another, which can lead to many types of risks.

Mitigating these risks will require some forethought such that users of these systems are properly authenticated and authorized and that information traveling between planes passes through narrowly defined interfaces that protect against unauthorized access.

SECURITY DESIGN

The security objectives and requirements in this document are aimed at analyzing and mitigating threats and improving resiliency to attacks on CGL systems. The requirements in this section attempt to implement security objectives for CGL systems and are based on an intersection of assumptions about CGL systems:

- Intended use
- Environment
- Security policies
- Exposure to expected threats and vulnerabilities

The security requirements are firmly rooted in sound security practices. These practices and terminology borrow heavily from [CSPP-OS03], an example Common Criteria profile for common off the shelf (COTS) operating systems.

Given the environment described in the previous section, the significant threat to carrier grade systems is unauthorized access to management and control interfaces by intruders.

The CGL Security Requirements have been based upon the Common Criteria Protection Profiles:

- Identify the assumptions about CGL systems based upon their use and their environment.
- Draft a set of security policies to which CGL systems shall adhere.
- Identify common threats to which CGL systems are exposed.
- Derive the set of functional objectives that CGL systems shall implement.
- Derive a coherent set of requirements that address the functional objectives.

DESIGN OBJECTIVES

This section identifies the security objectives met by the requirements in this specification. A more complete list from which these security objectives were

taken is found in section 10.7. A Target of Evaluation (TOE) is the system and environment to which these objectives are applied.

The following table specifies the security objectives met by requirements listed in section of this document.

Security Objective	**Description**
O.DETECT-SOPHISTICATED	The environment must provide the ability to detect sophisticated attacks and the results of such attacks (e.g. corrupted system state).
O.ENTRY-NON-TECHNICAL	The environment must provide sufficient protection against non-technical attacks by other than authenticated users.
O.PHYSICAL	Those responsible for the system must ensure that those parts of the system critical to security policy are protected from physical attack that might compromise security.
O.ACCESS-TOE	The system must provide public access and access by authenticated users to those resources and actions for which they have been authorized.
O.ACCOUNT-TOE	The system must ensure, for actions under its control or knowledge, that all users can subsequently be held accountable for their security relevant actions. It is anticipated that individual accountability might not be achieved for some actions.
O.AUTHORIZE-TOE	The system must provide the ability to specify and manage user and system process access rights to individual processing resources and data elements under its control, supporting the organization’s security policy for access control.

O.BYPASS-TOE	The system must prevent errant or non-malicious, authorized software or users from bypassing or circumventing security policy enforcement. NOTE: This objective is limited to 'non-malicious' because CSPP-OS controls are not expected to provide sufficient mitigation for the greater negative impact that 'malicious' implies.
O.DETECT-TOE	The system must enable the detection of a specified set of vulnerabilities.
O.ENTRY-TOE	The system must prevent logical entry to itself using unsophisticated technical methods by persons without authority for such access.
O.KNOWN-TOE	The system must ensure that, for all actions under its control and except for a well-defined set of allowed actions, all users are identified and authenticated before being granted access.
O.OBSERVE-TOE	The system must ensure that its security status is not misrepresented to the administrator or user. This is a combination of prevention and detection.
O.RESOURCES	The system must protect itself from user or system errors that result in shared resource exhaustion.
O.APPLICATION-TOOLS	The system must provide a reasonable, up-to-date set of security tools and libraries for use by applications.
O.ACCESS-MALICIOUS	System and environmental controls are required to sufficiently mitigate the threat of malicious actions by authenticated users.
O.DETECT-SYSTEM	The system, in conjunction with other entities in the environment, must enable the detection of system insecurities.
O.NETWORK	The system must be able to meet its security objectives in a distributed environment.

O.ENTRY-SOPHISTICATED	The system and environment must sufficiently mitigate the threat of an individual (other than an authenticated user) gaining unauthorized access via sophisticated, technical attack.
O.CONTAINMENT	The system and environment must provide the ability to contain the effect of a security failure of an application to that application.

The following table specifies the security objectives not met by requirements in section of this document.

Security Objective	**Rational for not including in specification**
O.ACCESS-NON-TECHNICAL	The environment must provide sufficient protection against non-technical attacks by authenticated users for non-malicious purposes.
O.AVAILABLE-TOE	The system must protect itself from unsophisticated denial-of-service attacks.
O.INFO-FLOW	The environment must ensure that any information flow control policies are enforced between system components and at the system external interfaces.
O.RECOVER-TOE, O.RECOVER-SYSTEM	Fail-secure is not something that OSDL CGL can provide.
O.COMPLY	There are many regulations that might apply to OSDL CGL. It is not the responsibility of this specification to enumerate requirements to conform to this myriad of regulations.
O.DUE-CARE	It is the responsibility of the administrative personnel to properly secure and maintain a system.

O.MANAGE	It is the responsibility of administrative personnel to properly secure and maintain a system. This includes periodic audits of system configuration (not log analysis). However, no such software is being required by OSDL CGL.
O.OPERATE	Mostly this is the responsibility of administrative personnel. Secure default configuration settings will not be listed in this specification.
O.DENIAL-SOPHISTICATED	OSDL CGL is not directly able to mitigate most denial of service attacks, as mitigating them would require redesign of protocols and interfaces.

SECURITY REQUIREMENTS

SEC.1.1 DYNAMIC KERNEL SECURITY MODULE MECHANISM

ID	Name	Category	Priority
SEC.1.1	Dynamic Kernel Security Module Mechanism	Security	P1
CGL specifies that carrier grade Linux shall support an interface that allows the addition of new access control policy implementations to the kernel without requiring patching or recompilation. This support must allow for the dynamic loading of such policy implementations. The mechanism must govern all of the kernel objects. This requirement does not specify any particular policies.			

SEC.1.2 PROCESS CONTAINMENT USING FILE SYSTEM RESTRICTIONS

ID	Name	Category	Priority
SEC.1.2	Process Containment using File System Restrictions	Security	P1
CGL specifies that carrier grade Linux shall provide support for constraining the privileges and access to system resources of a process independently of the user account under which the process runs by limiting a process' access to a subset of the file system hierarchy. This limits the effects of a security compromise of a process (such as a buffer overflow exploit).			

SEC.1.3 PROCESS CONTAINMENT USING MAC-BASED MECHANISM

ID	Name	Category	Priority
SEC.1.3	Process Containment Using MAC-based Mechanism	Security	P1
CGL specifies that carrier grade Linux shall provide support for constraining the privileges and access to system resources of a process independently of the user account under which the process runs, using a mandatory access control (MAC) mechanism. This limits the effects of a security compromise of a process, such as a buffer overflow exploit, even if it running as root.			

SEC.1.3.1 MAC-BASED POLICY ADMINISTRATION TOOLS

ID	Name	Category	Priority
SEC.1.3.1	MAC-based Policy Administration Tools	Security	P2
CGL specifies that carrier grade Linux shall provide tools for the administration of MAC-based access control policies. These tools should facilitate the creation, maintenance, and management of policies. The tools should provide at least one of a command line or graphical interface.			

SEC.1.4 BUFFER OVERFLOW PROTECTION

ID	Name	Category	Priority
SEC.1.4	Buffer Overflow Protection	Security	P1
CGL specifies that carrier grade Linux shall provide at least one mechanism to protect against the exploitation of software bugs that exploit the lack of boundary checking in many programs and give an attacker some access to a task's address space by writing outside of buffer bounds.			

SEC.1.5 ACCESS CONTROL LIST SUPPORT FOR FILE SYSTEMS

ID	Name	Category	Priority
SEC.1.5	Access Control List Support for File Systems	Security	P1
CGL specifies that carrier grade Linux shall provide access control list (ACL) capabilities on file systems that allow the specification of access rights for multiple users and groups.			

SEC.2.1 GENERIC AUTHENTICATION MODULES

ID	Name	Category	Priority
SEC.2.1	Generic Authentication Modules	Security	P1
CGL specifies that carrier grade Linux shall support a mechanism for implementing new operating system authentication mechanisms. This support must allow for the dynamic loading of authentication modules.			

SEC.2.2 PASSWORD INTEGRITY CHECKING

ID	Name	Category	Priority
SEC.2.2	Password Integrity Checking	Security	P1
CGL specifies that carrier grade Linux shall provide tools to check passwords to ensure they cannot be cracked using common attack methods. These tools shall support at least the DES cipher text format and allow the user to specify rules for rejecting passwords.			

SEC.3.1 AUDITING

ID	Name	Category	Priority
SEC.3.1	Auditing	Security	P1
CGL specifies that carrier grade Linux shall provide auditing mechanisms that flag security-relevant events and alert a system administrator.			

SEC.3.2 SECURE TRANSPORT OF LOG INFORMATION

ID	Name	Category	Priority
SEC.3.2	Secure Transport of Log Information	Security	P1
CGL specifies that carrier grade Linux shall provide secure transport of log information over a network to the log files. The transport mechanism shall ensure that the information remains confidential, cannot be modified, is not a replay of an earlier log message, and originated at the source it claims.			

SEC.3.3 PERIODIC AUTOMATED LOG ANALYSIS

ID	Name	Category	Priority
SEC.3.3	Periodic Automated Log Analysis	Security	P1
CGL specifies that carrier grade Linux shall provide a mechanism for periodically and automatically analyzing log files. This mechanism shall be able to generate reports if any suspicious or unrecognized log entry is detected.			

