Chapter 8

Discovery of Platforms and Information

Discovery consists of seeing what everybody has seen and thinking what nobody has thought.

—Albert Szent-Gyorgyi (1893–1986)

The act of doing discovery in the context of Intel® Active Management Technology (Intel AMT) means locating, connecting, and acquiring real time information about the remote computer regardless of its power state. Discovery is important because it's generally the necessary starting point for further use of Intel AMT features.

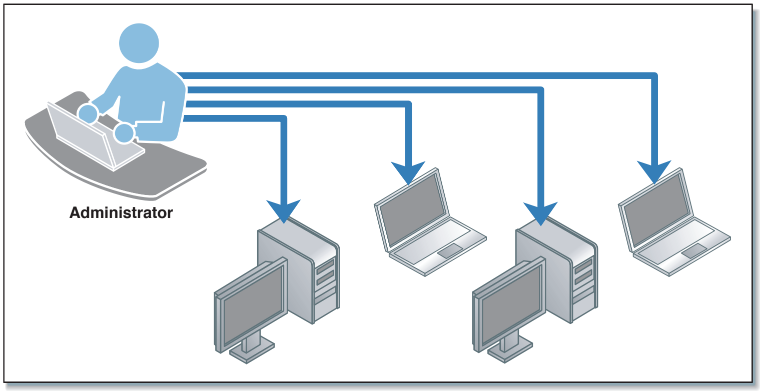


Figure 8.1 Administrators Must First Discover the Computers on the Network before Managing Them

Performing Intel AMT discovery can be summed up in three phases: The first phase involves scanning a network for computers with Intel AMT support. The second phase involves obtaining the list of supported Intel AMT features for this specific computer. The third phase is obtaining available management information from a given computer.

When using modern management software, the first step of this process will rarely have to occur. Most management consoles will have a database of all managed computers that support Intel AMT. This database will grow each time a new computer is provisioned or when a computer is added manually by the network administrator.

The second step is crucial since many different versions of Intel AMT are on the market and with more to come, being able to discover what version of Intel AMT a computer supports becomes increasingly important. Lastly, available management information about a computer is retrieved.

Network Scanning for Intel® AMT

It is sometimes useful to scan a range of IP addresses to find computers that support Intel AMT. Different approaches can be used with varying degrees of efficiency. The basic technique is to attempt a TCP connection to both ports 16992 and 16993, hoping that one of these ports successfully connects.

In the case where port 16992 successfully connects, one can send a basic HTTP get request on this connection and obtain the Intel AMT server in the response’s SERVER field. Figure 8.2 shows a typical HTTP HEAD requests and responses given back by the Intel AMT HTTP server.

HTTP Request to port 16992

```
HEAD / HTTP/1.1
Host: hostname.domain.com

HTTP response from Intel AMT 2.0 computer

HTTP/1.1 303 See Other
Location: http://hostname.domain.com/logon.htm
Content-Length: 0
Server: Intel(R) Active Management Technology 2.1.2

HTTP response from Intel AMT 3.0 computer

HTTP/1.1 303 See Other
Location: http://hostname.domain.com/logon.htm
Content-Length: 0
Server: Intel(R) Active Management Technology 3.0.1
```

Figure 8.2 Discovery Requests and Responses

Note that with Intel AMT 1.0, the version is simply omitted and must be assumed to be 1.0 by the management console.

If port 16992 fails to connect, a connection attempt to port 16993 can be attempted. This time a TLS session could also be attempted. A successful TCP connection to port 16993 could indicate that Intel AMT is unprovisioned and awaiting setup, or that Intel AMT is already set up with TLS security. If a TLS session successfully connects, the same HTTP request performed above can also be accomplished through TLS to determine the computer's Intel AMT information.

Attempting a TCP connection sweep of many IP addresses on a network can take a very long time, especially with the connection limitations included in Microsoft† Windows XP SP2 and Microsoft Vista[1]. The network sweep

---

[1] The TCP/IP stack now limits the number of simultaneous incomplete outbound TCP connection attempts. After the limit has been reached, subsequent connection attempts are put in a queue and will be resolved at a fixed rate. Under normal operation, when applications are connecting to available hosts at valid IP addresses, no connection rate-limiting will occur.—Microsoft TechNet.

can be significantly speeded up by using a combination of ARP and PING. In Figure 8.3 we have a possible state machine for network discovery of Intel AMT computers.

