Open and Secure Terminal Initiative (OSTI) Architecture Specification

Revision 1.00

Table of Contents

Abbreviations and Acronyms Used

Term	Expansion	Meaning
OSTI	Open and Secure Terminal Initiative	The architecture described in this document
OS	Operating System	A central piece of software that manages hardware resources and other software programs on a computing device
OOS	Operator OS	The OS that runs in the Operator Domain, typically, for personal use
EOS	Enterprise OS	The OS that runs in the Enterprise Domain, typically, for business use
VMM	Virtual Machine Monitor	Software that allows multiple OS to run simultaneously; also sometimes called a hypervisor
GUI	Graphical User Interface	Visual interface by which the user and software communicate
BSP	Board Support Package	A layer of software that adapts the hardware independent part of an OS to specific hardware
DAL	Domain Abstraction Layer	A layer of software in the OSTI architecture that presents an abstracted run-time environment to an OS
HID	Human Interface Device	A device that the platform uses to receives or provides stimuli directly to or from the user, such as a keypad, display or speaker.
SIM	Subscriber Identity Module	IC card with non-volatile memory that contains a subscriber's information such as telephone numbers, service details identifications, etc
SoC	System on a Chip	An integrated circuit that contains essential elements of an electronic system, including a processor and key peripherals.

1 Overview

1.1 OSTI goal

The Open and Secure Terminal Initiative (OSTI) will enable third-party vendors to develop new applications that enhance the value of the cellular phone to the end user while maintaining the quality and reliability of traditional Operator services.

1.2 Platform Architecture Overview

The OSTI platform supports two domains: an Operator Domain intended for personal use, and an Enterprise Domain for business use. Each domain has a different Graphical User Interface (GUI), and each domain has its own management policies, potentially instigated and enforced by different entities. Each domain may have a different Operating System (OS) running in it as well.

Under the OSTI architecture, the user experiences one domain at a time, but can switch between domains nearly instantly. The OSTI architecture provides separation between the domains, protecting quality of service and user privacy even in the presence of flawed or malicious software in the Enterprise Domain.

Figure 1 shows the architecture of an OSTI platform, and the role of the OSTI architecture in providing a standard interface between the Enterprise OS and the rest of the platform. This standard interface is provided by the Domain Abstraction Layer (DAL). It allows the vendors of the Enterprise OS and applications to develop without concern for the implementation of the Operator Domain. In some implementations, the DAL is similar in purpose and complexity to a Board Support Package (BSP), the code that allows an OS to run on a specific hardware configuration. In other implementations, some modification of the Enterprise OS may be necessary.

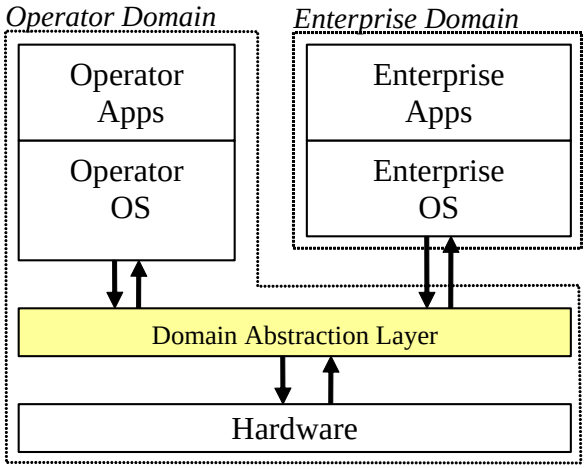

Figure 1: OSTI Platform Architecture

The components of the Operator Domain shown in Figure 1, other than the DAL, are only an example. The structure of the Operator Domain and the relationship between Operator OS and DAL are beyond the scope of this specification.

1.3 Intended Audience

This document is intended for cellular phone platform vendors and system software developers.

1.4 Purpose of this document

This revision of this specification only addresses the interface between the Enterprise Domain and the rest of the platform. For the platform vendors this document gives an overview of the capabilities the platform needs to provide. For Enterprise Domain software and OS developers it gives an overview of the environment their software will operate in and the capabilities the platform will provide to it.

The document currently considers two possible implementation technologies separately, to make sure that any unique concerns of either are addressed. In future versions, these two sections may be

merged. This document does not mandate a particular underlying implementation for OSTI, or specify any details of the Operator Domain.

2 OSTI Characteristics

2.1 New User Experience

OSTI gives the user the ability to quickly switch back and forth between a business-oriented Enterprise Domain and a personally-oriented Operator Domain. Rather than needing one phone to fulfill business needs and another phone for their personal life, the OSTI platform will allow a single phone to handle all of the user's mobile telecommunications needs, while protecting the quality of service and the confidentiality of data in the Operator Domain. It would be "One phone for both your lives."

The user interacts with only one domain at any one time. The domain that the user is interacting with is known as the "foreground" domain, and the domain that the user is not interacting with is the "background" domain. The user can change which domain they are interacting with using a domain switch button or other input device dedicated to that function. When the Enterprise domain is in the background, applications in that domain either continue to run, or will be suspended until the Enterprise domain becomes the foreground domain. Whether an Enterprise application continues to run or not may be determined by the implementation of OSTI used, or in some implementations may be configurable for each application or by the type of application.