SEC.3.4 ACTIVE LOG MONITORING

ID	Name	Category	Priority
SEC.3.4	Active Log Monitoring	Security	P1
CGL specifies that carrier grade Linux shall provide a mechanism for automatically analyzing security-relevant log information. This mechanism shall be able to generate alarms if criteria set by a system administrator are met.			

SEC.3.5 LOG INTEGRITY AND ORIGIN AUTHENTICATION

ID	Name	Category	Priority
SEC.3.5	Log Integrity and Origin Authentication	Security	P1
CGL specifies that carrier grade Linux shall provide a mechanism to check that log files have not been modified (integrity), even by most insiders. In addition, CGL specifies that carrier grade Linux shall provide a mechanism to verify the origin of a log message. CGL specifies that carrier grade Linux shall provide a mechanism to prevent replay attacks of a log message.			

SEC.4.1 IPSEC

ID	Name	Category	Priority
SEC.4.1	IPsec	Security	P1
CGL specifies that carrier grade Linux shall provide IPsec support for network level confidentiality and integrity. The implementation shall conform to RFC 2401, 2402, 2406 and at least one encapsulating security payload (ESP) algorithm such as specified by RFC 2451.			

SEC.4.2 IKE

ID	Name	Category	Priority
SEC.4.2	IKE	Security	P1
CGL specifies that carrier grade Linux shall provide an Internet Key Exchange (IKE) service to perform standards-based key exchange for IPsec. The service shall conform to RFC 2409.			

SEC.4.3 PF_KEY VERSION 2

ID	Name	Category	Priority
SEC.4.3	PF_KEY Version 2	Security	P1
CGL specifies that carrier grade Linux shall provide PF_KEY support, as defined by RFC 2367, for key management for the IPsec module and the IKE service.			

SEC.4.4 PKI SUPPORT FOR APPLICATIONS

ID	Name	Category	Priority
SEC.4.4	PKI Support for Applications	Security	P1
CGL specifies that carrier grade Linux shall provide basic PKI features, which shall conform to the IETF PKIX standards, specifically RFC 2527, 3279 and 3280. Support for processing certification revocation lists (CRLs) is required, although a specified delivery mechanism such as HTTP/FTP RFC 2585) is not specified.			

SEC.4.5 SSL/TLS SUPPORT FOR APPLICATIONS

ID	Name	Category	Priority
SEC.4.5	SSL/TLS Support for Applications	Security	P1
CGL specifies that carrier grade Linux shall provide basic SSL/TLS support, which shall conform to the legacy SSL and IETF TLS standards.			

SEC.4.6 PKI CERTIFICATE AUTHORITY (CA)

ID	Name	Category	Priority
SEC.4.6	PKI Certificate Authority (CA)	Security	P1
CGL specifies that carrier grade Linux shall provide a basic PKI CA service. This service shall conform to the IETF PKIX standards, specifically RFC 2527, RFC 3279 and 3280. Support for the management of certification revocation lists (CRLs) is required. Certificate management and request protocols as defined by RFC 2527 3279, and 3280, are not requirements.			

SEC.5.1 PERIODIC USER-LEVEL FILE INTEGRITY CHECKING

ID	Name	Category	Priority
SEC.5.1	Periodic User-Level File Integrity Checking	Security	P1
CGL specifies that carrier grade Linux shall provide a mechanism to enable a periodic checking of the integrity of files at user-level. Files to be checked are both binary files, which should not change after installation, and text files, such as configuration and log files, which may change. File integrity checks shall be able to be scheduled at any time of the day. The checking mechanism shall be able to send alarms to a system administrator when inconsistencies are detected.			

SEC.7.1 MEMORY LIMITS

ID	Name	Category	Priority
SEC.7.1	Memory Limits	Security	P1
CGL specifies that carrier grade Linux shall provide support for per-process limits for the use of system memory.			

SEC.7.2 FILE SYSTEM QUOTAS

ID	Name	Category	Priority
SEC.7.2	File System Quotas	Security	P1
CGL specifies that carrier grade Linux shall provide support for per-user file system quotas.			

SEC.7.3 PROCESS QUOTAS

ID	Name	Category	Priority
SEC.7.3	Process Quotas	Security	P1
CGL specifies that carrier grade Linux shall provide support for per-user quotas on the number of processes which may be created.			

SEC.8 TRUSTED PLATFORM MODULE (TPM) SUPPORT

ID	Name	Category	Priority
SEC.8	Trusted Platform Module (TPM) Support	Security	P2
CGL specifies that, if and only if it is installed and executing on a TPMenabled platform, carrier grade Linux shall provide OS support for the TPM hardware, as defined in TCG TPM Specification, version 2.			

SEC.9.1 ROLE-BASED ACCESS CONTROL

ID	Name	Category	Priority
SEC.9.1	Role-Based Access Control	Security	P1
CGL specifies that carrier grade Linux shall provide a mechanism to associate a name with a set of privileges and commands to be executed, defining a role within the system. It must be possible to assign a list of authorized users to a role, to remove users from a role and to log and audit actions performed within the role. Each role must have a symbolic name and be able to be uniquely identified within the system.			

SEC.9.2 ADVANCED ROLE-BASED ACCESS CONTROL

ID	Name	Category	Priority
SEC.9.2	Advanced Role-Based Access Control	Security	P2
CGL specifies that carrier grade Linux shall implement the Common Criteria Role-Based Access Control protection profile, version 1.0.			

SEC.10 TAMPER-RESISTANT STORAGE

ID	Name	Category	Priority
SEC.10	Tamper-Resistant Storage	Security	P2
CGL specifies that carrier grade Linux shall provide secure, tamper-resistant storage for security-relevant data such as keys and certificates. It must be possible for both kernel and user space to request validation of such data and to receive an assessment whether such data has been modified either via the operating system or some external source.			

SEC.11.1 FILE ACCESS TRACING

ID	Name	Category	Priority
SEC.11.1	File Access Tracing	Security	P1

CGL specifies that carrier grade Linux shall provide the ability to record and report, via the normal system event reporting mechanism, file access events. At least the following file access events must be recorded and reported:

- File open
- File close
- File read
- File write
- File deletion
- File attribute changes

The reports must at least include the event that is being recorded and some uniquely identifiable information about the issuer of the operation.

SEC.11.2 FILE ACCESS TRACING: LIMITING

ID	Name	Category	Priority
SEC.11.2	File Access Tracing: Limiting	Security	P2

CGL specifies that carrier grade Linux shall provide the ability to record and report file access events. It must be possible to include or exclude arbitrary files and/or directory hierarchies from the file access tracing and the types of events that shall be logged.

SECURITY DESIGN PRINCIPLES

Principle	Description
Relevance	The requirement must be relevant and implement the function CGL objectives.
Correctness of Implementation	The requirement must faithfully implement the security model upon which it is based.
Simplicity	The requirements should be simple to implement. Complexity is the enemy of security. Common uses should be easy to handle and defaults should be sensible.
Robustness	The implementations of the requirements should be difficult to configure incorrectly, fail in secure ways, and produce useful error messages.
Orthogonality	Requirements should be useful individually without significant overlap in functionality.
Interface Stability	Changes and additions to the Linux APIs should be done with backward compatibility in mind for both source code and binary code.
Provision of Defense-in-Depth	Multiple security mechanisms should exist to provide additional security protection.
Designed for Testing	A test suite should be provided for unit testing of the requirement implementations.

ITU-T RECOMMENDATION X.805 ET. AL.

The International Telecommunications Union (ITU) has published many standards that are relevant to the security of telecommunications systems. The specification defers to the ITU standards for telecommunications-specific security requirements. The CGL Security Requirements Definition is limited to issues relating to security of the underlying operating system.

THE X.805 SECURITY FRAMEWORK

X.805 defines security in terms of two major concepts which are layers and planes.

The three layers are:

1. **Infrastructure** - security of routers, switches, servers, communication links, etc.
2. **Services** - security of services offered to the customer, such as leased lines, e-mail, SMS.
3. **Application** - security of customer applications using services.

The three planes are:

1. **Management** - security of OAM&P
2. **Control** - security of signaling, i.e. Session creation and modification
3. **End-**user - security of end-user data flows

Layers and planes intersect, forming a 3 by 3 matrix. Orthogonal to this, X.805 defines eight security dimensions:

- Privacy and data confidentiality
- Authentication
- Integrity
- Non-repudiation
- Access Control
- Communication
- Availability

These dimensions touch each of the cells of the layers/planes matrix.

For brevity's sake, we refer to the definitions in [ITU03].

Many of the issues addressed by X.805 are not relevant to our analysis, because they are outside the scope of an operating system.

RISKS, THREATS, AND VULNERABILITIES

All discussion of security revolves around risk. Risks are created when a security vulnerability is combined with the threat of that vulnerability being exploited. In the common buffer overflow attack scenario vulnerability (the lack of input validation in the software) and a threat (the attacker using software that exploit the vulnerability), creates the risk of a successful attack. The risk can be mitigated in different ways. The vulnerability is removed by fixing the software. The vulnerability is also removed by preventing the attack.

Risks do not necessarily have to be mitigated in software, but that the environment in which a system is embedded can also mitigate them. This is an important point because it is nearly impossible to construct systems that are invulnerable to attack.