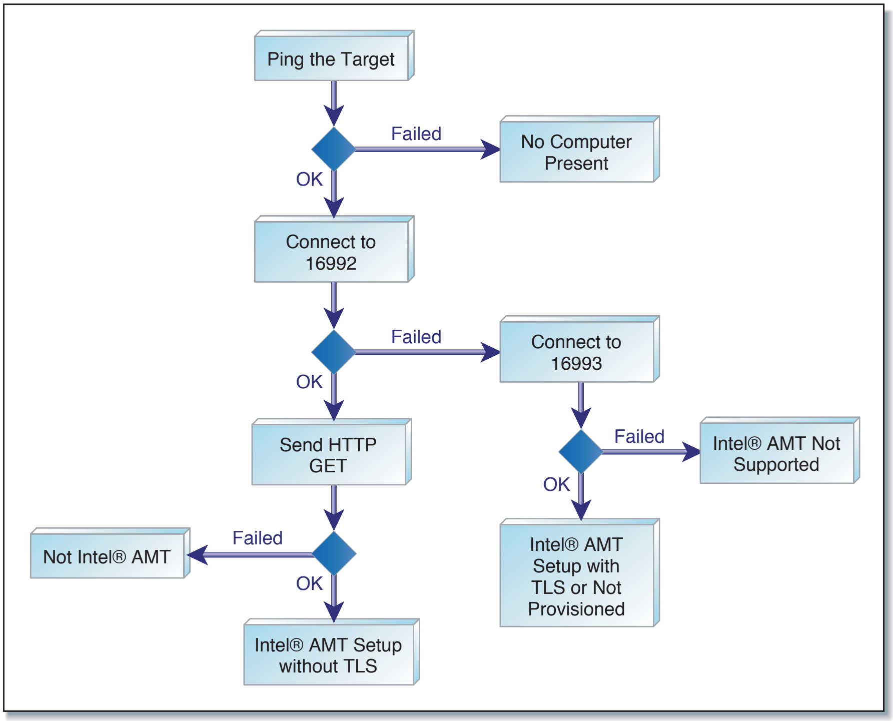


Figure 8.3 Basic Intel® AMT Network Scanning State Diagram

Basic discovery can be performed without having an authorized username and password. Note that this state machine assumes that Intel AMT is set to respond to PING requests, a feature that may be turned off by the administrator. Also, if a computer accepts TCP connections on port 16993, we can't conclude that the computer is provisioned and ready to go. Computers that are not provisioned will expect connections on port 16993 for a provisioning server. Once we know that computer with Intel AMT is present on the network at a given IP address, authentication is required to complete the second part of the discovery process.

Obtaining Intel® AMT Features

Once a management console has determined the presence of Intel AMT it must use appropriate credentials (username and password or Kerberos) to authenticate itself. This done, the second phase of the discovery involves obtaining information necessary to talk to Intel AMT itself. Two critical pieces of information are needed by a management console to talk to Intel AMT. First the management console needs to determine the Intel AMT version and second whether it communicates using the older EOI (SOAP based External Operations Interface) messages or the newer WS-Management based WSDL. Intel AMT 1.x and 2.x support only the older EOI while Intel AMT 3.x to 6.x support both EOI and WS-Management, with EOI having a progressively more limited set of features as WS-Management becomes the standard.

As illustrated in Figure 8.4, the first call a management console should make is the GetCoreVersion method in the SecurityAdmin SOAP service. By analyzing the response to this call, software can determine a lot of information. It should first be noted that the same GetCoreVersion method is also present in the GeneralInfo service, but that service was not present in Intel AMT 1.x, so we will use the one in SecurityAdmin. It is possible for a computer to be in WS-Management-only mode; in this case none of the EOI actions are available and Intel AMT will respond with a 404 “HTTP Not Found” error. Consoles receiving this error should try to switch over to using WS-Management on Intel AMT 3.0 or higher.

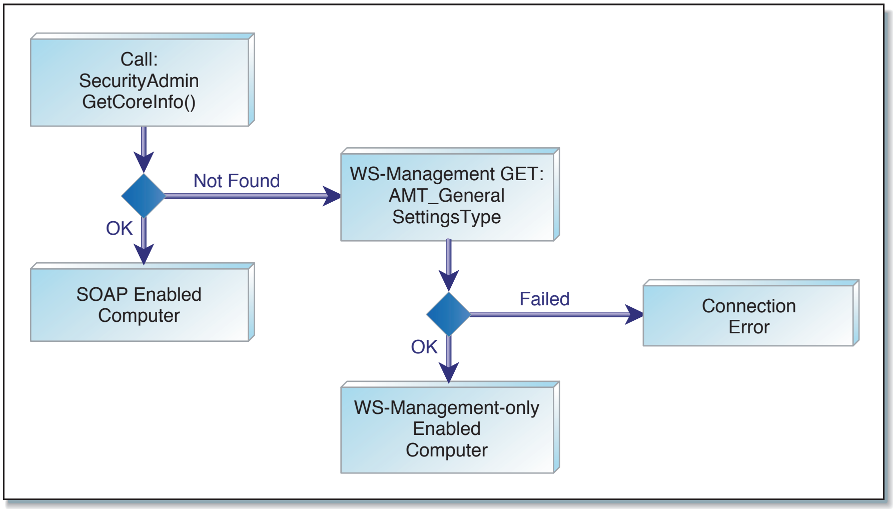


Figure 8.4 Basic Intel® AMT Connection State Diagram

The authentication used by the console could have limited access to Intel AMT features. In other words, the username and password that were used might not have had all security realms associated with it. A console could optionally then try to check if it is logged as administrator by calling GetAdminAclEntry. This method returns the name of the administrator account, generally "admin". If the console is not logged as administrator, the list of realms can be obtained using GetUserAclEntries method.

