iSCSI Initiator Configuration Reference
Release 1.0

Note:
Before using this information and the product it supports, read the information in “Notices”

First Edition (May 2006)

This edition includes information that specifically applies to the ISCSI Initiator Configuration.

Contents

FIGURES

TABLES

About this guide

This document defines the technical aspects of the definition and usage of the iSCSI configuration parameters as well as the methodologies for deploying these parameters to a given initiator.

Chapter 1: Technical Terms and Definitions

The following terms are used in this publication:

Terms	Definition
iSCSI	An IETF-certified protocol for transacting SCSI commands across a TCPIP network. At a high level, this protocol can be viewed as encapsulating SCSI commands and data within standard TCPIP frames. Please refer to the iSCSI protocol specification to acquire knowledge of the details.
Initiator	An iSCSI or FC initiator is responsible for issue storage requests such as reads or writes. Typically the initiator is a server or client system requesting information from an iSCSI or FC target
Target	An iSCSI or FC target is responsible for responding to storage requests such as reads or writes. Typically, the target is a storage controller containing disks that responds to storage request
IP Address	IP address is the TCP/IP level identifier of a node on the network.. There are two formats to this address: IPv4 (most prevalent today) where the address is comprised of 4 bytes presented as period demarked sequence decimal numbers such as 192.121.201.96. IPv6 where the address is comprised of 16 bytes presented as colon demarked sequence of hexadecimal numbers such as FED0:ABCD:1234:9A12:8888:EEFF:0001:0192
TCP Port	TCP port is the port number of a given node interface. The port number concept allows multiple applications to be easily segregated by port number. The format of the port number is a 4 digit number such as 3260
iSCSI IQN	iSCSI IQN is the iSCSI level identifier of a node on the network. There are two formats to this address: IQN format is a human friendly format where the 1st 4 characters present a formatting tag while the remaining characters are human friendly string identifying the node such as iqn.this_is_my_node. EUI format is a smaller 16 character hex value with a leading string such as eui.0123456790ABCDEF
CHAP ID/PW	CHAP is an authentication protocol using the ID, PW, and random number to create a one way hash. Both sides know the ID and PW while the random number and ID is transferred openly on the network. The resultant one way hash is exchanged on the open network after calculation. If the incoming hash agrees with the internal constructed hash, then the hash sender is, indeed, who he claims to be.
DHCP	DHCP is a discovery and configuration service potentially on the network. This service aids in the dynamic configuration of a given node at boot time.
SLP	SLP is another discovery and configuration service potentially on the network. This service, though more rich and complex, also aids in the configuration and discovery of services on the network.
iSNS	iSNS is another discovery and configuration service potentially on the network where a node can look up the location of a desired resource.
Out of Band	Out of band (OOB) is the paradigm where configuration information is passed to a resource outside the normal transport media. This path into a node is used to configure security settings or to configure the node in the absence of any network discovery services
In Band	In band is the paradigm where configuration information is passed to a resource using the normal transport media. This path into a node is used to configure public or non threatening settings and is used by network discovery services to configure the nod.
SCSI	SCSI is a block protocol used to access storage devices. The protocol is a command response protocol used to interact with the storage device directly, to determine status and maintain control, as well as to interact with the storage device to read and write data.
HBA	A HBA represents a storage adapter that resides in a server. HBAs interact with the server cpu to transmit and receive packets from a set of storage devices. There are iSCSI and FC HBAs
TOE	A Tcpip Offload Engine represents technology residing on a network interface controller (NIC) adapter that performs the entire network stack processing, thus relieving the server processing complex from performing all network processing.
Firmware iSCSI	Firmware iSCSI represents the concept of the iscsi protocol processing taking place on the host CPU and using standard EN NICs as the transport media.
iSCSI HBA	iSCSI HBA represents the concept of using an iSCSI HBA for protocol processing and using dediciated ports as the transport media. ISCSI HBAs also leverage TOE capabilities to enhance

Terms	Definition
	capabilities and performance
BIOS	Basic Input Output Software (BIOS) represents the software executed prior to the operating system loading. BIOS ensures the server hardware is operating correctly and the appropriate configuration is in place prior to loading the operating system's master boot record.
INT 13	The term INT13 represents the BIOS interrupt used to access storage whether it be floppy, local disk, PXE server, or iSCSI target. In essence, a storage service that complies with the INT 13 architecture will correctly "chain" itself into the INT 13 interrupt chain and assign itself the correct disk number starting with 80.
UNDI	Universal Network Driver Interface (UNDI) represents a class of low level pre-OS functions that provide network access and transport. UNDI functions are accessed by a family of calls defined in the PXE standard and provides UDP datagram transport onto the IP network.

Table 1: Key Definitions

Chapter 2. iSCSI Parameter Definition

The key to establishing an iSCSI session is the defining or discovering the network topology. This network topology defines relationship between a given initiator and a given target. These parameters can be defined in a static, user defined manner for implementations where there are limited network infrastructure services. The parameters can also be discovered from the network by several network discovery or service location services.

Since these parameters are used by the boot iSCSI service and the OS iSCSI service, they need to be provided in a consistent manner.

iSCSI Parameter Summary

The iSCSI parameters can be broken into three categories. The base parameters define the common basic parameters used to establish the connection between the initiator and target. The base options define how to interpret the base parameters. The extended iSCSI parameters define implementation specific or emerging parameters that are not universally applicable

iSCSI Base Parameter Summary

Table 2: iSCSI Base Parameter Data Structure summarizes the base parameters associated with the iSCSI parameter data structure. These parameters define the data structure format, iSCSI parameter discovery information, and basic iSCSI information necessary for establishing the iSCSI session between the initiator and target

Parameter	Name	Size (Bytes)	Definition
Signature	"ISAN"	4	FYI only, not part of data structure
	"PARM"	4	FYI only, not part of data structure
Version	VER[7:0]	1	Version of the parm data structure
Level	LVL]7:0]	1	Level of data structure within version
Data Structure Length	LEN[15:0]	2	Length in bytes of the total data
Checksum	CHK[7:0]	1	1 byte checksum of data structure total
Static/Dynamic	S_D[7:0]	1	Flavors of static and/or dynamic
Options	OPT[31:0]	4	Parameter structure options
Retry Count	RTRY[7:0]	1	Retries to attempt
Timeout Value	TOVAL[7:0]	1	Time out durations per attempt
Scope/Vendor ID	SVID[63:0]	8	Scope ID to aid in DHCP scope/usage
Client Alt ID	CAID[63:0]	8	Support Further DHCP refinement
Discovery IP address	DIP[63:0]	8	IP address of DHCP server for unicast
Initiator IP address	IIP[63:0]	8	iSCSI initiator IP address
Initiator name	IIQN[71:0][7:0]	72	iSCSI iqn tag
Initiator CHAP id	ISDI[15:0][7:0]	16	CHAP ID or 1st half of security key
Initiator CHAP pw	ISDP[23:0][7:0]	24	CHAP PW or 2nd half of security key
Subnet Mask	MSK[63:0]	8	IP network subnet mask
VLAN tag	VLAN[16:0]	2	VLAN tag for iSCSI VLAN
Gateway IP address	GRIP[63:0]	8	IP network gateway mask
1st Target IP address	T1IP[63:0]	8	iSCSI target IP address of storage
1st Target TCP port	T1PT[15:0]	2	iSCSI TCP port on target IP address
1st Target name	T1IQN[71:0][7:0]	72	iSCSI iqn tag
1st Target Boot LUN	T1BL[7:0]	1	Boot LUN residing on 1st Target - option
1st Target CHAP id	T1SDI[15:0][7:0]	16	CHAP ID or 1st half of security key
1st Target CHAP pw	T1SDP[23:0][7:0]	24	CHAP PW or 2nd half of security key
2nd Target IP address	T2IP[63:0]	8	iSCSI target IP address of storage
2nd Target TCP port	T2PT[15:0]	2	iSCSI TCP port on target IP address
2nd Target name	T2IQN[71:0][7:0]	72	iSCSI iqn tag
2nd Target Boot LUN	T2BL[7:0]	1	Boot LUN residing on 2nd target
2nd Target CHAP id	T2SDI[15:0][7:0]	16	CHAP ID or 1st half of security key
2nd Target CHAP pw	T2SDP[23:0][7:0]	24	CHAP PW or 2nd half of security key
Base Data Structure Size		400	From version to 2nd target CHAP PW
iSCSI Parameter Page Size		432	Total space per instanced in PSR
Extended parameters	EPID[7:0] EPCT[15:0] EPOPT[7:0] EPVAL[EPCT:0][7:0]		for ID+CT+OPT+values (5 byte min per)
Extended space in page		432	Extended Space in 1 BIOS iSCSI parm block

