- IPMI -

Intelligent Platform Management Interface Specification

Proposed extensions for IPMI v2.0:

RMCP+ and Serial Over LAN

DRAFT 9/15/03

Intel Hewlett-Packard NEC Dell

Lenovo
Ex. 1023 - Page 1

Revision History

Date	Modifications
9/15/03	Release for Adopter review

Copyright © 2003 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Computer Corporation, All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

INTEL, HEWLETT-PACKARD, NEC, AND DELL DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. INTEL, HEWLETT-PACKARD, NEC, AND DELL, DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

I^2C is a trademark of Philips Semiconductors. All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

I^2C is a two-wire communications bus/protocol developed by Philips. IPMB is a subset of the I^2C bus/protocol and was developed by Intel. Implementations of the I^2C bus/protocol or the IPMB bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel, Hewlett-Packard, NEC, and Dell retain the right to make changes to this document at any time, without notice. Intel, Hewlett-Packard, NEC, and Dell make no warranty for the use of this document and assume no responsibility for any error which may appear in the document nor does it make a commitment to update the information contained herein.

Table of Contents

1. Introduction 6
1.1 Reference Documents 6
1.2 SOL Overview 7
1.3 RMCP+ Overview 7

2. RMCP+ Protocols 7
2.1 Differences with IPMI v1.5 Authentication 8
2.2 IPMI v1.5 Discovery and Sessions under IPMI v2.0/RMCP+ 9
2.3 Support for IPMI v2.0 Capabilities under IPMI v1.5 9
2.4 VLAN Support 9
2.5 IPMI Messages Encapsulation Under RMCP 9
2.5.1 RMCP/ASF and IPMI Byte Order 10
2.6 Example IPMI over LAN Packet 11
2.7 Discovering Support For IPMI over IP Connections 14
2.8 Channel Security Discovery Commands 14
2.9 Get Channel Authentication Capabilities Command 14
2.10 Get Channel Cipher Suites Command 17
2.10.1 Cipher Suite Records 18
2.10.2 Cipher Suite IDs 19
2.11 RMCP/ASF Pong Message (Ping Response) 20
2.12 AES-CBC-128 Encrypted Payloads 21
2.12.1 Generating the Initialization Vector 21
2.12.2 Encryption with AES 22
2.12.3 CBC (Cipher Block Chaining) 22
2.13 RMCP+ Authentication 22
2.13.1 RSSP Open Session Request 22
2.13.2 RSSP Open Session Response 24
2.14 Close Session Command 25
2.15 Session Termination 25
2.16 RAKP Messages 26
2.16.1 RAKP Message 1 26
2.16.2 RAKP Message 2 27
2.16.3 RAKP Message 3 28
2.16.4 RAKP Message 4 29
2.17 RSSP and RAKP Message Status Codes 30
2.18 IPMI v2.0 RMCP+ Payload Types 31
2.19 Session Sequence Numbers 32
2.20 IPMI v1.5 Session Sequence Number Handling 32
2.20.1 IPMI v1.5 Inbound Session Sequence Number Tracking and Handling 33
2.20.2 IPMI v1.5 Out-of-order Packet Handling 33
2.20.3 IPMI v1.5 Outbound Session Sequence Number Tracking and Handling 34
2.21 IPMI v2.0 RMCP+ Session Sequence Number Handling 34
2.21.1 IPMI v2.0 RMCP+ Sliding Window 35
2.22 Authentication, Integrity, and Encryption Algorithm Numbers 35
2.23 RSSP Authenticated Key-Exchange Protocol (RAKP) 36
2.24 Generating Additional Keying Material 39
2.25 Random Number Generation 40
2.25.1 Random Number Key 40
2.25.2 Random Number Generator Counters 40
2.25.3 Random Number Generator Operation 40
2.26 Extended LAN Configuration Data 41

3. RMCP+ Support and Payload Commands 43
3.1 Activate Payload Command 43

3.2 Deactivate Payload Command 45
3.3 Suspend/Resume Payload Encryption Command 45
3.4 Get Payload Activation Status Command 46
3.5 Get Payload Instance Info Command 47
3.6 Set User Payload Access Command 48
3.7 Get User Payload Access Command 50
3.8 Get Channel Payload Support Command 51
3.9 Get Channel Payload Version Command 51
3.10 Get Channel OEM Payload Info Command 52
3.11 Set User Password Command 52
3.12 Set Channel Security Keys Command 54

4. Serial Over LAN 55
4.1 SOL and Serial Port Sharing 56
4.2 SOL Operation Overview 57
4.3 SOL Security 58
4.4 SOL Sequence Numbers 58
4.5 Flow Control 58
4.6 Bit Rate Handling 58
4.7 SOL Payload Data Format 59
4.8 Activating SOL using RMCP+ Authentication 61
4.9 SOL Packet Acknowledge and Retries 62
4.10 SOL Interaction with Windows.NET Escape Sequences 64
4.11 SOL Payload Activated with Serial Port Sharing 64

5. SOL Commands 66
5.1 SOL Activating Command 66
5.2 Set SOL Configuration Parameters Command 67
5.3 Get SOL Configuration Parameters Command 67

Appendix A - Previous Sequence Number Tracking 71

Appendix B - Command Assignments 72

List of Tables

Table 2-1, RMCP Packet Format for IPMI via Ethernet 11
Table 2-2, Channel Security Discovery Commands 14
Table 2-3, Get Channel Authentication Capabilities Command 15
Table 2-4, Get Channel Cipher Suites Command 18
Table 1-4, Cipher Suite Record Format 19
Table 2-6, Cipher Suite IDs 19
Table 2-7, RMCP Packet Fields for ASF Presence Pong Message (Ping Response) 20
Table 2-8, AES-CBC Encrypted Payload Fields 21
Table 2-9, RSSP Open Session Request 23
Table 2-10, RSSP Open Session Response 24
Table 2-11, Close Session Command 25
Table 2-12, RAKP Message 1 26
Table 2-13, RAKP Message 2 28
Table 2-14, RAKP Message 3 29
Table 1-14, RAKP Message 4 30
Table 1-15, RSSP and RAKP Message Status Codes 30
Table 2-17, Payload Type Numbers 32
Table 2-18, Authentication Algorithm Numbers 35
Table 2-19, Integrity Algorithm Numbers 35

Table 2-20, Confidentiality Algorithm Numbers 36
Table 2-21, Extended LAN Configuration Data 41
Table 3-1, RMCP+ Support and Payload Commands 43
Table 3-2, Activate Payload Command 44
Table 3-3, Deactivate Payload Command 45
Table 3-4, Payload-specific Encryption Behavior 46
Table 3-5, Suspend/Resume Payload Encryption Command 46
Table 3-6, Get Payload Activation Status Command 47
Table 3-7, Get Payload Instance Info Command 48
Table 3-8, Set User Payload Access Command 49
Table 3-9, Get User Payload Access Command 50
Table 3-10, Get Channel Payload Support Command 51
Table 3-11, Get Channel Payload Version Command 52
Table 3-12, Get Channel OEM Payload Info Command 52
Table 3-13, Set User Password Command 54
Table 3-14, Set Channel Security Keys Command 55
Table 4-1, Mux Settings 56
Table 4-2, SOL Payload Data Format 59
Table 4-3, Remote Console to BMC SOL Packet Handling 63
Table 4-4, Set Serial/Modem Mux Command Operation while SOL Active 65
Table 5-1, SOL Commands 66
Table 5-2, SOL Activating Command 66
Table 5-3, Set SOL Configuration Parameters Command 67
Table 5-4, Get SOL Configuration Parameters Command 67
Table 5-5, SOL Configuration Parameters 68

1. Introduction

This document presents proposed additions to the IPMI specification. At this time, it is planned that these proposals will be included in revised IPMI specification that will be issued as "IPMI v2.0". The document is being provided to IPMI Adopters for comment.

The proposals in this document are provided for information only. Within this document the term "IPMI v2.0" is used to denote or highlight elements that are different or in addition to the present IPMI v1.5 specification. It does not imply that the proposals have been formally reviewed and accepted by the IPMI Promoters, nor whether they will be included in any future IPMI specification in part or in whole. All proposals are subject to review and approval by the IPMI Promoters (Intel, HP, NEC and Dell) and are subject to change without prior notice.

The proposals in this document primarily cover the following areas: RMCP+ and SOL. Other proposal areas that have been publicly discussed are being worked on as well. These areas were not available in suitable form for IPMI Adopter review at this time.

1.1 Reference Documents

The following documents are companion and supporting specifications for this document:

[AES] *Advanced Encryption Standard*, FIPS 197, November 2001.
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[ASF 2.0] *Alert Standard Format (A S F) Specification Version 2.0, 23 April 2003*, ©2000-2003, Distributed Management Task Force, Inc.
http://www.dmtf.org

[RFC2104] *RFC 2104, HMAC: Keyed-Hashing for Message Authentication*, H. Krawczyk, IBM, M. Bellare, UCSD, R. Canetti, IBM, February 1997.
http://www.ietf.org/rfc/rfc2104.txt

[RFC2404] *RFC 2404, The Use of HMAC-SHA-1-96 within ESP and AH*, C. Madson, Cisco Systems Inc., R. Glenn, NIST, November 1998.
http://www.ietf.org/rfc/rfc2404.txt

[SHA-1] NIST, FIPS PUB 180-2: Secure Hash Standard, August 2002.
http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[MODES] *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, NIST Special Publication 800-38A, December 2001.
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

Lenovo
Ex. 1023 - Page 6

[MSFT EMS] *Building Hardware and Firmware to Complement Microsoft Windows Headless Operation, Version 1.00, July 16, 2002.*
http://www.microsoft.com/whdc/hwdev/platform/server/headless/

1.2 SOL Overview

SOL stands for "Serial over LAN" and is the working name for the specification of IPMI interfaces for redirecting baseboard serial controller traffic over an IPMI session via RMCP+. This can be used to enable asynchronous serial-based OS and pre-OS communication over a connection to the BMC. SOL is specified in *Section 4, Serial Over LAN* and *Section 4.10, SOL Interaction with Windows.NET Escape Sequences*.

1.3 RMCP+ Overview

RMCP+ is the working name for an enhanced protocol for transferring IPMI messages and other types of payloads to an IPMI-based BMC over IP. RMCP+ is specified in *Section 2, RMCP+ Protocols*, and *Section 3, RMCP+ Support and Payload Commands*.

RMCP+ includes:

- **Support for multiple payload types over an IPMI session.** These include both 'standard' and 'OEM' payloads. Payload support enables types of traffic other than IPMI messages to be simultaneously carried over an IPMI session. The specification also allows a separate session to be established for carrying payloads.
- **Enhanced user authentication algorithms.** RMCP+ includes more robust session set-up and key handling algorithms than those for IPMI over LAN in IPMI v1.5.
- **Encryption support.** IPMI messages and other payloads can be encrypted under an IPMI session. This enables confidentiality for remote operations such as setting user passwords and for SOL.
- **Better alignment with the ASF 2.0.** The RMCP+ follows many of the packet format and authentication elements defined for RMCP (Remote Management Control Protocol) as specified in the Distributed Management Task Force "ASF 2.0" specification. (See [ASF 2.0] in the *Reference Documents* section.) This helps make it easier to create remote management applications that can work with both ASF 2.0 and IPMI v2.0 -based remote management connections.

2. RMCP+ Protocols

This section presents the packet formats, algorithms, and protocols associated with RMCP+. The following lists the main characteristics of RMCP+:

- Uses an ASF 2.0 -based authentication algorithms and steps for establishing a session. (Commands corresponding to the ASF 2.0 RSP/RAKP messages, however, are defined under the IPMI message class rather than under the ASF message class. The commands vary slightly from their ASF 2.0 counterparts, but are close enough to

make it straightforward to create a single set of code that can connect to either IPMI or ASF -based systems.)

- Supports encrypted payloads.
- Authenticated sessions can be established on any UDP port, including port 26Fh
- Multiple payload types are supported over a single session
- A mix of authenticated and unauthenticated packets can be supported over a single session
- RMCP+ is specified under the IPMI message class in RMCP. The packets remain conformant with the RMCP specification at the RMCP packet level.
- There can be a mix of encrypted and unencrypted payloads over a single session

2.1 Differences with IPMI v1.5 Authentication

The following presents an overview of some of the key differences as well as similarities between IPMI v1.5 and RMCP+ user setup and authentication mechanisms:

- IPMI v1.5 had a hook that would allow packets under a session to have different authentication signatures than the type of signature that was negotiated to open the session. This hook was to include a 'Authentication Type' field that specified the type of authentication on a per-packet basis. This capability was not used in the specification. Thus, to simplify things a packet can only have two types of authentication: the type of authentication selected in the "Open Session" command or "none" - thus the 'Authentication Type' field is deleted and instead the presence or absence of the "Integrity Data" field is used to indicate whether a given packet in the session is authenticated or not.
- For RMCP+, the IPMI *Get Session Challenge* request and response messages and *Activate Session* messages are replaced by the *Open Session* and *RAKP*_x Messages:
    1. The RMCP *RSSP Open Session Request, RSSP Open Session Response,* and *RAKP_1* messages, combined, correspond roughly to the IPMI *Get Session Challenge* request.
    2. The RMCP *RAKP_2* messagecorresponds roughly to an IPMI *Get Session Challenge* response.
    3. The RMCP *RAKP_3* messagecorresponds roughly to an IPMI *Activate Session* request.
    4. Note that there's no corresponding response message to acknowledge that the RAKP_3 message. However, as an ASF message there will be an RMCP ACK that is sent back if the RMCP sequence number is not FFh.
- ASF 2.0 authentication defined 'Roles' such as User and Administrator, where a key (password) was associated with each role. IPMI v1.5 authentication associated a key with each User Name where a 'privilege level' (such as User or

Administrator) was configured for each user name. These two approaches are merged in RMCP+. The BMC can be configured with 'null' user names, whereby key lookup is done based on 'privilege level only', or with non-null user names, where the key lookup for the session is determined according to the user name.

- RMCP+ supports HMAC using SHA-1 for authentication. We expect that this type of authentication will be significantly slower than IPMI v1.5 Authentication for a given processor (which uses a simple MD2 or MD5 signature). First, RMCP uses SHA-1 which, while more robust, is computationally more intensive. Probably more significant is the use of HMAC which requires that the hashing algorithm is run twice on the data for each packet.

IPMI v1.5	Single Session ID for BMC and console	BMC and remote console pick unique, random starting 'incoming' and 'outgoing' sequence numbers
RMCP+	Separate Session IDs for BMC and console	BMC and remote console pick their own session IDs.

2.2 IPMI v1.5 Discovery and Sessions under IPMI v2.0/RMCP+

To enable backward compatibility, an IPMI v2.0 BMC that supports RMCP+ for IPMI messaging and standard payloads is required to simultaneously support IPMI v1.5 Packet formats. For a given BMC, users must be equally configurable for IPMI v2.0 or v1.5 access.

2.3 Support for IPMI v2.0 Capabilities under IPMI v1.5

IPMI v2.0-specific capabilities related to payloads (e.g. Serial Over LAN) are only available over IPMI v2.0/RMCP+ sessions. Otherwise, new IPMI v2.0 commands and command extensions are available under IPMI v1.5 sessions.

2.4 VLAN Support

VLAN support is optional, but recommended, for BMC access via IPMI 1.5 packet and IPMI v2.0 packet formats. A BMC that supports VLAN on a channel is required to support it for both packet formats. A BMC is not required to support VLAN equally across all LAN channels. This is allowed because some LAN connections may not have hardware that supports VLAN with a LAN connection to the BMC.

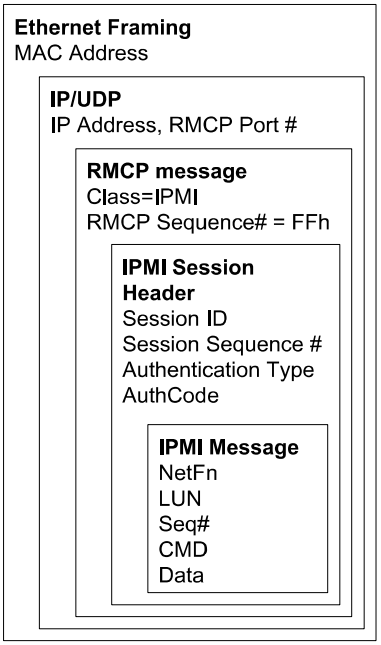
When a VLAN ID is configured into the LAN parameters, the BMC will only accept packets with that VLAN tag. This includes all RMCP and RMCP+ packets as well as DHCP and ARP packets. Conversely, all BMC-generated packets will include the given VLAN tag.

2.5 IPMI Messages Encapsulation Under RMCP

For LAN transfers, IPMI messages are a special class of data encapsulated in an IPMI Session packet. The IPMI Session packets are encapsulated in RMCP packets, which are encapsulated in UDP datagrams. This is illustrated in the following figure. The same type of encapsulation is used for IPMI

serial/modem messages via PPP, except the Ethernet Framing is replaced with a packet that uses PPP Framing and IP protocol type.

Figure 2-1, IPMI LAN Packet Layering



2.5.1 RMCP/ASF and IPMI Byte Order

Please take note of the following:

Multi-byte fields in RMCP/ASF fields are specified as being transmitted in 'Network Byte Order' - meaning most-significant byte first.
RMCP and ASF-specified fields are therefore transferred most-significant byte first.

The IPMI convention is to transfer multi-byte numeric fields least-significant Byte first. Therefore, unless otherwise specified:
Data in the IPMI Session Header and IPMI Message fields are transmitted least-significant byte first.

2.6 Example IPMI over LAN Packet

The following table shows the format and fields for IPMI messages encapsulated in an RMCP packet that is itself encapsulated within an IPv4 UDP packet delivered over Ethernet.

The table includes a set of columns under the label "Format". The first column shows the format used for IPMI over LAN as originally defined in the IPMI v1.5 specification, the second column shows the format for authenticated packets delivered using RMCP per the ASF 2.0 specification. This column is shown for reference only. The third column shows the format of packets for RMCP+.

