Architecture Guide: Intel® Active Management Technology

Published On: Thursday, February 15, 2007 | **Last Modified On:** Wednesday, September 19, 2007

Intel® AMT Capabilities Overview

Intel provides software developers with excellent support to take advantage of the next-generation manageability capabilities of Intel® Active Management Technology (Intel® AMT). This overview introduces developers to the hardware, firmware, and software architecture that underlie Intel AMT, preparing them to get started with the technology.

Intel® Active Management Technology (Intel® AMT) is a silicon-resident management mechanism for remote discovery, healing, and protection of computing systems. It provides the basis for software solutions to address key manageability issues, improving the efficiency of remote management and asset inventory functionality in third-party management software, safeguarding functionality of critical agents from operating-system (OS) failure, power loss, and intentional or inadvertent client removal:

- Remotely Discover Computing Assets in Any Operational State: Intel® AMT stores hardware asset information in flash memory that can be read anytime, even if the PC is powered off or has an inoperable OS. Intel AMT does not rely on software agents to prevent accidental data loss. It also provides management applications with a general-purpose, non-volatile data store that accepts local or network-based storage commands.
- Remotely Heal Computing Assets: Proactive alerting notifies IT of a system problem, even when the system is down. Intel AMT provides out-of-band (OOB) access to remotely diagnose, control, and repair PCs after software, OS, or hardware failures. Alerting and event logging assists IT to diagnose problems quickly to reduce end-user downtime. Intel AMT also supports IDE-Redirection and Serial-Over-LAN capabilities for management applications.
- Remotely Protect Computing Assets: Through Out of Band communication, each system's software version numbers are checked and, if necessary, system software and virus protection are remotely updated with the most recent patches and virus definitions. Viruses and worms can also be contained at their source, if needed, by means of built-in circuit-breaker functionality.

Intel AMT infrastructure supports the creation of setup and configuration interfaces for management applications, as well as network, security, and storage administration. The platform provides standards-1.1 based encryption support by means of Transport Layer Security (TLS), as well as robust authentication support via Kerberos.

1.1 Intel AMT Use Case Features

The technical capabilities and business value of Intel AMT are summarized in the use cases linked to the descriptions below:

Use Case	Purpose	Intel AMT Features Implemented (Typical)
UC1 (Discover): Platform Auditing	Reduce or eliminate manual inventory audits by being able to locate systems regardless of power state or health. Improve asset management.	Out of Band (OOB) access, Power Status Control/ Monitoring, Intel® AMT Flash, Remote platform inventory, Tamper-resistant agent, Network Admin Interface
UC2 (Discover): Software Inventory	Improve the software-inventory process; optimize maintenance contracts, licensing, and configurations inventory through firmware (FW) resident SW info.	Out-of Band (OOB) access, Remote software inventory, 3rd Party Data Store,Tamper-resistant agent, Network

Management		Admin Interface
UC3 (Discover): Hardware Inventory Management	Reduce manual audits and better manage hardware inventories, recalls, warranties. Efficiently manage hardware inventories.	Out-of Band (OOB) access, Intel® AMT Flash, Remote Hardware Inventory, Tamper-resistant agent, Network Admin Interface
UC4 (Heal): Remote Diagnosis, Remote Repair	Remotely diagnose and repair client machines, reducing on-site visits to resolve SW problems, even when OS is down.	Out-of Band (OOB) access, Remote troubleshooting and recovery, Tamper-resistant agent, Alert Handling, Read Event Logs, Network Admin Interface
UC5 (Heal): Remote Diagnosis, Local Repair	Reduce visits to resolve HW problems with improved remote diagnosis and hardware information.	Out-of Band access, Remote troubleshooting and recovery, Remote field-replaceable unit(FRU) inventory, Intel® AMT Flash, Tamper-resistant agent, Event Logs, Alert Handling, Network Admin Interface
UC6 (Protect): Software Version Compliance	Ensure up-to-date software versions, virus signatures, etc. Improve accuracy, speed and efficiency of anti-virus software updates regardless of OS or power state.	Out-of Band (OOB) access, IDE-R/SOL, 3rd Party Data Storage, System Defense, Agent Presence, Alert Handling, Read Event Logs, Network Admin Interface
UC7 (Protect): Hardware-based Isolation and Recovery	Detect and stop malware from propagating. Suspicious activity detected at a node, alert sent to console, IT quarantines system and updates policy out of band. Monitors out-bound traffic by comparing a timeslice of network traffic to enhanced filters in the system defense engine to obtain data on the timeframe and number of occurrences of a particular network traffic event.	IDE-R/SOL, System Defense, Alert Handling, Read Event Logs, Network Admin Interface, Wired and/or Wireless Network Filters, Flash Memory for Enhanced Filter Storage, Worm-Detection Filters
UC8 (Protect): Presence Checking of User Partition Agents	Virtually eliminate the ability of users or malware to circumvent protection. If the user disables agents, that action triggers alerts, quarantines the system, and re-initializes agent.	Agent Presence, Alert Handling, Read Event Logs, IDE-R/SOL, Network Admin Interface
UC9 (Protect): Endpoint Access Control (EAC)	Limit network access by visitor, rogue systems, and systems that do not conform to company policies for virus protection, OS patches, etc. Force systems that do not meet corporate policy onto a remediation network.	NAC server plug-in to read posture, verify AMT signature and return health statement; posture is created by Intel AMT firmware from system and BIOS data and then given to the Intel AMT Posture Plugin in Host OS
UC10 (Configure): One-Touch Configuration	Perform automated setup and configuration of an Intel AMT device, either using credentials stored on a USB key storage device or by keying credential information manually into BIOS.	Intel AMT firmware image, LMS driver, MEI driver, Intel Setup and Configuration Service (if a corresponding service is not provided by third-party software)
UC11 (Configure): Remote (Zero-Touch) Configuration	Automatically set up and configure an Intel AMT device upon connection to the network, either using a third-party management software agent resident on the client OS or from a 'bare-metal' state, without requiring a host OS.	Intel AMT firmware image, LMS driver, MEI driver, Intel Setup and Configuration Service (if a corresponding service is not provided by third-party software)