If at any time a call returns a forbidden error (HTTP 403) it is a good chance that the attempted call is not allowed because the account does not have the correct associated permission realm. The following diagram shows a possible algorithm a console could use to determine available features. For advanced developers, there is also a way to detect that a console is talking to Intel AMT locally, through the LMS service. This is useful in cases when the administrator is performing discovery on his own computer locally. A telltale sign that this is happening is that the call to GetCoreVersion will return an HTTP 401 unauthorized. In that case, other calls can be attempted to determine if a local LMS connection is indeed occurring.

Information about the computer's name, unique identifier, version, protocol, and permissions can be stored in a database for future use. When communicating again with the same computer, this information could be used to skip the first two phases of discovery with an important caveat: Since the Intel AMT settings of a computer could change without notice; it's probably good practice to perform the feature check upon each connection. After all, software developers should not assume that their own software is the only software being used to configure or manage Intel AMT.

Obtaining Management Information

Finally, now that we know more about this Intel AMT computer we are communicating with, we are in a position to start gathering useful management information about this computer itself. Here are four basic categories of management information that could be of use:

- Asset inventory
- Event log
- Power, battery, and lockup state
- Third party data storage

Other management information such as network filter counters, heuristic filter state, and agent presence state will be covered in Chapter 10.

Asset Inventory

One of the most basic features of Intel AMT allows the console to read hardware information about the computer that was gathered by the BIOS upon the last boot up. Hardware assets are often the first features developers write code for when first starting with Intel AMT since it's one of the simplest. These include:

- Computer system
- Base board
- BIOS
- Memory modules
- Hardware modules (such as PCI cards)

- Media devices (such as disk drives)
- Processors
- Batteries (mobiles only)

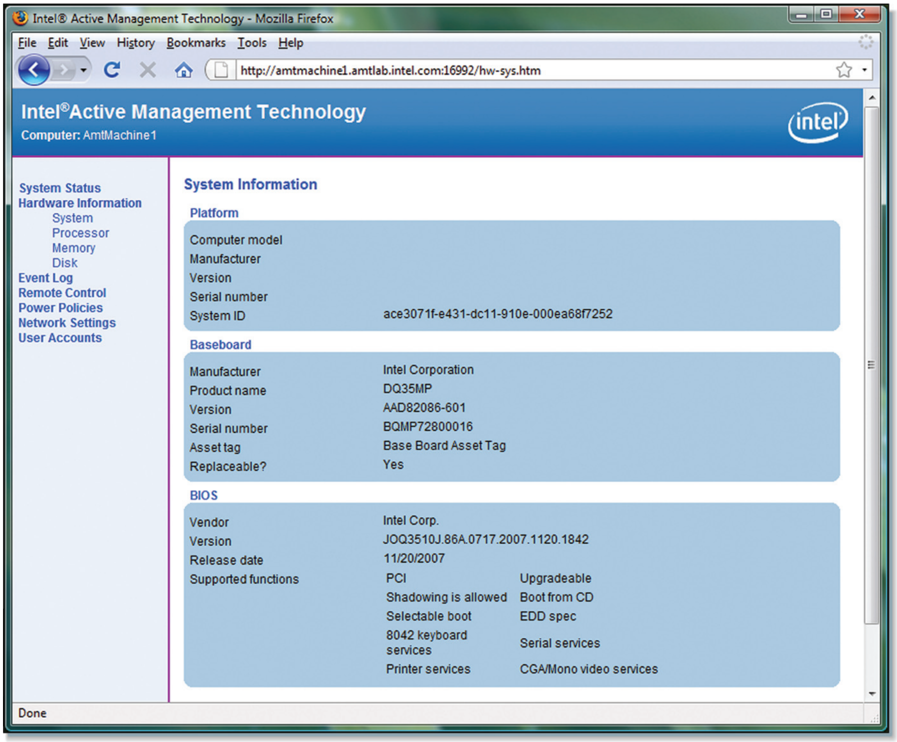
Developers must be aware that it is possible for all of this information to be completely missing in the rare case that Intel AMT was just provisioned and no host reboot has occurred since then. In this case, software should behave appropriately and indicate to the administrator that this information is not currently available.

When using EOI, this information is gathered using the Hardware Asset service. With WS-Management, similar information is obtained through many CIM objects. Examples of how to obtain asset information with both EOI and WS-Management are available in the Intel AMT SDK.

Hardware asset information can change when, for example, memory is changed, a PCI card is added, or the BIOS is updated. Most of these changes require a reboot of the computer, and the updated information will be reflected within Intel AMT once the reboot occurs. When a change to a hardware asset occurs, there are no events logged or notification given. It is therefore the management console's responsibility to occasionally get the hardware asset inventory of each computer and compare with the previously obtained one to see if anything has changed. Determining if such a change has occurred can be very useful to track inventory and detect theft.