Table 2-1, RMCP Packet Format for IPMI via Ethernet

		Format			Value
	Field	RMCP / IPMI 1.5 26Fh	ASF RMCP 298h	RMCP / IPMI 2.0 "RMCP+" 26Fh	
	Destination Address	6			Dest. MAC Address for 802.3
	Source Address	6			Source MAC Address for 802.3
802.1q[10]	*TPI*	2			Tag Protocol Identifier =8100h
	VLAN TAG - user priority	3-bits			User Priority[11] default = 000b, configurable
	VLAN TAG - CFI	1-bit			Canonical Format Indicator = 0b
	VLAN TAG - VLAN ID	12-bits			0's = no VLAN
	Frame Type	2			0800h
IP Header	Version	4-bits			4h for IPv4
	Header Length (length of IP header in units of 4-bytes)	4-bits			5h
	Precedence	3-bits			000b[4]
	Service Type (Type of Service)	4-bits			1000b[4] (minimize delay)
	reserved	1-bit			0b
	Total Length	2			
	Identification[5]	2			note[5]
	Flags	3-bits			010b[6] (don't fragment)
	Fragment Offset	13-bits			0_0000_0000_0000b[7]
	Time-to-Live	1			40h[3]
	Protocol	1			11h
	Header Checksum	2			
	Source IP Address	4			
	Destination IP Address	4			(28 bytes)
UDP Header	Source Port	2			
	Destination Port	2			26Fh, 298h
	UDP Length	2			
	UDP Checksum	2			(36 bytes) Some payloads may not use the UDP checksum, in which case this field gets set to 0's. The receiver must accept the packet and ignore the checksum field in this case.
RSP Header	Session ID		4		
	Session Sequence #		4		

RMCP Header	Version	1	1	1	06h (ASF 2.0)
	Reserved	1	1	1	
	RMCP Seq #	1	1	1	FFh for IPMI[2]
	Class of Message	1	1	1	07h for IPMI
IPMI Session Header	Auth Type / Format	1		**1**	[7:4] - reserved [3:0] - Authentication Type / Format 0h = none 1h = MD2 2h = MD5 3h = reserved 4h = straight password / key 5h = OEM proprietary 6h = Format = RMCP+ (IPMI v2.0 only) all other = reserved
	Payload Type			**1**	Payload Type [7] - 0b = payload is unencrypted 1b = payload is encrypted [6] - 0b = payload is unauthenticated (no AuthCode field) 1b = payload is authenticated (AuthCode field is present) [5:0] = payload type. See *Table 2-17, Payload Type Numbers*.
	OEM IANA			**4**	This field is only present when Payload Type = 02h (OEM Explicit) byte 1:3 - OEM IANA byte 4 - reserved
	OEM Payload ID			**2**	This field is only present when Payload Type = 02h (OEM Explicit). The definition and values of this field are specified by the company or body identified by the OEM IANA field.
	IPMI v2.0 RMCP+ Session ID			**4**	note[8] Session ID is 0000_0000h for messages that are sent 'outside' of a session.
	Session Sequence Number	4		**4**	note[8] For IPMI v2.0 "RMCP+" there are separate sequence numbers tracked for authenticated and unauthenticated packets. 0000_0000h is used for packets that are sent 'outside' of a session.
	IPMI v1.5 Session ID	4			note[8] Session ID is 0000_0000h for messages that are sent 'outside' of a session.
	Msg. Auth. Code Code (AuthCode) (not present when Authentication Type set to 'none'.)	16			
	IPMI Msg/Payload length	1	2	2	Payload length in bytes. 1-based.
IPMI Payload	Confidentiality Header			var	For encrypted payloads, based on encryption type for given payload. The confidentiality header is not encrypted.
	Payload Data	var	var	var	For IPMI v2.0: IPMI, SOL, KVM,

					etc. per Payload Type field.
	Confidentiality Trailer			var	For encrypted payloads, based on encryption type for given payload. The confidentiality trailer is typically encrypted along with the Payload Data.
IPMI Session Trailer[9] / RSP Trailer	Integrity PAD		var	var	Added as needed to cause the number of bytes in the data range covered by the AuthCode (Integrity Data) field to be a multiple of 4 bytes (DWORD). If present, each Integrity Pad byte is set to 00h.
	Pad Length		1	1	indicates how many pad bytes were added so that the amount of non-pad data can be determined.
	Next Header		1	1	Reserved in IPMI v2.0. Set to 00h when writing, ignore when read.
	AuthCode (Integrity Data)		**var**	**var**	For IPMI v1.5 this field is as specified by Auth Type. For IPMI v2.0 (RMCP+) if this field is present, then it is calculated according to the Integrity Algorithm that was negotiated during the session open process. See *Table 2-19, Integrity Algorithm Numbers.* This field is absent when the packet is unauthenticated.
	Legacy PAD[1]	1			legacy PAD not needed for IPMI v2.0
MAC level	CRC	4	4	4	

1. Some LAN adapter chips may have a problem where packets of overall lengths 56, 84, 112, 128, or 156 are not handled correctly. The PAD byte is added as necessary to avoid these overall lengths. Remote console software must use the PAD byte when formatting packets to any 10/100 Ethernet device that accepts RMCP packets.
2. RMCP Messages with class=IPMI should be sent with an RMCP Sequence Number of FFh to indicate that an RMCP ACK message should not be generated by the message receiver.
3. Default value for packets transmitted from the BMC. Can be overridden via a configuration parameter setting.
4. Value used for packets transmitted from the BMC. The BMC ignores the value of this parameter (except for checksum calculations) on received packets.
5. BMC should increment this field each time it sends a new packet.
6. Default value for packets transmitted from the BMC. Bit offset 1 (fragment bit) can be overridden via a configuration parameter setting.
7. Default value for packets transmitted from the BMC. The BMC is not required to support receiving fragmented packets. Packets with a non-zero fragment offset and/or a flags field bit 2 = 1b ("more fragments" may be silently discarded.)
8. The Session ID and Session Sequence Number must be non-zero for commands executed during an active session. All 0's (0000_0000h) for the Session ID and/or Session Sequence Number (null Session ID, null Session Sequence Number) are special values only used for messages and commands that can be executed prior to establishing a session, e.g. Get System GUID, Get Channel Authentication Capabilities, Get Session Challenge, and RSSP, or RAKP messages. When the Session ID is 0000_0000h, the Sequence Number field is ignored, however the Session Sequence Number should still be set to 0000_0000h. In addition, for IPMI v2.0 RMCP+ packets, unless otherwise indicated bits 7:6 in the payload type field must also indicate that the packet is both unauthenticated and unencrypted. Note that the IPMI v1.5 Activate Session uses a null (all 0's) Session Sequence Number before a session is activated, but does not use a null Session ID. Instead, it uses the Temporary Session ID given by the BMC in the response to the Get Session Challenge command.
9. For IPMI v2.0 RMCP+ packets, the IPMI Session Trailer is absent whenever the Session ID is 0000_0000h, or whenever bit 6 in the payload type field indicates the packet is unauthenticated
10. Four bytes only present for IEEE 802.1q "VLAN" formatted packets

11. The use and interpretation of this number is defined in ISO/IEC 15802-3.

2.7 Discovering Support For IPMI over IP Connections

There are two mechanisms for discovering whether a given system (IP Address) supports IPMI v1.5 and/or IPMI v2.0 connections. The first is the 'RMCP Ping discovery' mechanism where the BMC returns whether IPMI is supported via the *supported entities* field of the Ping response (a.k.a. the 'Pong' message). The BMC can optionally return values in the Ping response indicating what level of IPMI connection(s) (IPMI v1.5 and/or IPMI v2.0) it supports.

The second mechanism is an 'IPMI command discovery' mechanism where the remote console discovers that the system supports IPMI by issuing a *Get Channel Authentication Capabilities* command to determine support for IPMI v1.5/IPMI v2.0 connections. BMCs that support IPMI v2.0/RMCP+ must support the *Get Channel Authentication Capabilities* command in both the IPMI v1.5 and IPMI v2.0 packet formats. It is recommended that a remote console use the IPMI v1.5 formats until it has confirmed IPMI v2.0 support.

When the remote console decides to connect to the discovered system, it can use the *Get Channel Authentication Capabilities* and (for IPMI v2.0) the *Get Channel Cipher Suites* commands to determine which authentication, integrity, and confidentiality algorithm Cipher Suite(s) can be used for establishing the connection.

2.8 Channel Security Discovery Commands

Table 2-2, Channel Security Discovery Commands

Command	Section Defined	O/M
Get Channel Authentication Capabilities	2.9	O[1]
Get Channel Cipher Suites	2.10	O[2]

1. Mandatory if session-based channels are implemented.
2. Mandatory if IPMI v2.0 / RMCP+ session-based channels are implemented.

2.9 Get Channel Authentication Capabilities Command

This command is sent in unauthenticated (clear) format. This command is used to retrieve capability information about the channel that the message is delivered over, or for a particular channel. The command returns the authentication algorithm support for the given privilege level. When activating a session, the privilege level passed in this command will normally be the same Requested Maximum Privilege level that will be used for a subsequent *Activate Session* command.

BMC implementations of IP-based channels must support the *Get Channel Authentication Capabilities Command* using the IPMI v1.5 packet format. BMCs that support IPMI v2.0 RMCP+ must also support the command using the IPMI v2.0/RMCP+ format.

The *Get Channel Authentication Capabilities* command can also be used as a no-op 'Ping' to keep a session from timing out.

Table 2-3, Get Channel Authentication Capabilities Command

Session Request Data		authentication type = NONE
		session seq# = null (0's)
		session ID = null (0's)
		AuthCode = NOT PRESENT
IPMI Request Data	1	Channel Number [7] - 1b = get IPMI v2.0+ extended data. 0b = Backward compatible with IPMI v1.5. Result data only returns bytes 1:9, bit [7] of byte 3 (Authentication Type Support) returns as 0b, byte 5 returns 00h. [6:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	2	Requested Maximum Privilege Level [7:4] - reserved [3:0] - requested privilege level 0h = reserved 1h = Callback level 2h = User level 3h = Operator level 4h = Administrator level 5h = OEM Proprietary level

Session Response Data		authentication type = NONE
		session seq# = null (0's)
		session ID = null (0's)
		AuthCode = NOT PRESENT
IPMI Response Data	1	Completion Code
	2	Channel Number Channel number that the Authentication Capabilities is being returned for. If the channel number in the request was set to Eh, this will return the channel number for the channel that the request was received on.
	3	Authentication Type Support Returns the setting of the Authentication Type Enable field from the configuration parameters for the given channel that corresponds to the Requested Maximum Privilege Level. [7] - 1b = IPMI v2.0+ extended capabilities available. See Extended Capabilities field, below. 0b = IPMI v1.5 support only. [6] - reserved [5:0] - IPMI v1.5 Authentication type(s) enabled for given Requested Maximum Privilege Level All bits: 1b = supported 0b = authentication type not available for use. [5] - OEM proprietary (per OEM identified by the IANA OEM ID in the RMCP Ping Response) [4] - straight password / key [3] - reserved [2] - MD5 [1] - MD2 [0] - none
	4	[7:5] - reserved Following bits apply to IPMI v1.5 and v2.0: [4] - Per-message Authentication status 0b = Per-message Authentication is enabled. Packets to the BMC must be authenticated per Authentication Type used to activate the session, and User Level Authentication setting, following. 1b = Per-message Authentication is disabled. Authentication Type 'none' accepted for packets to the BMC after the session has been activated. [3] - User Level Authentication status 0b = User Level Authentication is enabled. User Level commands must be authenticated per Authentication Type used to activate the session. 1b = User Level Authentication is disabled. Authentication Type 'none' accepted for User Level commands to the BMC. [2:0] - Anonymous Login status This parameter returns values that tells the remote console whether there are users on the system that have 'null' usernames. This can be used to guide the way the remote console presents login options to the user. (see IPMI v1.5 specification sections *6.9.1, 'Anonymous Login' Convention* and *6.9.2, Anonymous Login*) [2] - 1b = Non-null usernames enabled. (One or more users are enabled that have non-null usernames). [1] - 1b = Null usernames enabled (One or more users that have a null username, but non-null password, are presently enabled) [0] - 1b = Anonymous Login enabled (A user that has a null username and null password is presently enabled)

5	For IPMI v1.5: - reserved For IPMI v2.0+: - Extended Capabilities [7:2] - reserved [1] - 1b = channel supports IPMI v2.0 connections. [0] - 1b = channel supports IPMI v1.5 connections.
6:8	OEM ID IANA Enterprise Number for OEM/Organization that specified the particular OEM Authentication Type. Least significant byte first. Return 00h, 00h, 00h if no OEM authentication type available.
9	OEM auxiliary data. Additional OEM-specific information for the OEM Authentication Type. Return 00h if no OEM authentication type available.

2.10 Get Channel Cipher Suites Command

This command can be executed prior to establishing a session with the BMC. The command is used to look up what authentication, integrity, and authentication algorithms are supported. The algorithms are used in combination as 'Cipher Suites'.

The data is accessed 16-bytes at a time starting from List Index field value of 0 in the request and then repeating the request incrementing the List Index field each time until fewer than 16-bytes of algorithm data (or no algorithm data) is returned in the response, or the maximum List Index value has been reached.

A given Cipher Suite may only be available for establishing a session at a particular maximum privilege level or lower. For example, a Cipher Suite that has a privilege level of 'Admin' can therefore be used for any privilege level, while a privilege level of User can only be used for establish sessions with a Maximum Requested Privilege Level of User or Callback.

Because the authentication algorithm specifies the steps for authenticating the user, it is a necessary part of session establishment. Therefore an authentication algorithm number is required for all Cipher Suites. It is possible that a given Cipher Suite may not specify use of an integrity or confidentiality algorithm. If the Cipher Suite has integrity and/or confidentiality of 'none', then all the same steps for establishing a session are used (open session request/response, RAKP messages) - but the integrity (AuthCode) and confidentiality fields will be absent in packets for that are sent under the session.

Table 2-4, Get Channel Cipher Suites Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	2	Payload Type. [7:6] - reserved [5:0] - Payload Type number Typically 00h (IPMI). The Payload Type number is used to look up the Security Algorithm support when establishing a separate session for a given payload type.
	3	List Index. [7] - 1b = list algorithms by Cipher Suite 0b = list supported algorithms [6] - reserved [5:0] - List index (00h-3Fh) 00h = get first set of algorithm numbers. BMC returns up to sixteen (16) bytes per index. 0h selects the first set of 16, 1h selects the next set of 16, and so on. The BMC will return 0 bytes of supported algorithm numbers when the indexed range exceeds the number of available algorithms. A return of 0 bytes of supported algorithm numbers, or a return of fewer than 16 numbers indicates that the full list has been read. Note that Cipher
IPMI Response Data	1	Completion Code
	2	Channel Number Channel number that the Authentication Algorithms are being returned for. If the channel number in the request was set to Eh, this will return the channel number for the channel that the request was received on.
	(3:18)	Cipher Suite Record data bytes, per *Table 1-4, Cipher Suite Record Format*. Record data is 'packed'; there are no pad bytes between records. It is possible that record data will span across multiple List Index values. The BMC returns Cipher Suite Record data 16-bytes at a time, starting from List Index 00h.

2.10.1 Cipher Suite Records

The data from the *Get Channel Cipher Suites* command is issued as Cipher Suite records. Tag bits are used to delimit different fields in the record. Each record starts off with header bytes that are tagged with the value 11b (for this version of the specification, there is only one header byte). Following the header bytes are algorithm number bytes for the different algorithms that form the Cipher Suite. Each byte is tagged with the type of algorithm the number is for. Cipher Suite records are required to list algorithms in the order: Authentication Algorithm number first, Integrity Algorithm numbers next, and Confidentiality Algorithm numbers last.

If more than one algorithm of a given type is listed in the Cipher Suite Record, then any one of the algorithms can be used in combination with the other types. For example, if a Cipher Suite response returns both MD5 and MD2 as Authentication and Integrity algorithms, and !RC4 for confidentiality, then the allowed combinations are [MD2, MD2, !RC4], [MD2, MD5, !RC4], [MD5, MD2, !RC4], and [MD5, MD5, !RC4]. I.e. a remote console can negotiate for those combinations when establishing a session.

Table 1-4, Cipher Suite Record Format

bytes	Tag bits [7:6]	Data bits [5:0]
1	11b	Header. Byte 1 - Cipher Suite ID. This value is used a numeric way of identifying the Cipher Suite on the platform. It's used in commands and configuration parameters that enable and disable Cipher Suites. See *Table 2-6, Cipher Suite IDs.*
(2:4)	11b	Byte 1:3 - OEM IANA Least significant byte first. These bytes are present in the record for an OEM Cipher Suite number. 3-byte IANA for the OEM or body that defined the Cipher Suite.
var	00b	Authentication Algorithm Number. A Cipher Suite is only allowed to utilize one Authentication algorithm. See *Table 2-18, Authentication Algorithm Numbers*
var	01b	Integrity Algorithm Number(s). See *Table 2-19, Integrity Algorithm Numbers*
var	10b	Confidentiality Algorithm Number(s). See *Table 2-20, Confidentiality Algorithm Numbers*

2.10.2 Cipher Suite IDs

The following table provides the number ranges and assignments for Cipher Suite IDs. The Cipher Suite ID values are used as a way to identify different Cipher Suites in configuration parameters and IPMI commands.

The OEM IDs do not correspond to a particular Cipher Suite, but are handles that can be used to identify the Cipher Suite on a particular implementation of a BMC. I.e. the OEM Cipher Suite corresponding to "80h" can be different from one BMC to the next. These handles can, however, be used in configuration parameters and commands the same way as the IPMI-defined Cipher Suite IDs.

The *Get Channel Cipher Suites* command will return the algorithms used to form a given Cipher Suite (those numbers can then be used by a remote console in the commands for establishing a session). For OEM defined Cipher Suites, the *Get Channel Cipher Suites* command will also return the IANA for the OEM or body that defined the Cipher Suite.