2. Intel AMT Hardware Architecture

Intel AMT's core hardware architecture is resident in firmware, as shown at a high level in Figure 1. The micro-controller within the chipset's graphics and memory controller hub houses the Management Engine (ME) firmware, which implements various services on behalf of management applications. Flash memory houses system BIOS, code used by the management engine, and a third-party data store (3PDS) that enables applications to store information as needed in non-volatile memory.

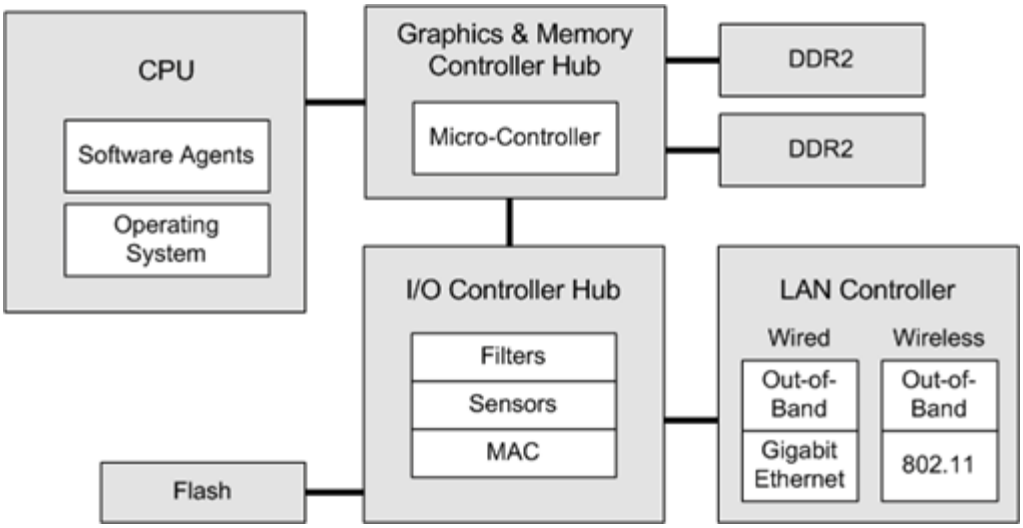

Figure 1. Intel AMT silicon architecture

The ME, which resides in the micro-controller within the graphics and memory controller hub, is shown in more detail with associated architectural components in Figure 2. Note that the ME runs on auxiliary power and is available at all system power states (S0-S5). The shared SPI interface allows multiple masters to use a single FLASH device, including BIOS, firmware, 3PDS, and communications.

While one of the key usage models for Intel AMT is that it allows management applications to access client computers when they are in a powered-off state, the radio in a wireless network interface card (NIC) is typically not operational in power states other than S0. Thus, no wireless Intel AMT functionality is available when laptops are powered down or in low-power modes (sleep, hibernate, etc.).

Note: *Intel AMT Releases 2.5 and 3.0 are concurrent releases, with Release 2.5 supporting wireless capabilities on mobile platforms and Release 3.0 supporting wired PCs. For complete details about the capabilities of each Release, see the Intel® Active Management Technology SDK Start Here Guide. For details about Intel AMT wireless functionality, see "Technical Considerations for Intel® AMT in a Wireless Environment."*