ALL SOFTWARE CONTAINS VULNERABILITIES

All software contains vulnerabilities and it is impractical to find and remove all of them in a system. Some methods for lowering the risks relating to vulnerabilities are:

- **Not exposing the system running the software to insecure networks.** This is practical for certain limited purposes, for instance controlling a power plant. In the CGL environment one could segregate network traffic from different security planes, which would eliminate the threat of intruders attacking software operating in the management and control plane.
- **Overflow detection through the use of programming languages and development tools.** One example is the **gcc** compiler using the stack protection (previously known as ProPolice) extension. Most stack buffer overflows will result in the premature termination of a program. This termination transforms the risk of a successful buffer overflow attack into a denial of service attack.
- **Limiting software privileges**. A common approach is the use of 'chroot' jails, a method of restricting a program's access to a very limited part of the file system. Another approach is the use of a security manager that decides whether an application is allowed to perform certain operations. A

common example is the Java sandbox which prevents access of applets to most system resources.

- **Restricting network access using a DMZ**. The application and the system running it may still be compromised, but the problem is somewhat contained.

The solution of many security problems will be a combination of the correct application of OS facilities, and a correct design of the environment in which the systems operate.

APPLICATIONS ACCESSING MULTIPLE PLANES

A particular issue exists where applications need to access multiple security planes. Many CGL services can be provisioned remotely by the end-user. Many ISPs that offer domain hosting allow the creation of new mailboxes by the customer. These facilities create new risks:

- Unauthorized rerouting of e-mail and telephone calls by disgruntled employees or unscrupulous competitors.
- Exploitation of vulnerabilities in software to 'jump' from one security plane to another.

Mitigating these risks requires forethought.

- The users of these systems need to be properly authenticated and authorized.
- Information traveling between planes should pass through narrowly defined interfaces that protect against unauthorized access to the control and management planes from the end-user plane. A security failure in an exposed part of the system should not result in failure of the system as a whole.

Facilities that limit information flow between planes are not commonly available. Possible approaches could be:

- Running software on multiple hosts, with very limited connectivity between them.
- Running multiple processes on the same host, using operating system facilities to contain each process in its own security domain.

PRIVILEGE MINIMIZATION AND FINE-GRAINED ACCESS CONTROLS

Unix-like systems such as Linux share a few common security facilities:

- Discretionary access control using user IDs, group IDs, and file system privileges.
- Restriction of processes to a portion of the filesystem.

Some Unix-like systems provide additional facilities which can be useful under certain circumstances, such as:

- Access Control Lists: Some access control policies are difficult to implement with the classical Unix access control mechanism. ACLs provide a more powerful mechanism to describe access rules. The lack of users on typical carrier grade equipment makes ACLs not overly useful.
- Role Based Access Control: Users of the system can be assigned 'roles' which grant privileges to resources. The role 'help desk' for example could include privileges to change passwords for non-administrative users. RBAC is most useful if there are many instances of the role. This is not commonly the case for CGL systems.

To mitigate risks precipitated by software design or implementation errors, CGL requires a much more fine-grained control over system privileges. The common way to handle programs that need certain privileges is to give them full privileges at start-up time and let the program drop all the privileges they don't need. This causes a few problems. The privileges that need to be dropped are not necessarily the same on all systems, and there becomes a proliferation of privilege-manipulation code on the system. Tools that allow the designer or administrator to start software with the minimal set of privileges is required.

Another issue is that Linux systems do not have a sufficiently fine-grained privilege model. For example, it is impossible to restrict the use of a specific IP address and/or port range to a limited number of processes. Ideally, it should be possible to allow a specific process to bind to port 80 (WWW) on a single interface. Multi-level security (MLS) implementations can be used to prohibit processes from accessing network interfaces they do not need to access.

SECURITY ENVIRONMENT

The following sections borrow heavily from [CSPP-OS03], an example Common Criteria profile for COTS operating systems.

TARGETS OF EVALUATION

Name	Assumption	Rationale
A.COTS	The TOE is constructed from near-term achievable off the shelf Linux technology.	This follows from the charter of CGL.
A.MALICIOUS-INSIDER	The TOE is not expected to be able to sufficiently mitigate the risks resulting from the malicious abuse of authorized privileges.	In CGL environments the primary threats are network-based attacks, so the focus is on this type of threat.
A.SOPHISTICATED-ATTACK	The TOE is expected to be able to mitigate risks resulting from the application of moderately sophisticated attack methods[1].	Internet-based CGL applications are subject to network-based attacks, and should be more resistant to attacks than general-purpose systems.
A.APPLICATION-HOSTILE	The network containing the TOE is used to provide a limited set of applications to an untrusted network, not to provide shell access to users at different trust levels.	Communications architectures are moving away from general-purpose computing to application servers in hostile environments.

1 Unlike the COTS draft CC profile.

ENVIRONMENT

Name	Assumption	Rationale
A.ADMIN	The security features of the TOE are competently administered on a continuous basis.	It is essential for security that administration is both competent and continuous.
A.ADMIN-ONLY	Authenticated access to the TOE is only provided to those charged with maintaining the TOE and the applications it provides.	CGL is not targeting general purpose computing.
A.USER-NEED	Authenticated users, such as administrators, recognize the need for a secure CGL environment.	Application administrators value security of applications which they maintain.
A.USER-TRUST	Authenticated users, such as administrators, are generally trusted to perform discretionary actions in accordance with security policies.	Access is restricted to administrators maintaining applications.
A.NET-SEGREGATION	Network connections in the management, control and end-user planes are adequately segregated. One approach is to use physically separate networks. Another approach is the use of cryptographic methods for authentication, integrity verification and data	The end user should not be able to gain access to either the control or management plane.

Name	Assumption	Rationale
	confidentiality.	
A.CLUSTER-SEGREGATION	If the TOE is part of a cluster the intra-cluster communications should be adequately segregated from any other traffic, either by physical separation or by the use of cryptographic methods for authentication, integrity verification and data confidentiality.	Results are likely to be disruptive if cluster traffic is tampered with or captured. For this reason, separate interconnect is preferable.
A.PROCESS-UNTRUSTED	Processes running on the TOE cannot always be trusted to perform their duties as designed, and may attempt to access resources it is not meant to access.	It is often impossible to run legacy code in restricted environments such as chroot jails. The TOE should support a safe way to run this type of code in such a way that program bugs or vulnerability exploits only have limited consequences.

ORGANIZATIONAL SECURITY POLICIES

Name	Policy	Rationale
P.ACCESS	Access rights to specific data objects are determined by object attributes assigned to that object, user identity, user attributes, and environmental conditions as defined by the security	Linux supports policies that grant or deny access to objects using rules driven by attributes of the user (such as user identity), attributes of the object (such as permission bits),

Name	Policy	Rationale
	policy.	type of access (such as read or write), and environmental conditions (such as time-of-day).
P.ACCOUNT	Users must be held accountable for security-relevant actions.	Organizational policies should require that users are held accountable for their actions. This facilities after-the-fact investigations and providing some deterrence to improper actions.
P.COMPLY	The implementation and use of the organization's CGL systems must comply with all applicable laws, regulations, and contractual agreements imposed on the organization.	The organization will meet all requirements imposed upon it from outside governmental or contractual obligations.
P.DUE-CARE	The organization's CGL systems must be implemented and operated in a manner that represents due care and diligence with respect to the risks to the organization.	It is important that the level of security afforded by the CGL system be in accordance with best practices within the business or government sector in which the organization is placed.
P.INFO-FLOW	Information flow between application components must be in accordance with established information flow policies.	This document includes information flow control as this is needed in many environments. While this might not be implemented by mechanisms within the

Name	Policy	Rationale
		Linux TOE, the CGL system, of which the TOE is a part, will likely have to meet this policy.
P.KNOWN	Except for well-defined set of allowed operations, users of the TOE must be identified and authenticated before TOE access is granted.	Beyond a well-defined set of actions such as read access to a public web-server, there is a finite community of known, authenticated users who are authenticated before being allowed access.
P.NETWORK	The organization's IT security policy must be maintained in the environment of distributed systems interconnected via insecure networking.	CGL system will likely connect through untrested networks and these connections should not compromise security of a CGL system.
P.PHYSICAL	The processing resources of the TOE that must be physically protected in order to ensure that security objectives are met will be located within controlled access facilities that mitigate unauthorized, physical access.	A TOE will not be able to meet its security requirements unless at least a minimum degree of physical security is provided.
P.SURVIVE	The IT system, in conjunction with its environment, must resist, be resilient to, and detect a security breach and recover from the breach	Linux systems will provide a measure of their resilience through functionality and assurances that resist,

Name	Policy	Rationale
	when possible.	detect, and recover from security breaches. For sophisticated attacks, a large portion of this resilience is provided by the TOE environment.
P.TRAINING	Authenticated user of the system must be adequately trained. This enables the users to effectively implement organizational security policies with respect to their discretionary actions. It also supports the need for non-discretionary controls implemented to enforce these policies.	Once granted legitimate access, authenticated users are expected to use CGL resources and information only in accordance with the organizational security policy. In order for this to be possible, these users must be adequately trained both to understand the purpose and need for security controls and to be able to make secure decisions with respect to their discretionary actions.
P.USAGE	The organization's IT resources must be used only for authorized purposes.	Linux systems must, in conjunction with its environment, ensure that the organization's information technology is only used for authorized purposes.