Table 2-6, Cipher Suite IDs

ID	characteristics	Cipher Suite	Authentication Algorithm	Integrity Algorithm(s)	Confidentiality Algorithm(s)
1	S	(01h, 00h, 00h)	RAKP-HMAC-SHA1	None	None
2	S, A	(01h, 01h, 00h)	RAKP-HMAC-SHA1	HMAC-SHA1-96	None
3	S, A, E	(01h, 01h, 01h)	RAKP-HMAC-SHA1	SHA1 HMAC-SHA1-96	AES-CBC-128
80h-BFh	OEM specified	OEM specified	OEM specified	OEM specified	OEM specified
C0h-FFh	reserved	-	-	-	-

Key:
S = authenticated session setup (correct role, username and password/key required to establish session)
A = authenticated payload data supported.
E = authentication and encrypted payload data supported

2.11 RMCP/ASF Pong Message (Ping Response)

Table 2-7, RMCP Packet Fields for ASF Presence Pong Message (Ping Response)

	Field	size in bytes	Value
UDP Header	Source Port	2	26Fh
	Destination Port	2	from Ping request
	UDP Length	2	per UDP
	UDP Checksum	2	per UDP
RMCP Header	Version	1	6 = RMCP Version 1.0
	Reserved	1	00h
	RMCP Sequence Number	1	FFh for IPMI[2]
	Class of Message	1	06h = ASF
ASF Message	IANA Enterprise Number	4	4542 = ASF IANA
	Message Type	1	40h = Presence Pong
	Message Tag	1	from Ping request
	Reserved	1	00h
	Data Length	1	16 (10h)
	IANA Enterprise Number	4	If no OEM-specific capabilities exist, this field contains the ASF IANA (4542) and the OEM-defined field is set to all zeroes (00000000h). Otherwise, this field contains the OEM's IANA Enterprise Number and the OEM-defined field contains the OEM-specific capabilities. For IPMI v2.0 this field can contain the IPMI IANA for 'quick discovery'. If the IANA = IPMI (00001BF2h, 7154 decimal) then the OEM-defined field is used to tell the remote console what level of IPMI over LAN command support is available, provided the 'IPMI supported' bit in the supported entities field (below) is also set. This is recommended for new implementations, but is not mandatory. If the IPMI IANA is not provided, the 'IPMI supported' bit indicates that the remote console must use the *Get Channel Authentication Capabilities* command to determine the level of IPMI over LAN support.
	OEM-defined	4	This field can contain OEM-defined values; the definition of these values is left to the manufacturer identified by the preceding IANA Enterprise number. For IANA = IPMI (byte 1 returned first): byte 1: [7:2] - reserved [1] - 1b = supports IPMI v2.0/RMCP+ connections. [0] - 1b = supports IPMI v1.5 connections.

Lenovo
Ex. 1023 - Page 20

		byte 2:4 - reserved (return as 00h)
Supported Entities	1	81h for IPMI [7] 1b = IPMI Supported [6:4] Reserved [3:0] 0001b = ASF Version 1.0
Supported Interactions[1]	1	Reserved for future definition by ASF specification, set to 00000000b
Reserved	6	Reserved for future definition by ASF specification, set to 00 00 00 00 00 00h

2.12 AES-CBC-128 Encrypted Payloads

The following table summarizes the contents of the IPMI Payload when AES-CBC encryption is used.

Table 2-8, AES-CBC Encrypted Payload Fields

Field	Size	Sub field	Description
Confidentiality Header	16	Initialization Vector	For the AES algorithm in CBC mode, this field must be a 16-byte random value generated uniquely for each message (packet).
Payload Data	variable	Payload Data	
Confidentiality Trailer	variable	Confidentiality Pad	Added to the Data field to be encrypted (including the Confidentiality Pad Length field) so that they have a length that is a multiple of the block size of algorithm being used. For the AES algorithm, the block size is 16 bytes.
	1	Confidentiality Pad Length	Defines the number of Confidentiality Pad bytes present in the message. For the AES algorithm, this number will range from 0 to 15 bytes. This field is mandatory. If no *Confidentiality Pad* bytes are required, the *Confidentiality Pad Length* field is set to 00h. If present, the value of the first byte of *Confidentiality Pad* shall be one (01h) and all subsequent bytes shall have a monotonically increasing value (e.g., 02h, 03h, 04h, etc). The receiver, as an additional check for proper decryption, shall check the value of each byte of *Confidentiality Pad*. Some messages may not require padding if the messages already provide the necessary alignment.

2.12.1 Generating the Initialization Vector

The initialization vector (IV) should be unpredictable. In particular, for any given plaintext, it must not be possible to predict what the next IV will be from the last IV. For AES-CBC-128, the IV is recommended to be a 16-byte random number generated by a high quality random number generation process. See *Section 2.25, Random Number Generation*.

2.12.2 Encryption with AES

AES-128 uses a 128-bit Cipher Key. The Cipher Key is the first 128-bits of key "K2", K2 is generated from the Session Integrity Key (SIK). See *Section 2.24, Generating Additional Keying Material*.

Once the Cipher Key has been generated it is used to encrypt the payload data. The payload data is padded to make it an integral numbers of blocks in length (a block is 16 bytes for AES). The payload is then encrypted one block at a time from the lowest data offset to the highest using Cipher_Key as specified in [AES].

2.12.3 CBC (Cipher Block Chaining)

When CBC is used, before a block of payload data is encrypted it is first exclusive-ORed with the previous ciphertext block (or in the case of the first block, with the initialization vector). For AES, this means that the each 16-byte block of plaintext payload data is exclusive-ORed with the previous 16-bytes of encrypted data before being encrypted. See [MODES] for information on CBC.

For AEC-CBC-128 encrypted payloads under IPMI v2.0 RMCP+, CBC does not span between packets, it only applies to blocks within a packet. Instead, each individual packet is encrypted using a different initialization vector. Thus, packets can be decrypted even if an intermediate block is lost.

2.13 RMCP+ Authentication

RMCP+ session establishment messages are specified under the IPMI Message class for RMCP. The following messages are specified:

1. RSSP Open Session Request
2. RSSP Open Session Response
3. RAKP Message 1
4. RAKP Message 2
5. RAKP Message 3

The Payload Type field identifies which message is being carried in the IPMI Payload portion of the packet.

2.13.1 RSSP Open Session Request

The remote console sends this RSSP message to the managed system to open a protected session. The client responds with an *RSSP Open Session Response* message. Following the *Remote Console Session ID* field, this message contains one or more *Authentication Payload* proposals, one or more *Integrity Payload* proposals, and one or more *Confidentiality Payload* proposals. If the *RSSP Open Session Request* is accepted, the BMC has found a Cipher Suite that matches up with the one or more combinations of the algorithm proposals.

Note: While the format of the *RSSP Open Session Request* incorporates payload fields that can support multiple algorithms of each type, this revision of the specification only requires supporting one (1) algorithm or empty 'wildcard' field for each security algorithm payload type in the request.

The format of this message is as follows:

Table 2-9, RSSP Open Session Request

	byte	Data field
IPMI Session Header		Payload Type = RSSP Open Session Request Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h
IPMI Payload	1	Message Tag - Selected by remote console.
	2	Requested Maximum Privilege Level (Role) [7:4] - Reserved for future definition by this specification, set to 0h [3:0] - Requested Maximum Privilege Level (Role). 0h = Highest level matching proposed algorithms. BMC will pick the Cipher Suite returned in the *RSSP Open Session Response* by checking the algorithms proposed in the *RSSP Open Session Request* against the Cipher Suites available for each privilege level, starting with the "OEM Proprietary level" and progressing to lower privilege levels until a match is found. The resultant match results in an 'effective' maximum privilege level for the session. The resultant level is returned in the *RSSP Open Session Response.* 1h = CALLBACK level 2h = USER level 3h = OPERATOR level 4h = ADMINISTRATOR level 5h = OEM Proprietary level
	3:4	reserved - write as 00_00h
	5:8	Remote Console Session ID. Selected by the remote console to identify packets that are received for the given session by the remote console
	9:13	Authentication Payload. Identifies the authentication type that the managed system wants to use for the session. byte 1 - Payload Type 00h = authentication algorithm byte 2:3 - reserved = 0000h byte 4 - Payload Length in bytes (1-based). 00h = Null field ("wildcard"). BMC picks algorithm based on Requested Maximum Privilege Level and that matches with the proposed Integrity and Confidentiality payloads. If the Requested Maximum Privilege Level is 'unspecified' the BMC picks algorithm based on the Integrity and Confidentiality algorithm proposals starting from the highest privilege level until a match is found. byte 5 - Authentication Algorithm [7:6] - reserved [5:0] - Authentication Algorithm (See *Table 2-18, Authentication Algorithm Numbers*) byte 6:8 - reserved
	14:18	Integrity Payload. Identifies the integrity type that the managed system wants to use for the session. byte 1 - Payload Type 01h = integrity algorithm

	byte 2:3 - reserved = 0000h byte 4 - Payload Length in bytes(1-based) 00h = Null field ("wildcard"). BMC picks algorithm based on Requested Maximum Privilege Level and that matches with the proposed Authentication and Confidentiality payloads. If the Requested Maximum Privilege Level is 'unspecified' the BMC picks algorithm based on the Authentication and Confidentiality algorithm proposals starting from the highest privilege level until a match is found. byte 5 - Integrity Algorithm [7:6] - reserved [5:0] - Integrity Algorithm (See *Table 2-19, Integrity Algorithm Numbers*) byte 6:8 - reserved
19:23	Confidentiality Payload. Defined confidentiality algorithms are: byte 1 - Payload Type 02h = confidentiality algorithm byte 2:3 - reserved = 0000h byte 4 - Payload Length in bytes (1-based) 00h = Null field ("wildcard"). BMC picks algorithm based on Requested Maximum Privilege Level and that matches with the proposed Authentication and Integrity payloads. If the Requested Maximum Privilege Level is 'unspecified' the BMC picks algorithm based on the Authentication and Integrity algorithm proposals starting from the highest privilege level until a match is found. byte 5 - Confidentiality Algorithm [7:6] - reserved [5:0] - Confidentiality Algorithm (See *Table 2-20, Confidentiality Algorithm Numbers*) byte 6:8 - reserved

2.13.2 RSSP Open Session Response

A managed system sends this RSSP message to the management console in response to an *RSSP Open Session Request* message. Following the *Status Code*, *Mgt Console and Managed System Session ID* fields, this message contains a single *Authentication* payload, a single *Integrity* payload, and a single *Confidentiality* payload. These payloads represent the proposals that the managed system selected from the list offered by the management console.

The format of this message's *Data* section is as follows:

Table 2-10, RSSP Open Session Response

	byte	data field
IPMI Session Header		Payload Type = RSSP Open Session Response Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h
IPMI Payload	1	Message Tag - Selected by remote console.
	2	RSSP Status Code - Identifies the status of the previous message. If the previous message generated an error, then only the Status Code, Reserved, and Remote Console Session ID fields are returned. See *Table 1-15, RSSP and RAKP Message Status Codes*. The session is then immediately closed at the BMC.

3	Maximum Privilege Level (Role) - Indicates the Maximum Privilege Level allowed for the session based on the security algorithms that were proposed in the *RSSP Open Session Request.* [7:4] - Reserved for future definition by this specification, set to 0h [3:0] - Requested Maximum Privilege Level (Role). 0h = unspecified (returned with error completion code). 1h = CALLBACK level 2h = USER level 3h = OPERATOR level 4h = ADMINISTRATOR level 5h = OEM Proprietary level
4	reserved - write as 00h
5:8	Remote Console Session ID The Remote Console Session ID specified by *RSSP Open Session Request* message associated with this response.
9:12	Managed System Session ID The Session ID selected by the Managed System for this new session. The Bypass Session ID (All 0's) is not valid in this context.
13:20	Authentication Payload This payload defines the authentication algorithm proposal selected by the Managed System to be used for this session (see *Table 2-9, RSSP Open Session Request* for the definition of this payload). A single algorithm will be returned. The 'Null field' is not allowed.
21:28	Integrity Payload This payload defines the integrity algorithm proposal selected by the Managed System to be used for this session (see *Table 2-9, RSSP Open Session Request* for the definition of this payload). A single algorithm will be returned. The 'Null field' is not allowed.
29:36	Confidentiality Payload This payload defines the confidentiality algorithm proposal selected by the Managed System to be used for this session (see *Table 2-9, RSSP Open Session Request* for the definition of this payload). A single algorithm will be returned. The 'Null field' is not allowed.

2.14 Close Session Command

This IPMI command is used to immediately terminate a session in progress. It is typically used to close the session that the user is communicating over, but it can also be used to other terminate sessions in progress (provided that the user is operating at the appropriate privilege level, or the command is executed over a non-authenticated channel - e.g. the system interface).

Table 2-11, Close Session Command

	byte	data field
Request Data	1:4	Session ID
Response Data	1	Completion Code 87h = invalid session ID in request

2.15 Session Termination

The following actions can terminate a session:

- The Close Session command
- Session Inactivity Timeout (Per IPMI v1.5 section 6.11.13, Session Inactivity Timeouts)

Terminating a session causes any payloads that were activated under that session to be automatically deactivated. Terminating one session will not cause other sessions to

terminate. If multiple sessions are opened by a remote console they will need to be terminated individually.

2.16 RAKP Messages

RAKP Messages are transferred in the payload portion of an IPMI over LAN packet with a Format field set to "RMCP+". The Payload Type field indicates which RAKP message is held in the IPMI Payload portion of the packet.

2.16.1 RAKP Message 1

A remote console sends this RAKP message to the managed system to begin the session authentication process. The remote console selects a *Remote Console Random Number*, a *Maximum Requested Privilege Level (Role)*, and an optional *User Name* and sends them to the managed system along with the *Managed System Session ID* specified by the client on the previous *RSSP Open Session Response*.

Upon receiving *RAKP Message 1*, the managed system verifies that the message contains an active *Managed System Session ID* and that a session can be created using the given user information and security algorithm proposal. The managed system responds with an *RAKP Message 2*.

FYI: RAKP Message 1 corresponds roughly to an IPMI v1.5 *Get Session Challenge* request message. RAKP Message 2 corresponds roughly to an IPMI *Get Session Challenge* response message.

The format of the IPMI Session Header, Payload, and Session Trailer for the *RAKP Message 1* is shown in the following table.

Table 2-12, RAKP Message 1

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 1 Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h
IPMI Payload	1	Message Tag - Selected by remote console.
	2:4	reserved - write as 00_00h
	5:8	Managed System Session ID The Managed System's Session ID for this session, returned by the Managed System on the previous *RSSP Open Session Response* message.
	9:24	Remote Console Random Number Random number selected by the Remote Console

25	Requested Maximum Privilege Level (Role) [7:5] - Reserved for future definition by this specification, set to 000b [4] - 0b = Username/Privilege lookup. Both the Requested Privilege Level and User Name are used to look up password/key. The BMC will search the user entries starting with USER ID 1 and use the first entry that matches the specified user name and has a Maximum Privilege Level that matches the Requested Privilege Level. This can be used in combination with 'null' user names to enable a password that is just based on the requested privilege level. 1b = Name-only lookup. User Name alone is used to look up password/key. Privilege Level field acts as a 'Maximum Requested Privilege Level' ala IPMI v1.5. The rules for privilege level handling are summarized as follows: • If the Requested Privilege Level is greater than the privilege limit for the channel/user, the user will be allowed to connect but will be restricted to the channel/user privilege limit that was configured for the user. • If the Requested Privilege Level is less than the channel/user privilege limit, the user will be allowed to connect and Request Privilege Level will become the effective privilege limit for the user. I.e. the user will not be able to raise their privilege level higher than the Requested Privilege Level. [3:0] - Requested Maximum Privilege Level 0h = reserved 1h = CALLBACK level 2h = USER level 3h = OPERATOR level 4h = ADMINISTRATOR level 5h = OEM Proprietary level
26:27	reserved - write as 0000h
28	User Name Length 00h No name present 01h-10h Name length 11h-FFh Reserved for future definition by this specification
29:44	User Name ASCII character Name that the user at the Remote Console wishes to assume for this session. No NULL characters (00h) are allowed in the name. Sixteen-bytes, max.

2.16.2 RAKP Message 2

The managed system sends this RAKP message to a remote console in response to the receipt of an *RAKP Message 1.* Once *RAKP Message 1* has been validated the managed system selects a *Managed System Random Number* and computes a *Key Exchange Authentication Code* over the values specified by the RAKP algorithm. The managed system sends those values along with the *Managed System Globally Unique ID* (GUID) and the *Remote Console Session ID* (sent by the console on the previous *RSSP Open Session Request*) to the remote console.

Upon receiving *RAKP Message 2*, the remote console verifies that the *Remote Console Session ID* is active and that the *Managed System GUID* matches the managed system that the remote console has associated with the session. The remote console then validates the *AuthCode (Integrity Check Value)* and responds with an *RAKP Message 3*.

The format of an *RAKP Message 2* message's *Data* section is as follows:

Table 2-13, RAKP Message 2

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 2 Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h ←**TBD:**need to verify whether this is 0 or not.
IPMI Payload	1	Message Tag - Selected by remote console.
	2	RSSP Status Code - Identifies the status of the previous message. If the previous message generated an error, then only the Completion Code, Reserved, and Remote Console Session ID fields are returned. If the Remote Console Session ID field is indeterminate (as would be the case if the Managed System Session ID in RAKP Message 1 were invalid) then the Remote Console Session ID field will be set to all zeros. On error, the remote console can attempt to correct the error and send a new RAKP Message 1. Note that the remote console must change the Message Tag value to ensure the BMC sees the message as a new message and not as a retry. See *Table 1-15, RSSP and RAKP Message Status Codes* for the status codes defined for this message.
	3:4	Reserved - write as 00_00h.
	5:8	Remote Console Session ID - The Remote Console Session ID specified by the *RSSP Open Session Request* message associated with this response.
	9:24	Managed System Random Number - Random number selected by the managed system.
	25:40	Managed System GUID - The Globally Unique ID (GUID) of the Managed System. This value is specified by the client system's SMBIOS implementation.
	41:N	Key Exchange Authentication Code An integrity check value over the relevant items specified by the RAKP algorithm for RAKP Message 2. The size of this field depends on the specific Authentication Algorithm (e.g. for RAKP-HMAC-SHA1 the Authentication Algorithm would be HMAC using SHA1) that was identified in the *RSSP Open Session Response*.