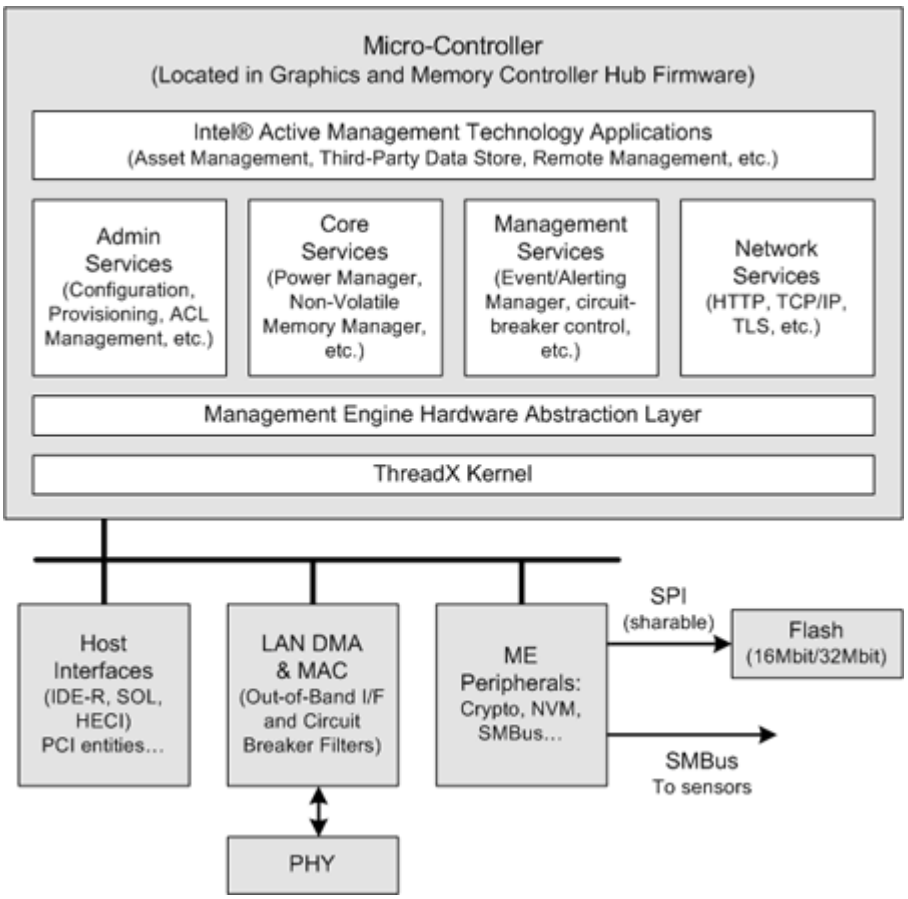

Figure 2. Management Engine (ME) architecture

2.1 ME External Memory (UMA)

A small amount of main memory (typically less than 1% of total system memory) is dedicated to execute ME code and store ME run-time data. This characteristic is similar in concept to UMA for Graphics, and this memory will be located adjacent to the Graphics UMA memory space. From the OS's perspective, the Graphics UMA space will simply appear to be slightly larger. ME code is stored compressed in Flash, so no hard drive access is required to make use of it.

The chipset protects this memory range from being accessed by the main CPU, preventing the ability of malicious software to access this space. Note that this space is always taken from memory channel 0; thus, the channel 0 DIMM slot must be populated. If memory slot 0 is not populated, no UMA is available to the ME.

The ME can access its dedicated memory space even when the system is in S3 state, and the graphics and memory controller hub can dynamically switch memory power state to allow ME access. This capability allows for low average power, since the memory is 'on' only when needed.

2.2 LAN Out-of-Band Communications Architecture

Intel AMT provides for remote communication of PCs with a central management console via SOAP, regardless of power state and OS condition, as shown in Figure 3. This mechanism allows the ME firmware to share a common LAN MAC, hostname, and IP address with the OS, helping to minimize the IT infrastructure cost to support functionality based on Intel AMT.