Essential Promise

The essential promise of OSTI is that no matter what happens in the Enterprise Domain, the Operator Domain will continue to provide the operator services that the user depends on with uncompromised security and quality.

In order to provide a standard method of returning to the Operator Domain even in situations where the Enterprise Domain is completely dysfunctional, the user may force the Operator Domain to be the foreground domain by holding the domain switch button down for certain amount of time.

2.2 Extensible Enterprise Solution

In the Enterprise Domain, the user will use a standard mobile OS, such as Microsoft Windows® Mobile. This will allow the user to download a large number of third-party applications that enhance the value of the phone. In a large corporation, the corporate IT department would manage this domain, allowing the corporation to control the device policies for this domain, including allowed applications, allowed access to the corporate network, security settings, and so on.

2.3 Reliable Operator Service

In the Operator Domain, the software used is provided by the cellular network operator, who will control the policies for this domain. The Operator Domain must be protected from the Enterprise Domain, so that no matter what policies are set in the Enterprise Domain, and no matter what programs, viruses or OS bugs might be encountered in the Enterprise Domain, the quality of the Operator Domain services would not be degraded, and the security of Operator Domain data would not be compromised.

2.4 Consistency between Operator Domain and Enterprise Domain

From the viewpoint of the cellular network operator, OSTI must meet the following requirements at all times, regardless of which domain is the foreground domain.

Communication Centric: As a cellular phone is essentially a communication device, the communication capability must be regarded as the most important functionality provided by the device.

Basic Phone Functionality: OSTI terminals must provide basic phone functionality with natural extension from the traditional cellular phone functionality. So while in some ways an OSTI platform behaves like two independent phones in the same package, in other ways it behaves like a single traditional cellular phone. For example, when OSTI terminals are put into silent mode (or manner mode) while one domain is the foreground domain, this silent mode must be reflected when the other domain becomes the foreground domain. To give another example, if the cellular radio transceiver is turned on or off while one domain is the foreground domain, then it must remain on or off when the other domain becomes the foreground domain. The OSTI architecture provides an inter-domain message passing capability that may be of use in meeting this requirement.

3 Architecture Overview

OSTI accommodates at least two technologies that allow multiple operating systems to share the same platform: OS Switching, which exploits advanced power saving features to enable co-existence of multiple OSes, and Virtualization technology using a Virtual Machine Monitor (VMM), which provides a different virtual machine for each OS.

OSTI aims to minimize the difference between the semantics of the interfaces that OS Switching and VMM implementations each provide to the Enterprise Domain. However, some differences in these semantics are intrinsic in the differences between the two technologies, and as a result this specification has separate sections for each. These differences in the semantics may also result in differences in user-visible platform behavior.

3.1 OS Switching

Modern operating systems support various “sleep” modes to save power when there are little or no activities in the system. A common “sleep” mode is for the OS to go through a “suspend” process that saves critical portions of the system’s state to the system memory. Once this state is saved, most or all of the hardware can be turned off. When the system is turned back on, the OS goes through a “resume” process that fetches the saved state from the memory and restores the system to the same operating state it was in before the “suspend” operation. After the resume, applications will begin running again from the same point where they were suspended. The “suspend” and “resume” operations can be accomplished very quickly: typically in a fraction of a second.

OS Switching exploits this capability to switch between operating systems. The OS being switched out goes through the “suspend” process, but the hardware is not turned off. Instead, the OS Switching software boots or resumes a different operating system. During the period when the second OS is running, none of the critical state saved by the first OS is modified. Therefore, when the second OS is suspended and the first OS is resumed out of “sleep”, it is as if the second OS was never run at all, as far as the first OS is concerned. Conversely, the second OS is essentially unaware that the first OS is running while it thinks it was in “sleep” state.

By exploiting the rapid suspend and resume capability described above, OS Switching allows the user to suspend one OS and its applications and switch to the other with the touch of a button, and then return to the first OS with another press of the button.

3.2 VMM

A Virtual Machine Monitor, also sometimes called a “hypervisor”, typically presents to the Enterprise Domain a CPU architecture that is similar but not identical to the underlying platform and a set of virtualized devices that can access peripherals. The VMM places constraints on which resources the Enterprise OS and its applications can use. Enterprise Domain code executes at a reduced privilege level, which prevents it from bypassing the restrictions placed on it by the VMM.

A VMM can switch which domain is active in a manner similar to how an operating system can switch processes. In some systems, this can make it appear to the user that both domains are active simultaneously. However, OSTI requires that the user only directly interact with the foreground domain. The foreground domain is the domain that currently owns the core Human Interface Devices (HID), as defined in section 5.2.4.1.

There are costs associated with using a VMM. For example, using a VMM can impose additional overhead on calls to system services. Also, some VMMs require modifications to any operating system that is run under them.