2.16.3 RAKP Message 3

A remote console sends this RAKP message to a managed system in response to the receipt of an *RAKP Message 2.* Once it validates *RAKP Message 2*, the remote console creates a *Session Integrity Key* using the values specified by the RAKP algorithm. The remote console then computes an *Integrity Check Value* over the values specified by the RAKP algorithm, and sends that along with the *Managed System Session ID* (sent by the managed system on the previous *RSSP Open Session Response* message) to the managed system.

After receiving *RAKP Message 3*, the managed system verifies that the *Managed System Session ID* is active and then validates the *Integrity Check Value.* If the *Integrity Check Value* is valid, the managed system creates a *Session Integrity Key* using the values specified by the RAKP algorithm. With the shared *Session Integrity Key* in place,

integrity protected messages can now be exchanged between the remote console and the managed system.

The format of an *RAKP Message 3* message's *Data* section is as follows:

Table 2-14, RAKP Message 3

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 3 Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h ←**TBD:**need to verify whether this is 0 or not.
	1	Message Tag - Selected by remote console.
	2	RSSP Status Code Identifies the status of the previous message. If the previous message generated an error, then only the Completion Code, Reserved, and Managed System Session ID fields are returned. If the BMC receives an error from the remote console, it will immediately close the session if the Key Exchange Authentication code (below) is valid. See *Table 1-15, RSSP and RAKP Message Status Codes* for the status codes defined for this message.
	3:4	Reserved - write as 00_00h.
	5:8	Managed System Session ID The Managed System's Session ID for this session, returned by the managed system on the previous *RSSP Open Session Response* message.
	9:N	Key Exchange Authentication Code An integrity check value over the relevant items specified by the RAKP algorithm for Message 3. The size of this field depends on the specific Authentication Algorithm that was selected when the session was created.

2.16.4 RAKP Message 4

A managed client sends this RAKP message to a management console in response to the receipt of an *RAKP Message 3.* Once *RAKP Message 3* has been validated, the managed client computes an *Integrity Check Value* over the values specified by the RAKP algorithm. The managed client then sends the *Mgt Console Session ID* and the *Integrity Check Value* to the management console.

Upon receiving *RAKP Message 4*, the management console verifies that the *Mgt Console Session ID* is active and then validates the *Integrity Check Value.*

The format of an *RAKP Message 4* message's *Data* section is as follows:

Table 1-14, RAKP Message 4

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 4 Session Sequence Number = 00_00_00_00h Session ID = 00_00_00_00h ←**TBD:**need to verify whether this is 0 or not.
	1	Message Tag - Selected by remote console.
	2	RSSP Status Code - Identifies the status of the previous message. If the previous message generated an error, then only the Status Code, Reserved, and Management Console Session ID fields are returned. See 2.1.3.6.1 for the status codes defined for this message.
	3:4	Reserved - Reserved for future definition by this specification set to 000000h.
	5:8	Mgt Console Session ID The Mgt Console Session ID specified by the RSSP *Open Session Request (83h)* message associated with this response.
	9:N	Integrity Check Value An integrity check value over the relevant items specified by the RAKP algorithm for Message 4. The size of this field depends on the specific algorithm that was selected when the session was created.

2.17 RSSP and RAKP Message Status Codes

The table below lists the status codes for specific RSSP and RAKP messages.

Table 1-15, RSSP and RAKP Message Status Codes

Status Code	Description	Message			
		RSSP Open Session Response	RAKP Msg 2	RAKP Msg 3	RAKP Msg 4
00h	No errors	X	X	X	X
01h	Insufficient resources to create a session	X			
02h	Invalid session ID	X	X	X	X
03h	Invalid payload type	X			
04h	Invalid authentication algorithm	X			
05h	Invalid integrity algorithm	X			
06h	No matching authentication payload	X			
07h	No matching integrity payload	X			
08h	Inactive session ID		X	X	X
09h	Invalid role		X		
0Ah	Unauthorized role		X		
0Bh	Insufficient resources to create a session at the requested role		X		
0Ch	Invalid name length		X		
0Dh	Unauthorized name		X		
0Eh	Unauthorized GUID			X	
0Fh	Invalid integrity check value			X	X
10h	Invalid confidentiality algorithm	X			
11h	No Cipher Suite match with proposed security algorithms	X			
12h-FFh	Reserved for future definition by this specification				

2.18 IPMI v2.0 RMCP+ Payload Types

The Payload Type field in the IPMI v2.0/RMCP+ packet carries a *Payload Type Number* that identifies what type of payload field is being carried in that particular packet.

The *Payload Type Numbers* (also referred to as *payload types*) are classified into three main categories: Standard Payload Types - used to identify payloads that are specified by the IPMI specifications, Session Setup Payload Types - used to identify payloads that are for session setup messages specified by the IPMI specifications, and OEM Payload Types that are used to identify payloads that are specified by a given OEM.

The following table lists the assignment and ranges of the Payload Type Numbers. The complete identification of an OEM Payload is given by the combination of a three-byte IANA ID for the OEM plus a two-byte OEM Payload ID that is assigned and defined by the given OEM. These can either be carried explicitly in each packet (adding six bytes of overhead) or an application can elect to use an OEM Payload Type Handle. The OEM Payload Type Handle in the Payload Type provides a value that represents a particular OEM IANA and OEM Payload ID on a system. The *Get Channel Payload Support* command is used discover what OEM Payloads (if any) are used on the managed system, and to associate the OEM Payload Type number with the OEM IANA ID and Payload ID.

OEM Payload Handle assignments can vary from system to system. For example, “OEM0” on one system may not correspond to the same type of OEM Payload as “OEM0” on another system. Software that uses OEM Payload Handles must not assume that a given OEM Payload Handle number will correspond to a particular OEM IANA and Payload ID combination across multiple systems. Software must use the *Get Channel Payload Support* command to discover the relationship.

Associated with each payload type is a format version number that provides information on the backward compatibility with different versions of the payload type. See *Section 3.9, Get Channel Payload Version Command,* for more information.

Table 2-17, Payload Type Numbers

number*	type	major format version	minor format version
	Standard Payload Types		
0h	IPMI Message	1h	0h
1h	SOL (serial over LAN)	1h	0h
2h	OEM Explicit (This payload type indicates that the packet includes explicit OEM IANA and OEM Payload ID fields, rather than using an OEM Payload Type Handle to identify the type of payload. When used, this option adds 6 bytes to the overhead of the packet.)	OEM specified according to OEM IANA and OEM Payload ID.	OEM specified according to OEM IANA and OEM Payload ID.
	Session Setup Payload Types		
10h	RSSP Open Session Request	1h	0h
11h	RSSP Open Session Response	1h	0h
12h	RAKP Message 1	1h	0h
13h	RAKP Message 2	1h	0h
14h	RAKP Message 3	1h	0h
15h	RAKP Message 4	1h	0h
	OEM Payload Type Handles		
20h-27h	OEM payloads OEM0 through OEM7, respectively.	OEM specified according to OEM IANA and OEM Payload ID.	OEM specified according to OEM IANA and OEM Payload ID.
all other	reserved		

* The payload type number is a 6-bits (00h-3Fh).

2.19 Session Sequence Numbers

The session sequence number is a 32-bit, unsigned, value. The session sequence number is used for protecting against replay attacks. The session sequence number is *not* used for matching IPMI requests and responses or tracking lost packets. The IPMI Sequence (Seq) field, or similar field in the particular payload is used for that purpose. The sender of the packet increments the session sequence number for every packet that gets transmitted even if the payload of the content is a 'retry'. Session Sequence Numbers are generated and tracked on a per-session basis. I.e. there are separate sets of sequence numbers and sequence number handling for each session.

Sequence numbers only apply to packets that are transmitted within the context of an IPMI session. Certain IPMI commands and protocol messages are accepted 'outside of a session'. For these packets, the sequence number fields are always set to 0000_0000h.

2.20 IPMI v1.5 Session Sequence Number Handling

For IPMI v1.5 sessions, there are just two Session Sequence Numbers associated with a given session: the *Inbound* Session Sequence Number and the *Outbound* Session Sequence Numbers. (The inbound and outbound directions are defined with respect to the BMC. Inbound messages are from the remote console to the BMC, while outbound messages are from the BMC to the remote console.) The inbound and outbound sequence numbers are updated and tracked independently. I.e. since the number of incoming packets and outgoing packets will typically vary, the inbound and outbound sequence numbers will not stay in lock step with one another.

The BMC and the remote console independently select the starting session sequence number for the messages they receive. Typically, this is done using a random number in order to further reduce the likelihood of a playback attack. The remote console sets the starting values for the outbound session sequence number when it sends the first *Activate Session* command for an authenticated session. The *Activate Session* response is the first authenticated outbound (BMC to remote console) message. This response message uses the initial outbound session sequence number value that the remote console delivered in the prior *Activate Session* command request. The BMC must increment the outbound session sequence number by one (1) for each subsequent outbound message from the BMC.

The first authenticated inbound message uses the initial inbound (remote console to BMC) session sequence number value that was returned by the BMC in the response to the *Activate Session* command. The remote console must increment the inbound session sequence number by one (1) for each subsequent message it sends to the BMC.

2.20.1 IPMI v1.5 Inbound Session Sequence Number Tracking and Handling

Session sequence numbers are tracked on a per-session basis. At a minimum, the BMC is required to track that the inbound sequence number is increasing, and to silently discard the packet if the sequence number is **eight** counts or more from than the last value received. (An implementation is allowed to contain a proprietary configuration option that enables a larger sequence number difference, as long as the standard of +eight can be restored.)

An implementation can elect to terminate the session if it receives a number of sequence numbers that are more than eight counts from the last value received.

Valid packets (packets with good data integrity checks and signature) to a given session that have the same inbound sequence number as an earlier packet are considered to be duplicate packets and are silently discarded (even if the message content is different).

2.20.2 IPMI v1.5 Out-of-order Packet Handling

In order to avoid closing a session because a packet was received out-of-order, the BMC must implement one of two options:

> Option 1: Advancing eight-count (or greater) window. Recommended. Track which packets have been received that have sequence numbers up to eight counts *less* than the highest last received sequence number, tracking which of the prior eight sequence numbers have been received. Also accept packet with sequence numbers that are up to eight counts greater than the last received sequence number, and set that number as the new value for the highest sequence number received. This option is illustrated in *Appendix A - Previous Sequence Number Tracking*.

Option 2: Drop any packets with sequence numbers that are lower than the last valid value received. While simpler than option 1, this option is not recommended except for resource-constrained implementations due to the fact that any out-of-order packets will require the remote console to timeout and retransmit.

Sequence number wrap-around must be taken into account. When a sequence number advances from FFFF_FFFFh to 0000_0000h, the an eight count window will range from FFFF_FFF9h to 0000_0000h. The next window would go from FFFF_FFFAh to 0000_00001h, etc.

When dropping packets because of sequence number, any packet with an illegal, duplicate, or out-of-range sequence can be dropped without having to verify the packet integrity data (AuthCode) signature first. When accepting packets, the BMC must apply any packet integrity and authentication code checks before accepting the packet's sequence number.

2.20.3 IPMI v1.5 Outbound Session Sequence Number Tracking and Handling

The remote console is required to handle outbound session sequence number tracking in the same manner as the BMC handles the inbound session sequence number, except that Option 2 (above) should not be used as a means of handling out-of-order packets.

2.21 IPMI v2.0 RMCP+ Session Sequence Number Handling

For IPMI v2.0 RMCP+ sessions, there are two *sets* of Session Sequence Numbers for a given session. One set of inbound and outbound sequence numbers is used for authenticated (signed) packets, and the other set is used for unauthenticated packets. The inbound and outbound sequence numbers for authenticated packets are updated and tracked independently from the inbound and outbound sequence numbers for unauthenticated packets.

IPMI v2.0 RMCP+ Session Sequence Numbers are used for rejecting packets that may have been duplicated by the network or intentionally replayed.

The individual Session Sequence Numbers is are initialized to zero whenever a session is created and incremented by one at the start of outbound processing for a given packet (i.e. the first transmitted packet has a '1' as the sequence number, not 0). Session Sequence numbers are incremented for every packet that is transmitted by a given sender, regardless of whether the payload for the packet is a 'retry' or not.

For IPMI v2.0 RMCP+ packets, sequence numbers don't wrap. If the sequence number reaches FFFF_FFFFh the session is automatically closed and must be re-established by the remote console.

When dropping packets because of sequence number, any packet with an illegal, duplicate, or out-of-range sequence can be dropped without having to verify the packet integrity data (AuthCode) signature first. When accepting packets, the BMC must apply

any packet integrity and authentication code checks before accepting the packet's sequence number.

2.21.1 IPMI v2.0 RMCP+ Sliding Window

IPMI v2.0 RMCP+ uses a 'sliding window' for tracking sequence numbers for received packets. This sliding window is used for rejecting packets that have sequence numbers that are significantly out-of-range with respect to the sequence number for the most recently accepted packet while allowing a number of out-of-order packets to be accepted.

In order for a packet to be accepted by the BMC, its sequence number must fall within a 32 count sliding window, where packets will be accepted if they are within plus 15 or minus 16 counts of the highest sequence number that was previously accepted, and they are not duplicates of any previously received sequence numbers.

2.22 Authentication, Integrity, and Encryption Algorithm Numbers

The Authentication Algorithm Number specifies the type of authentication "handshake" process that is used and identifies any particular variations of hashing or signature algorithm that is used as part of the process. For example, RAKP-HMAC-SHA1 specifies the use of RAKP messages for the key exchange portion of establishing the session, and that HMAC-SHA1 is used to create the Key Exchange Authentication Code that is a parameter in RAKP Message 2 and RAKP Message 3.

Table 2-18, Authentication Algorithm Numbers

number*	type	Mandatory / Optional[1]
00h	none	M
01h	RAKP-HMAC-SHA1	M
02h	RAKP-HMAC-MD5-128	O
03h	RAKP-MD5-128	O
C0h-FFh	OEM	O
all other	reserved	-

* The number range is limited to six (6) bits (00h-3Fh)

The Integrity Algorithm Number specifies the algorithm used to generate the contents for the AuthCode "signature" field that accompanies authenticated IPMI v2.0/RMCP+ messages once the session has been established. Unless otherwise specified, the algorithm applies to the packet data starting with the AuthType/Format field up to and including the field that immediately precedes the AuthCode field itself. The AuthCode field is a common part of the IPMI

Table 2-19, Integrity Algorithm Numbers

[1] Mandatory/Optional is with respect to BMC support. It is recommended that remote consoles support all specified algorithms in order to support maximum number of BMC implementations.

number*	type	Mandatory / Optional[1]
00h	none	M
01h	HMAC-SHA1-96	M
02h	HMAC-MD5-128	O
03h	MD5-128	O
C0h-FFh	OEM	O
all other	reserved	-

* The number range is limited to six (6) bits (00h-3Fh)

The Confidentiality Algorithm Number specifies the encryption/decryption algorithm field that is used for encrypted payload data under the session. The 'encrypted' bit in the payload type field being set identifies packets with payloads that include data that is encrypted per this specification. When payload data is encrypted, there may be additional "Confidentiality Header" and/or "Confidentiality Trailer" fields that are included within the payload. The size and definition of those fields is specific to the particular confidentiality algorithm.

Table 2-20, Confidentiality Algorithm Numbers

number*	type	Mandatory / Optional[1]
00h	none	M
01h	AES-CBC-128 (See Section 2.12, AES-CBC-128 Encrypted Payloads, for more information)	M
02h	RC4-128-MD5	O
C0h-FFh	OEM	O
all other	reserved	-

* The number range is limited to six (6) bits (00h-3Fh)

2.23 RSSP Authenticated Key-Exchange Protocol (RAKP)

RSSP can support a number of different authentication and key exchange protocols during its *Creation* phase. For this specification, the mandatory-to-implement authentication and key exchange protocol is the RSSP Authenticated Key-Exchange Protocol (RAKP). RAKP (defined below) was developed based on the Authenticated Key Exchange Protocol (AKEP) defined by Bellare and Rogaway in [BR1].

RAKP uses pre-shared symmetric keys and HMAC-based integrity algorithms to mutually authenticate a remote console to a given managed system and to generate pair-wise unique symmetric keying material that can be used with a number of integrity and confidentiality algorithms to provide protection for RMCP messages. For this specification, RAKP shall use the HMAC-SHA1 integrity algorithm defined in [RFC2104] and the HMAC-SHA1-96 integrity algorithm defined in [RFC2404].

RAKP also supports the concept of remote console user "roles" and optionally "usernames" (e.g. operator "x" or administrator "y"), which are established by RAKP when a session is created.

Examples of behavior that can be controlled include the roles that the managed system can use to establish sessions (e.g. operator-only sessions) and the roles and names

(optional) allowed to execute each RMCP message the managed system might receive during a given session.

Before a given managed system's RMCP implementation can become operational, it must be configured with various RMCP-related parameters. This includes installing user passwords (keys) and usernames, setting up access rights for the individual users, and configuring which Cipher Suites are used for authenticated and/or encrypted transfers with the managed system.

The managed system can be configured with keys for each username, or 'null' usernames can be used, in which case the key is associated solely with a given privilege level (role). The different user keys are specified using the notation $\mathbf{K_{[UID]}}$, where UID represents the User ID number that is used in the user-specific configuration commands in IPMI.

The user keys are 'shared secrets' between the BMC and the remote console(s). RAKP/RSSP does not include a secure, confidential mechanism for installing and distributing user keys between BMCs and remote consoles. The installation and distribution of user keys can typically be accomplished with a software utility that uses OS-provided mechanisms for the secure transfer of keys. If authentication and encryption are available, an 'Admin' level user can use IPMI commands such as *Set User Password* for remotely updating and configuring user key and privilege information over an authenticated and confidential session to the BMC.

An additional key, $\mathbf{K_G}$, is used for key generation operations. The scope of these keys (shared by multiple managed systems and the remote console or pair-wise unique for each managed system and the remote console) is a local policy issue that is determined by the equipment owner at the time of installation.

