Wired for Management Baseline 2.0 Review Request 77
Remote Lockout API

CLOSED	1	11/20/98	7/28/99
Status	**Priority**	**Received**	**Complete**
WfM	2.0	WfM	2.0 errata
Specification	**Release**	**Target Specification**	**Target Release**
Jithendra Bethur	Phoenix Technologies Ltd.	(408) 570 1376	Jithendra_bethur@phoenix.com
Requester	**Company**	**Voice**	**Email**

*The following text describes an approved change to the Wired for Management 2.0 Baseline Specification. Please substitute the **Changed Text** given below in the appropriate section of the specification. If you are interested in the history and rationale behind this Review Request, this information follows at the end of the document.*

Changed Text

Replace the entire Section 5.2.4 "Remote Lockout Considerations" with the following text:

5.2.4 Remote Lockout Interface (RLI) – Version 1.10

5.2.4.1 Overview

During remote management of a PXE client, it may be required to prevent an end-user from interrupting sensitive operations like a BIOS update. The Remote Lockout Interface (RLI) allows programmatic lockout of events that could interrupt such an operation. The interface is expected to be used by software executing on the client system.

The RLI hides the details of the underlying hardware implementation. This allows manufacturers to provide different lockout hardware implementations while providing a consistent control interface to system software.

5.2.4.2 Initial Conditions

The initial state of the Remote Lockout setting after power-on, cold boot or warm boot is all events are enabled. System software uses this interface to lockout events. System software should re-enable events when sensitive operations have been completed.

5.2.4.3 Invocation and Parameter Passing

The Remote Lockout Interface is only available in real mode and is invoked using INT 15H. All parameters are passed to and from the RLI functions using processor registers. The AH register is set to 25H for the RLI and the AL register indicates the desired RLI function.

The AX, BX CX, DX, SI and DI registers may be altered by the RLI. System software should save these registers before calling RLI functions and restore them on return (after retrieving any parameters returned by the RLI). All other processor registers are preserved.

If the function is successful, the RLI returns with the CF (carry flag) reset and the AH register set to zero.

If an error is encountered, the RLI functions return with the CF (carry flag) set and AH set to one of the following return codes:

86h ERR_FUNCTION_NOT_SUPPORTED

5.2.4.4 Remote Lockout Interface Functions

The RLI supports three functions:

- Inquire Lockout Capabilities
- Get Remote Lockouts
- Set Remote Lockouts

All of the RLI functions use a bitmap to describe lockouts. Inquire Lockout Capabilities and Get Remote Lockouts return this bitmap. Set Remote Lockout uses this bitmap to select events to lockout. The bits are numbered from zero to 15 with the right-most bit being the least significant and bit 0. The Lockout Bitmap is defined as follows:

Bit	Definition
0	Permanently reserved, was hard on/off switch for pre-v1.1 implementations of this interface.
1	Soft On/Off Switch (controlled via ACPI power-button interfaces)
2	Reset button
3	Mouse
4	Ctrl-Alt-Del key combination
5	All keyboard activity except Ctrl-Alt-Del.
15 .. 6	Reserved for future use (reset to zero for backward and forward compatibility)

5.2.4.4.1 Inquire Lockout Capabilities (2500H)

This function returns the lockout capabilities of the system. The bit-mapped value returned in the CX register indicates the lockouts managed by the RLI. All return values are static. This function always returns the same values. The current lockout setting does not affect the value returned by this function.

Input [AX] = 2500H

Output [CF] = status

Set = error

Reset = success

[AH] = Return code, if CF set

86h ERR_FUNCTION_NOT_SUPPORTED

If [CF] reset,

[AH] = Zero (00)

[BX] = Revision (BCD-encoded with an implied decimal point between the bytes. For example, 0110H is 1.10)

[CX] = Lockout capabilities (see Lockout Bitmap definition above)

If bit set (to 1), event lockout supported

If bit reset (to 0), event lockout not supported

Example

```
MOV     AX, 2500H               ; Inquire Lockout Capability
MOV     BX, 0                   ; Clear output registers
MOV     CX, 0
INT     15H                     ; Able to get capabilities?
JC      ERROR           ; No, error out


MOV     wRevision, BX           ; Yes, save interface revision
MOV     wCapabilities, CX       ;   and system lockout capabilities
```

Comments This function was unintentionally omitted from versions of the RLI prior to Version 1.1.