Table 2: iSCSI Base Parameter Data Structure Summary

iSCSI Parameter Option Summary

Table 3: iSCSI Parameter Option Data Structure Summary is the summary of the options available. The options assist in interpreting the base parameter structure in such areas of IP address format, discovery methodology, and target behavior and presence.

Parameter	Name	Definition
IP Address Format	OPT[31]	IPv4 vs IPv6 IP addressing format
DHCP Option Format	OPT[30]	DHCP vendor options 201-206 or option 17
Boot LUN Usage	OPT[29]	Use the boot LUN fields
Security persistance	OPT[28]	Erase any persistent security settings
1st Target Security Context	OPT[27:24]	Format of the 1st Target security fields
1st Target Security Transport	OPT[23:22]	IPSEC transport modes
2nd Target Security Context	OPT[21:18]	Format of the 1st Target security fields
2nd Target Security Transport	OPT[17:16]	IPSEC transport modes
Target Presences	OPT[15:14]	1st / 2nd target present
Client ID for IP Usage	OPT[13:12]	Client identifier for IP acquisition
Client ID for iSCSI Usage	OPT[11:10]	Client identifier for iSCSI acquisition
Discovery IP Address Usage	OPT[9:8]	Network discovery service targeted
DHCP Option 60 format	OPT[7]	DHCP option 60 format
(Unused)	OPT[6:0]	Unused at this time (VER=0x03/LVL=0x01)

Table 3: iSCSI Parameter Option Data Structure Summary

iSCSI Extended Parameter Summary

Table 4: iSCSI Extended Parameter Data Structure Summary is the summary of extended iSCSI parameters defined. These extended parameters are used in situations where a given parameter is not universal or where the given parameter is emerging in the iSCSI industry.

Parameter	Name	Max size in Bytes (Including id/len/opt)	Definition
NULL	EPID = 0x00	5	Base + Extended DS Termination
Import PKCS#12	EPID = 0x01	1338	Reserved: not in use
Import/store PKCS#12	EPID = 0x02	1338	Reserved: not in use
1st Target IPSEC secret	EPID = 0x03	36	32 Byte CHAP secrets
2nd Target IPSEC secret	EPID = 0x04	36	32 Byte CHAP secrets
IPv6 global addr info	EPID = 0x05	148	IPv6 Global addressing values
Disk Image Identification	EPID = 0x06	259	Up to 255 byte image identification
1st Tgt full LUN Definition	EPID = 0x07	12	Fully define LUN in 8 byte format
2nd Tgt full LUN Definition	EPID = 0x08	12	Fully define LUN in 8 byte format
iSCSI Initiator Extended Name	EPID = 0x09	155	Upper chrs of full 223 chr name
iSCSI 1st target Extended Name	EPID = 0x0A	155	Upper chrs of full 223 chr name
iSCSI 2nd target Extended Name	EPID = 0x0B	155	Upper chrs of full 223 chr name
Initiator Extended Secret	EPID = 0x0C	132	2 64 byte secrets if needed

Table 4: iSCSI Extended Parameter Data Structure Summary

A producer / consumer view of the iscsi parameters are presented in Table 5: Producer / Consumer View of iSCSI Parameter Data Structure. Several of the labels below need to be explained. BIOS represents the actual BIOS executable. BIOS Pnls represents the local F1/BIOS panels a user could use to configure iSCSI parameters. MM/RSA Pnls represents the panels that the MM or RSA would need to provide to allow the blade or server to be manually configured remotely. Wizard represents the iSCSI deployment wizard configuring the iSCSI parameters.

In terms of deployment paths and locations, there are several labels below that represent different methods for deploying the parameters. Local represents the local BIOS taking the local user panel input and placing into PSA. OOB represents out of band deployment where the parameters are pushed to PSA via RS485/BMC transfers. DHCP represents deploying the parameters on a DHCP server in the network where the initiator can acquire the parameters at boot time. PSA represents the non volatile storage space on the blade or server where the parameters would reside if configured by the BIOS or pushed OOB. LCL MEM represents local server memory used to pass the parameters to the initiator.

It should be noted that the static/dynamic field indicates which parameters are considered valid. See the S_D field definition in 0 **Error! Reference source not found.**. Also, only the wizard currently capable of defining extended parameters.

Parameter	Local Producer		Remote Producer		Location	Consumer
	BIOS	BIOS Panels	MM/RSA Panels	Wizard Tools	Perm Locale	Initiator Acquires
Signature	By rote					From Lcl mem
Version	Default		Default(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Level	Default		Default(OOB)	Calc to PSA (OOB)	PSA	From Lcl mem
Data Structure Len	Default		Calculate(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Checksum	Default		Calculate(OOB)	Calc to PSA (OOB)	PSA	From Lcl mem
Static/Dynamic		User to PSA(Local)	User to PSA(OOB)	Calculate(OOB)	PSA	From Lcl mem
Options		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Retry Count	Default		User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Timeout Value	Default		User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Scope/Vendor ID		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Client Alt ID		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Discovery IP address		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Initiator IP address		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Initiator name		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
Initiator CHAP id		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Initiator CHAP pw		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Subnet Mask		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
VLAN tag		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Gateway IP address		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
1st Target IP address		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
1st Target TCP port		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
1st Target name		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
1st Target Boot LUN		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
1st Target CHAP id		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
1st Target CHAP pw		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
2nd Target IP address		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP

Parameter	Local Producer		Remote Producer		Location	Consumer
2nd Target TCP port		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
2nd Target name		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
2nd Target Boot LUN		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB) or DHCP	PSA OR DHCP	From Lcl mem or DHCP
2nd Target CHAP id		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
2nd Target CHAP pw		User to PSA(Local)	User to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem
Extended Parameters		User? to PSA(Local)	User?to PSA(OOB)	Calc to PSA(OOB)	PSA	From Lcl mem

Table 5: Producer / Consumer View of iSCSI Parameter Data Structure

Presence of the iSCSI Parameter Data Structure

The initiator can determine the presence of the iSCSI parameter data structure by looking at the first 4 bytes presented at the memory region specified by the data structure pointer. These 1st 4 bytes contain the characters 'I''S''A''N'. Refer to the "ESW Boot Initiator Support Design" document written by Scott Dunham and owned by Christopher Cerbini for more details.

iSCSI Parameter Definition for Initiators

The iSCSI parameters for the initiator are defined below comprise the set of parameters involved with establishing the iSCSI session between initiator and target. Note that in the descriptions below, required indicates that a non-NULL value must always be defined for a given parameter while optional indicates that, in some cases, NULL values are acceptable.