Once this and other necessary RMCP-related data is installed in the managed system and the managed system is initialized, the remote console can initiate sessions with the managed system. Following the exchange of RMCP Presence Ping/Pong and RSSP Open Session Request/Response messages (exchanging session IDs and selecting RAKP for use), the remote console starts the RAKP protocol. First, the remote console selects a random number, $\mathbf{R_M}$, a requested role, $\mathbf{Role_M}$, a user name length, $\mathbf{ULength_M}$, a user name (optional – denoted by < > below), $\mathbf{UName_M}$, and the managed system's session ID, $\mathbf{SID_C}$, and sends them to the managed system as Message 1.

Message 1: Remote Console —► Managed System

$$\mathbf{SID_C, R_M, Role_M, ULength_M, < UName_M >}$$

After receiving Message 1, the managed system verifies that the value $\mathbf{SID_C}$ is active and that a session can be created using $\mathbf{Role_M}$, $\mathbf{ULength_M}$, and (optional), $\mathbf{UName_M}$ for the given selections for security algorithms.

If the request is valid, the managed system then selects a random number, $\mathbf{R_C}$, and sends

to the remote console as Message 2 the values $\mathbf{SID_M}$, $\mathbf{R_C}$, and $\mathbf{GUID_C}$ as well as the HMAC per [RFC2404] of the values ($\mathbf{SID_M}$, $\mathbf{SID_C}$, $\mathbf{R_M}$, $\mathbf{R_C}$, $\mathbf{GUID_C}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, $\mathbf{< UName_M >}$) generated using key $\mathbf{K_{[UID]}}$ selected by the given username, $\mathbf{UName_M}$, and role, $\mathbf{Role_M}$.

Message 2: Managed System —► Remote Console

SID_M, R_C, $GUID_C$,
$HMAC_{K[UID]}$ (SID_M, SID_C, R_M, R_C, $GUID_C$, $Role_M$, $ULength_M$, < $UName_M$ >)

Where:

SID_M	=	Remote_Console_Session_ID
SID_C	=	Managed_System_Session_ID
R_M	=	Remote Console Random_Number
R_C	=	Managed System Random Number
$GUID_C$	=	Managed_System_GUID
$Role_M$	=	Requested Privilege Level (Role) (this is the entire byte holding the Requested Privilege Level field)
$ULength_M$	=	User Name Length byte
$UName_M$	=	User Name bytes (if present)

After receiving Message 2, the remote console verifies that the value $\mathbf{SID_M}$ is active and that $\mathbf{GUID_C}$ matches the managed system that the remote console is expecting to communicate with. The remote console then validates the HMAC. If the HMAC is valid, the remote console creates the Session Integrity Key (SIK) by generating an HMAC per [RFC2104] of the concatenation of $\mathbf{R_M}$, $\mathbf{R_C}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, and (optional) $\mathbf{UName_M}$ using key $\mathbf{K_G}$ (note – no truncation).

$$\mathbf{SIK = HMAC_{KG}\ (R_M \mid R_C \mid Role_M \mid ULength_M \mid < UName_M >)}$$

Then the remote console sends to the managed system as Message 3 the value $\mathbf{SID_C}$ and the HMAC per [RFC2404] of the values ($\mathbf{R_C}$, $\mathbf{SID_M}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, $\mathbf{< UName_M >}$) generated using key $\mathbf{K_{[UID]}}$ selected by the username, $\mathbf{UName_M}$, and role $\mathbf{Role_M}$.

Message 3: Remote Console —► Managed System

$$\mathbf{SID_C,\ HMAC_{K[UID]}\ (R_C,\ SID_M,\ Role_M,\ ULength_M,\ < UName_M >)}$$

After receiving Message 3, the managed system verifies that the value $\mathbf{SID_C}$ is active and then validates the HMAC. If the HMAC is valid, the managed system creates the SIK by generating an HMAC per [RFC2104] of the concatenation of $\mathbf{R_M}$, $\mathbf{R_C}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, and (optional) $\mathbf{UName_M}$ using key $\mathbf{K_G}$ (note – no truncation).

Lenovo
Ex. 1023 - Page 38

$$\mathbf{SIK = HMAC_{KG}\ (R_M \mid R_C \mid Role_M \mid ULength_M \mid < UName_M >)}$$

The managed system then sends to the management console as Message 4 the values $\mathbf{SID_M}$, and the HMAC per [RFC2404] of the values ($\mathbf{R_M}$, $\mathbf{SID_C}$, $\mathbf{GUID_C}$) generated using key **SIK**. The managed system then transitions into the *Message Transfer* session state,

Message 4: Managed System —► Mgt Console

$$\mathbf{SID_M,\ HMAC_{SIK}\ (R_M,\ SID_C,\ GUID_C)}$$

After receiving Message 4, the management console verifies that the value $\mathbf{SID_M}$ is active and then validates the HMAC. If the HMAC is valid, the management console has verification that mutual authentication with the managed system was successful and that the same pair-wise unique SIK was successfully generated on both ends of the connection. The management console then transitions into the *Message Transfer* session state (the session is now active and, if authentication or authentication/encryption have been enabled, the transfer of authenticated and authenticated/encrypted payloads can commence).

The same RAKP steps are followed for session establishment even if the Cipher Suite indicates that there are no integrity or encryption algorithms required for the session.

2.24 Generating Additional Keying Material

Because this specification supports both integrity and confidentiality services for a session, RSP needs more keying material than can be provided by SIK alone. As a result, all keying material for the RSP integrity and confidentiality algorithms will be generated by processing a pre-defined set of constants using HMAC per [RFC2104], keyed by SIK.

$$\mathbf{K_1 = HMAC_{SIK}\ (const\ 1)}$$
$$\mathbf{K_2 = HMAC_{SIK}\ (const\ 2)}$$
$$\mathbf{K_3 = HMAC_{SIK}\ (const\ 3)}$$

These constants are constructed using a hexadecimal octet value repeated up to the HMAC block size in length starting with the constant 01h. This mechanism can be used to derive up to 255 HMAC-block-length pieces of keying material from a single SIK. For the mandatory-to-implement integrity and confidentiality algorithms defined in this specification, processing the first two (2) constants will generate the require amount of keying material.

```
Const 1   = 0x0101010101010101010  0101010101010101010
Const 2   = 0x0202020202020202020  0202020202020202020
Const 3   = 0x0303030303030303030  0303030303030303030
                          .
                          .
Const 255 = 0xFFFFFFFFFFFFFFFFFFF  FFFFFFFFFFFFFFFFFFF
```

2.25 Random Number Generation

Algorithms for authentication (data integrity) and confidentiality (for Initialization Vectors) depend on "quality" random numbers for their security. Quality in this context means that the numbers must be random in a cryptographic sense (i.e., they must be genuinely unpredictable). To ensure that a baseline-level of quality random numbers are provided for remote consoles and managed systems, this specification recommends the following algorithm be used for random number generation for use in authentication and data integrity algorithms if no other higher-quality source of random numbers is available (e.g., a hardware random number generator).

2.25.1 Random Number Key

Each BMC is configured with a 160-bit key, $\mathbf{K_R}$, which is unique for each managed system. The value of this key cannot be reused during the lifetime of $\mathbf{K_{[UID]}}$ and $\mathbf{K_G}$. The *Set Security Keys* command can be used for setting this key. Note that to avoid the possibility of run-time software changing this key, the BMC includes an option for the key to be locked so that run-time software cannot change the key.

2.25.2 Random Number Generator Counters

The managed system maintains two (2) 32 bit counters, $\mathbf{C_P}$ and $\mathbf{C_Q}$. $\mathbf{C_P}$ is used to count the number of device power cycles and its value is saved in non-volatile storage. $\mathbf{C_Q}$ is used to count the number of random number generation requests per power cycle and is volatile. Whenever a new $\mathbf{K_R}$ is installed in non-volatile storage, the counters are reset to zero (0). Once initialized, $\mathbf{C_P}$ is incremented by one (1) after each power cycle and its new value is again saved in non-volatile storage. $\mathbf{C_Q}$ is reset to zero (0) on each power cycle (i.e., its value is not saved across power cycles).

2.25.3 Random Number Generator Operation

The managed system creates a random number by generating an HMAC per [RFC2104] of the concatenation of $\mathbf{C_P}$ and $\mathbf{C_Q}$ using key $\mathbf{K_R}$ using the SHA1 hash function. The output of this generator is a 160-bit pseudo-random number. Uses that require fewer bits can draw the required number of bits from the 160-bit value. Since the value is random, it shouldn't matter which bits are used. Most implementations will simply take either the most-significant 'N' bits or least-significant, whichever is most convenient.

$$\textbf{Random Number} = \mathbf{HMAC_{KR}}\ (\mathbf{C_P} \mid \mathbf{C_Q})$$

$\mathbf{C_Q}$ is incremented by one (1) after each random number generation request. After each power cycle, the value of $\mathbf{C_Q}$ is reset to zero (0) (i.e., its value is not saved across power cycles). If during a given power cycle, $\mathbf{C_Q}$ rolls-over to zero, the managed system must increment $\mathbf{C_P}$ by one (1) and save its new value back into non-volatile storage.

Lenovo
Ex. 1023 - Page 40

2.26 Extended LAN Configuration Data

Table 2-21, Extended LAN Configuration Data, lists new configuration parameters for IPMI v2.0/RMCP+. Parameters are read-write and non-volatile unless otherwise specified. The IPMI *Set/Get LAN Configuration Parameters* commands are used for accessing this configuration data.

Table 2-21, Extended LAN Configuration Data

Parameter	#	Parameter Data (non-volatile unless otherwise noted) [1]
802.1q VLAN ID (12-bit)	20	Byte 1: [7:0] - Least significant 8-bits of the VLAN ID. 00h if VLAN ID not used. Byte 2: [7] - VLAN ID enable. 0b = disabled, 1b = enabled. If enabled, the BMC will only accept packets for this channel if they have 802.1q fields and their VLAN ID matches the [6:4] - reserved [3:0] - most significant four bits of the VLAN ID
802.1q VLAN Priority	21	Byte 1: [7:5] - reserved [2:0] - Value for Priority field of 802.1q fields. Ignored when VLAN ID enable is 0b (disabled) - See *802.1q VLAN ID* parameter, above. Setting is network dependent. By default, this should be set to 000b.
IPMI v2.0 Messaging Fixed Cipher Suite Entry Support (Read Only)	22	This parameter provides a count of the number (8 max.) of fixed Cipher Suites for use with IPMI Messaging on the given channel. Software can find out what security algorithms are associated with given Cipher Suite ID by using the *Get Channel Cipher Suites* command. In addition, there are Cipher Suite IDs assigned for standard Cipher Suites (see *Table 2-6, Cipher Suite IDs*) byte 1: [7:4] - reserved [3:0] - Fixed Entry count Number of configurable Cipher Suite entries, 1-based, 8h max. Configurable entries follow the non-configurable entries. E.g. if there are eight total entries, and three are configurable, then entries A:E would be non-configurable and F:H would be configurable.
IPMI v2.0 Messaging Fixed Cipher Suite Entries	23	This parameter contains zero to eight (8) bytes of fixed Cipher Suite IDs for Cipher Suites that can be used for establishing an IPMI messaging session with the BMC. The number of bytes supported is given in the preceding parameter. byte 1: Cipher Suite ID entry A. 00h = unspecified. byte 2: Cipher Suite ID entry B. 00h = unspecified. ... byte 8: Cipher Suite ID entry H. 00h = unspecified.

Parameter	#	Parameter Data (non-volatile unless otherwise noted) [1]
IPMI v2.0 Messaging Fixed Cipher Suite Privilege Levels	24	This parameter allows the configuration of which privilege levels are associated with each fixed Cipher Suite. The total number of nibbles supported (zero to eight) matches the number of fixed Cipher Suite IDs. byte 1: Maximum Privilege Level for 1st and 2nd Cipher Suites [7:4] - Maximum Privilege Level for 2nd Cipher Suite [3:0] - Maximum Privilege Level for 1st Cipher Suite 0h = Unspecified (given Cipher Suite is unused) 1h = Callback level 2h = User level 3h = Operator level 4h = Administrator level 5h = OEM Proprietary level byte 2: Maximum Privilege Level for 3rd and 4th Cipher Suites byte 3: Maximum Privilege Level for 5th and 6th Cipher Suites byte 4: Maximum Privilege Level for 7th and 8th Cipher Suites
IPMI v2.0 Messaging Configurable Cipher Suite Entry Support (Read Only)	25	This parameter provides a count of the number (8 max.) of configurable Cipher Suites for use with IPMI Messaging on the given channel. byte 1: [7:4] - reserved [3:0] - Configurable Entry Count. Total number of bytes of Configurable Cipher Suite entries in the following parameter. 1-based. 8h max.
IPMI v2.0 Messaging Configurable Cipher Suite Entries	26	This parameter contains zero to eight (8) bytes of configurable Cipher Suite IDs for Cipher Suites that can be used for establishing an IPMI messaging session with the BMC. Note: this parameter may be unsupported if the preceding parameter (Configurable Entry Count) = 0h. byte 1: Cipher Suite ID entry A. 00h = unspecified. byte 2: Cipher Suite ID entry B. 00h = unspecified. ... byte 8: Cipher Suite ID entry H. 00h = unspecified.
IPMI v2.0 Messaging Fixed Cipher Suite Privilege Levels	27	This parameter allows the configuration of which privilege levels are associated with each fixed Cipher Suite. The number of nibbles supported (zero to eight) matches the number of configurable Cipher Suite IDs. byte 1: Maximum Privilege Level for configurable Cipher Suites A & B [7:4] - Maximum Privilege Level for Cipher Suite B [3:0] - Maximum Privilege Level for Cipher Suite A 0h = Unspecified (given Cipher Suite is unused) 1h = Callback level 2h = User level 3h = Operator level 4h = Administrator level 5h = OEM Proprietary level byte 2: Maximum Privilege Level for configurable Cipher Suites C & D byte 3: Maximum Privilege Level for configurable Cipher Suites E & F byte 4: Maximum Privilege Level for configurable Cipher Suites G & H

Lenovo
Ex. 1023 - Page 42

3. RMCP+ Support and Payload Commands

The following sections list the commands associated with discovering, enabling, and activating payloads under IPMI v2.0/RMCP+. Also included in this section are updates and additions to IPMI commands to support IPMI v2.0/RMCP+ sessions, authentication, and configuration.

Table 3-1, RMCP+ Support and Payload Commands

Command	Section Defined	O/M
Activate Payload	3.1	O[1]
Deactivate Payload	3.2	O[1]
Suspend/Resume Payload Encryption Command	3.3	O[4]
Get Payload Activation Status	3.4	O[1]
Get Payload Instance Info	3.5	O[1]
Set User Payload Access Command	3.6	O[1]
Get User Payload Access Command	3.7	O[1]
Get Channel Payload Support Command	3.8	O[1]
Get Channel Payload Version Command	3.9	O[1]
Get Channel OEM Payload Info Command	3.10	O[1,2]
Set User Password	3.11	O[3]
Set Channel Security Keys	3.12	O

1. Mandatory if standard payloads other than IPMI Messaging are available on the channel.
2. Mandatory if OEM payloads are available on the channel.
3. Mandatory if session-based channels are supported.
4. Mandatory for standard payloads per *Table 3-4, Payload-specific Encryption Behavior*, when a channel supports CipherSuites that allow a session to be established with a standard Cipher Suite that supports encryption.

3.1 Activate Payload Command

This command is used for activating and deactivating a payload type under a given IPMI session. The ability to execute this command is determined via the user's privileges as assigned via the *Set User Payload Access* command.

The *Activate Payload* command may return a port number that is separate from the port number for the session that the command was issued under. In this case, the remote console must establish a session on the port number that the *Activate Session* command returned. The remote console must then issue the *Activate Payload* command on that port number in order to actually activate the payload. It is possible that the remote console already had a session active on the given port number. If the privileges associated with that session are sufficient (this will typically be the case unless the remote console activated the session at a privilege level that was lower than the maximum level for the user) the remote console can re-use the existing session and just use the *Activate Payload* command to activate the new payload type.

BMCs may have limited resources for handling multiple sessions. It is highly recommended that a remote console avoids creating multiple sessions and shares sessions for multiple payloads whenever possible.

Table 3-2, Activate Payload Command

	byte	data field
Request Data	1	[7:6] - reserved [5:0] - payload type (See *Table 2-17, Payload Type Numbers*)
	2	Payload Instance [7:4] - reserved [3:0] - payload instance. 1-based. 0h = reserved.
	3:6	Auxiliary Request Data. Additional payload-specific parameters to configure behavior of the payload when it becomes activated. Ignored if no auxiliary data is specified for given payload type. For Payload Type = SOL: byte 1 [7] - Encryption Activation 1b: Activate payload with encryption. All SOL payload data from the BMC will be encrypted, if encryption was negotiated at the time of session activation. 0b: Activate payload without encryption. BMC will send all SOL payload data unencrypted, if that option is allowed. (An SOL configuration parameter allows a system to be configured to require encryption for all SOL transfers) [6] - Authentication Activation [5:4] - reserved [3:2] - Shared Serial Alert Behavior The following settings are determine what happens to serial alerts if IPMI over Serial and SOL are sharing the same baseboard serial controller. 11b: Reserved 10b: Serial/modem alerts succeed while SOL active. 01b: Serial/modem alerts deferred while SOL active. 00b: Serial/modem alerts fail while SOL active. [1] - Reserved [0] - SOL Activation Request The BMC asserts RTS and DCD/DSR to the baseboard serial controller and starts accepting characters as soon as SOL is activated. 1b: Request Activate SOL (no-op if SOL already activated) 0b: Deactivate SOL (no-op if SOL already deactivated) byte 2:4 reserved - write a 00h
Response Data	1	Completion Code Generic plus the following command-specific completion codes: 80h: Payload already active on another session (required). This will be returned any time an attempt is made to activate a payload type when that type is already activated for another session, and when the BMC only supports one instance of that payload type running at a time. 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation. 82h: Payload activation limit reached. Cannot activate given payload type because the maximum number of simultaneous instances of that payload type are already running. 83h: Cannot activate payload with encryption. 84h: Cannot activate payload without encryption. BMC requires encryption for all payloads for given privilege level.
	2:5	Auxiliary Response Data. For Payload = SOL: reserved. Return as 00_00_00_00h.
	6:7	Inbound Payload Size Maximum size of payload data field from remote console to BMC. Excludes size of confidentiality header and trailer fields, if any. 1-based.
	8:9	Outbound Payload Size Maximum size of payload data field from BMC to remote console. Excludes size of confidentiality header and trailer fields, if any. 1-based.