Developers often ask if USB or 1394 devices are part of the hardware inventory and no, they are not. Listing these devices and other information can be done using the third party data storage (3PDS) and the assistance of OS software covered later in this chapter.

Figure 8.5 shows one of the Intel AMT Web pages for system inventory. This information is also available to management consoles using a programmatic interface- EOI or WS-Management interface.


Intel® Active Management Technology - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://amtmachine1.amtlab.intel.com:16992/hw-sys.htm
Intel®Active Management Technology
Computer: AmtMachine1
intel
System Status
Hardware Information
System
Processor
Memory
Disk
Event Log
Remote Control
Power Policies
Network Settings
User Accounts
Done

System Information

Platform

Computer model	
Manufacturer	
Version	
Serial number	
System ID	ace3071f-e431-dc11-910e-000ea68f7252

Baseboard

Manufacturer	Intel Corporation
Product name	DQ35MP
Version	AAD82086-601
Serial number	BQMP72800016
Asset tag	Base Board Asset Tag
Replaceable?	Yes

BIOS

Vendor	Intel Corp.	
Version	JOQ3510J.86A.0717.2007.1120.1842	
Release date	11/20/2007	
Supported functions	PCI	Upgradeable
	Shadowing is allowed	Boot from CD
	Selectable boot	EDD spec
	8042 keyboard services	Serial services
	Printer services	CGA/Mono video services

Figure 8.5 Intel® AMT Web Page for System Inventory

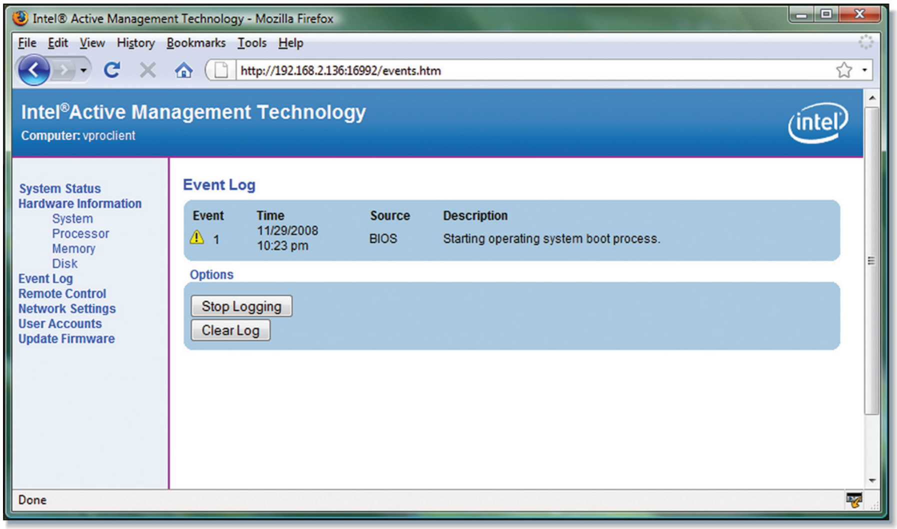
Intel® AMT Event Log

This log provides basic historical information about the computer: reboots, errors, case intrusion, and much more.

The event log can serve as a computer's "black box" and can sometimes be helpful in determining problems that occurred in the past. On most platforms, the event log keeps 390 events and once full, the older events are automatically deleted to make way for new events. For mobile computers the event log allows network administers to get historical data about the computer even if in the past it has not been connected to the office network.

The event log is cleared when Intel AMT is first provisioned and gathers event information from three main sources: Intel AMT, the BIOS, and system sensors. For most computers with Intel AMT on the market today, the case intrusion sensor, if connected, is the only sensor that will cause a logged event, but this can change depending on the board vendor.

It is important to note that events coming from the BIOS vary greatly from one BIOS vendor to another, and so management software should not make assumptions about events coming from the BIOS without first checking the computer's BIOS vendor. There is no formal list of what vendor supports what events. Figures 8.6 and 8.7 show the difference in the events logged between two different BIOS vendors.