Signature Field

The version field is not part of the iscsi parameter data structure. It is used by the initiator to determine that the data structure is present. It is provided here for clarity. It is not used in any length calculations or any checksum verification efforts.

Perspective	Definition	Description
Format	SIG[31:0]	"ISAN" string"
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	SIG[31:0]!= "ISAN" or 0x4E415349	The data structure is not present
	SIG[31:0]= "ISAN" or 0x4E415349	Data structure is present and ready for processing
Default	SIG[31:0]= "ISAN" or 0x4E415349	Version defined – data structure in use
Producer	BIOS: Created by rote and wrs to mem	Only BIOS creates this field
Consumer	Initiator: Rds from mem	Initiator validates data structure is present

Table 6: iSCSI Data Structure Signature Definition: "ISAN"

Perspective	Definition	Description
Format	SIG[63:32]	"PARM" string"
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	SIG[63:32]!= "PARM" or 0x4D524150	No parameters are provided
	SIG[63:32]= "PARM" or 0x4D524150	Parameters are provided
Default	SIG[63:32]= "PARM" or 0x4D524150	Parameters are provided
Producer	BIOS: Created by rote and wrs to mem	Only BIOS creates this field
Consumer	Initiator: Rds from mem	Initiator validates data structure is present

Table 7: iSCSI Data Structure Signature Definition: "PARM"

The usage matrix of this signature is defined below:

"ISAN"	**"PARM"**	**Meaning:**
Not present	Not present	iSCSI not invoked and no parameter data structure present
Present	Not present	iSCSI invoked but no parameter data structure Use full DHCP if BIOS configure selected…else use any local config settings
Not present	Present	iSCSI invoked but initiator should use any local config settings Note: if iSCSI HBA in use and set to use BIOS settings, then this defaults to full DHCP. If iSCSI HBA in use and set to use local settings, then use local settings.
Present	Present	iSCSI invoked and valid parameter data structure in place

Table 8: "ISAN" and "PARM" Usage matrix

Note that in the future, the "PARM" string may be replaced with other strings that denote other usage models related or unrelated to iSCSI. If "PARM" is changed in the context of iSCSI, appropriate interlock with initiator providers and updates such as version/level will be enforced to maintain interoperability.

Version Field

The version field is a required field that presents the version of the iSCSI parameter data structure.

The version field defines the base size and field definition. If the base size changes or a field is added or deleted or changed in terms of context usage, the version field would change value. The version field is 1 byte in size. Subsequent versions of the data structure may be altered as desired as long as the versions field is incremented.

Perspective	Definition	Description
Format	VER[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	VER[7:0]=0x00	Version not defined – data structure not in use
	VER[7:0]=0x01	1st generation of this data structure is in use
	VER[7:0]>= 0x01	Version is defined and a future version is in use
Default	VER[7:0]=0x01	1st generation of this data structure is in use
Producer	Wizard: Calculate value, wrs via OOB to PSA	Wizard may support several version values
	MM/RSA: Current default, wrs via OOB to PSA	Future MM/RSA updates may migrate default value
	BIOS: If local, use default, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use interpret subsequent bytes

Table 9: Version Field Definition

Level Field

The level field is a required field that presents the level of the iSCSI parameter data structure within a given version.

The level field defines the bit format of the base data structure. If, in a given field, bit definitions are changed in definition, the level field would change in value. The level field is 1 byte in size. Subsequent versions of the data structure may be altered as desired as long as the versions field is incremented. Note that with a new version, the level field resets to 0x01 i.e. the level field refers to the data structure level in a given version.

Perspective	Definition	Description
Format	LVL[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	LVL[7:0]=0x00	Level not defined – data structure not in use
	LVL[7:0]=0x01	Level of this data structure is 1
	LVL[7:0]>= 0x01	Level is defined and a future version is in use
Default	LVL[7:0]=0x01	Level not defined – data structure not in use
Producer	Wizard: Calculate value, wrs via OOB to PSA	Wizard may support several level values
	MM/RSA: Current default, wrs via OOB to PSA	Future MM/RSA updates may migrate default value
	BIOS: If local, use default, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use interpret subsequent bytes

Table 10:Level Field Definition

Data Structure Length

The lenght field is a required field that presents the total length of the base data structure + any extended parameters bytes that the BIOS is passing to the initiator. For example, with no extended parameters specified, the length covers all parameters from version to 2nd target CHAP PW. Conversely, with 3 24-byte extended parameters, the length covers the base data structure plus the 3 24-byte extended parameters.

An initiator that uses extended parameters starts with next byte after the 2nd target CHAP PW. In the event that the initiator wants to jump to the first extended parameter, the version and level of the data structure indicates the length of the base data structure component. For example, the 1st extended parameter after the base data structure for version 1, level 1 of the base data structure is located at offset **!Undefined Bookmark, BASEDS**d in memory (remember the "ISAN" signature that is not part of the actual iSCSI parameter data structure).

Perspective	Definition	Description
Format	LEN[15:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	LEN[15:0]=0x00	Length defined – data structure not in use
	LEN[15:0]>=0x01	Length of the data structure is 1 or more bytes in size
Default	LEN[15:0]=0x00	Length not defined – data structure not in use
Producer	Wizard: Calculate value, wrs via OOB to PSA	Wizard calculates for base+extended parms
	MM/RSA: Current default, wrs via OOB to PSA	MM/RSA uses length for current version/level
	BIOS: If local, use calculate, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use interpret subsequent bytes

Table 11: Data Structure Length Definition

Data Structure Checksum

This checksum is a required value used to ensure the iSCSI data structure is valid. The checksum field contains the value required for the entire checksum to add up to 0x00.

To be explicit, the checksum covers all fields starting with the version and ending with either the 2nd target CHAP PW if not extended parameters are present or with the last byte of the value of the last extended parameter. The signature is not included in the checksum.

Perspective	Definition	Description
Format	CHK[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	CHK[7:0]=0x00	Checksum valid
	CHK[7:0]!= 0x00	Checksum valid
Default	CHK[7:0]=0x00	Checksum valid
Producer	Wizard: Calculate value, wrs via OOB to PSA	Wizard calculates for base+extended parms
	MM/RSA: Calculate, wrs via OOB to PSA	MM/RSA calculates value based on user input
	BIOS: If local, use calculate, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use interpret subsequent bytes

Table 12: Data Structure Checksum Definition

Static/Dynamic Usage Flag

The static/dynamic flag is a required flag comprising of 1 byte and is used by BIOS to determine if the iSCSI parameters are located in VPD space, for static mode, or should be acquired by a discovery service, in dynamic mode.

With a size of one byte, there can be many flavors and variations of all static, all dynamic, or a mixtures of static and dynamic approaches specified by the is byte flag.