10:11	Payload UDP Port Number UDP port number that payload can be transferred over. If the port number is same as the port that was used to establish the IPMI session, then SOL payload transfers are now available under that IPMI session on that port. Otherwise, the remote console will need to establish a separate IPMI Session to the specified Port Number using the same IP Address, username and password/key information that was used to establish the IPMI session. SOL Ppayload transfers will then be available over that session. If the remote console already has an IPMI session established on that port for a different payload type, the SOL payload type will now also be available over that session - provided that the session was established at a privilege level that matches the privilege level and authentication required for SOL. Otherwise, the remote console will need to close that session and re-establish it at the necessary privilege level.
12:13	Payload VLAN Number - FFFFh if VLAN addressing is not used.

3.2 Deactivate Payload Command

This command is used to terminate use of a given payload on an IPMI session. This type of traffic then becomes freed for activation by another session, or for possible re-activation under the present session. The *Deactivate Payload* command does not cause the session to be terminated. The *Close Session* command should be used for that purpose. A remote console application does not need to explicitly deactivate payload(s) prior to terminating a session. When a session terminates all payloads that were active under that session are automatically deactivated by the BMC.

Table 3-3, Deactivate Payload Command

	byte	data field
Request Data	1	[7:6] - reserved [5:0] - payload type (See *Table 2-17, Payload Type Numbers*)
	2	Payload Instance [7:4] - reserved [3:0] - payload instance. 1-based. 0h = reserved.
	3:6	Payload Auxiliary Data. Additional parameters to configure behavior of the payload when it becomes deactivated. Ignored if no auxiliary data is specified for given payload type. For Payload Type = SOL: (no auxiliary data) write as 0000_0000h
Response Data	1	Completion Code Generic plus the following command-specific completion codes: 80h: Payload already deactivated. 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation.

3.3 Suspend/Resume Payload Encryption Command

This command enables a remote console to control whether payload data from the BMC is sent encrypted or not. Since encryption can be a significant burden on software, this command provides a mechanism to allow higher performance by operating without encryption and only activating encryption when it is required for data confidentiality.

The extent at which this command can control encryption of data from the BMC is dependent on the payload definition. Some payload definitions may use a mix of encrypted and unencrypted payload data transfers. For example, a payload may implement a 'request/response' protocol, where the BMC would return an encrypted or unencrypted response based on whether the request from the remote console was encrypted or unencrypted. In this case, the command may only affect data that is autonomously generated by the BMC. Other payload definitions may just use whatever encryption the session was activated with, and offer no 'run-time' control of encryption/decryption, while other payload definitions may be 'stream based' where it is desirable for the remote console to be able to select when payload data is from the BMC is encrypted or not.

Table 3-4, Payload-specific Encryption Behavior

Payload Type = IPMI Messaging
• Encrypted requests from the remote console will get encrypted responses from the BMC. • The *Suspend/Resume Payload Encryption* command controls whether asynchronous (unrequested) messages from the BMC are encrypted or not. • PET Traps (which are actually separate from IPMI Messaging) are always sent unencrypted.
Payload Type = SOL
• The SOL configuration parameters allow configuring the system to require that SOL data be encrypted. • The BMC will transmit SOL payload data according to encryption settings that were selected when the payload was activated unless over-ridden by SOL configuration parameters. • The *Suspend/Resume Payload Encryption* command controls whether SOL Payload data is encrypted or not.

Table 3-5, Suspend/Resume Payload Encryption Command

	byte	data field
Request Data	1	[7:6] - reserved [5:0] - payload type (See *Table 2-17, Payload Type Numbers*)
	2	Payload Instance [7:4] - reserved [3:0] - payload instance. 1-based. 0h = reserved.
	3	[7:2] - reserved [4:0] - Operation 1h = Resume/Start encryption on all transfers of specified payload data from the BMC 0h = Suspend encryption on all transfers of specified payload messages from the BMC
Response Data	1	Completion Code Generic plus the following command-specific completion codes: 80h: Operation not supported for given payload type. 81h: Operation not allowed under present configuration for given payload type. 82h: Encryption is not available for session that payload type is active under. 83h: The payload instance is not presently active.

3.4 Get Payload Activation Status Command

This command returns what and how many instances of a given payload type are presently activated, and how many total instances can be activated.

Table 3-6, Get Payload Activation Status Command

	byte	data field
Request Data	1	Payload Type Number - number of the payload type to retrieve status for. See Table 2-17, Payload Type Numbers.
	2:4	OEM IANA (used to look up information by IANA and OEM Payload ID when Payload Type Number = "OEM Explicit", 00_00_00h otherwise.)
	5:6	OEM Payload ID (used to look up information by IANA and OEM Payload ID when Payload Type Number = "OEM Explicit" 0000h otherwise.)
Response Data	1	Completion Code
	2	Instance capacity [7:4] - reserved. [3:0] - Number of instances of given payload type that can be simultaneously activated on BMC. 1-based. 0h = reserved.
	3	[7] - 1b = instance 8 is activated. 0b = instance 8 is deactivated. [6] - 1b = instance 7 is activated. 0b = instance 7 is deactivated. … [0] - 1b = instance 1 is activated. 0b = instance 1 is deactivated.
	4	[7] - 1b = instance 16 is activated. 0b = instance 16 is deactivated. [6] - 1b = instance 15 is activated. 0b = instance 15 is deactivated. … [0] - 1b = instance 9 is activated. 0b = instance 9 is deactivated.

3.5 Get Payload Instance Info Command

This command returns information about a specific instance of a payload type. It is primarily used by software that may want to negotiate with an application that is presently using the given payload type. It accomplishes this by using the Session ID returned from this command with the *Get Session Info* command to look up the addressing information for the party that activated the payload. The application may then use that information to establish a direct dialog with the application that presently 'owns' the payload (note that this inter-application communication is not defined in the IPMI specifications).

Table 3-7, Get Payload Instance Info Command

	byte	data field
Request Data	1	Payload Type Number - number of the payload type to retrieve status for.
	2:4	OEM IANA (used to look up information by IANA and OEM Payload ID when Payload Type Number = "OEM Explicit", 00_00_00h otherwise.)
	5:6	OEM Payload ID (used to look up information by IANA and OEM Payload ID when Payload Type Number = "OEM Explicit" 0000h otherwise.)
	7	Payload Instance. 1-based. 0h = reserved.
Response Data	1	Completion Code
	2:5	Session ID - ID of session that instance is presently activated on. 00_00_00_00h if given instance is not activated. Remote software can use this information with the *Session Info* command to
	4:11	Payload-specific information (8-bytes) For Payload Type = SOL: Byte 1: Port Number A number representing the system serial port that is being redirected. 1-based. 0h = unspecified. Used when more than one port can be redirected on a system. Byte 2:8 = reserved.

3.6 Set User Payload Access Command

IPMI v2.0 introduces the *Set User Payload Access* and *Get User Payload Access* commands. These commands can be thought of as extensions to the *Set User Access* and *Get User Access* commands, respectively. The *Set User Payload Access* command controls whether the specified user has the ability to activate the specified payload type on the given channel.

The *Set User Payload Access* command uses bitfields to allow a configuration utility to use a single command to set enable/disable multiple payloads at a time. Standard payloads are set separately from OEM payload enables. The command would be issued at least once with Standard payloads selected to set the configuration for Standard payloads, and then at least once with OEM Payloads selected to set the configuration for OEM payloads.

Table 3-8, Set User Payload Access Command

	byte	data field
Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
	2	User ID Standard/OEM Select [7] - 1b = set OEM Payload Enables 0b = set Standard Payload Enables [6] - reserved. User ID [5:0] - User ID. 000000b = reserved.
	3	Payload enables 1 For Standard/OEM Select = 0b (standard): [7:6] - reserved for standard payload enable 3 [5:4] - reserved for standard payload enable 2 [3:2] - standard payload enable 1 (SOL Payload enable) Controls whether the user can activate SOL Payloads. [1:0] - standard payload enable 0 (IPMI v2.0 messaging payload enable) Controls whether the user can use IPMI messages beyond those required for supporting any other enabled payloads. 00b = don't change present payload enable setting 01b = enable user to activate payload on this channel 10b = reserved 11b = disable user ability to activate payload on this channel For Standard/OEM Select = 1b (OEM): [7:6] - OEM 3 payload enable 00b = don't change present payload enable setting 01b = enable user to activate payload on this channel 10b = reserved 11b = disable user ability to activate payload on this channel [5:4] - OEM 2 payload enable [3:2] - OEM 1 payload enable [1:0] - OEM 0 payload enable
	4	Payload Enables 2 For Standard/OEM Select = 0b (standard): [7:6] - reserved for standard payload enable 7 [5:4] - reserved for standard payload enable 6 [3:2] - reserved for standard payload enable 5 [1:0] - reserved for standard payload enable 4 For Standard/OEM Select = 1b (OEM): [7:6] - OEM 7 payload enable [5:4] - OEM 6 payload enable [3:2] - OEM 5 payload enable [1:0] - OEM 4 payload enable
	5	Payload Enables 3 reserved - write as 00h.
	6	Payload Enables 4 reserved - write as 00h.
Response Data	1	Completion Code. Note: an implementation will not return an error completion code if the user access level is set higher than the privilege limit for a given channel. If it is desired to bring attention to this condition, it is up to software to check the channel privilege limits set using the *Set Channel Access* command and provide notification of any mismatch.

3.7 Get User Payload Access Command

The *Get User Payload Access* command returns the user payload enable settings that were set using the *Set User Payload Access* command.

Table 3-9, Get User Payload Access Command

	byte	data field
Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
	2	User ID Standard/OEM Select [7] - 1b = get OEM Payload Enables 0b = get Standard Payload Enables [6] - reserved. User ID [5:0] - User ID. 000000b = reserved.
Response Data	1	Completion Code
	2	Payload enables 1 For Standard/OEM Select = 0b (standard): [7:6] - reserved for standard payload enable 3 [5:4] - reserved for standard payload enable 2 [3:2] - standard payload enable 1 (SOL Payload enable) Controls whether the user can activate SOL Payloads. [1:0] - standard payload enable 0 (IPMI v2.0 messaging payload enable) Controls whether the user can use IPMI messages beyond those required for supporting any other enabled payloads. 00b = don't change present payload enable setting 01b = enable user to activate payload on this channel 10b = reserved 11b = disable user ability to activate payload on this channel For Standard/OEM Select = 1b (OEM): [7:6] - OEM 3 payload enable 00b = don't change present payload enable setting 01b = enable user to activate payload on this channel 10b = reserved 11b = disable user ability to activate payload on this channel [5:4] - OEM 2 payload enable [3:2] - OEM 1 payload enable [1:0] - OEM 0 payload enable
	3	Payload Enables 2 For Standard/OEM Select = 0b (standard): reserved - write as 00h. For Standard/OEM Select = 1b (OEM): [7:6] - OEM 7 payload enable [5:4] - OEM 6 payload enable [3:2] - OEM 5 payload enable [1:0] - OEM 4 payload enable
	4	Payload Enables 3 reserved.
	5	Payload Enables 4 reserved.

3.8 Get Channel Payload Support Command

This command enables local and remote console software to determine what payloads are enabled on the given BMC. The command returns a bitfield indicating which Payload Type numbers can be activated on the given channel.

Table 3-10, Get Channel Payload Support Command

	byte	data field
Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
Response Data	1	Completion Code
	2	[7] = Standard payload type #7 supported … [0] = Standard payload type #0 supported
	3	[7] = Standard payload type #15 (0Fh) supported … [0] = Standard payload type #8 supported
	4	[7] = Session Setup payload type #7 supported … [0] = Session Setup payload type #0 supported
	5	[7] = Session Setup payload type #15 (0Fh) supported … [0] = Session Setup payload type #8 supported
	6	[7] = Payload type 27h (OEM7) used … [0] = Payload type 20h (OEM0) used
	7	[7] = Payload type 2Fh (OEM15) used … [0] = Payload type 28h (OEM8) used
	8:9	reserved. Return as 0000h

3.9 Get Channel Payload Version Command

This command returns version information for the given payload type. The version number has major and minor parts. The major part of the version should only increment when there are significant changes to the payload format, commands, or payload-specific protocols that break backward compatibility with earlier versions. The minor part of the version increments when there are extensions to the payload format that are significant but are backwards compatible with earlier versions under the same major version number. An example of a major change would be a change to the payload activation process that would prevent earlier applications from activating the given payload type. An example of a minor format version change would be the definition of commands for new functions that did not exist under the previous format, but if unused, do not interfere with the operation of older applications.

Table 3-11, Get Channel Payload Version Command

Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
	2	Payload Type Number. (See *Table 2-17, Payload Type Numbers*)
	3:5	OEM IANA. When Payload Type Number is 02h (OEM Explicit) this field holds the OEM IANA for the OEM payload type to return version information for. Otherwise, this field is set to 00_00_00h.
	6:7	OEM Payload ID. When Payload Type Number is 02h (OEM Explicit) this field holds the OEM Payload ID for the OEM payload type to return version information for. Otherwise, this field is set to 0000h.
Response Data	1	Completion Code
	2	Format Version [7:4] - Major Format Version. BCD encoded (0 to 9) [3:0] - Minor Format Version. BCD encoded. (0 to 9) Software should present version data to the user in the format "major.minor" - e.g. 10h → "1.0"

3.10 Get Channel OEM Payload Info Command

This command provides a mechanism for software to determine the OEM Payload Type Number that corresponds to a particular type of OEM Payload, or vice versa. The command also returns the format version of the payload.

Table 3-12, Get Channel OEM Payload Info Command

Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
	2	Payload Type Number. (See *Table 2-17, Payload Type Numbers*). Use "OEM Explicit" to look up information by OEM IANA/OEM Payload ID.
	3:5	OEM IANA. When Payload Type Number is 02h (OEM Explicit) this field holds the OEM IANA for the OEM payload type to look up information for. Otherwise, this field is set to 00_00_00h.
	6:7	OEM Payload ID. When Payload Type Number is 02h (OEM Explicit) this field holds the OEM Payload ID for the OEM payload type to look up information for. Otherwise, this field is set to 0000h.
Response Data	1	Completion Code 80h = OEM Payload IANA and/or Payload ID not supported.
	2	Payload Type Number. (See *Table 2-17, Payload Type Numbers*) This is always returned as the OEM Payload Type number (OEM0-OEM7). "OEM Explicit" is *not* returned for this parameter.
	3:5	OEM IANA. IANA for the OEM that has defined the OEM payload type.
	6:7	OEM Payload ID. Payload ID value, specified by the OEM the defined the payload type.
	8	Format Version [7:4] - Major Format Version. BCD encoded (0 to 9) [3:0] - Minor Format Version. BCD encoded. (0 to 9) Software should present version data to the user in the format "major.minor" - e.g. 10h → "1.0"

3.11 Set User Password Command

This command is used to set and change user passwords and to enable and disable User IDs. If no password protection is desired for a given user, the password must be stored as an ASCII null-string. The management controller firmware will force the remaining fifteen bytes to 00h and store the password as sixteen bytes of 00h.

If this command is not supported, then the implication is that only User 1 with a fixed, null password is supported.

The password is stored as a 16-byte or 20-byte (for IPMI v2.0) ‘octet string’. All values (0-255) are allowed for every byte. The management controller does not check the format or interpret values that are passed in with the Set User Password command.

Software is allowed to place additional restrictions on what passwords can be entered, in which case it is the responsibility of configuration software and console software to stay in synch with that definition. For example, remote console software could restrict passwords to the printable ASCII character set in order to simplify direct keyboard entry.

If this is done, any companion configuration utility should ensure that the user does not configure the managed system with non-printable passwords. Otherwise, it would be possible for the management controller to be configured with passwords that could not be entered via the remote console utility.

Table 3-13, Set User Password Command

	byte	data field
Request Data	1	User ID. For IPMI v2.0, the BMC shall support 20-byte passwords for all supported user IDs that have configurable passwords. The BMC shall maintain an internal tag that indicates whether the password was set as a 16-byte or as a 20-byte password. A 16-byte password can be used in algorithms that call for a 20-byte password. In this case, the 16-byte password is padded with 0's to 20-bytes. The 'test password' operation shall return the 'test failed' error completion code if an attempt is made to test a password that was stored as a 20-byte password as a 16-byte password (per 'password size' bit 7), and vice versa. [7] - password size 1b = set 20-byte user password/key. 0b = set 16-bytes user password/key (IPMI v1.5 backward compatible) [6] - reserved. [5:0] - User ID. 000000b = reserved. (User ID 1 is permanently associated with User 1, the null user name).
	2	[7:2] - reserved [1:0] - operation 00b = disable user 01b = enable user 10b = set password 11b = test password
		For password size = 16 bytes:
	3:18	16-byte Password data. This is a required, fixed length field when used for the set- and test password operations. If the password is entered as an ASCII string, it must be null (00h) terminated and 00h padded if the string is shorter than 16 bytes. This field need not be present if the operation is 'disable user' or 'enable user'. If this field is present for those operations, the BMC will ignore the data.
		For password size = 20 bytes:
	3:22	20-byte Password data. This is a required, fixed length field when used for the set- and test password operations. If the password is entered as an ASCII string, it must be null (00h) terminated and 00h padded if the string is shorter than 16 bytes. This field need not be present if the operation is 'disable user' or 'enable user'. If this field is present for those operations, the BMC will ignore the data.
Response Data	1	Completion Code. Generic, plus following command-specific completion codes: 80h = password test failed (mandatory)

3.12 Set Channel Security Keys Command

The *Set Channel Security Keys* command provides a standardized interface for initializing system unique keys that are used for the pseudo-random number generator key (K_R) and the key-generation key (K_G) used for RMCP+. This is mainly to provide a common interface for BMCs that are not pre-configured with these values.