Name	Policy	Rationale
P.CONTAINMENT	The TOE must be able to mitigate the risks of common threats to the integrity of applications and data caused by security-relevant errors in applications.	Linux systems should limit the damage done by buffer overflows and other common attacks. This is achieved through privilege minimization and process containment mechanisms such as jails.
P.PRIVILEGE-MIN	The TOE must be able to run applications with a minimal set of necessary privileges.	Linux systems should allow granting of privileges on a need-only basis. The nothing-or-everything model of 'root' privileges is not acceptable.
P.NET-SEGREGATION	The TOE must be configured to provide adequate segregation between the management, control and end-user planes, using separate networks, cryptographic methods, or both.	As per the requirements in X.805, the planes should be adequately segregated.
P.CLUSTER-SEGREGATION	If the TOE is part of a cluster the intra-cluster traffic must be adequately segregated from any other traffic.	As per the requirements in X.805, the planes should be adequately segregated including intra-cluster traffic.
P.PROCESS-NET-SEGREGATION	The TOE must allow the configuration of access controls on network resources in such a way that	Network resources should be segregated such that access is limited to the planes required for the

Name	Policy	Rationale
	a process's network access can be restricted to the minimum subset necessary.	network process's operation.
P.PROCESS-FILE-SEGRAGATION	The TOE must allow the configuration of access controls on files in such a way that the process can only access necessary files.	Limit the impact of process subversion of a process through buffer overflow attacks, insertion attacks and other common attacks.
P.TRACEABLE-TOE	The TOE should log sufficient information for security-relevant events.	Information such as user and process identifiers are needed for forensics and log file analysis.

SECURITY THREATS

This section borrows from a published example Common Criteria protection profile. According to [CSPP-OS03] the following threats do not have to be addressed by the target of evaluation. We believe that given some of the intended uses of this document we do need to address these two threats where possible.

Threat	Description of Threat
P.ACCESS	Access rights to specific data objects are determined by object attributes assigned to that object, user identity, user attributes, and environmental conditions as defined by the security policy Linux supports organizational policies that grant or deny access to objects using rules driven by attributes of the user (such as user identity), attributes of the object (such as permission bits), type of access (such as read or

Threat	Description of Threat
	write), and environmental conditions (such as time-of-day).
T.DENIAL-SOPHISTICATED	Sophisticated denial of network attacks include such threats as: • SYN flooding • IP fragmentation attacks
T.ENTRY-SOPHISTICATED	Sophisticated technical attacks by unauthenticated users, such as: • Buffer overflow attacks • Brute force or dictionary attacks on password • Network sniffing attacks • Man-in-The-Middle attacks • Session hijacking

The following threats must be addressed by the target of evaluation:

Threat	Description of Threat
T.ACCESS-TOE	An authorized user may gain non-malicious access to a resource or information controlled by the TOE. Such attacks include:

Threat	Description of Threat
	• Exploitation of improperly configured access permissions. • Information exposure through system errors. • Simple exploitation of vulnerabilities.
T.AUDIT-CONFIDENTIALITY-TOE	Disclosure of security event records to unauthorized users or processes. This is caused by: • Improperly configured permissions for log files. • Exploitable SUID programs.
T.AUDIT-CORRUPTED-TOE	Unauthorized modification or destruction of security event records. This is caused by: • Improperly configured access permissions for log files. • Easily exploitable SUID programs.
T.CRASH-TOE	Compromise of secure state when system crashes because the system does not fail securely.
T.DENIAL-TOE	Unsophisticated denial-of-service attacks. Examples include: • Creating enough Telnet or SSH sessions to lock out other users.

Threat	Description of Threat
	• Flood ping a system.
T.OBSERVE-TOE	Security compromise going undetected, for example: • The installation of a 'root kit'[2] goes undetected. • A buffer overflow and the following security compromise goes undetected. • Auditing is not configured to store all relevant security events.
T.RECORD-EVENT-TOE	Security-relevant events going unrecorded which is caused by: • Overloading the auditing system. • Large quantities of log events that 'rotate' files containing a security-relevant event out of existence.
T.RESOURCES	Exhaustion of system resources, which can be caused by: • Failing to configure the system resource limits for number of processes, memory or other resources.

2 A root kit is a set of programs that compromise security and usually hide their own existence.

Threat	Description of Threat
	• Underpowered systems.
T.TOE-CORRUPTED	The security of the TOE is intentionally corrupted, enabling future attack. This can include back doors left by programmers or intentional improper configuration of security-relevant systems (e.g. through the use of unauthenticated install media)

According to [CSPP-OS03] the following set of threats does not have to be addressed by the OS (TOE) alone. The environment should also play a role in addressing these vulnerabilities:

SECURITY OBJECTIVES

ENVIRONMENTAL SECURITY OBJECTIVES

Objective	Description	Threat or Policy
O.ACCESS-NON-TECHNICAL	The IT other than the TOE environment must provide sufficient protection against non-technical attacks by authenticated users for non-malicious purposes. This will be accomplished primarily via prevention with a goal of high effectiveness. Personnel security and user training and awareness will provide a major part of achieving this objective.	P.TRAINING
O.ACCESS-NON-TOE	The IT other than the TOE must provide public access and access by authenticated users to the resources	P.ACCESS

Objective	Description	Threat or Policy
	and actions for which they have been authorized and over which the TOE does not exercise control. The focus is on prevention with a high degree of effectiveness.	
O.ACCOUNT-NON-TOE	The TOE must ensure, for actions under its control or knowledge, that all users can subsequently be held accountable for their security relevant actions. This is expected with a high degree of effectiveness.	P.ACCOUNT T.TRACEABLE-NON-TOE T.RECORD-EVENT-NON-TOE T.AUDIT-CORRUPTED-NON-TOE T.AUDIT-CONFIDENTIALITY-NON-TOE
O.APPLICATION-TOOLS	The TOE must provide a reasonable, current set of security tools and libraries for use by applications.	P.DUE-CARE T.INSTALL T.OPERATE
O.AUTHORIZE-NON-TOE	The TOE must provide the ability to specify and manage user and system process access rights to individual processing resources and data elements under its control, supporting the organization’s security policy for access control. This is expected with	P.ACCESS

Objective	Description	Threat or Policy
	a high degree of effectiveness. NOTE: This includes initializing, specifying and managing (1) object security attributes, (2) active entity identity and security attributes, and (3) security relevant environmental conditions.	
O.AVAILABLE-NON-TOE	The IT other than the TOE must protect itself from unsophisticated, denial-of-service attacks. This is a combination of prevention, detection and recovery with a high degree of effectiveness.	P.SURVIVE T.DENIAL-NON-TOE
O.BYPASS-NON-TOE	For access not controlled by the TOE, IT other than the TOE must prevent errant or non-malicious, authorized software or users from bypassing or circumventing security policy enforcement. This will be accomplished with high effectiveness. NOTE: This objective is limited to 'non-malicious' because IT controls in the notional CSPP system are not expected to provide sufficient mitigation for the greater negative impact that 'malicious' implies.	T.ACCESS-NON-TOE
O.DETECT-SOPHISTICATED	The TOE environment must provide the ability to detect sophisticated attacks and the results of such attacks (e.g., corrupted system state). The goal is for moderate	P.SURVIVE T.SYSTEM-CORRUPTED

Objective	Description	Threat or Policy
	effectiveness.	
O.ENTRY-NON-TECHNICAL	The TOE environment must provide sufficient protection against non-technical attacks by other than authenticated users. This will be accomplished primarily via prevention with a goal of high effectiveness. User training and awareness will provide a major part of achieving this objective.	P.TRAINING
O.ENTRY-NON-TOE	For resources not controlled by the TOE, IT other than the TOE must prevent logical entry using unsophisticated, technical methods, by persons without authority for such access. This is clearly a prevent focus and is to be achieved with a high degree of effectiveness.	P.USAGE T.ENTRY-NON-TOE
O.INFO-FLOW	The TOE environment must ensure that any information flow control policies are enforced - (1) between system components and (2) at the system external interfaces. This will be accomplished by preventing unauthorized flows with high effectiveness.	P.INFO-FLOW
O.KNOWN-NON-TOE	The IT other than the TOE must ensure that, for all actions under its control and except for a well-defined set of allowed actions, all users are identified and authenticated before being granted access. This is	P.KNOWN

Objective	Description	Threat or Policy
	expected with a high degree of effectiveness.	
O.OBSERVE-NON-TOE	The IT other than the TOE must ensure that its security status is not misrepresented to the administrator or user. This is a combination of prevent and detect and, considering the potentially large number of possible failure modes, is to be achieved with a moderate, verses high, degree of effectiveness.	T.OBSERVE-NON-TOE
O.PHYSICAL	Those responsible for the TOE must ensure that those parts of the TOE critical to security policy are protected from physical attack that might compromise IT security. This will be accomplished primarily via prevention with a goal of high effectiveness.	P.PHYSICAL T.PHYSICAL

TOE SECURITY OBJECTIVES

Objective	Description	Threat or Policy
O.ACCESS-TOE	The TOE must provide public access and access by authenticated users to those TOE resources and actions for which they have been authorized. This will be accomplished with high effectiveness.	P.ACCESS
O.ACCOUNT-TOE	The TOE must ensure, for actions under its control or knowledge, that	P.ACCOUNT

Objective	Description	Threat or Policy
	all TOE users can subsequently be held accountable for their security relevant actions. This will be done with moderate effectiveness, in that it is anticipated that individual accountability might not be achieved for some actions.	T.TRACEABLE-TOE T.RECORD-EVENT-TOE T.AUDIT-CORRUPTED-TOE T.AUDIT-CONFIDENTIALITYTOE
O.AUTHORIZE-TOE	The TOE must provide the ability to specify and manage user and system process access rights to individual processing resources and data elements under its control, supporting the organization's security policy for access control. This will be accomplished with high effectiveness.	P.ACCESS
O.AVAILABLE-TOE	The TOE must protect itself from unsophisticated, denial-of-service attacks. This will include a combination of protection and detection with high effectiveness.	P.SURVIVE T.DENIAL-TOE
O.BYPASS-TOE	The TOE must prevent errant or non-malicious, authorized software or users from bypassing or circumventing TOE security policy enforcement. This will be accomplished with high effectiveness.	T.ACCESS-TOE