Perspective	Definition	Description
Format	S_D[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	S_D[7:0]=0x00	All parameters acquired via DHCP acquisition
	S_D[7:0]=0x01	All parms acquired via DHCP except: - Security parameters
	S_D[7:0]=0x02	All parms acquired via DHCP except: Security parameters Initiator IP address and Discovery IP address (unicast DHCPINFORM transaction)
	S_D[7:0]=0x03	All parameters are present in data structure except: - Target name parms (retrieved by sendtarget)
	S_D[7:0]=0x04	All parameters are present in the data structure
Default	S_D[7:0]=0x00	All parameters acquired via DHCP
Producer	Wizard: Set value, wrs via OOB to PSA	Wizard determines parm method and sets
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input for parm method and sets
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine how to find parms

Table 13: Static/Dynamic Field Definition

The matrix of parameters present in the data structure versus the value of the S_D field is in Table 14: S_D Influence on Parameter Presence in Data Structure. Note the term valid indicates that a value MAY reside in this field location while a blank indicates that no value will be present in the field location. The use of the word MAY is meant to indicate that, due to some options, some field may not be used. For example, in any of these cases, the 2nd target parameters may not be specified because the customer has chosen not to have a 2nd target in the network.

Parameter	S_D[7:0]=0x04	S_D[7:0]=0x03	S_D[7:0]=0x02	S_D[7:0]=0x01	S_D[7:0]=0x00
Signature	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Version	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Level	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Data Structure Length	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Checksum	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Static/Dynamic	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)	Valid (req'd)
Options	Valid	Valid	Valid	Valid	
Retry Count	Valid	Valid			
Timeout Value	Valid	Valid			
Scope/Vendor ID			Valid	Valid	Valid(canned)
Client Alt ID			Valid	Valid	
Discovery IP address			Valid		
Initiator IP address	Valid	Valid	Valid		
Initiator name	Valid	Valid			
Initiator CHAP id	Valid	Valid	Valid	Valid	
Initiator CHAP pw	Valid	Valid	Valid	Valid	
Subnet Mask	Valid	Valid			
VLAN Tag	Valid	Valid	Valid	Valid	
Gateway IP address	Valid	Valid			
1st Target IP address	Valid	Valid			
1st Target TCP port	Valid	Valid			
1st Target name	Valid				
1st Target Boot LUN	Valid	Valid			
1st Target CHAP id	Valid	Valid	Valid	Valid	
1st Target CHAP pw	Valid	Valid	Valid	Valid	
2nd Target IP address	Valid	Valid			
2nd Target TCP port	Valid	Valid			
2nd Target name	Valid				
2nd Target Boot LUN	Valid	Valid			
2nd Target CHAP id	Valid	Valid	Valid	Valid	
2nd Target CHAP pw	Valid	Valid	Valid	Valid	

Table 14: S_D Influence on Parameter Presence in Data Structure

Option Field

The option field is an optional field presents option and configuration information for the initiator to use in BIOS execution and in establishing the iSCSI session.

The option field defines the state of options available to the BIOS and iSCSI initiator. The option field is 4 bytes in size and currently includes the options listed in subsequent sub headings. Below is the definition of the options field.

In the case of S_D=0x00, the initiator should check datastructure option bit 30 to determine the DHCP Options format that should be used. If this bit is set to 0b1, the DHCP options 201 through 206 should be requested. This request is facilitated by the DHCP client using the vendor ID option (Option 60). In the absence of a value in the vendor ID field in the iSCSI Parameter data structure, the value for option 60 should be "IBM ISAN". The DHCP server will respond with option 43 that encapsulates option 201 thru 206 under the option scope

provided by Option 60. Ie if 201 thru 206 are defined for vendor ID = “IBM ISAN”, then a valid response will be issued, otherwise NULLs will be issued. If the iSCSI parameter datastructure option bit 30 is set to 0b0, then the iscsi information for a single target is defined using the option 17 format. Since DHCP Option 17 is a “public” option, the response to option 17 can be a non NULL value but invalid in terms of iSCSI. As a result, the iSCSI parameter option bit 30 should be set to 0b1 in most cases.

Note that when using Vendor-scoped options in DHCP (Options 201 - 206), the size of the options may require RFC 3396 compliance since the overall size of th option 43 response may be larger than 255 bytes.

Perspective	**Definition**	**Description**
Format	OPT[31:0]	Data structure options
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[31:0]	See options listed below
Default	OPT[31:0]	See options listed below
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 217 to identifies DHCP format)

Table 15: Data Structure Option Field Definition

Below is the summary of options

Definition	Description
OPT[31]	IPV4 vs IPV6 (local link) address format
OPT[30]	DHCP format (option 17 vs site specific options)
OPT[29]	Boot LUN usage option (use or ignore)
OPT[28]	Security Persistance (erase or don't erase)
OPT[27]	1st Target security context
OPT[26]	
OPT[25]	
OPT[24]	
OPT[23]	1st Target security transport mode (no longer in use)
OPT[22]	
OPT[21]	2nd Target security context
OPT[20]	
OPT[19]	
OPT[18]	
OPT[17]	2nd Target security transport mode (no longer in use)
OPT[16]	
OPT[15]	Target presence
OPT[14]	
OPT[13]	Alt client ID for IP parameter acquisition
OPT[12]	
OPT[11]	Alt client ID for iSCSI parameter acquisition
OPT[10]	
OPT[09]	Discovery Address Usage
OPT[08]	
OPT[07]	DHCP Option 60 Format
OPT[06]	Undefined at this time
OPT[05]	
OPT[04]	
OPT[03]	
OPT[02]	
OPT[01]	
OPT[00]	

Table 16: iSCSI Parameter Option Summary

IPV4/IPV6 Option

This option field is an optional field that aids in the parsing of the iSCSI parameter data structure for IPv4 or IPv6 addresses. Does not change field size just where the address sits in the 8 bytes

Perspective	Definition	Description
Format	OPT[31]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[31]=0b0	IPV4 addressing used
	OPT[31]=0b1	IPV6 addressing used (local subnet format)
Default	OPT[31]=0b0	IPV4
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 17:IPV4/IPV6 Address Format Option Definition

DHCP Format Option

This option field is an optional field that defines the appropriate DHCP options to use to acquire iSCSI parameters. Namely, whether to use the internet draft using DHCP Option 17 to acquire iSCSI path information or whether to used customer/site specific options defined in this document. This option may be superceded by the initiator and DHCP functionality. Specifically, the initiator can choose to ignore this option and ask DHCP for both Option 17 and vendor specific Options. In turn, the DHCP server may respond with the valid options leaving the initiator to interrogate the DHCP server response to determine which options are valid. When using the vendor options, the options are located in option 205. In option 205, this bit should be considered reserve.

Note that when using Vendor scoped options in DHCP (Options 201 thu 206), the size of the options may require RFC 3396 compliance since the overall size of th option 43 response may be larger than 255 bytes.

Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[30]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[30]=0b0	DHCP parameters supplied by Option 17
	OPT[30]=0b1	DHCP parameters supplied by site specific options See **Error! Reference source not found.** section for details (note that scope/vendor field can aid in DHCP context)
Default	OPT[30]=0b1	DHCP parameters supplied by site specific options
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format) Note this bit is a reserved in option 205…aka a don't care condition since the 17 vs 20x usage is handled elsewhere.

Table 18: DHCP Option Definition

Boot LUN Field Usage Option

This option field is an optional field indicates whether to use the 1st target boot LUN field and 2nd target boot LUN field. If an initiator is to use the defined LUN number in the boot LUN fields, then this option should be disabled. If the initiator is to determine the boot LUN through other means such as intelligence or discovery, then this option should be enabled.