The command includes a mechanism that allows specified keys to be 'locked'. Once locked, the key value cannot be read back or rewritten via standard IPMI commands. It is possible, however, that a firmware update or re-installation procedure may cause the keys

to be cleared or unlocked. Software utilities responsible for BMC initial installation and setup should check to see whether keys have been locked and if not, should initialize them appropriately and lock them.

If this command is not supported, it indicates that the keys are either permanently pre-configured, or that they are only configurable via an OEM/BMC-specific mechanism.

Table 3-14, Set Channel Security Keys Command

	byte	data field
Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number
	2	Operation [7:2] - reserved [1:0] - Operation 00b = read key BMC returns value of specified key, provided key has not yet been locked. Some BMCs may allow the key to be re-written if it does not match the expected value. Other BMCs may only allow one 'set' operation. If the key value has not yet been initialized, the BMC will return 0's for the key value. Utility software responsible for BMC installation and initial setup can use this Operation to also check to see whether keys have been initialized and locked. 01b = set key BMC writes given key value to non-volatile storage. 10b = lock key BMC locks out modification or reading the key value. Once a key has been locked, it is not cannot be rewritten or read via IPMI specified commands. 11b = reserved
	3	Key ID [7:0] - key ID. 00h = RMCP+ "KR" key 01h = RMCP+ "KG" key all other = reserved
	(4:M)	Key value. Value for specified key. Used for "set" Operation only. Otherwise, this field is not used in the request. The BMC will ignore any bytes following the 'Key ID' byte.
Response Data	1	Completion Code. Generic, plus following command-specific completion codes: 80h = Cannot perform set / confirm. Key is locked (mandatory) 81h = insufficient key bytes 82h = too many key bytes 83h = key value does not meet criteria for specified type of key
	(2:N)	Key value. The BMC returns data the specified key value when the Operation is set to "confirm key". Otherwise, the BMC returns no additional bytes past the completion code.

4. Serial Over LAN

Serial over LAN (SOL) is the specification of packet formats and protocols for transmitting serial data over LAN using IPMI over LAN packets. The typical goal of this

capability is to redirect the traffic to/from a local asynchronous serial controller interface. This enables communication over LAN with local software that only understands how to communicate through a local serial controller. This can be used for implementing a virtual remote serial terminal for enabling the user or remote software interaction with serial-based interfaces for operating systems (e.g. “command-line” interfaces and Linux console) and management services (e.g. Microsoft’s serial-based Emergency Management Services (EMS)).

4.1 SOL and Serial Port Sharing

Serial-over-LAN has been designed to be able to work alongside the Serial Port Sharing capability of IPMI. This supports an implementation where traffic to/from the serial controller interface on the baseboard can be routed either to the BMC, SOL, or to the system’s serial port connector. To accomplish this, the specifications define commands to control the routing of the serial stream using the serial multiplexer logic, also referred to as the ‘mux’. *Table 4-1, Mux Settings* describes the basic connections that support Serial-over-LAN. When Serial Port Sharing is used with SOL, the mux has three settings:

Table 4-1, Mux Settings

Setting	Description
System	The mux connects the serial connector on the chassis to the system serial controller. The RxD line is routed so the BMC can snoop incoming serial traffic for escape sequences and character patterns.
BMC	The mux connects the serial connector on the chassis to the BMC. The mux circuitry also places the hardware handshake lines to the baseboard serial controller into a steady state.
SOL	The mux is set to connect the baseboard serial controller directly to the BMC.

Note that there is no requirement that SOL is used with serial port sharing. SOL can be implemented with a dedicated serial controller interface where the interface only provides traffic for SOL connections. Serial Port Sharing can be implemented on a separate port if desired, or not at all.

The IPMI specifications do not specify the hardware design and implementation of SOL or Serial Port Sharing. *Figure 4-1, SOL with Serial Port Sharing*, presents an example block diagram for the purposes of illustrating the concept of SOL when used with Serial Port Sharing.

The example figure shows the signal routing when the mux is set to ‘SOL’. The bold lines represent the flow of data. The interconnections and blocks shown are to illustrate the functional relationships between the system management elements, and do not map directly to the exact circuit implementation of the architecture.

When the mux is set to ‘SOL’, the serial connection at the back of the box is isolated from both the baseboard serial controller and the BMC and the serial connector cannot be used for communicating with the BMC or the system. However, when SOL is not in use, the BMC can allow the serial connector to be used for functions such as the communicating with the BMC via IPMI over Serial, or as a regular serial port.

Refer to *Section 4.10, SOL Interaction with Windows.NET Escape Sequences*, and Section *4.11, SOL Payload Activated with Serial Port Sharing* for additional information on the interaction between SOL and Serial Port Sharing.

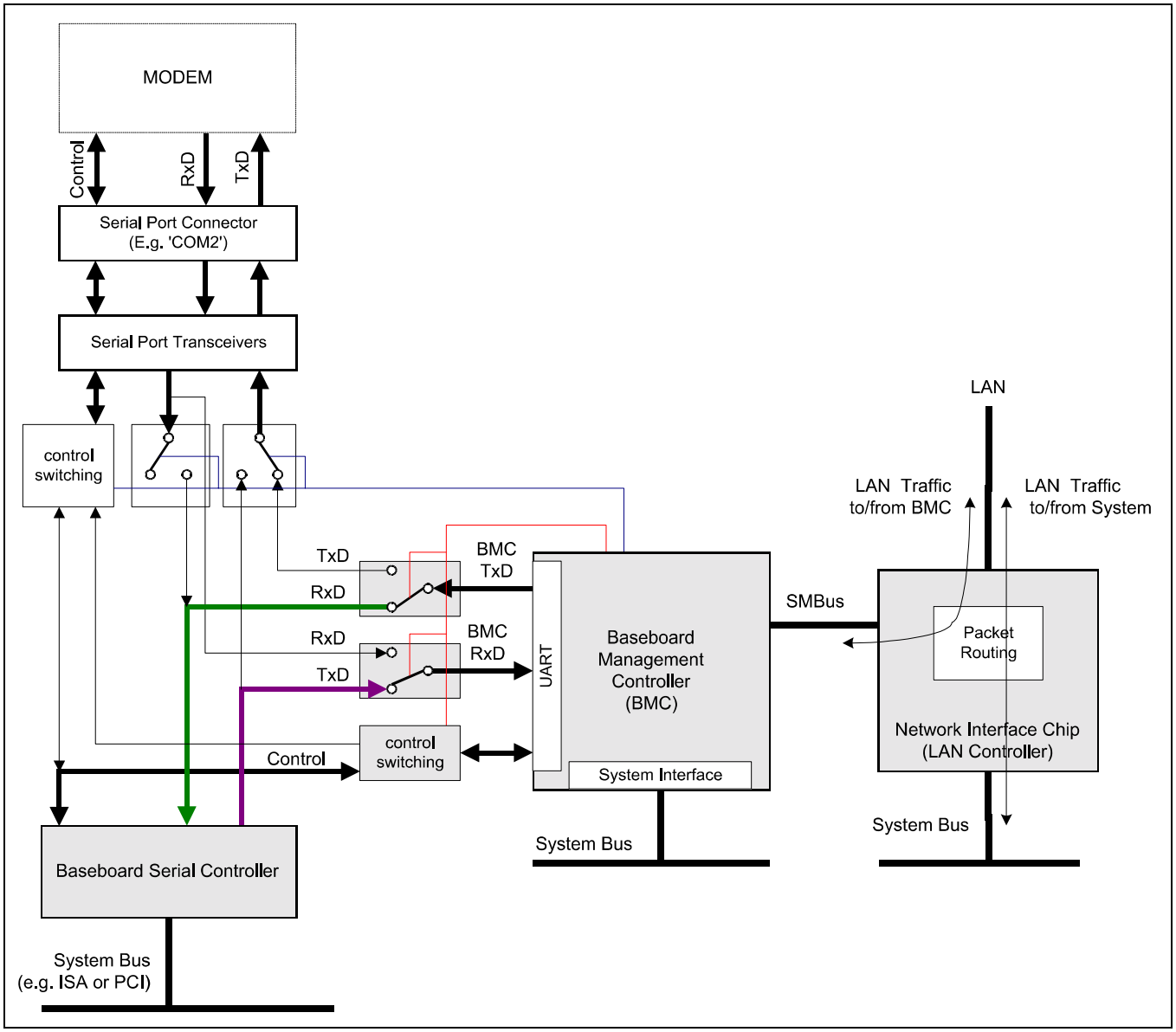


Figure 4-1, SOL with Serial Port Sharing

4.2 SOL Operation Overview

SOL operation is conceptually straightforward. A remote management application can establish an IPMI-over-LAN session with the BMC. Once the session is established, the remote console can request that the SOL be activated. If SOL is used with Serial Port Sharing, this causes the BMC to set the mux to 'SOL'.

From this point, any outgoing characters from the baseboard serial controller are assembled into packets by the BMC and sent to the remote console over the LAN. Conversely, in-bound LAN packets carrying characters for the system serial controller have their character data extracted by the BMC and delivered to the baseboard serial controller.

The SOL character data is carried as SOL Messages that are carried in UDP datagrams. The packet format is that for IPMI v2.0 RMCP+ with the Payload Type set to "SOL". The SOL Payload includes fields that are used for supporting acknowledge and retries of SOL messages, and for supporting functions such as flushing buffers or temporarily suspending serial traffic using flow control.

4.3 SOL Security

Authentication and encryption are handled at the RMCP+ level.

4.4 SOL Sequence Numbers

At the session level, SOL Payloads share the session sequence numbers for authenticated and unauthenticated packets with other packets under the IPMI session. At the payload level, SOL packets include their own message sequence numbers that are used for tracking missing and retried SOL messages.

4.5 Flow Control

Flow Control is used to help ensure that serial data is not lost because of differences in the throughput between the serial controller interface and the network interface. When SOL is used with a physical serial controller, flow control on the serial controller side is accomplished by the BMC controlling and monitoring the hardware handshake signal lines (RTS, CTS, DCD/DSR, DTR) on the serial controller.

Some implementations may have the serial controller function integrated into the BMC or another device. Such implementations may not have physical hardware flow control lines, but there must be internal control capabilities that accomplish the same thing.

Flow control on the network side is handled by use of acknowledges (ACKs) and negative-acknowledges (NACKs) that indicate whether the BMC is ready to accept more data or not.

4.6 Bit Rate Handling

Some implementations of SOL will connect pins for a serial interface on the BMC to the pins for the serial interface of a serial controller for the system. For standard microcontroller serial interfaces, the BMC would need to know the bit rate setting of the motherboard serial controller in order for the BMC's serial controller to be in synch. Thus, there is a non-volatile configuration setting for setting the bit rate for SOL in the BMC.

Other implementations may incorporate an embedded serial controller in the BMC or may have hardware mechanisms that allow the BMC to get the bit rate setting directly from the system serial controller. In this case, the non-volatile configuration bit rate setting is not used.

4.7 SOL Payload Data Format

Table 4-2, SOL Payload Data Format, specifies the fields that make up the SOL Payload in an RMCP+ packet.

Table 4-2, SOL Payload Data Format

Field	Size	Description	
Packet Sequence Number	1	Sequence Number for this packet. Sequence numbers must be non-zero. Multiple outstanding sequence numbers are not supported in this version of the specification. Retried packets use the same sequence number as the first packet. [7:4] Reserved [3:0] Packet sequence number. 0h = ACK-Only packet.	
Packet ACK /NACK Sequence Number	1	Sequence Number for packet being ACK'd or NACK'd. [7:4] Reserved. Write as 0h. (Future spec may use this to specify a range of packets being acknowledged) [3:0]: Packet sequence number being ACK'd/NACK'd. 0h: Informational packet. No request packet being ACK'd or NACK'd.	
Accepted Character count	1	Accepted Character Count 1-based. This field indicates the number of characters accepted from the packet, if any. 00h = Packet received but no data bytes accepted. **For BMC-to-Console:** In order to improve throughput, the BMC is allowed to append additional characters to a packet when it resends it. To support this, the remote console must accept retried packets (packets with the same packet sequence number) and check to see if the packet contains additional data. If the packet does contain addition data, the remote console should accept the data and acknowledge the packet using the packet sequence number and return the count of the number of characters that it had received. *The console must either accept all packet data sent to it or NACK the entire packet. It is not allowed to accept partial packet data.* **For Console-to-BMC:** The BMC is allowed to accept partial packet data by NACK'ing the packet and returning a character offset that is less than the data length sent by the remote console. The remote console would then send the next packet starting with the first byte of data that the BMC rejected.	
Operation / Status	1	**BMC to Remote Console:**	**Remote Console to BMC:**
		Operations are executed *before* character data is transferred. [7] reserved [6]	Note: Operations are executed *before* character data is transferred. [7] reserved [6] ACK/NACK

Field	Size	Description	
		1b: Packet is being NACK'd. The BMC is unable to accept all character data from packet. Note: Operation field is still accepted even if packet is NACK'd. 0b: ACK. BMC ready to accept next packet of character data. [5][1] A NACK packet with this status will be automatically sent one time after this bit changes state. (Whenever the system enters or leaves a power state where character transfers to the system serial controller are possible) A NACK packet with "Character transfer is unavailable" status will also be sent for each character transfer request from the remote console when the system is in a powered-down or sleep state. 1b: Character transfer is unavailable because system is in a powered-down or sleep state. 0b: SOL character transfer is available. [4][2] A NACK packet with this status will be automatically sent one sent once, just before the BMC deactivates SOL because of a front panel power-button or a reset. 1b: SOL is deactivated/deactivating. [Remote console can use this to tell if SOL was deactivated by some other party, or by local pushbutton reset or power on/off]. 0b: SOL is active. [3] Transmit Overrun 1b: characters were dropped between transmitting this packet and the previous packet, because the system did not pay attention to hardware flow control. 0b: no characters were lost between this packet and the preceding packet. [2] Break 1b: A break condition from the system has been detected. The BMC will generate this	1b: NACK. Packet is being NACK'd by the remote console. 0b: ACK. Packet is being ACK'd by the remote console. [5] Ring/WOR Assert RI (may not be supported on all implementations) - Goal is to allow this to be used for generating a WOR. [4] Break 1b: Generate BREAK (300 ms, nominal) [3] CTS Pause 1b: Deassert CTS (clear to send) to the baseboard serial controller. This is the default state when SOL is deactivated. 0b: Let BMC control CTS. [2] Drop DCD/DSR 1b: Deassert DCD/DSR to baseboard serial controller0b: Assert DCD/DSR to baseboard serial controller. [1] Flush Inbound 1b: Drop (flush) data from remote console to BMC [not including data carried in this packet, if any] [0] Flush Outbound 1b: Flush Outbound Character Data (flush data from BMC to remote console)

Field	Size	Description
		only on one packet at the start of the break. 0b: no break detected [1:0] Reserved
Character Data	varies	Data length, in bytes, is equal to the IPMI Message/SOL Message Length field in the Session Header, minus the bytes for the Packet Sequence Number, through and including the Operation Field.

1. If the system is powered down or in certain sleep states, the baseboard serial controller will not be available for transferring characters. When the BMC receives data from the remote console, and the system is powered down or sleeping, the BMC can use this bit to return status to indicate to the remote console why the characters it received may not be able to be transferred to the system. Note that this is an 'advisory' bit. The BMC will still attempt to put characters into the system serial controller in case that the serial controller is configured to wake the system under this condition. It is mandatory that the BMC returns this bit when the system is powered down (S4/S5). The BMC may not be able to differentiate other system sleep states, in which case the bit should be returned as 0b (transfers available).

2. The BMC issues this status when the payload is deactivated for due to the following conditions: a. The payload is deactivated because of a manual power down or reset. (An implementation is recommended to have a local manual "pushbutton" reset or power-off to deactivate an SOL payload. This is provided as a way of terminating remote control connections for local servicing.) b. The SOL payload is deactivated via the *Deactivate Payload* or *Close Session* commands. These command may have been issued from another session. (System software operating through the system interface, and users with Admin privilege have the ability to issue the *Deactivate Payload* and *Close Session* commands to other sessions).

4.8 Activating SOL using RMCP+ Authentication

To use SOL a remote console or remote application must first establish an IPMI Session with the BMC. This is accomplished by sending the specified IPMI and RMCP+ RSSP/RAKP messages to the BMC with the appropriate user name, role, and password/key information. If the remote console plans to use encryption with SOL, the console must also negotiate an encryption algorithm at the time that the IPMI session is established.

Once the IPMI session has been established, the *Get Channel Payload Support* command can be used to retrieve the present availability of SOL and the version of SOL.

If SOL is not already active on another session, the next step is to issue the *Activate Payload* command. The command will return information about what serial input and output data buffer sizes are available on the BMC as well as the UDP port number over which SOL packets can be transferred.

The port that was used for establishing the IPMI session may not be the same port number that SOL is available over. Some implementations transfer SOL payloads available over a separate UDP port in order to provide better performance. If the *Activate Payload* command returns a port number that is different than the port number that was used to establish the IPMI session, the remote console must establish a separate IPMI Session to the specified port number using the same IP Address, username and password/key information that was used to establish the IPMI session.

Note that if a second session has already been established on that port number for a different payload type, the existing session can also be used for SOL payloads, provided

that the session was established at a privilege level that matches the privilege level and authentication required for SOL. Otherwise, the remote console will need to close that session and re-establish it at the necessary privilege level.

4.9 SOL Packet Acknowledge and Retries

A packet acknowledge is of one of two types:

- An ACK, indicating that the packet has been received and all its data has been accepted
- A NACK, indicating that the packet was received but some or all of the data could not be accepted

To improve efficiency, the packet acknowledgment information can be carried in a packet that also carries the SOL character data. Conversely, a packet can be an ACK-only packet that carries ACK or NACK information, but no data. However, packets are only acknowledged if they contain character data. Therefore, ACK-only packets themselves are not acknowledged.

Except for ACK-only packets, the remote console must acknowledge each SOL packet that it receives. If the BMC does not receive an ACK packet within a timeout interval, the BMC will resend (retry) the packet. The number of retries and the amount of time between retries are configurable through the SOL Configuration Parameters. **Once the number of retries has been met the BMC will drop the packet and the data will be lost.**

If the remote console wants to temporarily stop the BMC from accepting characters from the system, it should use the “CTS Pause” bit in the control/status byte. Whether this stops the system from transmitting is dependent on whether the system software pays attention to the CTS (hardware handshake) status or not. If the system continues to send characters to the BMC, the BMC will attempt to transmit them to the remote console.