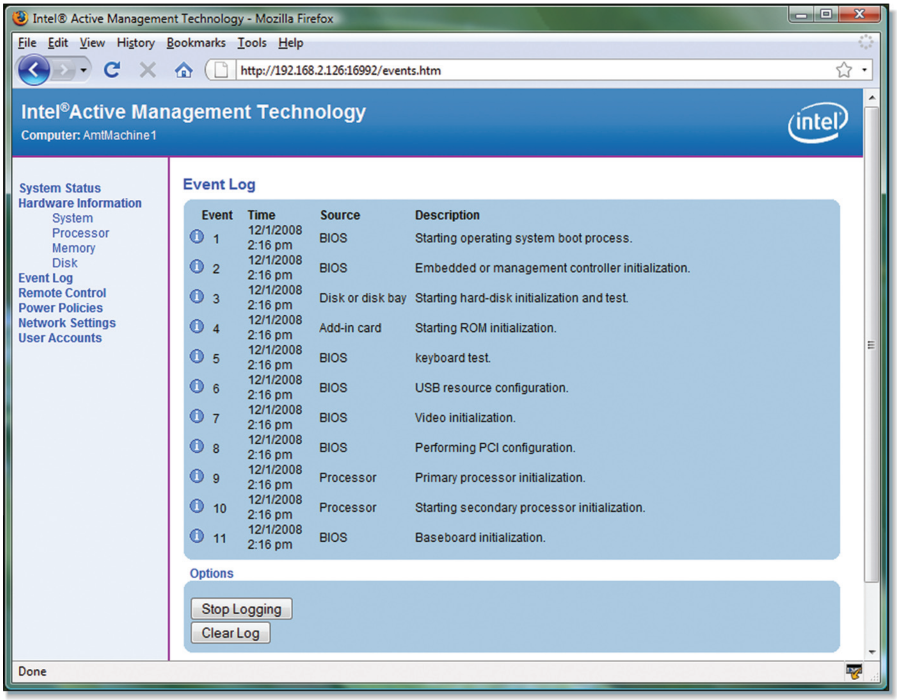


Computer: vproclient

Event Log

Event	Time	Source	Description
⚠ 1	11/29/2008 10:23 pm	BIOS	Starting operating system boot process.

Figure 8.6 Single Reboot on a Third Party Motherboard



Event Log

Event	Time	Source	Description
1	12/1/2008 2:16 pm	BIOS	Starting operating system boot process.
2	12/1/2008 2:16 pm	BIOS	Embedded or management controller initialization.
3	12/1/2008 2:16 pm	Disk or disk bay	Starting hard-disk initialization and test.
4	12/1/2008 2:16 pm	Add-in card	Starting ROM initialization.
5	12/1/2008 2:16 pm	BIOS	keyboard test.
6	12/1/2008 2:16 pm	BIOS	USB resource configuration.
7	12/1/2008 2:16 pm	BIOS	Video initialization.
8	12/1/2008 2:16 pm	BIOS	Performing PCI configuration.
9	12/1/2008 2:16 pm	Processor	Primary processor initialization.
10	12/1/2008 2:16 pm	Processor	Starting secondary processor initialization.
11	12/1/2008 2:16 pm	BIOS	Baseboard initialization.

Figure 8.7 Single Reboot on an Intel Motherboard

Like the hardware asset inventory feature, the Intel AMT SDK has samples for retrieving all events using both EOI and WS-Management. Developers will notice that when retrieving events using EOI, the events are ordered sequentially with the most recent event being retrieved first. This is not the case when retrieving using WS-Management and as a result, developers using WS-Management must sort the list of events using the event's time stamp before displaying the events to the administrator. The time stamp used to mark each event is based on the Intel AMT clock and so management consoles should take care to set the Intel AMT clock correctly. If the computer is provisioned in enterprise mode, Intel AMT makes use of UTC time and the console may need to perform appropriate time zone conversion.

A few administrative operations are allowed on the event log. First, an administrator with the proper rights can clear the event log. Since the event log can grow to be very long and take a long time to retrieve, network administrators may opt to clear the event log from time to time, especially if the log is stored in a central database. The other management operation allowed on the event log is a freeze. In this case, the log no longer records events. This could be useful if the network administrator is about to perform a long series of operations on a computer and does not need this information to be logged into the Intel AMT event log.

One question that is often asked regarding the event log is how does an event entry, which is a short set of numbers, convert to a string readable by humans? A prime example of this is the Intel AMT Web UI, which has a human readable event log.

When reading events using WS-Management, the same human-readable string that is visible in the Web UI is provided. This string is only in English and so does not help with internationalization and contributes to making the event log in WS-Management very slow to retrieve.

With EOI, the event log can be retrieved much faster but no human-readable string is provided. When building the Manageability DTK and Manageability Commander, each new EOI event was manually compared to the Web UI and appropriate code was added to perform the conversion. This also has the benefit of being available in many different languages.

Because of a known bug in some versions of Intel AMT, it's not always possible to retrieve the event log using WS-Management and perform the same value-to-text conversion that is possible when using EOI. As a result, avid international users of Intel AMT Commander will notice that the event log will be displayed in, for instance, Japanese when using EOI, but only in English when using WS-Management.