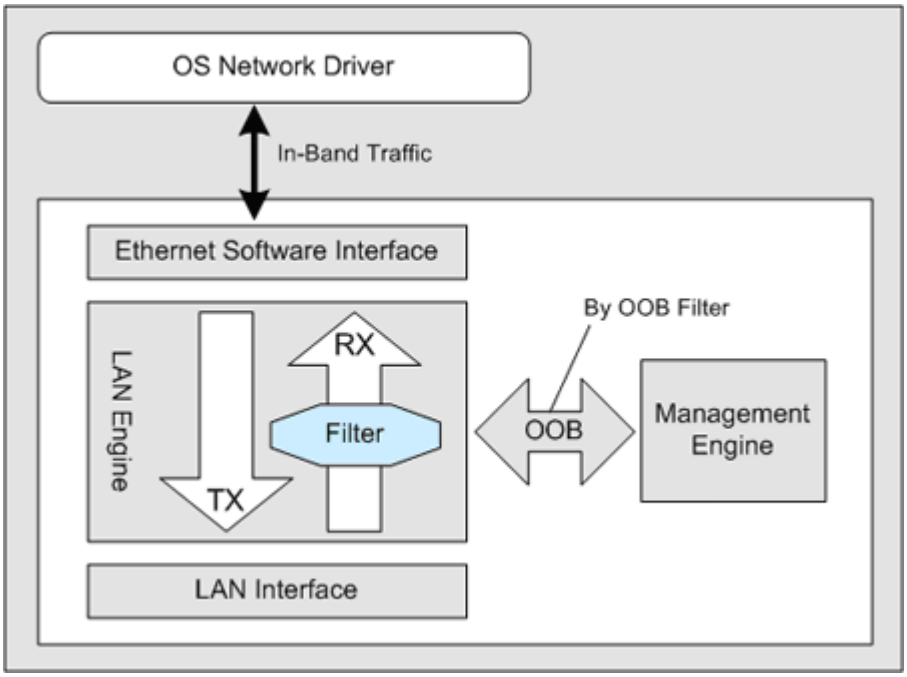

Figure 3. Intel AMT out-of-band communications architecture

The out-of-band communications architecture supports the following filters:

- **ARP:** Forwards ARP packets containing a specific IP address to the host and/or the micro-controller
- **DHCP:** Forwards DHCP Offer and ACK packets to the host and/or the micro-controller
- **IP Port Filters (HTTP and Redirection):** Redirects incoming IP packets on a specific port to the micro-controller

2.3 Intel AMT Host and Network Access Framework

As shown in Figure 4, communication between the host OS and the ME is accomplished by means of the Host Embedded Controller Interface (HECI). HECI is bi-directional, and either the host or Intel AMT firmware can initiate transactions. In addition, transactions can be completed asynchronously by the firmware and then synchronized later.

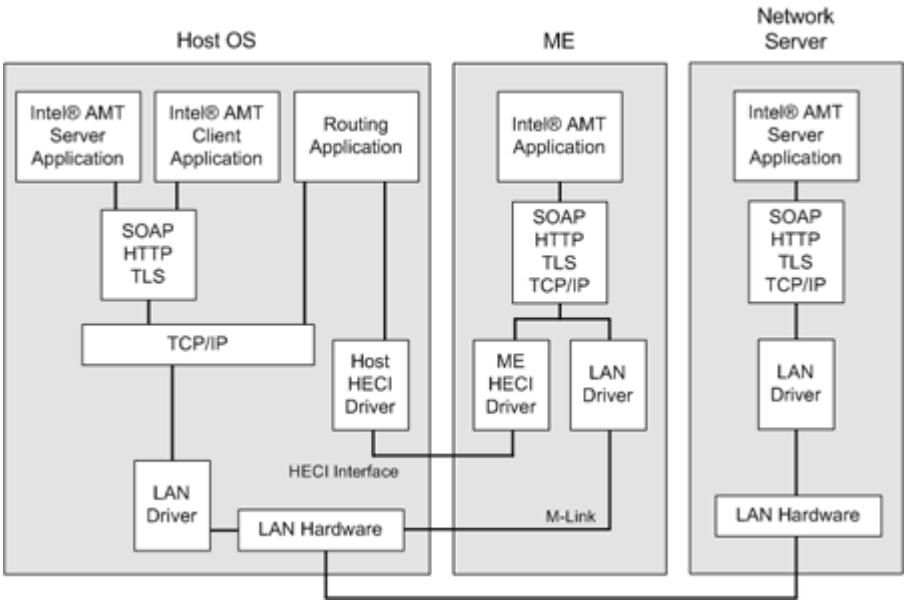

Figure 4. Intel AMT host and network access

Message flow between one client pair does not impede message flow between a separate client pair, and messages may be of any length, subject to the limitations of the client's receive buffer (rather than limitations of the HECI drivers). HECI software and firmware drivers can break messages into packets in order to support long messages. Flow control is communicated by HECI bus messages, and the HECI driver will not transmit a message until the associated client has a buffer ready to receive it.

2.4 Shared Flash Architecture and Permissions

The Flash memory associated with Intel AMT is shared by multiple masters (Host, ME, and LAN). The Flash protection scheme does not allow any master to perform a direct write to Flash, and read/write permissions to each Flash region are enforced in hardware. Each master has a Grant Override register that can override its descriptor permissions, giving other masters access to the region they own. A security-override strap is used during initial manufacturing and service returns to program (or re-program) the Flash.

Region boundaries are defined for BIOS, ME, GbE, and the Flash Descriptor. Master requester IDs are defined for BIOS, GbE, and ME, and read/write access is defined for each master in each region. The I/O controller hub hardware reads the Flash Descriptor at offset 0 at power-on reset. A 32-bit Flash signature is used to determine whether the system is operating in Descriptor Mode (with security). If an invalid signature is read, Descriptor Mode is disabled, and any master can have access to the entire Flash.

2.5 Third-Party Data Storage (3PDS)

Intel AMT provides a general-purpose non-volatile data store for use by applications that provides security equivalent to that provided by the OS for the file system. This data store is not a trusted-platform module; it is provided through a Storage Manager implemented in the ME firmware.