It is possible that additional characters could be received from the system serial controller while the BMC was waiting for a retry timeout. In order to improve throughput, the BMC is allowed the option of appending additional characters to a packet whenever it resends it. To support this, the remote console must accept retried packets (packets with the same packet sequence number) and check to see if the packet contains additional data. If the packet does contain addition data, the remote console should accept the data and acknowledge the packet using the packet sequence number and character offset value that it had received.

Lenovo
Ex. 1023 - Page 62

Table 4-3, Remote Console to BMC SOL Packet Handling

Packet fields from remote console:			Function / BMC Action
ACK/ NACK	**ACK'd/NACK'd Seq#**	**Accepted Character Count**	
ACK	Seq # from BMC data packet	Non-zero matches character count from BMC	**"Completion ACK"** BMC processing for specified packet is done. A packet from the remote console with an ACK and an Accepted Character Count for the full amount of data for the packet indicates that the remote console has successfully accepted the packet.
ACK	Seq # from BMC data packet	Non-zero less than character count from BMC	**"Partial ACK"** BMC immediately retransmits specified packet It is possible that the remote console may have missed a BMC retry where the BMC had appended more data to the packet (retry intervals should be configured to avoid this scenario). If the BMC receives an ACK for less than the last amount of transmitted data, the BMC will cease appending data to the packet and will retransmit the packet until it receives an ACK from the remote console with an Accepted Character Count for the full amount of packet data.
NACK	Seq # from BMC data packet	0	**"Suspend NACK"** BMC stops sending specified packet. The remote console would use the Suspend NACK if it were temporarily out of buffer space for characters already queued up in the BMC and did not want those characters to get dropped. The BMC stops sending / retrying specified packet and waits until it gets either a "Partial ACK", "Completion ACK", "Resume ACK" or a Flush Outbound operation from the remote console. The BMC will deassert CTS when it gets near running out of buffer space for characters from the system. If characters continue to come in (CTS is ignored by the system) a transmit overrun condition can occur.
ACK	Seq # from BMC data packet	0	**"Resume ACK"** BMC immediately retransmits specified packet. Cancels a "Suspend NACK". A packet from the remote console with an ACK and an Accepted Character Count of zero (0) bytes will cause the BMC to immediately retransmit the packet with the corresponding sequence number to the remote console. This can be used as a way to get the BMC to restart transmission after a Suspend NACK from the remote console.
ACK	0h	0	**"Control-only Packet"** BMC performs operation specified in the control/status field. For FLUSH operation: A packet from the remote console with a "Flush Outbound" operation will cause the BMC to cease any retries in progress and the BMC will start accumulating character data anew. The remote console can use this as a recovery mechanism if it gets 'lost' in the sequence from the BMC. See packet format table for info on other functions.

4.10 SOL Interaction with Windows.NET Escape Sequences

The Microsoft .NET Emergency Management Services specification (See [MSFT EMS]) defines certain character sequences for performing the following operations:

System Hard Reset	<ESC>R<ESC>r<ESC>R
Invoke Service Processor (i.e. switch to BMC)	<ESC>(
Exit Service Processor (with optional prompt)	Q
Exit Service Processor (without confirmation prompt)	<ESC>Q

The specification also requires that a switch to the Service Processor is acknowledged by sending an <ESC>* sequence to the remote console:

Acknowledge Switch to Service Processor	<ESC>*

Typically, the input escape sequences would be received by the BMC over the serial/modem connection. In this case, the sequences for Invoking and Exiting the service processor would control the serial/modem mux setting associated with serial port sharing. However, there are separate streams for SOL and BMC traffic, so unlike serial port sharing there is no need for a mux to switch between console traffic to the system serial controller and traffic to the BMC.

Therefore, since you can always send commands to the BMC the BMC itself does not snoop for or handle Windows .NET Escape sequences in the SOL character data. However, a remote console application can emulate support for the Windows .NET headless sequences by filtering for the Windows .NET escape sequences prior to placing the data in an SOL packet. If the Windows .NET escape sequences are detected, the remote console can then take the appropriate actions. For example, if the Reset sequence is detected, the remote console would send the IPMI command for a system reset.

4.11 SOL Payload Activated with Serial Port Sharing

The following lists the behavior of certain IPMI commands while SOL is activated. This applies only when the SOL is being used in conjunction with Serial Port Sharing.

- The *Set Channel Access* command is accepted for the serial channel, but while SOL is activated the Channel Access Mode is forced to 'disabled'. Attempting to change the access mode while SOL is activated will result in a D5h completion code. The access mode shall be saved when SOL is activated, and restored when SOL is deactivated.
- The *Set Serial/Modem Configuration Parameters* and *Get Serial/Modem Configuration Parameters* commands are accepted for the serial/modem channel and have no special behavior while SOL is activated.
- The UDP proxy commands, *Set PPP UDP Proxy Transmit Data*, *Get PPP UDP Proxy Transmit Data*, *Send PPP UDP Proxy Packet*, *Get PPP UDP Proxy Receive Data*, will receive a D5h error completion code while SOL is activated.

Lenovo
Ex. 1023 - Page 64

- The *Callback* command will receive a D5h error completion code when the command is targeted to the serial/modem channel being used for SOL and SOL is activated. The *Set User Callback Options* and *Get User Callback Options* commands will be accepted.
- The *Send Message Command* will receive a D5h error completion code when the command is targeted to the serial/modem channel being used for SOL and SOL is activated.
- The *Alert Immediate Command* will receive a D5h error completion code when the command is targeted to the serial/modem channel being used for SOL and SOL is activated.
- While SOL is activated, the *Set Serial/Modem Mux* command will respond to the requested operations as follows:

Table 4-4, Set Serial/Modem Mux Command Operation while SOL Active

Operation	Response
0h = get present mux setting/status only	Accepted. BMC returns that mux is set to 'System'
1h = request switch of mux to system	Accepted
2h = request switch of mux to BMC	Rejected (see response data for command)
3h = force switch of mux to system	Allowed
4h = force switch of mux to BMC	BMC returns D5h completion code
5h = block requests to switch mux to system	BMC returns D5h completion code
6h = allow requests to switch mux to system	Accepted
7h = block requests to switch mux to BMC	BMC returns D5h completion code
8h = allow requests to switch mux to BMC	Accepted

5. SOL Commands

The following commands are specific to the SOL Payload type.

Table 5-1, SOL Commands

Command	Section Defined	O/M
SOL Activating	5.1	O[1]
Set SOL Configuration Parameters	5.2	O[1]
Get SOL Configuration Parameters	5.3	O[1]

1. Mandatory if the SOL Payload type is implemented.

5.1 SOL Activating Command

This command provides a mechanism for the BMC to notify a remote application that a SOL payload is activating on another channel. The primary use for this command is in conjunction with Serial Port Sharing to notify an application that is connected to the system serial port that they have lost their connection because another application has activated SOL and the baseboard serial controller is now being used for SOL.

Table 5-2, SOL Activating Command

	byte	data field
Request Data	1	SOL Session State [7:4] - reserved [3:0] - Session State 0h = SOL Activating (Shared serial connection about to be lost because BMC is about to switch it over for SOL use)
	2	SOL Payload Instance
	3	SOL format version, major (SOL Payload Format version. See *Table 2-17, Payload Type Numbers*.
	4	SOL format version, minor (SOL Payload Format version. See *Table 2-17, Payload Type Numbers*.
Response Data	1	Completion Code The request message is a message that is asynchronously generated by the BMC. The BMC will not wait for a response from the remote console before dropping the serial connection to proceed with SOL, therefore the remote console does not need to respond to this command.

Lenovo
Ex. 1023 - Page 66

5.2 Set SOL Configuration Parameters Command

This command is used for setting parameters such as the network addressing information required for SOL payload operation.

Table 5-3, Set SOL Configuration Parameters Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel number.
	2	Parameter selector
	3:N	Configuration parameter data, per *Table 5-5, SOL Configuration Parameters*
Response Data	1	Completion Code 80h = parameter not supported. 81h = attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters) 82h = attempt to write read-only parameter

5.3 Get SOL Configuration Parameters Command

This command is used for retrieving the configuration parameters from the *Set SOL Configuration* command.

Table 5-4, Get SOL Configuration Parameters Command

	byte	data field
Request Data	1	[7] - 0b = get parameter 1b = get parameter revision only. [6:4] - reserved [3:0] - Channel number.
	2	Parameter selector
	3	Set Selector. Selects a given set of parameters under a given Parameter selector value. 00h if parameter doesn't use a Set Selector.
	4	Block Selector (00h if parameter does not require a block number)
Response Data	1	Completion Code. Generic codes, plus following command-specific completion code(s): 80h = parameter not supported.
	2	[7:0] - Parameter revision. Format: MSN = present revision. LSN = oldest revision parameter is backward compatible with. **11h** for parameters in this specification.
		The following data bytes are not returned when the 'get parameter revision only' bit is 1b.
	3:N	Configuration parameter data, per *Table 5-5, SOL Configuration Parameters* If the rollback feature is implemented, the BMC makes a copy of the existing parameters when the 'set in progress' state becomes asserted (See the Set In Progress parameter #0). While the 'set in progress' state is active, the BMC will return data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the BMC returns parameter data from non-volatile storage.

Table 5-5, SOL Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
Set In Progress (volatile)	0	data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software than some other software or utility is in the process of making changes to the data. An implementation can also elect to provide a 'rollback' feature that uses this information to decide whether to 'roll back' to the previous configuration information, or to accept the configuration change. If used, the roll back shall restore all parameters to their previous state. Otherwise, the change shall take effect when the write occurs. [7:2] - reserved [1:0] - 00b = set complete. If a system reset or transition to powered down state occurs while 'set in progress' is active, the BMC will go to the 'set complete' state. If rollback is implemented, going directly to 'set complete' without first doing a 'commit write' will cause any pending write data to be discarded. 01b = set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The BMC does not provide any interlock mechanism that would prevent other software from writing parameter data while. 10b = commit write (optional). This is only used if a rollback is implemented. The BMC will save the data that has been written since the last time the 'set in progress' and then go to the 'set in progress' state. An error completion code will be returned if this option is not supported. 11b = reserved
SOL Enable	1	Byte 1: [7:1]: Reserved [0]: SOL Enable. Note, this controls whether the SOL payload type can be activated. Noter that whether an SOL stream can be established is also dependent on the Access Mode and Authentication settings for the corresponding LAN channel. The enabled/disabled state and access mode settings for the serial/modem channel have no effect on SOL. 1b = enable SOL payload 0b = disable SOL payload

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
SOL Authentication	2	Byte 1: SOL Authentication Enable [7]: Force SOL Payload Encryption 1b: Force encryption. If the Cipher Suite for the session supports encryption this setting will force the use of encryption for all SOL payload data. 0b: Encryption controlled by remote console. Whether SOL Packets are encrypted or not is selectable by the remote console at the time the payload is activated (using the *Activate Payload* command) and can be changed during operation via the *Suspend/Resume Payload Encryption* command. [6]: Force SOL Payload Authentication 1b: Force Authentication. If the Cipher Suite for the session supportes authentication this setting will force the use of authentication on all SOL Payload data. 0b: Authentication controlled by remote software. Note that for the standard Cipher Suites, if encryption is used authentication must also be used. Therefore, while encryption is being used software will not be able to select using unauthenticated payloads. :4]: Reserved [3:0]: SOL Privilege Level. Sets the minimum operating privilege level that is required to be able to activate SOL using the *Activate Payload* command. 0h: reserved 1h: reserved 2h: USER level 3h: OPERATOR level 4h: ADMINISTRATOR level 5h: OEM Proprietary level all other: Reserved
Character Accumulate Interval & Character Send Threshold	3	Byte 1: Character Accumulate Interval in 5 ms increments. 1-based. This sets the typical amount of time that the BMC will wait before transmitting a partial SOL character data packet. (Where a partial packet is defined as a packet that has fewer characters to transmit than the number of characters specified by the Send Threshold - see next field in this parameter). A packet will not be sent 00h = reserved Byte 2: Character Send Threshold. 1-based. The BMC will automatically send an SOL character data packet containing this number of characters as soon as this number of characters (or greater) has been accepted from the baseboard serial controller into the BMC. This provides a mechanism to tune the buffer to reduce latency to when the first characters are received after an idle interval. In the degenerate case, setting this value to a '1' would cause the BMC to send a packet as soon as the first character was received. This can be useful if the character accumulate interval is large. If the BMC is waiting for an acknowledge from the previous packet, it will ignore this threshold and continue to collect data until it has a full packet's worth.

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
SOL Retry	4	Byte 1: Retry Count [7:3] - Reserved [2:0] - Retry count. 1-based. 0 = no retries after packet is transmitted. Packet will be dropped if no ACK/NACK received by time retries expire. Byte 2: Retry Interval. 1-based. Retry Interval in 10 ms increments. Sets the time that the BMC will wait before the first retry and the time between retries when sending SOL packets to the remote console. 00h: Retries sent back-to-back
SOL non-volatile bit rate (non-volatile)	5	This configuration parameter is not supported if the implementation does not have a BMC serial controller that can be potentially configured Serial communication with the BMC when SOL is activated always occurs using 8 bits/character, no parity, 1 stop bit, and RTS/CTS (hardware) flow control. Note: If SOL is enabled for multiple LAN channels, the BMC uses the serial communication settings for the channel over which the Activate SOL command was initially received. The settings for other channels are ignored. Data 1 [7:4] - Reserved [3:0] - Bit Rate. 1-5h = reserved. Support for bit rates other than 19.2 kbps is optional. The BMC must return an error completion if a requested bit rate is not supported. It is recommended that the parameter out-of-range (C9h) code be used for this situation. 0h: Use setting presently configured for IPMI over serial channel used by BIOS serial redirection. The setting will be used even if the access mode for the serial channel is set to 'disabled'. Note: IPMI specification can allow more than one serial channel. 6h: 9600 bps 7h: 19.2 kbps 8h: 38.4 kbps 9h: 57.6 kbps Ah: 115.2 kbps all other = reserved
SOL volatile bit rate (volatile)	6	Set volatile version of SOL Serial Settings. Data follows that for the SOL non-volatile bit rate parameter.

Lenovo
Ex. 1023 - Page 70

Appendix A - Previous Sequence Number Tracking

The following illustrates how a method for tracking the last eight received sequence numbers can be used for handling out-of-sequence packet reception. The method illustration assumes that the receiver tracks the highest received sequence number that has been accepted, shown as 'Highest Received' column, and which of the previous eight sequence numbers have been received, shown as the 'Previously Received List'.

Note that for implementation, the previously received list can be implemented as a bit field where 1=received, 0=not received, and the bit positions correspond to sequence numbers for Highest Received-1, Highest Received-2, Highest Received-3, etc. The handling of wrap-around of the sequence number is not shown and is left to the reader.

1. Startup initialization. Highest Received session sequence number is set to value that was sent with the *Activate Session* command. The Previously Received List is initialized as if the preceding eight sequence numbers were received. This prevents a packet with a sequence number less than the initial value from being accepted. For this example, assume the initial sequence number is 40.

   Initial values are:

Highest Received	Previously Received List							
40	39	38	37	36	35	34	33	32
	Y	Y	Y	Y	Y	Y	Y	Y

2. Packet 41 is received. Packet is accepted because it is no more than 8 counts greater than the Highest Received value from step 1. Highest Received becomes 41. Previously received list gets 'pushed up'. Updated values are:

Highest Received	Previously Received List							
41	40	39	38	37	36	35	34	33
	Y	Y	Y	Y	Y	Y	Y	Y

3. Packet 44 received. Accepted because it is within 8 counts of the last Highest Received. Highest Received becomes 44. Previous received list gets pushed up by 2. Note that packets 43 and 42 are listed as unreceived. Updated values are:

Highest Received	Previously Received List							
44	43	42	41	40	39	38	37	36
	N	N	Y	Y	Y	Y	Y	Y

4. Packet 39 received. Dropped because it is a duplicate with previously received packet. No update to Highest Received and Previously Received List. Updated values are:

Highest Received	Previously Received List							
44	43	42	41	40	39	38	37	36
	N	N	Y	Y	Y	Y	Y	Y

5. Packet 42 received. Accepted because it is listed as unreceived on the Previously Received List. Packet 42 listed as received. No update to Highest Received because 42 is not higher than 44. Updated values are:

Highest Received	Previously Received List							
44	43	42	41	40	39	38	37	36
	N	Y	Y	Y	Y	Y	Y	Y

Appendix B - Command Assignments

The following is to be added to Table G-1 to cover the new command assignments for IPMI v2.0. (See Appendix G in IPMI v1.5 specification for key.)

Table B-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
BMC Device and Messaging Commands							
…							
Activate Payload		App	48h	X[5]			
Deactivate Payload		App	49h	X[5]			
Get Payload Activation Status		App	4Ah		X		
Get Payload Instance Info		App	4Bh		X		
Set User Payload Access		App	4Ch				
Get User Payload Access		App	4Dh		X		
Get Channel Payload Support		App	4Eh		X		
Get Channel Payload Version		App	4Fh		X		
Get Channel OEM Payload Info		App	50h		X		
unassigned		App	51h	-	-	-	-
Master Write/Read		App	52h			X	
Get Channel Authentication Capabilities		App	53h	p	p	p	p
Get Channel Cipher Suites		App	54h	p	p	p	p
Suspend/Resume Payload Encryption		App	55h		X[9]		
Set Channel Security Keys		App	56h				X
SOL Commands							
SOL Activating		Transport	20h				
Set SOL Configuration Parameters		Transport	21h				X
Get SOL Configuration Parameters		Transport	22h		X		

5. A session operating at Callback, User, or Operator can only use this command to terminate their own session. An Administrator or system software can use the command to terminate any session.
9. The Suspend/Resume command may be blocked by a configuration option for the particular payload type that forces encryption to be used. In this case, an Admin level command would typically be required to change the configuration.

Lenovo
Ex. 1023 - Page 72