This specification does not mandate which VMM to use, but OSTI-compliant terminals must select a VMM with specific capabilities for protecting the Operator OS's resources, including memory, peripherals, etceteras, from the Enterprise OS.

4 OS Switching Architecture

4.1 Enterprise OS Launch, Pause and Resume

4.1.1 Enterprise OS Launch

The initial launch of the Enterprise Domain may occur as soon as the Operator Domain has booted, or it may be delayed until the user initiates the switch to the Enterprise Domain. For the initial launch of the Enterprise OS, code in the DAL performs the functions of an OS Bootloader. This may include running integrity and/or authenticity checks on the OS image. The Enterprise OS will launch using a memory map identical to what the Operator Domain software believes the Enterprise OS is using. This memory map will exclude memory and memory-mapped devices that the Enterprise Domain is not permitted to directly access. As far as the Enterprise OS can tell, the excluded memory and devices do not exist.

4.1.2 Enterprise OS Suspend

When the Enterprise OS is signaled to suspend, it must save away the state needed for later resumption, just as it would if the device was being put into a deep sleep or hibernate state. This code must be sure to synchronize all active file systems and wait for any outstanding hardware operations to complete. It may power down any parts of the system except those necessary to accomplish the OS Switch: the processor, memory and so on. When the suspend process reaches the point where the hardware would normally be powered off, the DAL resumes the Operator OS instead.

When the Enterprise Domain is suspended, applications in that domain, such as MP3 playback, are suspended as well.

4.1.3 Enterprise OS Resume

When the DAL invokes the Enterprise OS resume code, the Enterprise OS resumes execution just as if it was resuming from a deep sleep or hibernate state.

4.2 Enterprise OS System Management Interface

4.2.1 Platform Management Functions

4.2.1.1 Interrupts

Some interrupts will be allocated to the Enterprise OS when it is running, but others will be reserved for use by the rest of the platform. The DAL must provide facilities that allow the Enterprise OS to enable or disable interrupts that are allocated to it, but do not allow the Enterprise OS to enable or disable any other interrupts. The DAL must provide facilities that allow the Enterprise OS to configure interrupts that it is using, such as setting the relative priority of an interrupt or the address of its service routine, but do not allow the Enterprise OS to alter the configuration of interrupts not allocated to it. The DAL will not create significant new interrupt-handling latencies.

Although the OSTI concept primarily switches domains in response to a user command, some event interrupts outside the control of the Enterprise OS may cause domain switching as well. An example of such an interrupt could be a watchdog timer, such as might be used to guarantee that Operator Domain housekeeping functions were periodically executed. When such an interrupt occurs the DAL will request a suspension of the Enterprise Domain and then, after the Enterprise Domain suspends, switch back to the Operator Domain. If the Enterprise Domain fails to suspend in a timely fashion after such a request is made, the switching back to the Operator Domain must occur anyway, and the ability to resume (as opposed to restarting) the Enterprise Domain is not guaranteed.

4.2.1.2 Power States

The Enterprise OS may manage device power arbitrarily while it is running. When suspending, the Enterprise OS shall not switch off devices which are needed by OS Switcher such as the CPU and memory system. When resuming, the Enterprise OS should switch on all devices that it needs to have on.

4.2.2 Operator OS / Enterprise OS Data Communication

The DAL provides the interface for Operator OS / Enterprise OS Data Communication. By defining the high-level semantics of this interface, the OSTI architecture provides applications with a guaranteed minimum level of functionality they can rely on across all OSTI platforms, even though how the functionality is implemented may vary.

In the OS Switching architecture, messages are exchanged in an asynchronous fashion as OSes run in a mutual exclusiveness manner. The semantics of these variable-length messages will be defined, but will not specify the functionality that the Operator Domain provides via this interface. Multiple messages may be outstanding simultaneously. All messages are acknowledged with at least a success or error indication.

An example of Operator OS / Enterprise OS Data Communication usage is sharing terminal setting information between the two domains such as the silent mode described in Section 2.4. Another example is making an integrated list of applications running in the two domains via the inter-domain communication facility.

The high-level semantics will be the same regardless of whether the implementation uses OS Switching or a VMM. There will be two lists of messages visible to the Enterprise Domain: incoming messages from the Operator Domain, and outgoing messages to the Operator Domain. The Operator Domain will remove messages from the outgoing message list as the messages are processed. The Enterprise Domain should remove messages from incoming list as they are processed. Messages are not guaranteed to be processed in order. Serializing messages if required is the responsibility of the applications. The interface will return an error if it is unable to add a message to a list (for example, because of insufficient memory or a violation of message size limits).

Every message will have a standard header that provides the basis for the following Enterprise Domain capabilities: unique identification of the message; distinguishing data messages from acknowledgement messages; determining the message length; determining whether another message follows this message in the list, and locating the next message in the list if it exists.

Data messages use the rest of the message for the payload. The syntax and semantics of the payload are defined at higher protocol levels beyond the scope of this document.

Acknowledgement messages provide the following additional capabilities to the Enterprise Domain: unique identification of the data message the message is a response to, and determination of whether they are an acknowledgement of successful acceptance (ACK) or not (NAK). Acknowledgement messages also contain a response payload; the syntax and semantics of this payload are defined at higher protocol levels beyond the scope of this document.