5.2.4.4.2 Get Remote Lockouts (2501H)

This function returns the current lockout setting. The bit-mapped value returned in the CX register indicates which events are enabled and which events are locked out.

Input [AX] = 2501H

Output [CF] = status

Set = error

Reset = success

[AH] = Return code, if CF set

86h ERR_FUNCTION_NOT_SUPPORTED

If [CF] reset,

[AH] = Zero (00)

[CX] = Current lockout setting for all supported events
(see Lockout Bitmap definition above)

If bit set (to 1), event locked out

If bit reset (to 0), event enabled (or unsupported)

Example

```
n       MOV   AX, 2501H          ; Get Remote Lockouts
        MOV   CX, 0              ; Clear output registers
        INT   15H                ; Able to get remote lockouts?
        JC    ERROR              ; No, error out

        MOV   wInitialSetting, CX ; Save initial setting

        TEST  CX, 4              ; Is Reset event locked out?
        JZ    RESET_ENABLED      ; No, continue
                                 ; Yes
```

Comments Only supported events are reported. Attempts to lockout unsupported events are not reflected in the Lockout Bitmap returned by this function. Only bits for supported events are ever set in the Lockout Bitmap.

5.2.4.4.3 Set Remote Lockouts (2502H)

This function locks out or enables the specified events. The bit-mapped value passed in the CX register indicates which events to allow and which events to lock out.

Input [AX] = 2502H

[CX] = Desired lockout setting (see Lockout Bitmap definition above)

If bit set (to 1), requesting event be locked out

If bit reset (to 0), requesting event be enabled

Output [CF] = status

Set = error

Reset = success

[AH] = Return code, if CF set

86h ERR_FUNCTION_NOT_SUPPORTED

If [CF] reset,

[AH] = Zero (00)

Example

```
MOV   AX, 2502H          ; Set Remote Lockouts
MOV   CX, 2              ; Disable Soft On/Off events, enable all others
INT   15H                ; Able to set remote lockouts
JC    ERROR        ; No, error out
```

Comments If supported, this request always succeeds. Requesting lockout of an unsupported event is not an error. The interface simply ignores the request for that event and sets supported events as requested (locked out or enabled).

Review Request History *– this section documents the discussion of this review request. The full discussion has been included in order to document those changes that have been explicitly rejected by the Wired for Management Review Team, which includes those companies listed at http://developer.intel.com/ial/wfm/list.htm.*

> ********* *Only those changes captured explicitly in the* ***Changed Text*** *section above have been approved and are now officially part of the specification.* *********

Summary

(Jithendra Bethur – Phoenix) WfM 2.0 specification specifies 'Remote Lockout' as one of the requirements for the platform firmware. It defines an interface via INT15 (function 2501H and 2502H) which the PXE client (platform firmware) should provide for the executables/boot-image downloaded over the network to 'lock out' certain devices/operations.

(Jose Ramirez- Intel) Results of Int 15h AX=2501h Get Remote Lockout Settings are ambiguous. . Results of Int 15h AX=2502h Set Remote Lockout Settings are ambiguous. Functionality is missing for determining currently available lockout features.

Background

(Jithendra Bethur – Phoenix) Since this feature (remote-lockout) is used by a downloaded executable only during PreBoot', the platform firmware might have to inhibit any further 'remote lockout' API calls after the system boots to an OS (OS here means local OS and not remote boot OS). This is to avoid any inadvertent calls being made to this API during run-time which may also pose some security issues. But how and when does the platform firmware decide to inhibit the 'remote-lockout' functionality is not defined here. This may be addressed by the review committee or may be left to the implementers of the platform firmware.

Also, it would be more desirable for a download image to do a 'Remote Lockout Installation check' before calling the remote lockout APIs. The 'Changed Text' below suggests one way of doing that.

Also, functions 2501H and 2502H may require an extension for OEMs to define/install their own additional devices capable of being locked-out. While the register CX serves for the bit fields for predefined devices, register DX may be used by OEMs to define their own.