The data store accepts storage commands over local host and network interfaces. Applications are uniquely identified using a concatenation of strings selected by the software vendor and platform owner, plus a unique user ID. It uses allocation lists to 'over-subscribe' the right to allocate, while only allocating actual storage to applications that are registered with the system, protecting the space allocated by one application from other applications unless the owning application grants permission.

The structure, meaning, and sensitivity of data placed into the non-volatile data store is transparent to the Storage Manager. Applications are responsible for any security mechanisms necessary to protect their stored data (e.g., encryption of sensitive data or keys). Applications are also responsible for

backup and recovery of their Application ID, data-store configuration, and any stored data.

The current minimum Flash size is 2MB, defined as the sum of space allocated to BIOS, ME firmware, and 3PDS. It supports partner space for four partners at 48KB each, with no support for non-partner space.

2.6 Management Capability Evolution

The following table summarizes the evolution of management capabilities in Intel AMT, relative to previous-generation management technologies.

Capabilities	Alert Standard Format (ASF): Client	Intelligent Platform Management Interface (IPMI): Server	Intel® AMT
Event Alerting	Yes	Yes	Yes
Event Logging	No	Yes	Yes
Remote Reboot	Yes	Yes	Yes
Secure Communications	Simple Authentication	RMCP+	HTTP Digest Authentication, TLS Encryption
Connection Protocol	RMCP	IPMI = RMCP+Advanced = HTTP	HTTP
Layer 4 Stack	UDP	UDP	TCP
Persistent Asset Information	No	Yes	Yes
OOB Management (OS State Independent)	No	Yes	Yes
Remote Control Capabilities	Remote Reboot Only	Serial Over LAN, KVM (with additional hardware)	Serial Over LAN
Remote Media Capabilities	PXE	IDE/USB Redirect	IDE Redirect
Remote BIOS Update	No	In Some Servers	Yes

3. Intel AMT Platform Security

Intel AMT integrates comprehensive security measures to protect data integrity throughout the system.

3.1 ME Firmware Security: Firmware Image Protection

The primary goal of ME firmware security is to ensure that only Intel-approved firmware images can run on the Intel AMT subsystem hardware, and that only IT administrators can apply approved Intel firmware update images.

During the design phase, a Firmware Signing Key (FWSK) public/private pair is generated at a secure Intel Location, using the Intel Code Signing System. The Private FWSK is stored securely and confidentially by Intel. Intel AMT ROM includes a SHA-1 Hash of the public key, based on RSA, 2048 bit modulus fixed. Each approved production firmware image is digitally signed by Intel with the private FWSK. The public FWSK and the digital signature are appended to the firmware image manifest.

At runtime, a secure boot sequence is accomplished by means of the boot ROM verifying that the public FWSK on Flash is valid, based on the hash value in ROM. The ROM validates the firmware image that corresponds to the manifest's digital signature through the use of the public FWSK, and if successful,

the system continues to boot from Flash code.

3.2 Network and Local Host Traffic Security

Network security is provided by TLS, and XML-encoded messages are encapsulated in SOAP over HTTP. TLS mutual authentication is carried out using the cipher suites TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_NULL_SHA (export/import), and RSA certificates and keys generated off-line and provisioned (2048 bit modulus). Mutual authentication is required by means of preinstalled certificates on both the client and server.

Two local Intel AMT features exist: 3PDS and Local Agent Presence. The traffic between these two features running on the host and Intel AMT goes over SOAP/TLS. The local interface is aligned with network interface security.

3.3 Authentication

Each Intel AMT device must be provisioned with at least one username/password pair, preferably unique. Because it is difficult to guarantee uniqueness, common username/passwords are a potential vulnerability. In response to that challenge, Intel AMT systems support Kerberos in order to achieve integration with Windows* domain authentication. This mechanism is based on a well-accepted set of Internet standards, including Kerberos v5 (RFC 1510), GSS-API (RFC 1964), and SPNEGO (RFC 2478).

This approach simplifies User ID management by using the group-based Windows authorization approach, rather than placing responsibility for creating a new approach on administrators. IT administrators are allowed or denied privileges to manage Intel AMT devices based on their group memberships in Active Directory.

3.4 Intel AMT Wireless Security

Wireless profiles, including network keys and other authentication information, must be programmed directly into the firmware, since the ME cannot synchronize directly with the host-resident set of wireless profiles.

Management applications that support the configuration of the wireless Intel AMT interface must address the variety of security topologies employed by their customers' wireless networks. The Intel AMT wireless management interface does not support open wireless networks, nor does it support Wireless Equivalency Protocol (WEP). Use of Intel AMT wireless connectivity typically requires the use of security included in or related to the 802.11i specification, such as Wi-Fi Protected Access (WPA) or Robust Security Network (RSN). It also optionally supports 802.1x authentication.

Note that, when the wireless Intel AMT interface is initially accessed for setup and configuration, it will by definition not yet have any wireless security profiles configured on it. For this reason, initial setup and configuration of the wireless Intel AMT interface must be accomplished by a wired, rather than a wireless connection.