Operators and vendors determine the minimum levels of each capability provided by the interface, such as the number of uniquely identifiable messages supported, payload and response payload size, and the number or total size of messages that can be in the messages lists. Agreement on minimum levels across the industry will facilitate use of the interface.

4.2.3 Storage

4.2.3.1 Fixed (i.e. built-in) Persistent Storage

The DAL will provide the Enterprise Domain with access to portions of the platforms built-in persistent storage, such as Flash devices, hard disk drives, or battery-backed up RAM. The amount of fixed persistent storage made available to the Enterprise Domain will be constant. The format and file system used on this persistent storage may differ from those used on the persistent storage allocated to the Operator Domain. The Enterprise Domain should not attempt to access fixed persistent storage beyond that allowed by the DAL. DAL providers should enforce this restriction.

4.2.3.2 Direct-access memory (RAM, etcetera)

The memory map provided to the Enterprise OS at launch will inform the Enterprise OS of what memory it may access. The Enterprise OS must not attempt to access memory addresses outside those ranges permitted to it by the memory map, nor permit applications to do so. DAL providers should enforce this restriction.

4.2.3.3 Removable Storage

The Enterprise Domain has the same access to removable storage, such as SD cards, as the Operator Domain. Conceptually, each time the Enterprise OS is switched to background and switched back to foreground again, the Enterprise OS should behave as if the card was removed and was re-inserted while the Enterprise OS was suspended.

During the suspend period the Operator OS may modify the content on the removable storage.

The Enterprise OS should therefore “sync” the file system of any removable storage as part of the suspend process, to ensure that it is consistent and that all pending operations are complete. The Enterprise OS should poll for the presence of a removable card at resume. Because the card or its contents might have been altered since the suspend occurred, the Enterprise Domain must not rely on any file system information it obtained before the suspend-resume cycle.

4.2.4 Peripherals

4.2.4.1 HID (keyboard, cursor, touch screen, display, sound in/out)

When it is the foreground domain, the Enterprise Domain has complete access to and control of all Human Interface Devices (HID), with two exceptions described below. HIDs include keyboards, LCDs, audio outputs, touch screens and pointing devices. When the Enterprise Domain is the background domain, it has no access to any HIDs.

The OSTI architecture allows the user to switch domains manually, such as by pressing a dedicated button. The Enterprise OS should not prevent this domain switch from occurring. DAL providers should hide the domain-switch input device from Enterprise OS and should enforce the restriction.

The OSTI architecture supports an optional trusted indicator of whether the Operator or Enterprise domain is the foreground domain. An example would be a bi-color LED that was one color for the Operator Domain and another for the Enterprise Domain. Similar to the case of domain-switch input device, the Enterprise Domain would have no access or control over this indicator.

4.2.4.2 Real-Time Clock (RTC)

The DAL provides a Real-Time Clock (RTC) device to the Enterprise Domain. The Enterprise Domain will be able to control its RTC, but will not be able to affect the Operator Domain’s RTC. The Enterprise Domains RTC will keep time correctly regardless of domain switching.

There are many ways that an implementation may fulfill this requirement. For example, if platform has more than one RTC, the Enterprise Domain may be allocated an actual RTC for its exclusive use, which it may manipulate at will. On other platforms, the RTC provided to the Enterprise Domain may be a virtual device that merely scales and adds an offset to the Operator Domain's RTC, and the Enterprise Domain will only be able to alter the offset and scale factor.

4.2.4.3 Network connection

An OSTI platform may provide a networking device (e.g., a WiFi interface) that is exclusively for the use of the Enterprise Domain, and not usable by the Operator Domain. Such a device may be managed by the Enterprise Domain in any manner it wishes. The DAL should make such a device accessible to the Enterprise domain and does not impose any restrictions on its usage.

The Operator Domain and Enterprise Domain may share the same network device but that does not necessarily mean that they are sharing a connection: For example, if each domain uses a different MAC address when using WiFi hardware, they are not sharing a connection. OSTI allows this scenario.

However, the Enterprise Domain may be able to connect to data networks that are also accessed from the Operator Domain, such as WCDMA networks. Even if domain switching occurs, the terminal-side state of network connection should be consistent with the network-side state. There are at least two options that an operator may mandate:

Option 1: The Enterprise Domain should terminate all network connections before switching. The Operator Domain then reestablishes the network connections. Implementing this option may leave the platform without a data connection for a substantial amount of time after an OS Switch, but if it can be guaranteed that the Operator Domain network connection will not have the same TCP/IP address as the Enterprise Domain had, the complexity of sharing a TCP/IP address between the domains can be avoided. However, if a different address can not be guaranteed, the problem of sharing the address still exists, as discussed for the second option, below.

Option 2: The Enterprise domain shares a state of network connection with the Operator Domain to keep network connections. A single TCP/IP address handles external network communications for both domains.