Objective	Description	Threat or Policy
	NOTE: This objective is limited to ‘non-malicious’ because CSPP-OS controls are not expected to be sufficient mitigation for the greater negative impact that ‘malicious’ implies.	
O.DETECT-TOE	The TOE must enable the detection of TOE specific insecurities. The goal is high effectiveness for lower grade attacks.	P.SURVIVE T.TOE-CORRUPTED
O.ENTRY-TOE	The TOE must prevent logical entry to the TOE using unsophisticated, technical methods, by persons without authority for such access. This will be accomplished with high effectiveness.	P.USAGE T.ENTRY-TOE
O.KNOWN-TOE	The TOE must ensure that, for all actions under its control and except for a well-defined set of allowed actions, all users are identified and authenticated before being granted access. This will be accomplished with high effectiveness.	P.KNOWN
O.OBSERVE-TOE	The TOE must ensure that its security status is not misrepresented to the administrator or user. This is a combination of prevent and detect and, considering the potentially large number of possible failure modes, is to be achieved with a moderate,	T.OBSERVE-TOE

Objective	Description	Threat or Policy
	verses high, degree of effectiveness.	
O.RECOVER-TOE	The TOE must provide for recovery to a secure state following a system failure, discontinuity of service, or detection of an insecurity. This will be accomplished with a high effectiveness for specified failures and a low effectiveness for failures in general.	P.SURVIVE T.CRASH-TOE
O.RESOURCES	The TOE must protect itself from user or system errors that result in shared resource exhaustion. This will be accomplished via protection with high effectiveness.	P.SURVIVE T.RESOURCES

JOINT SECURITY OBJECTIVES

Objective	Description	Threat or Policy
O.ACCESS-MALICIOUS	The TOE controls will help in achieving this objective, but will not be sufficient. Additional, environmental controls are required to sufficiently mitigate the threat of malicious actions by authenticated users. This will be accomplished by focusing on deterrence, detection, and response with a goal of moderate effectiveness.	T.ACCESS-MALICIOUS
O.COMPLY	The TOE environment, in conjunction with controls implemented by the TOE, must support full compliance with	P.COMPLY

Objective	Description	Threat or Policy
	applicable laws, regulations, and contractual agreements. This will be accomplished via some technical controls, yet with a focus on non-technical controls to achieve this objective with high effectiveness.	
O.DETECT-SYSTEM	The TOE, in conjunction with other IT in the system, must enable the detection of system insecurities. The goal is high effectiveness for lower grade attacks.	P.SURVIVE T.SYSTEM-CORRUPTED
O.DUE-CARE	The TOE environment, in conjunction with the TOE itself, must be implemented and operated in a manner that clearly demonstrates due-care and diligence with respect to IT-related risks to the organization. This will be accomplished via a combination of technical and non-technical controls to achieve this objective with high effectiveness.	P.DUE-CARE
O.MANAGE	Those responsible for the system (in conjunction with mechanisms provided by the TOE) must ensure that it is managed and administered in a manner that maintains IT security. This will be accomplished with moderate effectiveness.	T.ADMIN-ERROR
O.NETWORK	The system must be able to meet its security objectives in a distributed environment. This will be accomplished with high effectiveness.	P.NETWORK

Objective	Description	Threat or Policy
O.OPERATE	Those responsible for the system (in conjunction with mechanisms provided by the TOE) must ensure that the system is delivered, installed, and operated in a manner which maintains IT security. This will be accomplished with moderate effectiveness.	T.INSTALL T.OPERATE P.TRAINING
O.RECOVER-SYSTEM	The system must provide for recovery to a secure state following a system failure, discontinuity of service, or detection of an insecurity. This will be accomplished with some prevention and a majority of detect and respond, with high effectiveness for specified failures. For general failure, this will be accomplished with low effectiveness.	P.SURVIVE T.CRASH-SYSTEM
O.ENTRY-SOPHISTICATED	The TOE and the environment must sufficiently mitigate the threat of an individual unauthenticated user gaining unauthorized access via sophisticated, technical attack. This is accomplished by focusing on prevention, detection and response with a goal of high effectiveness.	T.ENTRY-SOPHISTICATED
O.DENIAL-SOPHISTICATED	The TOE and the environment must maintain system availability in the face of sophisticated denial-of-service attacks. The focus is on prevention, detection and response with a goal of high effectiveness.	P.SURVIVE T.DENIAL-SOPHISTICATED
O.DETECT-	The TOE and the environment must provide the ability to detect	P.SURVIVE

Objective	Description	Threat or Policy
SOPHISTICATED	sophisticated attacks and the results of such attacks such as corrupted system state. The goal is for high effectiveness.	T.SYSTEM-CORRUPTED
O.CONTAINMENT	The TOE and the environment must provide the ability to constrain the effect of a security failure of an application to that application.	P.CONTAINMENT P.PRIVILEGE-MIN P.SURVIVE T.SYSTEM-CORRUPTED

SECURITY REFERENCES

- ITU03: ITU-T, Security in Telecommunications and Information Technology, 2003

- CSPP-OS03: Gary Stoneburner, COTS Security Protection Profile - Operating Systems (CSPP-OS), 20

11. CGL GAPS

Following are the features or aspects of Carrier Grade Linux that, at the time of this publication, the CGL Workgroup has identified as un-implemented in the open source community or has not been widely adopted and proven ready for carrier grade applications. These features are listed here to provide information for developers and distribution vendors on key areas of differentiation that are of particular interest to carriers.

AVL.3.2 FORCED UN-MOUNT

ID	PID	Name
GAP.1.0	AVL.3.2	Forced Un-mount
CGL specifies that carrier grade Linux shall provide support for forced unmounting of a file system. The un-mount shall work even if there are open files in the file system. Pending requests shall be ended with the return of an error value when the file system is unmounted.		

AVL.3.3 FORCED UN-MOUNT APPLICATION NOTIFICATION

ID	PID	Name
GAP.2.0	AVL.3.3	Forced Un-mount Application Notification
CGL specifies that carrier grade Linux shall provide a notification mechanism when a forced un-mount of a file system occurs.		

AVL.14.0 EXCESSIVE CPU CYCLE USAGE DETECTION

ID	PID	Name
GAP.3.0	AVL.14.0	Excessive CPU Cycle Usage Detection

CGL specifies that carrier grade Linux shall provide a mechanism that detects excessive CPU cycle usage by any process or thread. To enable detection, the following capabilities shall be provided:

- Communication between the monitoring process and the kernel.
- Registering a list of processes or threads and their allowed CPU cycle thresholds.
- Ability to define policy based on process events including process/thread creation and exit.
- Ability to take action whenever an event occurs.
- Ability to set the CPU cycle threshold to a resolution of one millisecond.

AVL.28.0 SUPPORT OF MLOCKED PAGE LIMITS

ID	PID	Name
GAP.4.0	AVL.28.0	Support of Mlocked Page Limits

CGL specifies that carrier grade Linux shall support system wide limits on mlocked pages. This shall be configurable and enforced when the mlock page count exceeds the maximum setting. Either explicitly through a system call or implicitly through a page fault. The behavior shall be identical to per process mlocked limit when this system wide limit is exceeded.

AVL.29.0 COARSE RESOURCE ENFORCEMENT

ID	PID	Name
GAP.5.0	AVL.29.0	Coarse Resource Enforcement

The CGOS needs to provide mechanisms that allow resource consumption constraints to be applied to an individual thread, a process and all processes running with a particular user ID or group ID, when resource consumption limits are exceeded.

These resource consumption constraints should follow today's mechanisms for resource exhaustion for individual processes and groups of processes. Constraints must have actions that can be selected when an application is first started. Such actions include "log", "signal process" and "terminate process".

This requirement applies to CPUs as well as memory.

CAF.2.3 DELIBERATE TCP SESSION TAKEOVER

ID	PID	Name
GAP.6.0	CAF.2.3	Deliberate TCP Session Takeover

CGL specifies a mechanism to synchronize TCP sockets, buffer structures, and sequence numbers so that redundant nodes may take over TCP sessions originated on other nodes. A deliberate TCP session takeover assumes that TCP session(s) are transferred deliberately and not as the result of unexpected node failure(s).

CAF.2.4 TCP SESSION TAKEOVER ON NODE FAILURE

ID	PID	Name
GAP.7.0	CAF.2.4	TCP Session Takeover on Node Failure
CGL specifies a mechanism to synchronize TCP sockets, buffer structures, and sequence numbers so that when a critical resource fails, such as a CPU, memory, or kernel, a redundant node may take over TCP sessions originated on the failed node. Note that when the TCP session(s) are assumed by a redundant node, the sessions will resume from the last checkpoint. TCP traffic should continue even if there is a conflict between the last TCP state of the failed node and the checkpointed TCP state on the redundant node.		

CMON.1.4 CLUSTER-WIDE APPLICATION MONITOR

ID	PID	Name
GAP.8.0	CMON.1.4	Cluster-Wide Application Monitor
CGL specifies that carrier grade Linux shall provide a cluster-wide logging mechanism. A cluster-wide log shall contain node identification, message type, and cluster time identification. This cluster-wide log may be implemented as a central log or as the collection of specific node logs.		