Perspective	Definition	Description
Format	OPT[29]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[29]=0b0	Use Boot LUNs defined in Boot LUN fields
	OPT[29]=0b1	Ignore Boot LUN fields under all conditions
Default	OPT[29]=0b0	Use Boot LUNs defined in Boot LUN fields
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 19: Boot LUN Usage Option Definition

Clear Persistent Credential Store

This option field is an optional field used to indicate to iSCSI service (iSCSI HBA based services namely) whether to clear IPsec certificates if they are in persistent storage. This option bit can be used in conjunction with extended parameters to implement an "import" or an "import and store" function. Note if the iSCSI service does not have persistent storage, this bit is meaningless regardless of the value.

Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[28]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[28]=0b0	Do not clear credential store
	OPT[28]=0b1	Clear credential store and accept EPID credentials
Default	OPT[28]=0b0	Do not clear credential store
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 20: Clear Persistent Credential Store Option Definition

1st Target Security Context

This option field is an optional field indicates the security context to be used with the 1st target. Specifically, the scenarios supported are no security to be used, one way authentication where the target authenticates the initiator, or mutual authentication where the initiator and target authenticate each other or IPsec based (Pre-shared key and certificates). The security context is used at iSCSI login to determine the level of security to apply to the session establishment.

Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[27:24]	1st Target Security context
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[27:24]=0b0000	No security context to be used
	OPT[27:24]=0b0101	One way security to be used in logging into target (target authenticates initiator) CHAP only
	OPT[27:24]=0b0110	Reserved
	OPT[27:24]=0b0111	Mutual security to be used in logging into target (tgt authenticates initiator / initiator authenticates tgt) CHAP only
	OPT[27:24]=0b1000	Preshared key based IPsec authentication See 1st Target Security Transport Section The pre-share key is supplied via the EPID defined as 1st Target IPsec Secret.
	OPT[27:24]=0b1001	X.509 Certificate based IPsec authentication See 1st Target Security Transport section The Distinguished Name is supplied via the EPID defined as 1st Target IPsec Secret.
	OPT[27:24]=0b1100	Pre-shared key based IPsec authentication See 1st Target Security Transport Section. The pre-share key value is supplied as a re-mapping of the CHAP ID. Note: see usage rules in 1st Target CHAP ID section
	OPT[27:24]=0b1101	reserved
Default	OPT[27:24]=0b0000	No security context to be used
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options

Table 21: 1st Target Security Context Option Definition

1st Target Security Transport Mode

This option field is an optional field indicates the security transport mode to be used when IPsec security context is selected. If CHAP authentication is specified, then these option bits are meaningless.
Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[23:22]	Target security transport
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[23:22]=0b00	reserved
	OPT[23:22]=0b01	reserved
	OPT[23:22]=0b10	reserved
	OPT[23:22]=0b11	reserved
Default	OPT[23:22]=0b00	Means nothing
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options

Table 22: 1st Target Security Transport Option Definition

2nd Target Security Context

This option field is an optional field indicates the security context to be used with the 2nd target. Specifically, the scenarios supported are no security to be used, one way authentication where the target authenticates the initiator, or mutual authentication where the initiator and target authenticate each other or IPsec based (Pre-shared key and certificates). The security context is used at iSCSI login to determine the level of security to apply to the session establishment.

Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[21:18]	2nd Target Security context
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[21:18]=0b0000	No security context to be used
	OPT[21:18]=0b0101	One way security to be used in logging into target (target authenticates initiator) CHAP only
	OPT[21:18]=0b0110	Reserved
	OPT[21:18]=0b0111	Mutual security to be used in logging into target (tgt authenticates initiator / initiator authenticates tgt) CHAP only
	OPT[21:18]=0b1000	Pre-shared key based IPsec authentication See 2nd Target Security Transport section. The pre-share key is supplied via the EPID defined as 2nd Target IPsec Secret.
	OPT[21:18]=0b1001	X.509 Certificate based IPsec authentication See 2nd Target Security Transport section. The Distinguished Name is supplied via the EPID defined as 2nd Target IPsec Secret.
	OPT[21:18]=0b1100	Pre-shared key based IPsec authentication See 2nd Target Security Transport section Note: see usage rules in 2nd Target CHAP ID section
	OPT[21:18]=0b1101	reserved
Default	OPT[21:18]=0b0000	No security context to be used
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 23 2nd Target Security Context Option Definition

2nd Target Security Transport Mode

This option field is an optional field indicates the security transport mode to be used when IPsec security context is selected. If CHAP authentication is specified, then these option bits are meaningless.

Note when option 205 is coming from the DHCP server, this bit is reserved and should be ignored regardless of value.

Perspective	Definition	Description
Format	OPT[17:16]	Target security transport
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[17:16]=0b00	reserved
	OPT[17:16]=0b01	reserved
	OPT[17:16]=0b10	reserved
	OPT[17:16]=0b11	reserved
Default	OPT[17:16]=0b00	Means nothing
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 24: 2nd Target Security Transport Option Definition

Target Presence

This option field is an optional field indicates whether any information is specified for the 1st and 2nd targets. I.e. querying these bits provides a fast path to determine if the target information is valid. If not present, then the parameters must be acquired from DHCP. In the future, these bits will allow the OOB target definitions to be preserved but ignored and, thus, use the acquired target definitions from a network service.

Perspective	Definition	Description
Format	OPT[15:14]	Target presence
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[15:14]=0b00	No parms for 1st or 2nd target is present in data structure
	OPT[15:14]=0b01	2nd target parms are present in data structure
	OPT[15:14]=0b10	1st target parms are present in data structure
	OPT[15:14]=0b11	1st and 2nd target parms are present in data structure
Default	OPT[15:14]=0b00	No parms for 1st or 2nd target is present in data structure
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 25: Target Presence Option Definition

Client ID for IP Parameter Acquisition

This option field is an optional field indicates what to use as the client ID for IP parameter acquisition. I.e. querying these bits determines whether to use the EN MAC address, Scope/vendor ID, or Client Alternate ID when querying a DHCP server for IP parameters. If not present, then the parameters must

be acquired from DHCP. Note that usage of Scope/Vendor ID as an additional usage scope tool is independent of this option.

Note that windows DHCP servers only support client identification via Ethernet MAC address. As a result, the option bits below should be set to select Ethernet MAC address when a windows DHCP server is in use.

Perspective	Definition	Description
Format	OPT[13:12]	ID for IP parameter acquistion
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[13:12]=0b00	Use EN MAC address of current port
	OPT[13:12]=0b01	Use Scope/Vendor ID as ID
	OPT[13:12]=0b10	Use Client Alternate ID as ID
	OPT[13:12]=0b11	Reserved
Default	OPT[13:12]=0b00	Use EN MAC address of current port
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 26: Client ID for IP Acquisition Option Definition

Client ID for iSCSI Parameter Acquisition

This option field is an optional field indicates what to use as the client ID for iSCSI parameter acquisition. I.e. querying these bits determines whether to use the EN MAC address, Scope/vendor ID, or Client Alternate ID when querying a DHCP server for iSCSI parameters. If not present, then the parameters must be acquired from DHCP. Note that usage of Scope/Vendor ID as an additional usage scope tool is independent of this option.

Note that windows DHCP servers only support client identification via Ethernet MAC address. As a result, the option bits below should be set to select Ethernet MAC address when a windows DHCP server is in use.