3.5 3PDS Security

Command Path Security utilizes security mechanisms contained in the local and remote OS network stacks (i.e., TLS with mutual authentication) to secure the path over which an application's storage commands travel. Access to administrative commands is controlled by a separate HTTP authentication ACL (StorageAdministration). Access to registration and storage commands is controlled by another separate HTTP authentication ACL (Storage).

Physical protection and isolation of the Flash device is provided by the chipset hardware. Because Flash devices provide a limited number of write cycles (~100K operations per 4Kb Flash block), the chipset also provides mechanisms to detect and prevent flash wear-out, as well as to prevent Flash wear-out attacks by malware and non-partner applications. This functionality is augmented by mechanisms to prevent Application ID masquerade attacks (ID/interface binding).

3.6 Export / Import Considerations

Intel AMT systems are classified as 'commodity' for purposes of export from the United States, and as such, they are not subject to export restrictions. If an importing country objects to confidentiality, a SKU can be created with confidentiality disabled by setting the silicon fuse CRYPTO_ENA = FALSE, and for TLS, using cipher suite RSA_WITH_NULL_SHA.

4 Intel AMT Interfaces

Intel AMT provides two general types of interfaces: network and local. Network interfaces consist of two types: a SOAP interface and an embedded web user interface. The SOAP interface is the enterprise model, enabling robust functionality designed to be controlled by management console applications created by third-party software makers. The full range of APIs associated with the SOAP interface are documented in the freely available Intel AMT SDK. The embedded web user interface has more limited functionality and is intended for use without enterprise management software, such as in small-to-medium business environments. The local host interface is used by software agents to access 3PDS and to support agent presence.

4.1 Network Interface Overview

The Intel AMT network interface, represented in Figure 5, is OS-independent and always available, assuming that the system is connected to the network and auxiliary power (with the caveat noted above that the wireless management interface is not available in low system-power states). It is manifested as a SOAP-based API based on Web Services Description Language (WSDL) 1.1. Each service supported by the network interface is provided by a distinct WSDL file. Security measures for the network interface include the use of HTTP Digest (RFC 2617) authentication by username/password credentials. The interface also supports TLS-secured connections and mutual authentication. Intel AMT Releases 2.5 and higher also support 802.11x., as well as Cisco Network Access Control (NAC).

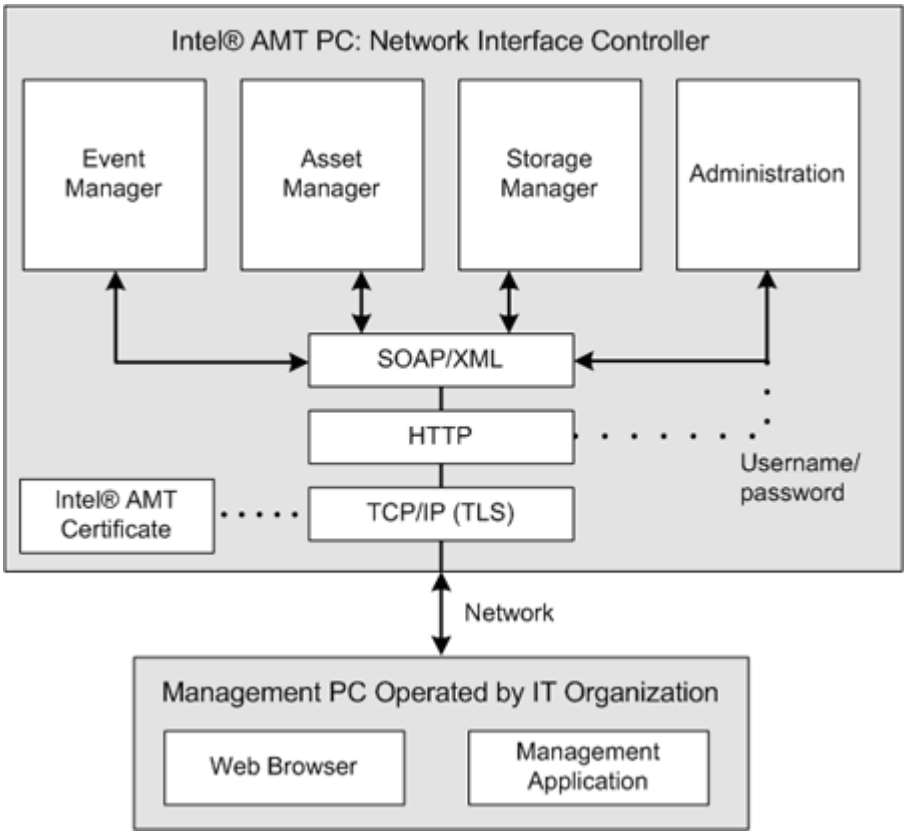

Figure 5. Intel AMT network interface topology

4.2 SOAP API Realms

Realms are a method of partitioning areas of responsibility within the administration of the firmware. Authentication occurs over HTTP or HTTPS, and Access Control Lists are maintained in firmware.