SFA.14.0 PER THREAD CPU TIME LIMITS AND SIGNALING

ID	PID	Name
GAP.9.0	SFA.14.0	Per Thread CPU Time Limits and Signaling
CGL specifies that carrier grade Linux shall provide a method to accurately track CPU time consumed by an individual thread. It shall also provide a method to set CPU threshold time used by an individual thread. This method shall also include the ability to send a signal to an individual thread if its CPU threshold time is exceeded.		

SMM.6.0 BOOT CYCLE DETECTION

ID	PID	Name
GAP.10.0	SMM.6.0	Boot Cycle Detection
CGL specifies that carrier grade Linux shall provide support for detecting a repeating reboot cycle due to recurring failures. This detection should happen in user space before system services are started. This type of failure requires a response due to the negative impact of repeatedly bringing up and taking down services. A configurable policy is needed to set thresholds of cycling and desired shutdown actions, such as exponential back off, shutdown, or notifying administrators.		

SMM.7.8 SUPPORT FOR USER LOCKED PAGE REPORTING

ID	PID	Name
GAP.11.0	SMM.7.8	Support for User Locked Page Reporting
CGL specifies that in addition to current memory usage reporting, the OS shall report the count of mlocked pages to accurately determine how much memory may be reclaimed by the page frame reclaim algorithm. Based on mlocked page count and current memory usage reporting, a more accurate amount of free physical memory may be determined. In addition current overcommit policies shall take mlocked pages into account to accurately enforce memory overcommit policies for which the count of mlocked pages is applicable.		

SMM.7.9 SUPPORT FOR PRECISE PROCESS ACCOUNTING

ID	PID	Name
GAP.12.0	SMM.7.9	Support for Precise Process Accounting

CGL specifies that carrier grade Linux shall support precise process accounting of CPU usage. This shall be accomplished by time stamping various kernel execution paths using the native platform high resolution counter. This accounting activity shall be run-time configurable, including partial or total disabling, via the proc file system. When totally disabled no additional overhead will be measurable. Disabling or enabling precise accounting shall not affect Linux native tick accounting. All data shall be accessible from the proc file system. For task perCPU metrics, a range of 1 through N rows may be configured such that each row accrues metrics for one CPU, a range in between 1 and N CPUs (all metrics summed together). Where N is the number of logical CPUs. Additional Sub-requirements follow.

Sub-requirement 1: The following metrics shall be accrued on per-CPU basis:

- Per task CPU usage user, system, interrupt (in tasks context), and time spent on run queue
- System wide CPU usage idle, user, system, interrupt, softirq
- Per task occurrence counts of system calls, signals, reschedules, voluntary blocks, preemption due to higher priority task and preemptions due to time slice expirations.
- System wide occurrence counts of interrupts, system calls, signals, and softirqs, with softirqs grouped by types.

Sub-requirement 2: A per task table of schedule latency counts shall be implemented such that a schedule latency value is indexed into predetermined ranges, and the count for that range is incremented. For example a table size of three will correspond to three scheduling latency ranges such as:

- index 0: 0-10 milliseconds
- index 1: 10-100 milliseconds
- index 2: greater than 100 milliseconds The table size and ranges may be build time configurable

Sub-requirement 3: Certain OS timers and CPU caps may be configured to increment or

ID	PID	Name
expire precisely with the initial list being SIGXCPU, SIGVTALARM, SIGPROF.		

SMM.10.0 SYSTEM INITIALIZATION ERROR HANDLING ENHANCEMENTS

ID	PID	Name
GAP.12.0	SMM.10.0	System Initialization Error Handling Enhancements

CGL specifies that carrier grade Linux shall provide a mechanism to detect errors during system initialization. When such an initialization error occurs, this mechanism shall be able to report the event to a remote system over the network. CGL further specifies the following error conditions shall apply to this requirement:

- The kernel image fails before init is started
- The init process fails to fully complete the startup initialization to the point where the conventional error reporting mechanisms are available

SPM.5.0 MANUAL SOFTWARE ROLLBACK

ID	PID	Name
GAP.13.0	SPM.5.0	Manual Software Rollback

CGL specifies that carrier grade Linux shall provide mechanisms that allow manual rollback to a previous version of software without having to reinstall the previous version.

SPM.6.0 AUTOMATIC SOFTWARE ROLLBACK

ID	PID	Name
GAP.14.0	SPM.6.0	Automatic Software Rollback

CGL specifies that carrier grade Linux shall provide mechanisms that allow automatic rollback with configurable triggers to a previous version of software without having to reinstall the previous version.

PMS.5.2 ISCSI INITIATOR IPV6 SUPPORT

ID	PID	Name
GAP.15.0	PMS.5.2	iSCSI Initiator IPv6 Support
CGL specifies that the iSCSI Initiators implemented by carrier grade Linux should support the IPv6 protocol. This would enable the iSCSI Initiator nodes to connect to iSCSI targets only supporting IPv6 addresses.		

PRF.1.6 PROTECTING AGAINST PRIORITY INVERSION ON MUTEX

ID	PID	Name
GAP.16.0	PRF.1.6	Protecting Against Priority Inversion On Mutex
CGL specifies that carrier grade Linux shall support a mechanism for protecting against priority inversion when using a mutex to synchronize tasks. This mechanism shall support transitive priority inheritance and resolve cases where several mutexes are owned by the same task. It shall be supported in UP and SMP contexts.		

PRF.2.4 SUPPORT FOR TASK EXCLUSIVE BIND TO LOGICAL CPU

ID	PID	Name
GAP.17.0	PRF.2.4	Support for Task Exclusive Bind to Logical CPU

CGL specifies that carrier grade Linux shall support exclusive bind of processes or threads to any number of logical CPUs. Once the binding is established the logical CPU(s) become exclusively dedicated to the execution of the bound processes/threads, and idle. CGL further specifies the following conditions shall also apply to this requirement:

- There must be at least one logical CPU available for unbound tasks. Because of this, binding need not be supported on systems with only one logical CPU
- A logical CPU is defined as any CPU or part of a CPU/node that Linux represents as a single processing unit to the user

PRF.11.1 APPLICATION (PRE)LOADING NON-ROOT

ID	PID	Name
GAP.18.0	PRF.11.1	Application (Pre)loading Non-Root

CGL specifies that carrier grade Linux shall provide support for the preloading of an application even when the application is not executing as root. A configuration capability must exist to allow the system loader to determine an application's eligible for preloading. The action of preloading an application must not overload the system memory. The configuration capability must provide a control that allows the application to specify what is to be done if it can't be pre-loaded. Options are:

- Load anyway as a normal (pageable) application.
- Fail and don't load the application.

Regardless of the option used, any failure to pre-load the application must be logged.

PRF.11.2 APPLICATION (PRE)LOADING LIMITS

ID	PID	Name
GAP.19.0	PRF.11.2	Application (Pre)loading Limits

CGL specifies that carrier grade Linux shall provide mechanisms to avoid overloading a system when preloading applications. Specifically, it shall be possible to specify the total amount of memory reserved (pinned) by preloading applications.

SEC.7.4 EXECUTION QUOTAS

ID	ID	Name
GAP.20.0	SEC.7.4	Execution Quotas

CGL specifies that carrier grade Linux shall provide support for per-user CPU execution quotas.

SEC.9.0 UNIFIED CRYPTOGRAPHIC FRAMEWORK

ID	PID	Name
GAP.21.0	SEC.9.0	Unified Cryptographic Framework

To provide a cryptographic framework that supports encryption and message hashing for both kernel and user applications, secure tamper-proof storage for security-relevant data such as keys, and registration of cryptographic capabilities.

The CGOS needs to provide a unified framework for optimized implementations of common cryptographic (encryption and message hashing) algorithms.

Carrier grade solutions rely on communication protocols that have stringent security requirements. Typically, these protocols are based on standard security application providers such as SSL, SSH, IKE and JCE.

Data integrity is accomplished through mechanisms (message hashing) that check that data transmitted across the network or stored on/retrieved from disk without encryption are not modified. Data confidentiality is accomplished through mechanisms (encryption) that convert the data to a form not easily reversible, before being transmitted or stored. The use of both encryption and message hashing for data that are transmitted or stored demands a cryptographic framework that is available to both the kernel and user applications and that transparently makes use of whatever hardware encryption capabilities are available.

A prerequisite to the security capabilities described above is the ability to store in a secure, tamper-proof way security-relevant data, such as keys used to verify the integrity of downloaded data. Keys can be loaded during system assembly, and additional keys can be provided using a secure mechanism after the system is started. Such a mechanism is almost always a combination of hardware, operating system and firmware. See also Trust Mechanisms (CGOS-3.1).

A unified cryptographic framework must expose to security providers a common interface to algorithms not only for various encryption algorithms (at the very minimum 3DES and AES) but also for message hashing (MD5, SHA1), message signing (RSA, DSA, DH) and random number generation. See the RSA cryptographic token interface standard PKCS #11 [19].

Hardware acceleration is also desirable for carrier grade components that use encryption. The cryptographic framework must offer mechanisms whereby device drivers can register the cryptographic hardware. A device with a cryptographic capability (key store, encryption algorithm) must be able to register the capability with the cryptographic framework. Registration includes, for example, the type of cryptographic capability,

ID	PID	Name
available algorithms, and number of contexts. When a driver initializes, it must register any cryptographic capabilities possessed by the device(s) it controls. When a kernel thread or user process requests that a particular algorithm be used, the cryptographic framework must try to use the most efficient implementation based on the availability of resources in a transparent manner. Algorithms must be easy to export/import. Cryptographic keys must be easily reduced to 56 bits, or cryptography must be easy to switch off.		