Perspective	Definition	Description
Format	OPT[11:10]	ID for iSCSI parameter acquistion
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[11:10]=0b00	Use EN MAC address of current port
	OPT[11:10]=0b01	Use Scope/Vendor ID as ID
	OPT[11:10]=0b10	Use Client Alternate ID as ID
	OPT[11:10]=0b11	Reserved
Default	OPT[11:10]=0b00	Use EN MAC address of current port
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (For DHCP: Option 205 to identifies DHCP format)

Table 27 Client ID for iSCSI Acquisition Option Definition

Discovery IP Address Usage

This option field is an optional field indicates whether the discovery IP address should be used to access a DHCP server or should be used to access a SLP server (DA). It should be noted that if this bit indicates the discovery should with a SLP server and the iSCSI client does not support SLP, the initiator should return control to the caller with an error indicating unsupported option.

Perspective	Definition	Description
Format	OPT[9:8]	ID for iSCSI parameter acquistion
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[9:8]=0b00	Discovery IP address points to a DHCP server
	OPT[9:8]=0b01	Discovery IP address points to a SLP server
	OPT[9:8]=0b10	Discovery IP address points to a iSNS server
	OPT[9:8]=0b11	Reserved
Default	OPT[9:8]=0b00	Discovery IP address points to a DHCP server
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine options (DHCP or SLP or iSNS)

Table 28 Client ID for iSCSI Acquisition Option Definition

DHCP Option 60 Format

This option field is an optional field indicates the format of DHCP option 60 when using DHCP servers. The creation of this bit stems from ambiguous information on how Option 60 is defined and used in Microsoft DHCP servers. Namely, while the PXE standard states the option 60 format is "PXEClient:ARCH:xxxx:UNDI:yyyzzz", the typical creation of option 60 in MS DHCP servers use the value "PXEClient". As of release of 3.3 of this document, it is not clear whether MS DHCP servers substring the option 60 coming from the client or whether the industry abbreviated the PXE specification. Accordingly, this bit, with its default being the "short" form can coexist in either situation.

In essence, making the default the "short" format, the initiator is actual substringing on the behalf of the DHCP server.

Perspective	Definition	Description
Format	OPT[7]	ID for DHCP option 60 format
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	OPT[7]=0b0	DHCP option 60 format is "short" Example: "IBM ISAN"
	OPT[7]=0b1	DHCP option 60 format is "long" Example" "IBM ISAN:03:01:QLGC:28:51
Default	OPT[7]=0b0	DHCP option 60 format is "short"
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use determine option 60 format for any DHCP activity

Table 29 DHCP Option 60 Format Option Definition

Undefined/Future Options

The remaining bits in the options field are undefined at this time. In the future versions of the data structure, these bits can be used for additional options

Retry Count

This retry count is an optional value used to indicate the number of iSCSI login retries to attempt before giving up on establishing the iSCSI session with target. Bit [7:4] define the retry count to 1st target while bits [3:0] define retry count to 2nd target. The initiator flow should try the 1st target the number of times specified then try the 2nd target the number specified then it should exit out and return control to the BIOS.

Perspective	Definition	Description
Format	RTRY[7:0]	RTRY field
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	RTRY[7:4]=0x0	0 retries to be attempted to 1st target
	RTRY[7:4]=!0x0	At least 1 retry is to be attempted to 1st target
	RTRY[3:0]=0x0	0 retries to be attempted to 2nd target
	RTRY[3:0]!=0x0	At least 1 retry is to be attempted to 2nd target
Default	RTRY[7:0]=0xFF	16 retries for 1st target and 2nd target
Example	RTRY[7:0]=0x33	3 retries are to be attempted to 1st and 2nd targets
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user dfts, else PSA to mem	Local access overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine how to find parms

Table 30: Retry Count Field Definition

Timeout Value

This timeout value is an optional value used to indicate the wait time to expire before making another attempt to access the targets. Note that the time out values can be indicated for the 1st target and 2nd target separately

Perspective	Definition	Description
Format	TOVAL[7:0]	Timeout value field
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	TOVAL[7:4]=0x0	Timeout value for 1st target access is 100 ms
	TOVAL[7:4]=0x1	Timeout value for 1st target access is 200 ms
	TOVAL[7:4]=0x2	Timeout value for 1st target access is 500 ms
	TOVAL[7:4]=0x4	Timeout value for 1st target access is 2000 ms
	TOVAL[7:4]=0x8	Timeout value for 1st target access is 20000 ms
	TOVAL[3:0]=0x0	Timeout value for 2nd target access is 100 ms
	TOVAL[3:0]=0x1	Timeout value for 2nd target access is 200 ms
	TOVAL[3:0]=0x2	Timeout value for 2nd target access is 500 ms
	TOVAL[3:0]=0x4	Timeout value for 2nd target access is 2000 ms
	TOVAL[3:0]=0x8	Timeout value for 2nd target access is 20000 ms
Default	TOVAL[7:0]=0x88	Timeout value for 1st and 2nd targets is 20 secs each
Example	TOVAL[7:0]=0x44	Timeout values for 1st and 2nd target is 2 seconds
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user dfts, else PSA to mem	Local access overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine how to find parms

Table 31: Timeout Value Field Definition

Scope/Vendor ID

This Scope/vendor ID is an optional address used in cases where parameters are acquired from a DHCP service and some scope or vendor casting is needed to aid the DHCP service in determining the parameters to return to the DHCP client. For example, this field can be used to identify the DHCPREQUEST or DHCPINFORM transaction is within the scope of iSCSI parameter acquisition. Note that since this field is per instance and per initiator, finer levels of scoping are possible.

The format of these 8 bytes is 8 characters to use in any scoping entries in DHCP transactions. Note that any spaces must be presented in the string as 0x20 and that if 8 valid characters are used, the length of the field (being 8 bytes) defines the end of the string (ie no /n or trailing 0x00s beyond the 8th character).

Perspective	Definition	Description
Format	SVID[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	SVID[63:0]=0x00	Scope/Vendor ID is not valid
	SVID[63:0]>= 0x00	Scope/Vendor ID is valid
Default	SVID[63:0]='IBM ISAN' or 0x4E415349 204D4249	Scope/Vendor ID is the canned value
Examples	SVID[63:0]='IBM ISAN'	Scope/Vendor ID is "IBM ISAN" (Global)
	SVID[63:0]='IBM13402'	Scope/Vendor ID is "IBM13402" (server 134 port 2)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 32: Scope/Vendor ID Field Definition

Client Alternate ID

This Client Alternate ID is an optional address used in cases where parameters are acquired from a DHCP service and a client ID different from EN MAC address or Scope/Vendor casting is needed to aid the DHCP service in determining the parameters to return to the DHCP client. For example, this field can be used to identify the DHCPREQUEST or DHCPINFORM transaction is within the scope of IP parameter or iSCSI parameter acquisition. Note that since this field is per instance and per initiator, finer levels of scoping are possible. Also note that the initiator may use this ID for either IP or iSCSI and use Scope/Vendor for iSCSI or IP to segment the context of acquisition.

The format of these 8 bytes is 8 characters to use in any scoping entries in DHCP transactions. Note that any spaces must be presented in the string as 0x20 and that if 8 valid characters are used, the length of the field (being 8 bytes) defines the end of the string (ie no /n or trailing 0x00s beyond the 8th character).

Note that when using a windows DHCP server, the client identification is only done by Ethernet MAC address preceded by a byte with the value of 0x01. As a result, when using a client ID in DHCP option 61, the actual value should be 0x01 || CAID[63:0] via out of band definition and 0x01 || DHCP Opt 204[ID] via in band definition. This permits compatibility between Windows DHCP servers and Linux DHCP servers.