4.2.4.3.1 Port Number Partitioning for shared connections

One possible method being investigated at present is port number partitioning. Both domains would use the same IP address, but would use different port numbers. For "well known" and registered port numbers (those associated with a particular standard service such as FTP or Voice-over-IP (VoIP)), it should be possible to allocate port numbers to domains at build time. For ephemeral port numbers, each domain would use a different range between 1024 and 65534; for example, if the Enterprise OS is Microsoft Windows® CE, it assigns ephemeral port numbers form the range 1025 through 5000, leaving the port numbers above 5000 for the Operator OS to use.

However, Enterprise Domain applications may be free to request any port number, and this could lead to an Enterprise application and an Operator application having the same port number and IP address, which would lead to confusion and possible compromise of Operator domain functionality. Software in the Enterprise Domain should not use port numbers being used in the Operator Domain. To prevent this, the Enterprise OS could, at socket creation, reject any requests for port numbers outside of a predefined range. Such a capability can be provided under Microsoft Windows® CE using a Layered Service Provider, for example. A Linux® OS could provide this capability via customization of the socket code. Alternatively, an Enterprise OS could simply prevent messages to or from a port number outside the allowed address from being transmitted, such as with a firewall.

In some cases, for some particular applications, it may be possible for the Enterprise and Operator Domain to share a port by using the IP address of the remote system. For example, the system might

have a list of servers which were allowed to manage the Operator Domain, and remote management requests received by the Operator Domain from any other servers would be routed to the Enterprise Domain or ignored, depending on platform policies.

4.2.4.4 SIM

A Subscriber Identity Module (SIM) consists of persistent storage and optionally an externally accessible cryptographic engine. Enterprise Domain access to the SIM's persistent storage will be handled in the same manner as access to any other fixed persistent storage would be, as described in section 4.2.3.1. If the SIM has an externally accessible cryptographic engine, Enterprise Domain access to that engine will be provided in the default manner used for most peripheral devices, as described in section 4.2.4.5.

4.2.4.5 Other Devices

Except for the devices described in the sections above, the Enterprise Domain is given complete access to and control over devices in the platform it is aware of. Note that the Enterprise Domain may not be made aware of the existence of some devices on the platform. Also note that for bus-mastering devices, the DAL may prevent any attempt to program a bus-mastering device to access memory beyond that specified to the Enterprise OS in the memory map provided to it.

On suspend, the Enterprise OS should treat these devices in the same manner that it would when going into a deep-sleep or hibernate state. On resuming from suspend state, the Enterprise OS should re-initialize these devices as if it was resuming from a deep-sleep or suspend-to-RAM state.

4.3 OS Switching Architecture Specific Considerations

In this section, we'll describe some issues related solely to the systems adopting OS Switching Architecture.

4.3.1 Protecting Operator Domain Data from the Enterprise Domain

As is described earlier, the OS Switching architecture allows the OS in each domain to run as if it were the only OS in the system. This means that each OS kernel has the highest privilege provided by the CPU and therefore can do anything in the system, including, but not limited to, investigating the actual configuration of system hardware such as the installed physical memory size (as opposed to the size communicated by the DAL at OS launch), reading or modifying the contents of memory portions assigned to the other domain or the OS Switcher, and so on. Normally the preinstalled Enterprise OS supplied with a terminal operates within the restrictions set for it at launch and does not access resources that are not assigned to it, and constrains applications to not do so as well. But if a malicious or defective application program gains kernel privilege, such a program might access or modify Operator Domain resources, potentially exposing secrets kept by the Operator Domain or interfering with the operation of the Operator Domain. It is necessary to implement countermeasures in order to mitigate this threat.

In general, static data such as files stored in flash or mini SD card can be protected well by introducing standard file encryption techniques, such as Encrypting File System (EFS) in the case of Microsoft Windows® 2000 and later. This document concentrates on how to protect data stored in physical memory. The data that must be protected falls into two categories. There are Operator Domain secrets that must not be readable by the Enterprise Domain, an OSTI-compliant terminal with OS Switching architecture must protect this data from Enterprise Domain eavesdropping. There is also data that the Enterprise Domain must not be able to modify without detection, and an OSTI-compliant terminal with OS Switching architecture must either protect important data in the Operator Domain's memory from modification or detect the unauthorized modification of such important data. Typical data that needs these protections includes, but is not limited to, the user's address book, DRM license material, decryption keys, etceteras.

In the current revision of this document, we provide two types of candidate mechanisms to satisfy the above requirement. These are 1) Obfuscation-hardened encryption and integrity protection, and 2) TrustZone-based protection.

4.3.1.1 Obfuscation-hardened Encryption and Integrity Protection

One way to protect the Operator OS's in-memory data from the Enterprise OS's surveillances is to encrypt them when the execution of the Operator Domain is suspended by the OS Switcher. The OS Switcher can also add integrity check data (such as a keyed cryptographic hash) so that it can detect any tampering with the Operator OS's memory data by the Enterprise Domain. During the resumption phase of the Operator OS, the OS Switcher can decrypt and validate the contents of important memory data so that they can be freely used by the Operator OS. Since we assume that domain switching occurs rather infrequently, and the amount of data that must be protected is not large, the overhead involved in the encryption and integrity validation calculations is not significant.