Intel® AMT Network Alerts

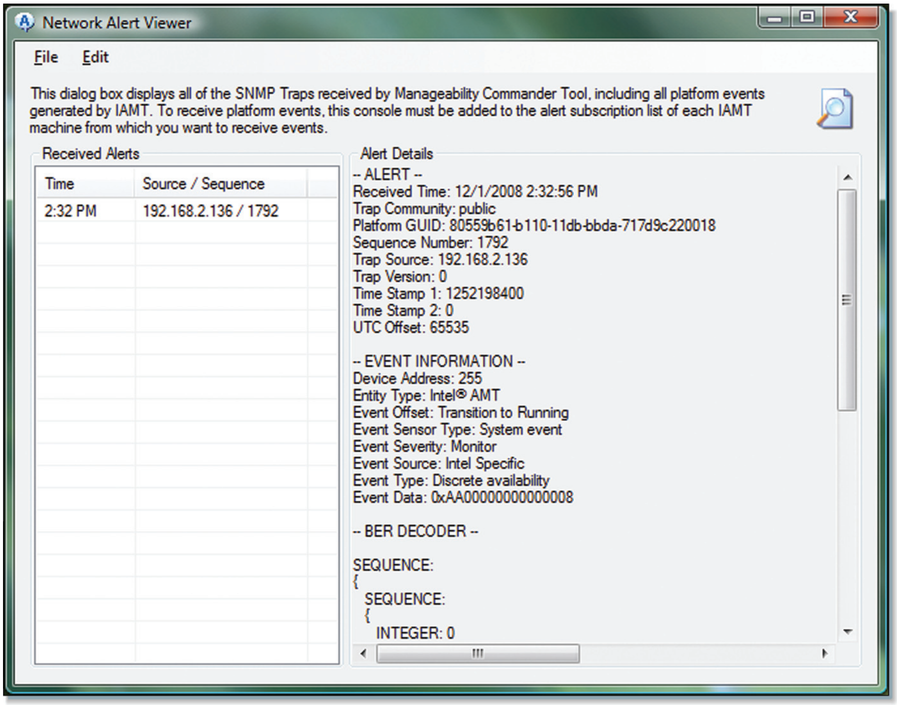
Since we just covered the Intel AMT event log in some detail, it is a good place to talk about the Intel AMT support for network alerts. A console could opt to regularly read the event log and take note of new events, but this would be very inefficient. Intel AMT supports network alerts. A management console can subscribe to network alerts by placing the console's IP address in the Intel AMT alert subscription list along with an event filter. Once this is done, every new event that matches the filter will cause a network alert to be sent to the console.

There are two ways Intel AMT can send a network alert: SNMP traps and WS-Eventing.

SNMP Traps

All versions of Intel AMT support sending alerts using SNMP traps. They are UDP packets sent to port 162 of the console. The packet contains all the basic information about the alert. Since it is UDP and not reliable, the packet is sent three times at a few second intervals with an identical sequence number. The management console must then remember what packets it got from each IP address and what sequence number it already received to remove any duplicates. Because SNMP trap packets are un-authenticated, are sent in the clear, and can be spoofed, they are not considered secure. One solution is to go back to the computer with Intel AMT that sent the alert and read the event log to confirm that the alert truly came from this computer. In any case, SNMP trap alerts should not be used if the administrator does not wish anyone on the network to see these events in plain text. Lastly, it's often assumed that because Intel AMT supports SNMP traps, that it also supports SNMP. This is not the case: Intel AMT does not have any support for SNMP.

In Figure 8.8, we show the SNMP trap viewer built into the Manageability Commander tool. This screen gives a good idea of what information is encoded in the SNMP trap packet. In Manageability Commander, this screen is accessible through the Alert Viewer on the File menu.



This dialog box displays all of the SNMP Traps received by Manageability Commander Tool, including all platform events generated by IAMT. To receive platform events, this console must be added to the alert subscription list of each IAMT machine from which you want to receive events.

Time	Source / Sequence
2:32 PM	192.168.2.136 / 1792

```
-- ALERT --
Received Time: 12/1/2008 2:32:56 PM
Trap Community: public
Platform GUID: 80559b61-b110-11db-bbda-717d9c220018
Sequence Number: 1792
Trap Source: 192.168.2.136
Trap Version: 0
Time Stamp 1: 1252198400
Time Stamp 2: 0
UTC Offset: 65535

-- EVENT INFORMATION --
Device Address: 255
Entity Type: Intel® AMT
Event Offset: Transition to Running
Event Sensor Type: System event
Event Severity: Monitor
Event Source: Intel Specific
Event Type: Discrete availability
Event Data: 0xAA00000000000008

-- BER DECODER --

SEQUENCE:
{
  SEQUENCE:
  {
    INTEGER: 0
```

Figure 8.8 Manageability Commander SNMP Alert Viewer

WS-Eventing

Starting with Intel AMT 3.0, a management console using WS-Management to talk to Intel AMT can subscribe to alerts using the much more secure WS-Eventing standard. Unlike SNMP traps, which are rather simple, WS-Eventing assumes that a Web server is located on the console and ready to receive events in the form of a full HTTP or HTTPs connection. Such alerts are much more involved since they require opening a TCP connection and possibly performing TLS negotiation before the alert it sent to the console. This said, the alert is much more reliable and securely sent to the console.

Event Log and Alert Filters

The event log and network alerts are only as useful as the information they store or carry. In order to prevent excessive events from being logged or alerts to be sent to a management console, Intel AMT supports the concept of an event filter. Up to 16 of these can usually be present and by default, most or all of these 16 slots are populated with default filters.