Realm	Controls access to:	Network	Local
Administration	All of the Intel® AMT interfaces	X	
General Info	General Information Interface	X	X
Hardware Asset	Hardware Asset Interface	X	
Remote Control	Remote Control Interface	X	
Event Manager	Event Manager Interface	X	
Redirection	Redirection interfaces (SOL/IDER)	X	
Storage (3PDS)	ISV Storage Interface	X	X
Storage Admin	Storage Admin Interface	X	
Local Agent Presence	Local Agent Presence Interface		X
Remote Agent Presence	Remote Agent Presence Interface	X	
Circuit Breaker	Circuit Breaker Interface	X	

4.3 API Overview

The following table provides an API overview:

GeneralInfo APIs:

The GeneralInfo APIs provides general (read only) information for various (local or network access) management applications.

Method	Description & Compatibility
GetCoreVersion()	Reads the firmware version information from the Intel AMT. Supported in Intel AMT Release 1.0 and later.
GetCodeVersions()	Reads the BIOS and firmware information from the Intel AMT Supported by Intel AMT Release 2.0 and later.
GetProvisioningMode()	Gets the current provisioning mode (Enterprise or Small Business) from the Intel AMT device. Supported in Intel AMT Release 1.0 and later
GetProvisioningState()	Gets the current provisioning (configuration) state from Intel AMT. Supported by Intel AMT Release 2.0 and later
GetVlanParameters()	Gets the VLAN mode and ID used by the Intel AMT device. Supported by Intel AMT Release 1.0 and later
	Gets the host name currently used by the Intel AMT device.

GetHostName()	Supported by Intel AMT Release 1.0 and later
GetConfigServerInfo()	Gets Configuration Server Information from Intel AMT. Suuported by Intel AMT Release 2.0 and later
GetAdminAclEntryStatus()	Reads Admin ACL entry status from Intel AMT. Supported by Intel AMT Release 2.0 and later.
GetAdminNetAclEntryStatus()	Reads remote Network Admin ACL entry status from Intel AMT Supported by Intel AMT Release 2.0 and later.
GetPasswordModel()	Gets the BIOS password mode of work from Intel AMT. Supported by Intel AMT Release 2.0 and later
GetEnabledInterfaces()	Gets enabled interfaces information of Intel AMT device Supported by Intel AMT Release 2.0 and later
GetNetworkState()	Reads Network State information from Intel AMT Supported by Intel AMT Release 2.0 and later
GetSecurityParameters()	Reads local interface security parameters. Supported by Intel AMT Release 2.0 and later.
GetIderSessionLog()	reads the IDER session log. Supported by Intel AMT Release 2.0 and later.

HardwareAsset APIs:

The HardwareAsset APIs perform operations that return hardware asset data.

Method	Description & Compatibility
EnumerateAssetTypes()	Enumerates the names of hardware asset types supported by the Intel AMT device. Supported in Intel AMT Release 1.0 and later.
GetAssetData()	Returns hardware asset data of Intel AMT device Supported by Intel AMT Release 1.0 and later.

Remote Control APIs:

The Remote Control APIs managing the power and booting state of the Intel AMTmanaged system.

Method	Description & Compatibility
GetRemoteControlCapabilities()	Gets the remote control capabilities supported by the Intel AMT device Supported in Intel AMT Release 1.0 and later.
RemoteControl()	Remotely controls the boot and power state of the Intel AMT-managed PC

	Supported by Intel AMT Release 1.0 and later
GetSystemPowerState()	Returns the power state of the Intel AMT-managed PC system Supported by Intel AMT Release 1.0 and later.

ISV Storage APIs:

The ISV storage APIs are used by ISVs to access the Intel AMT non-volatile storage feature

Method	Description & Compatibility
ISVS_GetAPIVersion()	Gets the ISVS API version supported by the Intel AMT device (deprecated since AMT 2.0).
ISVS_GetAPIVersionEx()	Gets the ISVS API version supported by the Intel AMT device. Extended version of ISVS_GetAPIVersion.

SOL Handling APIs:

Function	Description
IMR_SOLOpenTCPSession ()	Opens an SOL session with the specified client over a new TCP connection
IMR_SOLCloseSession()	Closes an open SOL session with the specified client
IMR_SOLSendText()	Sends text (keyboard input) to the client, where it will be received as incoming data from the serial controller
IMR_SOLReceiveText()	Data sent by the client on the serial controller is received by the library and stored in an internal buffer. This function retrieves SOL data that has been stored

IDER Handling APIs:

Function	Description
IMR_IDEROpenTCPSession()	Opens an IDER session with the specified client over a new TCP connection
IMR_IDERCloseSession()	Closes an open IDER session with the specified client
IMR_IDERClientFeatureSupported()	Queries the client about the special features that it supports. Currently the only special feature defined is an ability to disable/enable host IDE devices.
IMR_IDERGetDeviceState()	Queries the state of client IDE devices.
IMR_IDERSetDeviceState()	Controls the client IDE device(s) state. Devices can be disabled and enabled through this function.
IMR_IDERGetSessionStatistics()	Polls the active IDER session

Event Manager APIs:

Event Manager APIs include operations that can be used by a remote application to subscribe for events, set event filters and manage the event log

SubscribeForAlert()	Adds an alert subscription to the Intel AMT device.
EnumerateAlertSubscriptions()	Enumerates alert subscriptions in the Intel AMT device.