One obstacle to implementing the above mechanism is that software running at the highest privilege levels in the Enterprise OS can do almost anything while it is running. Such software can freely examine or modify all the physical memory, including the area possessed by the Operator OS or the Switcher, thus allowing the secrets used by the OS Switcher for encryption and integrity protection of the important Operator Domain data to eventually be recovered if they are stored in memory. Storing such secrets in independent tamper-resistant hardware (such as smartcard or TPM) cannot help much, because the Enterprise Domain can use such independent hardware to recover the encrypted contents.

One countermeasure to mitigate such threats is the use of so-called obfuscation techniques, which rather than storing secrets in memory, regenerate them as they are needed using a complicated algorithm that is hard for an attacker to reverse-engineer. If encryption and validation checks are used to protect Operator Domain data when the Enterprise Domain is active, this specification mandates the use of such obfuscation techniques to make it harder for an attacker to discover or reproduce the secrets used to provide that protection. A variety of such obfuscation techniques are available, and the selection of which techniques to use is up to the Operator and the platform vendor.

4.3.1.2 TrustZone®-based Protection

ARM processors that incorporate the TrustZone® architecture can partition the system into two domains, one called Secure and another called Non-Secure. At any given time, the processor operates either in the Secure domain or the Non-Secure domain, and can switch between the two of them using a mechanism similar to a software interrupt. The Secure domain has complete control of all platform resources accessible to the processor. The Non-Secure domain, however, can only access those resources that the TrustZone mechanisms allow it to access. The TrustZone mechanisms can be configured only by privileged code executing in the Secure domain. By using the Non-Secure domain for the Enterprise Domain, and the Secure Domain for the Operator Domain, TrustZone can provide the protection called for in section 4.3.1.

Note that the "Secure" domain is the "legacy" domain: the processor boots into the Secure domain, and if the software executing on the processor is not written with TrustZone in mind, it will execute on a TrustZone-enabled processor just as it would on a non-TrustZone-enabled processor. The key novelty of TrustZone is the Non-Secure domain, which is a new de-privileged level of execution.

Mechanisms on a TrustZone-enabled SoC can partition available on-chip and off-chip RAM into Secure and Non-Secure regions. Programs executing in the Enterprise/Non-Secure domain can only read or write memory that is designated as Non-Secure, while programs executing in the Operator/Secure domain can read or write any memory. Note that this means that memory-based communication between the Operator and Enterprise domain will use Non-Secure memory.

Mechanisms on a TrustZone-enabled SoC also allow making some or all peripherals Secure, and thus limit which peripherals Non-Secure programs can directly use. The TrustZone architecture can also eliminate the Non-Secure domain's ability to mask some interrupts, and can also configure the processor to direct interrupts to the Secure domain for handling even if the NonSecure domain is currently active.

TrustZone is exclusive to ARM architecture processors, but other processor architectures may also provide similar capabilities to build isolated domains, which could be used in a similar fashion.

4.3.2 OS Switching Control

Because of its mutual exclusiveness nature of the OS Switching architecture, when the Enterprise OS is active, the Operator OS is in a sleep state. Network Operators desire that the Enterprise OS should pass control to the Operator OS when the Enterprise OS detects events that should be handled by the Operator Domain. Examples of such events could include receiving a call or receiving a Short Messaging Service (SMS) message. But in some circumstances the Enterprise Domain may desire that the switching to the Operator Domain be temporarily disabled, to maintain active services such as a VoIP call in the Enterprise Domain.

To meet these requirements, the DAL provides an interface to receive OS Switch Requests including information of events that should be handled by the Operator Domain, and to receive Switch Lock/Unlock Requests. Both types of requests have an associated priority and a "queue-able" field specifying whether the request should be queued for later handling if it can not be handled immediately. How the "queue-able" field is treated is determined by operator/vendor policy, and the field may be ignored. If the DAL does not handle or queue a request, it must return a failure code to the requestor.

When the DAL receives a Switch Lock/Unlock Request that demands locking or unlocking the switching to the Operator Domain, it updates the status of the lock. If the Switch Lock Request removes the highest priority lock or reduces its priority below that of a pending OS Switch request, then the DAL will determine whether an OS switch is required according to the priorities and policies in the same manner as when an OS Switch Request is received.

When the DAL receives an OS Switch Request, it invokes the mechanisms that will deliver the information of the event that caused the request to the Operator Domain. These mechanisms may use Operator OS / Enterprise OS data communication functionality described in 4.2.2. Then, the DAL determines whether to switch to the Operator Domain or not, based on priority assignment and event-based policy. Next, if the decision is 'switch', the DAL orders OS switching to the Operator OS. Once switching is completed and the Operator OS is resumed, the Operator OS handles the event.