STD.3.2.7 SCTP SIGNING CHUNKS

ID	PID	Name
GAP.22.0	STD.3.2.7	SCTP signing chunks
CGL specifies that carrier grade Linux shall provide the functionality listed in the Internet draft below. • draft-ietf-tsvwg-sctp-auth-04.txt: allows an SCTP sender to sign chunks using shared keys between the sender and receiver to prevent blind attacks against static Verification tag.		

GAP.23.0 FILE SYSTEM BLOCK MIRRORING

ID	PID	Name
GAP.23.0		File System Block Mirroring
CGL specifies that carrier grade Linux shall provide support for a file system that provides RAID-1 style mirroring support where alternate mirrors can be consulted if the checksum fails for any specific block prior to reporting a failure to the file system client.		

GAP.24.0 ONLINE FILE SYSTEM INTEGRITY AND CONSISTENCY CHECKING

ID	PID	Name
GAP.24.0		Online File System Integrity and Consistency Checking
CGL specifies that carrier grade Linux shall provide support for a file system that allows data and metadata consistency and integrity checking on a file system while mounted and in use with the fsck or similar tool. This consistency and integrity checking should be more detailed than the fast recovery integrity checks done from a partially completed update described in AVL.X.2.		

GAP.25.0 FILE SYSTEM RESOURCE ALLOCATION GUARANTEES

ID	PID	Name
GAP.25.0		File System Resource Allocation Guarantees
CGL specifies that carrier grade Linux shall provide support for a file system that allows for pre-allocation of space for files, better ensuring data is not overly fragmented on the storage media, with an API similar to the posix_fallocate() POSIX function without incurring the performance overhead associated with that API. Deviation from the posix_fallocate() is permissible provided the API is mechanically translatable.		

GAP.26.0 FILE SYSTEM ONLINE DE-FRAGMENTATION

ID	PID	Name
GAP.26.0		File System Online De-fragmentation
CGL specifies that carrier grade Linux shall provide support for a file system that allows for de-fragmentation of on-disk data while the file system is mounted and in use.		

GAP.27.0 ONLINE FILE SYSTEM EXPANSION

ID	PID	Name
GAP.27.0		Online File System Expansion
CGL specifies that carrier grade Linux shall provide the ability to expand a mounted file system without service interruption.		

GAP.28.0 ONLINE FILE SYSTEM REDUCTION

ID	PID	Name
GAP.28.0		Online File System Reduction
CGL specifies that carrier grade Linux shall provide the ability to reduce the size of a live file system without service interruption.		

GAP.29.0 REGISTRATION OF CRYPTOGRAPHIC CAPABILITIES

ID	PID	Name
GAP.29.0		Registration of Cryptographic Capabilities
CGL specifies that carrier grade Linux shall provide a method for registering and advertising the cryptographic capabilities of the system to local and remote clients.		

GAP.30.0 FILE ACCESS TRACING: LOGGING

ID	PID	Name
GAP.30.0		File Access Tracing: Logging
CGL specifies that carrier grade Linux shall provide the ability to record and report file access events, preserving them to persistent / recoverable media that will be preserved across system crashes and/or reboots.		

GAP.31.0 ASYNCHRONOUS HARDWARE ACCELERATED CRYPTO SUPPORT

ID	PID	Name
GAP.31.0		Asynchronous Hardware Accelerated Crypto Support
CGL specifies that carrier grade Linux shall provide facilities for applications to asynchronously perform encryption when a hardware crypto engine is available.		

GAP.32.0 ASYNCHRONOUS HARDWARE ACCELERATED CRYPTO SUPPORT: IPSEC

ID	PID	Name
GAP.32.0		Asynchronous Hardware Accelerated Crypto Support: IPSec
CGL specifies that carrier grade Linux shall provide facilities for applications to asynchronously perform IPSec Authentication Header (AH) and Encapsulating Security Protocol (ESP) encryption as defined in RFC 4301 and RFC 4309 when a suitable hardware crypto engine is available.		

GAP.33.0 ASYNCHRONOUS HARDWARE ACCELERATED CRYPTO SUPPORT: SNOW 3G

ID	PID	Name
GAP.33.0		Asynchronous Hardware Accelerated Crypto Support: SNOW 3G
CGL specifies that carrier grade Linux shall provide facilities for applications to asynchronously perform SNOW 3G cipher for both Confidentiality (UEA2) and Integrity (UIA2) modes when a suitable hardware crypto engine is available.		

GAP.34.0 ASYNCHRONOUS HARDWARE ACCELERATED CRYPTO SUPPORT: AES

ID	PID	Name
GAP.34.0		Asynchronous Hardware Accelerated Crypto Support: AES
CGL specifies that carrier grade Linux provide facilities for applications to shall asynchronously perform Advanced Encryption Standard cipher when a suitable hardware crypto engine is available.		

GAP.35.0 THREAD NAMING: DEBUGGING

ID	PID	Name
GAP.35.0		Thread Naming: Debugging
CGL specifies that carrier grade Linux shall provide the ability to uniquely identify threads with a symbolic name in addition to the existing process and thread ID mechanism. Assigned symbolic names must be able to be displayed in addition to all other information normally presented about threads in the Gnu Debugger (GDB). It must be possible to use symbolic names rather than thread ID to address individual threads within GDB.		

GAP.36.0 THREAD NAMING: MONITORING

ID	PID	Name
GAP.36.0		Thread Naming: Monitoring
CGL specifies that carrier grade Linux shall provide the ability to uniquely identify threads with a symbolic name in addition to the existing process and thread ID mechanism. Assigned symbolic names must be able to be displayed in addition to all other information normally presented about threads in system status applications such as top.		

GAP.37.0 PROCESS CORE DUMP FILTERING

ID	PID	Name
GAP.37.0		Process Core Dump Filtering
CGL specifies that carrier grade Linux shall implement custom core dump behavior for processes. An API must be provided that will allow a process to request specialized handling in the event that the size of a resulting core dump would exceed the system-defined limit. If the core dump will exceed the limit, individual segments will be dumped in the following priority order: 1 Stack 2 Heap 3 Shared Memory 4 BSS Data 5 Initialized Data		

GAP.38.0 PROCESS CORE DUMP FILTERING: COMPATIBILITY

ID	PID	Name
GAP.38.0		Process Core Dump Filtering: Compatibility
CGL specifies that carrier grade Linux shall implement custom core dump behaviour for processes. The resulting core dump must be compatible with current versions of the Gnu Debugger, GDB, even if not all segments have been included.		

GAP.39.0 EFFICIENT MULTI-THREADED APPLICATION CPU USAGE MONITORING

ID	PID	Name
GAP.39.0		Efficient Multi-Threaded Application CPU Usage Monitoring
CGL specifies that carrier grade Linux shall provide a summary of overall CPU usage for highly threaded applications. This summary will include user, system and interrupt mode execution statistics as well as the time spent in userspace waiting for locks and time spend handling page faults for each thread and for the containing process. This summary must accurately reflect the usage of the system at the time the summary is requested and gathering these statistics must not result in any noticeable performance degradation. The mechanism must also facilitate retrieval of process time usage and enforcement of CPU exhaustion limits in context switching code. These statistics must not rely on periodic sampling, each state transition within a thread must be recorded for the individual thread and for the process containing the thread.		

GAP.40.0 PERSISTENT SHARED MEMORY

ID	PID	Name
GAP.40.0		Persistent Shared Memory
CGL specifies that carrier grade Linux shall provide a mechanism for applications to store and retrieve critical data without depending on a locally attached disk. This mechanism must preserve such data from system crashes and across system reboots.		

GAP.41.0 COARSE RESOURCE ENFORCEMENT

ID	PID	Name
GAP.41.0		Coarse Resource Enforcement

CGL specifies that carrier grade Linux shall provide a mechanism that will impose resource consumption limits on one or more threads, processes or groups of processes. It must be possible to address individual threads, groups of threads, whole processes or groups of processes identified by the effective or real user or group ID with which they are running. Limits must have actions associated with them that can be selected when the process or thread is first started. These actions must at least include:

- Log - Allow the resource overstep to continue but report it via the normal system event reporting mechanism.
- Signal - Allow the resource overstep to continue but send a pre-defined signal to the thread/process.
- Terminate - Do not allow the resource overstep to occur, instead terminate the thread/process.

The resource consumption limits must be applied to at least CPU time and memory usage.

GAP.42.0 API for Non-Uniform Memory Architectures: Domain Binding

ID	PID	Name
GAP.42.0		API for Non-Uniform Memory Architectures: Domain Binding

CGL specifies that carrier grade Linux shall implement the notion of a latency domain, defined as a set of CPUs with directly attached, local memory. All systems shall have at least one latency domain, representing a uniform memory architecture. Additional latency domains can exist for non-uniform memory architectures, in which case carrier grade Linux will provide an API that allows a process to bind to a specific latency domain. An application must be able to specify the binding policy, with at least the following policies available:

- Opportunistic - A process will only migrate to a new latency domain if it is unable to execute in the current latency domain.
- Strict - A process will never migrate to a new latency domain even if it would otherwise be unable to continue execution.

11. DEPRECATED REQUIREMENTS

The following sections list previous CGL requirements that have been deprecated since they are now considered ubiquitous and essential parts of any modern Linux distribution.

REQUIREMENTS DEPRECATED IN CGL 4.0

AVL.3 deprecated in CGL 4.0.

AVL.4 deprecated in CGL 4.0.

AVL.5 deprecated in CGL 4.0.

AVL.5.2 deprecated in CGL 4.0.