Perspective	Definition	Description
Format	CAID[63:0]	Client Alternate ID
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	CAID[63:0]=0x00	Client Alternate ID is not valid
	CAID[63:0]>= 0x00	Client Alternate ID is valid
Default	CAID[63:0]=0x00	Client Alternate ID is the canned value
Examples	CAID[63:0]='Svr 0013'	Client Alternate ID is "Svr 0013" (server specific)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 33: Client Alternate ID Definition

Discovery IP Address

This discovery IP address is an optional address used in cases where the initiator IP address is defined via the static/parameter push approach or via F1 BIOS Panels. This option aids in the dynamic or parameter acquisition approach where, for a variety of reasons, the initiator must access a specific IP address to acquire the parameters.

The discovery IP address (with the appropriate Discovery IP Address Usage set to use DHCP option) is used by the initiator to identify and unicast to a specific DHCP server to acquire some or all of the iSCSI parameters. With the support of unicast, DHCP broadcast storms can be eliminated.

If the discovery IP address is 0x00000000 00000000, the address should be considered invalid and, thus, not used. If the discovery IP address is 0x00000000 FFFFFFFF, the address indicates a broadcast method of transmission should be used (such as a broadcast of DHCPINFORM). If the the discovery IP address is equal to some other value, the address indicates a unicast method of transmission should be used (such as a unicast of DHCPINFORM).

In the future, solutions using SLP or iSNS discovery services, this discovery IP address is used for identifying the SLP or iSNS server in the network.

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	DIP[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	DIP[63:0]=0x00	Discovery address not valid
	DIP[63:0]> 0x00 DIP[63:0]< 0xFFFFFFFF FFFFFFFF	Discovery address is valid
Default	DIP[63:0]=0x00	Discovery address not valid
	DIP[63:0] != 0x00 & < 0x00000000 FFFFFFFF	Discovery address is valid and unicast should be used
	DIP[63:0] >= 0x00000000 FFFFFFFF	Discovery address is valid and broadcast should be used
Examples	DIP[63:0]=0x00000000 11223344	IPV4 address of 33.66.97.132
	DIP[63:0]=0x11223344 11223344	IPV6 address of FE80::1122:3344:1122:3344 Note FE80:: = FE80:0000:0000:0000 (canned link local format for high 8 bytes)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 34: Discovery IP Address Field Definition

Initiator IP Address

This Initiator IP address is an optional address used in cases where the initiator IP address is defined via the static/parameter push approach or via F1 BIOS Panels. This IP address defines the IP address for the initiator to use for iSCSI traffic. In the case of multihomed systems, this IP address may be limited to iSCSI traffic only, while application traffic may use one or more other IP addresses.

This Initiator IP address is an IETF compliant IP address that represents the iSCSI initiator IP address. It is used during an iSCSI session as the transaction requestor IP address. There may be more than one IP address associated with a given initiator through use of portal groups on the initiator. This one or more IP address is associated with an iSCSI IQN string that represents the identity of the iSCSI initiator.

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	IIP[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	IIP[63:0]=0x00	Initiator IP address not valid
	IIP[63:0]>0x00 IIP[63:0]< 0xFFFFFFFF FFFFFFFF	Initiator IP address is valid
Default	IIP[63:0]=0x00	Initiator IP address is not valid
Examples	IIP[63:0]=0x00000000 11223344	IPV4 address of 33.66.97.132
	IIP[63:0]=0x11223344 11223344	IPV6 address of FE80::1122:3344:1122:3344 Note FE80:: = FE80:0000:0000:0000 (canned link local format for high 8 bytes)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 35: Initiator IP Address Field Definition

Initiator Name

This initiator name is an optional string used in cases where the initiator IQN string is defined via static/parameter push approach or via F1 BIOS panels. This IQN string identifies the initiator in the context of iSCSI throughout the iSCSI session.

This initiator name is an iSCSI RFC compliant string in terms of formatting that represents the iSCSI target. The iSCSI RFC specifies that a given name can be up to 223 bytes in length. However, due to space limitations in the static/parameter push mode and reasonably screen space on the BIOS/F1 panels, this architecture limits size of the string to 72 bytes. It is used during an iSCSI sessions as the transaction requestor iSCSI identity. Only one initiator name can be associated with a given iSCSI session while multiple initiator IP addresses may be defined for that given iSCSI session.

The IQN format for the name is a 72 character string where the 1st 4 characters are 'iqn.' while the remaining 68 characters are alphanumeric using UTF-8 encoding. The EUI format for the name is a 20 character string where the 1st 4 characters are 'eui.' while the remaining 16 characters represent 16 characters representing 16 hex characters.

Note that any spaces must be presented in the string as 0x20 and that if 72 valid characters are used, the length of the field (being 8 bytes) defines the end of the string (ie no /n or trailing 0x00s beyond the 72nd character).

Note, as part of the 4.10 and later revision of the specification, a construct has been added to provide the full 223 characters for the iSCSI name. In essence, several EPIDs have been designated to provide the remaining characters in the name such that least significant 72 characters are defined in the field and the most significant

characters are defined in the appropriate EPID. These fields are concatentated together in the form of EPID(0x09)||IIQN in the case of the initiator name. Below illustrates several examples.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
IIQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x09) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI initiator name = EPID(0x09)||IIQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr099…chr000>” ie all 104 characters…”iqn.”+100 chars
IIQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x09) = 0x00…0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI initiator name = EPID(0x09)||IIQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	IIQN[71:0][7:0]	Up to a 72 byte string (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	IIQN[71:0][7:0]=0x00	Initiator IQN is not valid
	IIQN[71:0][7:0]!= 0x00	Initiator IQN is valid
Default	IIQN[71:0][7:0]=0x00	Initiator IQN is not valid
Examples	IIQN[71:0][7:0]=’iqn.date.blade1’	IQN format – note 1st 4 chrs are iqn. (note string format)
	IIQN[71:0][7:0]=’eui.F0E1D2C3B4A59678’	EUI format – note 1st 4 chrs are eui. (note string format0
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine how to value

Table 36: Initiator Name Field Definition

Initiator CHAP ID

This initiator CHAP ID is an optional string. It is optional from several perspectives. First, it is used in cases where the initiator CHAP ID is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication and there is no requirement for an explicit initiator ID to be used in the discovery process. In the case of the dynamic or parameter acquire approach; this field represents the CHAP ID for authentication as well as the Client Identifier for DHCP scoping. In a true broadcast scenario with no authentication, this field can be NULL.

This initiator CHAP ID in an IETF compliant ID used to aid in the authentication of the initiator to the target. Specifically, the initiator would encode this information during iSCSI session establishment and send it to the target where the target, running agreed to authentication algorithms, would validate the initiator is actually valid. When just the initiator CHAP ID and PW are used, it is in the context of one way authentication.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	ISDI[15:0][7:0]	16 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	ISDI[15:0][7:0]=0x00	Initiator security descriptor ID not valid
	ISDI[15:0][7:0]!=0x00	Initiator security descriptor ID valid
Default	ISDI[15:0][7:0]=0x00	Initiator security descriptor ID not valid
Example	ISDI[15:0][7:0]='blade_1_id'	Chap ID = "blade_1_id"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use for security

Table 37: Initiator CHAP ID Field Definition

It should be noted that complex security contexts are not currently applicable to the initiator security parameters. As a result, the current architecture does not take advantage of concatenated initiator CHAP and initiator PW field. Future solutions may choose to leverage this construct in a yet to be defined way. So in the current architecture, while the concatenated field is defined, it is currently unused. When larger more complex security constructs are required, there two possible solutions available to the solution. First solution is use the option bit in the target security context field to indicate that the given CHAP ID and PW fields be concatenated into one single field. For example, with the option bits set, initiator CHAP ID field and initiator CHAP PW field are concatenated into on initiator security field. Refer to Table 38: Initiator Complex Security Field Definition for format details. The second solution is using the extended parameter construct to specify a larger more complex security parameter.