When an event triggers an OS Switch Request that causes a switch to the Operator Domain, the Operator Domain may optionally handle any other queued OS Switch Requests, and may optionally return control to the Enterprise Domain once some or all such requests are handled, possibly contingent on whether there are any outstanding locks. Which options are taken is determined by Operator Domain policies that are outside the scope of this document. The Enterprise Domain should therefore not assume that the Operator Domain will implement any particular policy regarding how many requests are handled once a switch is triggered, or how quickly control will return to the Enterprise Domain.

Optionally, the DAL may support using callback functions to inform the Enterprise OS that an OS Switch Request has been fulfilled. The DAL always provides a means by which the Enterprise OS may attempt to register a callback function, but if the DAL does not support this capability, then any attempt to use this interface will return an error.

5 VMM Architecture

5.1 Enterprise OS Launch, Pause and Resume

These functions are the responsibility of the VMM, and will vary according to the VMM technology used. The architecting of these functions therefore falls to the VMM vendor and is outside the scope of this specification.

5.2 Enterprise OS System Management Interface

5.2.1 Platform Management Functions

5.2.1.1 Interrupts

In addition to the normal interrupt management and servicing requirements imposed on the VMM, the VMM must also guarantee that interrupts used by the Operator Domain, and the events triggered by the user's request to switch domains, are guaranteed to be serviced in a timely fashion regardless of what may be happening in the Enterprise Domain.

5.2.1.2 Power States

In an implementation using a VMM, the background OS may or may not be suspended. The VMM should provide the option of suspending the background domain, and provide the Enterprise OS with properly-functioning device power management commands for all the devices visible to the Enterprise OS that are not essential to the VMM's own operation.

5.2.2 Operator OS / Enterprise OS Data Communication

The Enterprise OS may exchange messages with the Operator OS via an asynchronous memory-mapped message passing facility. The OSTI specification will define the semantics of these variable-length messages, but will not specify the functionality that the Operator Domain provides via this interface. Multiple messages may be outstanding simultaneously. All messages are acknowledged with at least a success or error indication.

The semantics of this capability will be the same regardless of whether the implementation uses OS Switching or a VMM, and are outlined in section 4.2.2.

5.2.3 Storage

5.2.3.1 Fixed (i.e. built-in) Persistent Storage

The DAL will provide the Enterprise Domain with access to portions of the platforms built-in persistent storage, such as Flash devices, hard disk drives, or battery-backed up RAM. The amount of fixed persistent storage made available to the Enterprise Domain will be constant. The format and file system used on this persistent storage may differ from those used on the persistent storage allocated to the Operator Domain. The Enterprise Domain should not attempt to access fixed persistent storage beyond that allowed by the DAL. DAL providers should enforce this restriction.

5.2.3.2 Direct-access memory (RAM, etcetera)

The memory map provided to the Enterprise OS at launch will inform the Enterprise OS of what memory it may access. The Enterprise OS must not attempt to access memory addresses outside those ranges permitted to it by the memory map, nor permit applications to do so.

5.2.3.3 Removable Storage

Removable storage devices may only be usable (i.e. owned) by one Domain at a time. The Enterprise Domain can enumerate certain removable storage devices currently allocated to the Operator Domain, and can request the Operator Domain to relinquish the device so that the Enterprise Domain can use it. This request may be denied by the Operator Domain. Similarly, the Operator Domain may request that the Enterprise Domain relinquish control of a removable storage device it is using. The Enterprise Domain may not deny such a request. Ownership is managed using the facility described in section 5.3.1.

The DAL will allow the Operator to configure the system so that some or all removable storage devices are only visible to the Enterprise Domain, or only visible to the Operator Domain.

5.2.4 Peripherals

5.2.4.1 HID (keyboard, cursor, touch screen, display, sound in/out)

Human Interface Devices (HIDs) are divided into three classes: core HID, shared HID, and on-demand HID.

The core HID consists of the devices whose ownership belongs exclusively to the domain that is currently the foreground domain. The core HID includes, at minimum, the main application area of the primary display, the keypad except for the domain switching key, and the touch panel. It could also include other devices such as a secondary display or devices which are typically in other classes.

The second class is shared HID whose ownership is shared simultaneously by both foreground domain and background domain. Typical shared HID include sound output and vibration devices.

The third class is on-demand HID, whose ownership is transferable between foreground domain and background domain on a demand basis. Typically on-demand HID may include the camera, fingerprint reader, sound input, etceteras. Note the ownership of these devices is individually transferable. Refer to section 5.3.1 for the details on the mechanism used for ownership transfer.

For a particular phone the exact partition of HID classes are determined by the platform vendors or operator. For one extreme example, all devices can belong to core HID while no device belong to shared HID and on-demand HID, the end result being the same user experience as in an OS switching based platform.

The effects of ownership switching of a HID to the Enterprise Domain depend on the implementation of the virtual driver for the HID, which is part of the DAL implementation. If the virtual driver conceals any impact of ownership changes (such as a denial of device access to the Enterprise Domain) the Enterprise Domain can deal with the device as usual. Otherwise, the virtual driver may return errors when ownership is lost. The Enterprise Domain should properly handle any such device errors from virtual drivers. Virtual drivers may also provide mechanisms that can notify an application if ownership of a device it is using is lost.