AVL.7 deprecated in CGL 4.0.

AVL.7.3 deprecated in CGL 4.0.

AVL.8 deprecated in CGL 4.0.

AVL.8.2 deprecated in CGL 4.0

AVL.11.0 deprecated in CGL 4.0.

AVL.13 deprecated in CGL 4.0.

AVL.16.0 deprecated in CGL 4.0.

AVL.19.0 deprecated in CGL 4.0.

AVL.20.0 deprecated in CGL 4.0.

CCM.2 deprecated in CGL 4.0.

CAF.2 deprecated in CGL 4.0.

CMON.1 deprecated in CGL 4.0.

CDIAG.2 deprecated in CGL.4.0.

CCM.4.0 deprecated in CGL 4.0.

CCM.4.1 deprecated in CGL 4.0.

CCM.4.2 deprecated in CGL 4.0.

CCM.4.3 deprecated in CGL 4.0.

CCON.1 deprecated in CGL 4.0.

CDIAG.1 deprecated in CGL 4.0.

PLT.1.0 deprecated in CGL 4.0.

PMT.1.0 deprecated in CGL 4.0.

PMT.1.2 deprecated in CGL 4.0.

PMT.1.4 deprecated in CGL 4.0.

PMT.2.0 deprecated in CGL 4.0.

PIC.1.0 deprecated in CGL 4.0.

PIC.1.2 deprecated in CGL 4.0.

PIC.1.4 deprecated in CGL 4.0.

PMS.2.0 deprecated in CGL 4.0.

PMS.3.0 deprecated in CGL 4.0.

PMS.3.1 deprecated in CGL 4.0.

PMS.3.2 deprecated in CGL 4.0.

PMS.3.3 deprecated in CGL 4.0.
PMS.4.0 deprecated in CGL 4.0.
PMS.5.0 deprecated in CGL 4.0.
PRF.1 deprecated in CGL 4.0
PRF.1.10 deprecated in CGL 4.0
PRF.1.12 deprecated in CGL 4.0
PRF.2 deprecated in CGL 4.0
PRF.3 deprecated in CGL 4.0
PRF.3.3 deprecated in CGL 4.0
PRF.4 deprecated in CGL 4.0
PRF.4.5 deprecated in CGL 4.0
PRF.9.0 deprecated in CGL 4.0
PRF.11 deprecated in CGL 4.0
PRF.12.0 deprecated in CGL 4.0
PRF.13.0 deprecated in CGL 4.0
SEC.1 deprecated in CGL 4.0.
SEC.2 deprecated in CGL 4.0.
SEC.3 deprecated in CGL 4.0.
SEC.4 deprecated in CGL 4.0.
SEC.5 deprecated in CGL 4.0.
SEC.6 deprecated in CGL 4.0.
SEC.7 deprecated in CGL 4.0.
SMM.3 deprecated in CGL 4.0
SMM.7 deprecated in CGL 4.0
SMM.8 deprecated in CGL 4.0
SFA.2 deprecated in CGL 4.0
SFA.11.0 deprecated in CGL 4.0
SFA.12.0 deprecated in CGL 4.0

STD.9.0 as it concerns the IPMI v1.0 level of the specification has been deprecated in CGL

STD.10.0 as it concerns 802.1Q VLAN Bridging has been deprecated in CGL 4.0.

STD.12.0 has been deprecated in CGL 4.0.

STD.13.0 has been deprecated in CGL 4.0.

STD.14.0 has been deprecated in CGL 4.0.

STD.14.2 has been deprecated in CGL 4.0.

STD.15.0 has been deprecated in CGL 4.0.

STD.21.0 has been deprecated in CGL 4.0.

STD.23.0 has been deprecated in CGL 4.0.

STD.24.0 has been deprecated in CGL 4.0.

REQUIREMENTS DEPRECATED IN CGL 5.0

AVL.1.0 has been deprecated in CGL 5.0.

AVL.18.0 has been deprecated in CGL 5.0.

AVL.3.1 has been deprecated in CGL 5.0.

AVL.4.2 has been deprecated in CGL 5.0.

AVL.4.3 has been deprecated in CGL 5.0.

AVL.4.4 has been deprecated in CGL 5.0.

AVL.5.1 has been deprecated in CGL 5.0.

AVL.7.2 has been deprecated in CGL 5.0.

AVL.8.3 has been deprecated in CGL 5.0.

AVL.9.1 has been deprecated in CGL 5.0.

CAF.1.0 has been deprecated in CGL 5.0.

CCM.1.0 has been deprecated in CGL 5.0.

CCM.2.1 has been deprecated in CGL 5.0.

CCM.2.3 has been deprecated in CGL 5.0.

CCM.2.4 has been deprecated in CGL 5.0.

CCM.2.5 has been deprecated in CGL 5.0.
CCM.3.0 has been deprecated in CGL 5.0.
CCON.1.1 has been deprecated in CGL 5.0.
CCON.1.2 has been deprecated in CGL 5.0.
CCON.1.3 has been deprecated in CGL 5.0.
CCON.1.4 has been deprecated in CGL 5.0.
CCS.1.0 has been deprecated in CGL 5.0.
CCS.2.0 has been deprecated in CGL 5.0.
CDIAG.1.1 has been deprecated in CGL 5.0.
CDIAG.1.2 has been deprecated in CGL 5.0.
CES.1.0 has been deprecated in CGL 5.0.
CLS.1.0 has been deprecated in CGL 5.0.
CMON.1.1 has been deprecated in CGL 5.0.
CMON.1.2 has been deprecated in CGL 5.0.
CMON.1.3 has been deprecated in CGL 5.0.
CMS.1.0 has been deprecated in CGL 5.0.
CMS.2.0 has been deprecated in CGL 5.0.
CMS.3.0 has been deprecated in CGL 5.0.
CSM.3.0 has been deprecated in CGL 5.0.
CSM.5.0 has been deprecated in CGL 5.0.
PIC.1.1 has been deprecated in CGL 5.0.
PIC.1.5 has been deprecated in CGL 5.0.
PIC.1.6 has been deprecated in CGL 5.0.
PIC.2.0 has been deprecated in CGL 5.0.
PIC.3.0 has been deprecated in CGL 5.0.
PLT.1.1 has been deprecated in CGL 5.0.
PLT.1.1-a has been deprecated in CGL 5.0.
PLT.1.1-c has been deprecated in CGL 5.0.

PLT.1.2 has been deprecated in CGL 5.0.
PLT.1.2-a has been deprecated in CGL 5.0.
PLT.1.2-c has been deprecated in CGL 5.0.
PLT.1.3 has been deprecated in CGL 5.0.
PLT.1.3-a has been deprecated in CGL 5.0.
PLT.1.3-c has been deprecated in CGL 5.0.
PMT.1.1 has been deprecated in CGL 5.0.
PMT.1.3 has been deprecated in CGL 5.0.
PRF.1.1 has been deprecated in CGL 5.0.
PRF.1.11 has been deprecated in CGL 5.0.
PRF.1.2 has been deprecated in CGL 5.0.
PRF.1.3 has been deprecated in CGL 5.0.
PRF.1.5 has been deprecated in CGL 5.0.
PRF.1.8 has been deprecated in CGL 5.0.
PRF.1.9 has been deprecated in CGL 5.0.
PRF.10.0 has been deprecated in CGL 5.0.
PRF.3.1 has been deprecated in CGL 5.0.
PRF.3.2 has been deprecated in CGL 5.0.
PRF.4.1 has been deprecated in CGL 5.0.
PRF.4.3 has been deprecated in CGL 5.0.
PRF.4.4 has been deprecated in CGL 5.0.
SFA.13.0 has been deprecated in CGL 5.0.
SFA.3.1 has been deprecated in CGL 5.0.
SFA.5.0 has been deprecated in CGL 5.0.
SFA.6.0 has been deprecated in CGL 5.0.
SFA.7.0 has been deprecated in CGL 5.0.
SFA.9.0 has been deprecated in CGL 5.0.
SMM.1.0 has been deprecated in CGL 5.0.

SMM.11.0 has been deprecated in CGL 5.0.
SMM.14.1 has been deprecated in CGL 5.0.
SMM.14.2 has been deprecated in CGL 5.0.
SMM.2.0 has been deprecated in CGL 5.0.
SMM.2.1 has been deprecated in CGL 5.0.
SMM.6.1 has been deprecated in CGL 5.0.
SPM.7.0 has been deprecated in CGL 5.0.
SPM.8.0 has been deprecated in CGL 5.0.
STD.16.0 has been deprecated in CGL 5.0.
STD.19.0 has been deprecated in CGL 5.0.
STD.2.0 has been deprecated in CGL 5.0.
STD.2.1 has been deprecated in CGL 5.0.
STD.2.2 has been deprecated in CGL 5.0.
STD.2.3 has been deprecated in CGL 5.0.
STD.22.0 has been deprecated in CGL 5.0.
STD.25.0 has been deprecated in CGL 5.0.
STD.3.2.8 has been deprecated in CGL 5.0.
STD.8.2 has been deprecated in CGL 5.0.
STD.8.3 has been deprecated in CGL 5.0.
STD.8.4 has been deprecated in CGL 5.0.
STD.8.5 has been deprecated in CGL 5.0.
STD.8.6 has been deprecated in CGL 5.0.
STD.8.7 has been deprecated in CGL 5.0.

12. REFERENCES

Background information useful to readers of this document can be found in the following places:

- The Linux Foundation home page: http://www.linuxfoundation.org
- The Carrier Grade Linux web page: http://cgl.linuxfoundation.org