(Jose Ramirez- Intel) The Remote Lockout implementation details and functionality as specified in the "Wired For Management Baseline, Version 2.0 Release" are ambiguous and open to interpretation in many aspects. WFM review requests were submitted in an attempt to address some of the issues current at that time but closure was never achieved. In addition terminology used was not clearly defined.

1. The persistence of the remote lockout capability is undefined.
2. The get remote lockout settings request does not state whether the values returned indicate the current state of available lockout features or the current state of the locked out features.
3. The possible returned error codes returned in AH by the get and set remote lockout settings request are undefined.
4. The set remote lockout request doesn't specify if the capability allows the lockout of a single feature or multiple features at one time.
5. The definition of hard on/off switch is not stated.
6. The definition of soft on/off switch is not stated.
7. The behavior of setting an unsupported lockout feature along with setting supported lockout features on a system is undefined.

I am proposing clear definition of soft switches, removal of the hard switch function, and behavior clarifications for setting and getting remote lockout settings and status. I am also proposing two new functions. One allows for the re-enabling of locked out features. The second locks the settings of the remote lockout features until reboot. This prevents tampering with features after pre – boot if desired. I am also adding the additional definition of 4 optional OEM/BIOS vendor specific bits to support special features provided by OEM/BIOS vendors.

Response

(Jose Ramirez –– Intel)

Consolidation of the previous related RRs and moving ahead with this RR is a good idea. Remote Lockout definitely needed some rework. Here is my input and some important points to be considered:

NOTE: In the Changed Text section below, to make it more readable, I have ACCEPTED all changes first (to remove the original) and then done my changes over the proposed changes

1. Function 2501, Get Remote Lockout Settings should not be changed from it's previous definition. That is because there are earlier implementations out in the field which follows the old definition. At least I know Phoenix has Remote Lockout feature implemented in it's core BIOS since January 28th 1999. Any extensions to the previous definitions is fine but changing the very basic definition of the function should not be done.
   - Addition of optional OEM defined support to the previous definitions is ok (changes proposed. See changed text).
   - The new return code 0x86 for ERR_FAILURE_UNSUPPORTED is a good point. That's the way INT 15h works.

2. I would recommend the following set of functions for Remote Lockout.
   - **Int 15h Get Remote Lockout Capabilities or Remote Lockout Installation Check: AX = 2500H**

     This returns a constant bit-field structure for a given platform as to which devices are LOCKABLE and UNLOCKABLE. This constant is irrespective of whether all the lockable/unlackable devices has permanently been INHIBITED (until next bott) or not. The way to find out if Remote-Lockout has been permanently Inhibited until next boot is to do a ***Get Current Remote Lockout Settings*** and checking Bit15. In addition this function would return if an OEM defined function is defined or not and also if entire Remote Lockout feature has been inhibited until next boot.
   - **Int 15h Get Current Remote Lockout Settings: AX = 2501H**

     This always returns the current state of all the Lockable/Un-lockable devices. Same as the old definition, only that there are 2 additions and 1 change in the bit-field structure definition. See changed text below.
   - **Int 15h Set Remote Lockout Settings: AX = 2502H**

     This will LOCK/UNLOCK individual devices as per the Remote-Lockout Bit-Field structure. Same as the old definition, only that there are 2 additions and 1 change in the bit-field structure definition. See changed text below

3. The usage of the term "feature" is not appropriate for the individual lockable items. It's sometimes very confusing during reading. It could be changed to "device".
   - The term "feature" is already being used to address the Remote Lockout feature itself in its entirety. For example, 'Remote Lockout is a BIOS Feature'. Or

     ERR_FAILURE_UNSUPPORTED is returned to indicate Remote Lockout feature is not supported in this BIOS.
   - The term "device" could be used to address individual items which are to be LOCKED or UNLOCKED using the

Remote Lockout Feature. Example:

- Misleading Usage: "If a **feature** has been previously **enabled** ....."
- Suggested Usage: "If a **device** has been previously **locked** ....."

It makes sence since most of the items which are LOCKED or UNLOCKED are devices (like Keyboard, Mouse, On/Off Switch) except for the Ctrl-Alt Del key combination!