Event filters can be added, removed, and changed at will by an authorized administrator. They can be used to filter the types of events to be stored in the event log or sent as network alerts. In general, it's best to stick to the default filters if possible. Since, if many consoles and monitoring applications access the same Intel AMT computer, they may conflict in how they want to use alerts and the event log.

In EOI, the event log, event filters, and network alerts are all controlled by the EventManager service. This service and its features are fairly straightforward.

Computer's Power, Battery, and Lockup State

During the discovery phase, it's often useful to obtain the computer's current power state. As we will see in this section, this little piece of information can be very powerful.

Using EOI, the Remote Control service offers a method called `GetSystemPowerState`. In WS-Management the same information is obtained using the Power State property of the CIM_AssociatedPowerManagementService object. In both cases, the management console can retrieve the current power state of a computer. The Intel AMT SDK has a table of possible return values, but in practice, only a few are really used:

- S0 – Fully on
- S1 – Sleeping with power on, devices off
- S3 – Sleeping, suspend to memory
- S4 – Hibernating, suspend to disk, auxiliary power only
- S5 – Off, auxiliary power only

Retrieving the current power state can be important during discovery because other Intel AMT operations do not makes sense when the computer is off.

This information can also be used to track how effective power savings are or make sure that a critical server has not been turned off.

In the Manageability DTK, a sample tool called Intel AMT Monitor can be used to poll the power state of many computers every few minutes. The tool draws a colored graph showing the power state of each computer over time making it easy to determine which computers are staying on all night and which are saving the most power. Intel AMT can't tell the administrator exactly how much power is being used by a given computer; the exact power use of a computer changes depending on many factors including disk usages and CPU work load. Still, if we give each power state a relative power efficiency score, we can build a tool that computes the overall energy savings of an entire network of computers with Intel AMT. All of this is thanks to Intel AMT allowing us to query the computer's power state over the network without waking the computer up.

Querying for the power state also gives us two other pieces of interesting information. Two bits in the `GetSystemPowerState` method allow us to determine if a laptop is running on battery power and if a computer is locked up. The battery power bit is not always implemented by vendors, but when on, it should give the administrator a little warning not to perform operations on this computer that may excessively drain the battery. The lockup bit is actually implemented in cooperation with the Intel MEI driver running in the OS. If for any reason the Intel MEI driver fails to shutdown correctly, Intel AMT will turn on this bit, indicating a possible incorrect shutdown of the computer.

Third Party Data Storage (3PDS)

So Intel AMT does not provide all of this information you want when the computer is powered off? No problem. Third Party Data Storage allows developers to extend what information is available by allowing software to store data into the computer's onboard flash for later retrieval by a trusted administrator.

Imagine a small OS agent that starts each time the computers boots into the operating system and stores such things as:

- List of connected USB devices
- Installed software
- Summary of last blue screen kernel dump
- Currently logged user
- OS boot up and shutdown times
- OS errors
- Location of the latest backup

Such information would be stored periodically in the computer's flash memory and an administrator could retrieve this set of OS level historical information and use it to diagnose current problems. If a computer has a critical error, information contained in the flash could indicate what was going on before the error occurred. It could also be used for USB hardware and general software asset inventory. Being able to read from the platform's flash where to get the latest backup in the case of a complete hard disk failure could also be a lifesaver.

In any way you chose to use 3PDS, there are a few things to know before starting to write software. First, Intel AMT has 192 KB2 of flash space available. This space must be shared between all applications that want to make use of it and so, Intel AMT includes a space arbitration system of sorts, to make sure no single application is getting greedy.

Because the way by which local applications read and write to 3PDS is very different on AMT 1.x[3] compared to AMT 2.x and beyond, it is generally accepted for applications to only support 3PDS in AMT 2.x and beyond.

The 3PDS space is allocated to applications in pages of 4 KB. A single block of space can be composed of one or more pages of flash memory. When allocating and writing into a block, the unused space is simply not written into. When reading back the block, the unused space can contain leftover data. As a result, it's important that an application zero out the leftover space or make

---

[2] Intel® AMT 1.x only has 96 KB of 3PDS space available.

[3] Intel® AMT 1.x uses the Intel MEI driver to locally read and write to 3PDS. These 3PDS Intel MEI commands are not supported in Intel AMT 2.x and beyond which uses EOI/WS-Management through LMS instead.

sure to indicate how much data has actually been stored in the beginning of the block to make reading back the data easier. Once a block is allocated, the entire block must be read or written. It is not possible for software to write into only a portion of the block without rewriting the entire block.

Since 3PDS is flash storage, it not as fast as regular memory. As a result, it's generally recommended to start reading or writing to 3PDS on a separate thread and the larger the block, the longer it will take to read or write. Because of the limited space and slower speed, compressing the data before storing it into 3PDS is highly recommended. For example: the GZIP compression methods available in Microsoft .NET makes this very easy.

One of the biggest mistakes developers and administrators make when using 3PDS is to assume that 3PDS is freely accessible like a file system and the Microsoft Windows registry. Before starting to use 3PDS, users must have a good understanding of the 3PDS allocation system. Even if Intel AMT is provisioned and accessible, if not set up by the administrator first, it can't be accessed locally at all.