Note that the ring tone or vibration that announces an incoming call must be audible regardless of which domain is currently in the foreground. However, there are special modes (e.g., silent mode) that have higher priority than this requirement.

5.2.4.2 Real-Time Clock (RTC)

The DAL provides a Real-Time Clock (RTC) device to the Enterprise Domain. The Enterprise Domain will be able to control its RTC, but will not be able to affect the Operator Domain's RTC. The Enterprise Domain's RTC will keep time correctly regardless of domain switching.

There are many ways that an implementation may fulfill this requirement. For example, if platform has more than one RTC, the Enterprise Domain may be allocated an actual RTC for its

exclusive use, which it may manipulate at will. On other platforms, the RTC provided to the Enterprise Domain may be a virtual device that merely scales and adds an offset to the hardware RTC, and the Enterprise Domain will only be able to alter the offset and scale factor.

5.2.4.3 Network connection

The DAL shall provide a virtualized networking interface to the Enterprise domain. The DAL implementer can decide on the level of abstraction presented by this interface (e.g., emulation of an existing Ethernet controller, PPP over a virtual serial port, or emulation of high-level IP devices). Platform providers decide which underlying networking connections are to be used to provide the networking services and how both domains share those connections. For example, the platform may incorporate a system-wide NAT router that allows both domains to share a single network connection.

5.2.4.4 SIM

A SIM consists of persistent storage and optionally an externally accessible cryptographic engine. Enterprise Domain access to the SIM's persistent storage will be handled in the same manner as access to any other fixed persistent storage would be, as described in section 5.2.3.1. If the SIM has an externally accessible cryptographic engine, Enterprise Domain access to that engine will be provided in the default manner used for most peripheral devices, as described in section 5.2.4.5.

5.2.4.5 Other Devices

Handling of other devices is left to the VMM. Those devices that can only be used by one OS at a time will have their ownership managed through the Peripheral Ownership Switching facilities described in section 5.3.1.

5.3 VMM Specific Consideration

5.3.1 Peripheral Ownership Switching

Peripherals can be categorized into three classes, depending on whether and how they can be shared. Some peripherals can be shared with both domains simultaneously, such as sound output. Other peripherals can only be used by one domain or the other, and are not shared at all. Finally, some peripherals, such as input devices, can be used by either domain but only by one of the domains at any one time. This section describes how the ownership of this last class of peripherals is managed. The architecture is similar to the Domain Switching mechanisms described in section 4.3.2.

Locking or switching ownership of the core HID (described in section 5.2.4.1) is very similar, from the user's perspective, to the OS locking and OS Switching that can be done on a platform using the OS Switch technology. The granularity of peripheral ownership is determined by vendor/operator policy, except that the peripherals that make up the core HID must always be managed as a group. Other peripherals in this class can be managed individually or in groups, or all peripherals may be managed together, producing a user experience similar to that provided by an OS Switch-based platform.

The DAL provides the interface to receive Ownership Switch Requests and Ownership Lock/Unlock Requests. Both types of requests have an associated priority and "queue-able" field specifying whether the request should be queued for later handling if it can not be handled immediately. How the "queue-able" field is treated is determined by operator/vendor policy, and the field may be ignored. If the DAL does not handle or queue a request, it returns a failure code to the requestor.

When the DAL receives a Ownership Lock/Unlock Request that demands locking or unlocking the ownership of a peripheral, it updates the status of the lock. If the Ownership Lock/Unlock Request removes the highest priority lock or reduces its priority below that of a pending Ownership Switch request, then the DAL will determine whether an ownership switch is required according to the priorities and policies in the same manner as when an Ownership Switch Request is received.

When the DAL receives a Ownership Switch Request, the DAL determines whether to switch the ownership or not, based on priority assignment and event-based policy. Next, if the decision is 'switch', the DAL sends a request to the OS that currently owns the peripheral(s), asking it to relinquish ownership. Once that is complete, the DAL informs the OS that will receive ownership that it now owns the peripheral and may proceed with whatever initialization the peripheral requires. The Enterprise OS may not refuse a request from the DAL to relinquish ownership, and mechanisms in the platform will enforce this requirement if the Enterprise OS does not relinquish a peripheral within a reasonable amount of time after the request is issued. The Operator OS, however, is allowed to refuse requests.

Optionally, the DAL may support using callback functions to inform the Enterprise OS that an Ownership Switch Request has been fulfilled. The DAL always provide a means by which the Enterprise OS may attempt to register a callback function for this purpose, but if the DAL does not support this capability, then any attempt to use this interface will return an error.

As an example of how this facility might be used, when the Enterprise Domain being in the background receives an incoming call, it can submit Ownership Switching Request to the DAL to get core HID and sound I/O peripheral access. If that ownership is granted, the Enterprise Domain becomes the foreground domain, since foreground/background is determined by core HID ownership. The Enterprise Domain may then present caller information and output a ring tone. During the phone call, it may request the DAL lock the sound input device so that the phone application can maintain the access to those peripherals (and thus continue the call) even if the Operator Domain becomes the foreground domain.