4. **It is needless to mention that:**

   "Lockout of individual features must not lock out other supported features. For example, locking out all key press (CX & 20h) must not lock out Ctrl+Alt+Del (CX & 10h)."

   Bit-wise settings is pretty much a standard and it's implied that a bit stands for itself and not for it's neighboring bits.

   But one little clarification about "All Key Press" and "Ctrl-Alt-Del" is required **here** (but may not for the spec).

   'All Key Press Lockout' will lockout the entire keyboard. So it's obvious that "Ctrl-Alt-Del" will NOT work and that is the way it should be. It's, of course, not that 'Locking out All Key Press will in-turn execute the path/code for the Ctrl-Alt-Del' But the very fact that 'All Key Press Locked out means 'the entire keyboard is locked out' will make the Ctrl-Alt-Del key combination not to work. How can it possibly work? And the purpose of each is also to be bore in mind.

   An image downloaded via PXE may want to LOCKOUT the "Ctrl-Alt-Del" only if say it needs user interaction like choosing a menu, answering Y/N or pressing ENTER key etc, but it does NOT want to allow the user to reboot.

   Say if an app. doesn't want the user press any key at all (say in a case where a management app. will in-turn download and execute a 3rd party software like BIOS Flash update and say the software will pause or abort if any key is pressed) then it would first LOCKOUT All Key Press. Where as there is no case where you need the other way around.

   Again, this doesn't have to be mentioned explicitly in the spec. I think it's implied and will not cause any confusion.

5. The optional OEM defined functions needs further clarification. It's unclear how a 3rd party management app., an image downloaded via PXE would find out whether the OEM defined lockout function it is **aware** of, is actually present in this BIOS or not, and if present which one of the 4 assigned. Say OEM-1 assigns OEM-Lockout 1-4 as Serial Port, Parallel Port, HDD and FDD whereas OEM-2 assigns OEM-Lockout 1-4 as HDD, FDD, USB and resenved. How would an app. find out?

   May be using another register like DX to return an unique OEM signature would help. In that case there is no need to assign 4 different bits for OEMs is not required. Also it is like a limitation. Just one bit for OEM-defined functions and another register, say BL/BX, for their own sub-functions.

   Again, as (Dave Lawrence – STE) suspected, this would hinder development of standard management apps. for sure.

   Note that the Changed Text below still does not address the OEM defined functions or how to SET and GET the OEM defined device's lockout states. One option would be to use register BX (16 devices!) for OEM definitions.

6. It is better to have the Bit-Map (now called bit-field structure which is the common terminology) defined only once on the top (of all function definitions) and within every function definition, define clearly what it means when a BIT is 0 and when a BIT is 1.

NOTE: In the Changed Text section below, to make it more readable, I have ACCEPTED all changes first (to remove the original) and then done my changes over the proposed changes

July 15, 1999 (Cindy Merkin – Dell):

Per discussion with (Dave Lawrence – STE), the following changes are proposed:

1. Change the version number in the 2500h call to 1.1, v1.0 of the interface didn't include the "Get Capabilities" function.
2. Removed the "inhibit" function. It seems that this function could cause more problems than it's solving, unless

someone can identify a specific usage model.

3. Added a usage guideline for users of the interface, in an attempt to provide an industry solution for the v1.0 implementation differences.
4. Removed the OEM bits, per Dave Lawrence's previous comments. The management of the usage seems high for the value this function is providing.

Status

11/20/98: New RR received. No change text submitted.

6/29/99: (Jose Ramirez – Intel) change text submitted.

7/15/99: (Jose Ramirez – Intel) updated per discussions with (Dave Lawrence – STE) and (Cindy Merkin – Dell)

7/16/99: updated by (Dave Lawrence – STE) and (Cindy Merkin – Dell)

7/21/99 (telecon): agreed, with the following changes: (Cindy Merkin – Dell) separate Ctrl-Alt-Del from keyboard function overall sothat, e.g., during PXE operation, local boot can be inhibited while still allowing users to answer critical questions (e.g. user name). (Jose Ramirez – Intel) 1. Insert "25" for completeness in the function code definition 2. Clarify that "set" is "1" and "clear" is "0".

7/28/99: verification complete. Review Request is CLOSED and fully approved as an erratum to the Wired for Management 2.0 Baseline Specification.