3PDS Allocation System

In order to properly use 3PDS, the administrative console must first set up 3PDS correctly. This operation can only be performed remotely. It can't be done locally, even if the administrator username and password is known.

First, the administrator must set up to four "Enterprises." These are top level administrative domains; if a computer is owned and operated by a single entry, there should generally be only one enterprise. With Intel, we would use the string "Intel" or "Intel IT" as an enterprise name. Having many different enterprise strings means that a single computer can be administered by different entities without conflict. Without at least one enterprise, no data can be stored into 3PDS.

Next, 3PDS has a list of partner vendors and applications. By default this list is populated with a few well known industry management software vendors. The partner list can be deleted and/or changed by the remote administrator. The partner list limits the maximum amount of flash pages that can be allocated by a single application. If an application is not on this list, it is limited by a separate non-partner global setting. This setting is generally set to 2 pages or 8 KB. The partner and non-partner settings may allow for more than the total 192 KB of available flash space to be used, but this is okay since it's unlikely that all of the applications will each use all of their maximum allowed space.

Once the enterprises, partner table, and non-partner settings are all set, allocation, deletion, reading and writing of blocks can then start. Before performing any of these operations to blocks locally or remotely, the software must first register to 3PDS. This is a separate step from the Intel AMT connection with the username and password, which must always be done first. The 3PDS registration is performed using the `RegisterApplication` call. It must include the enterprise, vendor name, application name, and a unique identifier for this instance of the software. Once registered, each created block is be tagged with the information of the owner who created the block. This information is used to calculate allocation limits and determine who has read and write access to this block. Each block in 3PDS has the following attributes:

- Enterprise (string)
- Application Vendor (string)
- Application Name (string)
- Block Name (string)
- Owner (GUID)
- Number of 4-KB pages
- Permissions set
- Visibility (visible or hidden)

Block permissions allow the application to set who has the rights to read and write into this block, based on their own 3PDS registration. For example, this block can be set to be read by anyone who is registered with the same vendor name.

When the block visibility is set to false, only the owner of this block can see it. As a result, only an application that registers with the same enterprise, vendor, application, and GUID will be able to see this block.

Probably the best way to learn about this is to use Intel AMT Commander remotely and Intel AMT Outpost locally to get hands-on experience with these settings. The Manageability DTK also provides a set of tutorial videos that walk you through the steps of using 3PDS. As users get more familiar with how 3PDS works, we often get this following question: If registration is needed to read and write blocks for a given vendor and application, why is it the Intel AMT Commander can see reads and writes of all of the blocks in 3PDS regardless of the owner?

Intel AMT Commander will first enumerate all of the enterprises, vendors and applications that have been registered into 3PDS. It will then *register as each of them*, registering as many times as it needs, using the software UUID of all zeros. This trick allows Commander to gain access to most of the blocks in 3PDS. Still, this trick has its limits. If a local application sets a block to be hidden, Commander will no longer see it. Also, a local application can set block permissions to allow only the owner of the block to read and write the block, effectively locking Commander out.

Is it not possible to register both locally and remotely using the same enterprise, vendors, application, and UUID. This is why Commander will always use a GUID of all zeros[4] and Intel AMT Outpost by default uses a GUID of all zeros and a 1 at the end[5]. If a UUID is used locally, it later can't be used remotely and vice-versa. Any attempt will result in an error.

At this point, we want to dispel myths about 3DPS that come up from time to time. Intel AMT never looks at or interprets the 3PDS data. As a result, you can't put code into 3PDS and have Intel AMT execute it. Intel AMT will also never read or write data or results into 3PDS unless it's instructed to do so by local or remote software. If the data stored into 3PDS is highly sensitive and should not be read by other applications, the application must encrypt the data itself. 3PDS should not be considered to be a security storage area like the storage area provided by tamper-resistant modules.

To conclude with 3PDS, you may be thinking of placing an MP3 file into 3PDS so that you can stream music while your computer is off! The 192 KB space limitation would severely limit that type of usage, but it was a good idea while it lasted. Other ideas include placing instructions into 3DPS that could be read and executed by an OS agent the next time the computer is booted up. You could also push results such as "the backup is complete" or "finished computing this weather simulation" into 3PDS just before putting the computer to sleep. Regardless of what you chose to do with 3PDS, it is one of the Intel AMT features with many opportunities for innovation.

---

[4] Intel® AMT Commander always uses {00000000-0000-0000-0000-000000000000}

[5] Intel® AMT Outpost uses by default {00000000-0000-0000-0000-000000000001}

Summary

In this chapter we reviewed how to scan, connect, and gather data about a computer using Intel AMT. When done correctly, a management console should be able to handle connections to previously unknown computers regardless of the Intel AMT version or communication standard used and correctly acquire information about this computer. Discovery is also a required step before moving into the two following chapters where Intel AMT will be used to actively protect and heal a computer remotely.