GetAlertSubscription()	Gets value of one alert subscription from the Intel AMT device.
CancelAlertSubscription()	Removes an alert subscription from Intel AMT device
EnumerateAlertPolicies()	Enumerates alert policies in the Intel AMT device
SetAlertCommunityString()	Sets the community string in the Intel AMT device PET alerts.
GetAlertCommunityString()	Gets the community string used in the Intel AMT device PET alerts.
AddEventFilter()	Adds an event filter to the Intel AMT device
EnumerateEventFilters()	Enumerates the event filters in the Intel AMT device
GetEventFilter()	Gets the value of one event filter from the Intel AMT device
UpdateEventFilter()	Updates value of one event filter in the Intel AMT device
RemoveEventFilter()	Removes an event filter from the Intel AMT device
GetEventLogStatus()	Gets the attributes of the event log status
ReadEventLogRecords()	Reads all event log records stored in the Intel AMT device.
ClearEventLog()	Clear the event log in the Intel AMT device
FreezeEventLog()	Freezes the event log in the Intel AMT device to prevent modification
SetEventLogTimestampClock()	Sets the time used to timestamp the event log
GetEventLogTimestampClock()	Gets the current time used to timestamp the event log
EnumerateSensors()	Enumerates all sensors controlled by the Intel AMT Device
GetSensorAttributes()	Gets sensor attributes from a sensor controlled by the Intel AMT device
SubscribeForGeneralAlert()	Register to receive a selected alert type
EnumerateGeneralAlertSubscriptions()	Enumerate subscriptions for events created by the user
GetGeneralAlertSubscription()	Returns details of a selected general alert

5. Intel AMT Software Development Kit and Other Developer Tools

The Intel AMT Software Development Kit (SDK) provides tools and capabilities that enable developers to develop manageability applications that take full advantage of Intel AMT. The latest version of the Intel AMT SDK is freely downloadable from http://www.intel.com/cd/ids/developer/asmo-na/eng/321157.htm. The SDK includes the following components:

- Intel AMT Storage Library and API provides abstraction of Intel AMT non-volatile storage calls. The library enables local and remote access to the non-volatile store on Intel AMT machines in all power states, and it includes a static library, headers, and sample code, as well as a C-based API.
- Intel AMT Redirection Library and API provides abstraction of redirection calls, Serial over LAN (SoL) console redirection, and IDE Redirection (IDER). The library enables remote management of Intel AMT machines through SoL and IDER sessions, and it includes a dynamic library (Windows), static library (Linux*), header files, and sample code, as well as a C-based API.
- Intel AMT Network Interfaces include the SOAP-based network interfaces defined in WSDL files.
- Documentation includes a User Guide, Storage Design Guide, Network Design Guide, and Validation Design Guide. Also included are a Redirection Library User Guide, Developer Guide to the sample setup, Small Business Configuration User Guide, and System Defense Feature & Agent Presence Overview.

The Intel AMT SDK can be used with any language that includes a SOAP stack, including gSOAP (C++), ATL SOAP (MS C++, although optional parameters are not supported and a memory leak has been reported), and C#. The SDK requires the Microsoft .NET Framework v1.1 (some samples and other components require the .NET Framework 2.0), and Windows Storage library requires the Microsoft Platform SDK (for WinHTTP).

Additional freely downloadable tools are also provided by Intel for use by developers in creating management applications that support Intel AMT. For a full discussion of these tools, please see the Intel® AMT Developer Resource Guide.

6. Intel AMT Developer Support

Developers creating products that take advantage of Intel AMT are entitled to technical support from Intel. A Manageability developer forum exists for the purpose of providing support for architect/developer questions regarding Intel manageability technologies, including the Intel AMT Software Development Kit (SDK), Developer Tool Kit (DTK), and Setup and Configuration Service (SCS). Intel is in the process of building a community around manageability to help foster growth and promote development efforts. Questions are answered by both peers and Intel representatives that monitor this forum.

7. Additional Resources

- The Intel Manageability Community is a core developer resource for manageability technologies from Intel that provides tools, documentation, use cases, blogs, and user forums.
- Intel AMT Technology & Research provides in-depth information about the hardware and software features and capabilities that underlie Intel AMT.
- Intel AMT Technology Brief (PDF 424KB) provides a concise overview of the technology from a business perspective, with a focus on features and benefits to IT organizations and software vendors.

Post a comment If you have any questions, please contact our support team.