Perspective	Definition	Description
Format	ISDX[39:0][7:0]	ISDI || ISDP (40 bytes of space via 16||24)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	ISDX[39:0][7:0]=0x00	Initiator security descriptor ID not valid
	ISDX[39:0][7:0]!=0x00	Initiator security descriptor ID valid
Default	ISDX[39:0][7:0]=0x00	Initiator security descriptor ID not valid
Example	ISDX[39:0][7:0]= 0xFEDCBA98 76543210 F0E1D2C3 B4A59687	32 byte security key
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 38: Initiator Complex Security Field Definition

Initiator CHAP PW

This initiator CHAP PW is an optional string. It is optional from several perspectives. First, it is used in cases where the initiator CHAP PW is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication.

This initiator CHAP PW in an IETF compliant PW used to aid in the authentication of the initiator to the target. Specifically, the initiator would encode this information during iSCSI session establishment and send it to the target where the target, running agreed to authentication algorithms, would validate the initiator is actually valid. When just the initiator CHAP ID and PW are used, it is in the context of one way authentication.

To enforce strong CHAP authentication, the password is 24 bytes in size which provides the minimum of 128 bits of randomness on the secrets. The format of the password is an ASCII field.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	ISDP[23:0][7:0]	24 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	ISDP[23:0][7:0]=0x00	Initiator security descriptor PW not valid
	ISDP[23:0][7:0]!=0x00	Initiator security descriptor PW valid
Default	ISDP[23:0][7:0]=0x00	Initiator security descriptor PW not valid
Example	ISDP[23:0][7:0]='blade_1_pw'	Chap PW = "blade_1_pw"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 39: Initiator CHAP Password Field Definition

Network Subnet Mask

This network subnet mask is an optional mask used in cases where the subnet mask is defined via static/parameter push approach or via F1 BIOS panels. The mask defines the local network scope of all the IP addresses on this particular subnet. Specifically, this mask defines the local network containing stations that may be accessed directly from this station (i.e. no router or gateways involved).

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	MSK[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	MSK[63:0]=0x00	Subnet mask is not valid
	MSK[63:0]> 0x00 MSK[63:0]< 0xFFFFFFFF FFFFFFFF	Subnet mask is valid
Default	MSK[63:0]=0x00	Subnet mask is not valid
Examples	MSK[63:0]=0x00000000 FFFFFF00	IPV4 mask of 255.255.255.0
	MSK[63:0]=0xFFFFFFFF FFFFFF00	IPV6 mask of FFFF:FFFF:FFFF:FFFF (This would be concatenated with the IPV6 address) (e.g. FFFF:FFFF:FFFF:FFFF:1122:3344:1122:3344)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 40: Subnet Mask Field Definition

VLAN Tag

This network subnet mask is an optional mask used in cases where the VLAN tag is defined via static/parameter push approach or via F1 BIOS panels. The VLAN tag defines the VLAN virtual LAN to use for the iSCSI traffic within the subnet. The VLAN construct allows the logical or virtual segmentation of traffic to a subset of network stations on a given physical local network. Specifically, this VLAN tag defines the VLAN tag parameters to be used by network stations when 802.1Q is implementation on a given local network.

The BIOS can use this information to pre-configure a given NIC to enable the desired VLAN tag “lane” to be used for iSCSI traffic. No assumption is made on using this VLAN tag for other kinds of network or application traffic, there it should not be assumed that this VLAN is applicable for all network traffic eminating from this network station.

The VLAN configuration cannot be supported in a DHCP full DHCP acquisition environment since the initiator can not see the DHCP server on the defined VLAN in order to acquire the VLAN tag (a conundrum). In a DHCP partial acquisition, the VLAN configuration can be support if the VLAN tag settings are defined by BIOS or a deployment wizard and passed to the NIC and initiator directly. In this case, since the initiator has the VLAN information defined before starting the DHCP exchange, the initiator can acquire the iSCSI parameters from the DHCP server sitting on the given VLAN..

Perspective	Definition	Description
Format	VLAN[15:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	VLAN[15:0]=0x00	Untagged packets are to be sent and received for iSCSI
	VLAN[15:12]=!0x0	VLAN tag control information per 802.1Q (user priority and canonical form bits)
	VLAN[11:0]> 0x00 VLAN[11:0]< 0xFFF	Valid VLAN tag to use when sending and receiving iSCSI packets
Default	VLAN[15:0]=0x0000	Untagged packets are to be used for iSCSI transactions
Examples	VLAN[15:0]=0x0000	Untagged packets are to be used for iSCSI transactions
	VLAN[15:0]=0xF001	VLAN tags of 0x001 with the highest user priority and of canonical form
Producer	Wizard: Set value, wrs via OOB	Wizard determines options, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem	Initiator to use determine locale (mem)

Table 41: VLAN Tag Field Definition

Network Gateway/Router IP Address

This network gateway/router IP address is an optional address used in cases where the subnet mask is defined via static/parameter pus approach or via F1 BIOS panels. Note that this address defines either the gateway or the router to reach outside the current subnet and it is IETF compliant.

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	GRIP[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	GRIP[63:0]=0x00	Gateway/router IP address not valid
	GRIP[63:0]> 0x00 GRIP[63:0]< 0xFFFFFFFF FFFFFFFF	Gateway/router IP address is valid
Default	GRIP[63:0]=0x00	Gateway/router IP address is not valid
Examples	GRIP[63:0]=0x00000000 11223344	IPV4 address of 33.66.97.132
	GRIP[63:0]=0x11223344 11223344	IPV6 address of FE80::1122:3344:1122:3344 Note FE80:: = FE80:0000:0000:0000 (canned link local format for high 8 bytes)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 42: Gateway/Router IP Address Field Definition

1st Target IP Address

This 1st target IP address is an optional address is used in cases where the 1st target IP address is defined via static/parameter push or via F1 BIOS panels. This IP address defines the IP address for the target to use for iSCSI traffic. In the case of multihomed systems, several IP addresses may be employed for a variety of reasons such as multiple classes of iSCSI traffic, iSCSI and mgmt traffic coexisting on a given NIC, or for failover scenarios. The initiator should use this address to access the 1st target.

This 1st target IP address is an IETF compliant IP address that represents the iSCSI target IP address. It is used during an iSCSI session as the transaction responder IP address. There may be more than one IP address associated with a given target through use of portal groups on the target. This one or more IP address is associated with an iSCSI IQN string that represents the identity of the iSCSI target node.

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	T1IP[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1IP[63:0]=0x00	1st Target IP address not valid
	T1IP[63:0]> 0x00 T1IP[63:0]< 0xFFFFFFFF FFFFFFFF	1st Target IP address is valid
Default	T1IP[63:0]=0x00	1st Target IP address is not valid
Examples	T1IP[63:0]=0x00000000 11223344	IPV4 address of 33.66.97.132
	T1IP[63:0]=0x11223344 11223344	IPV6 address of FE80::1122:3344:1122:3344 Note FE80:: = FE80:0000:0000:0000 (canned link local format for high 8 bytes)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 43: 1st Target IP Address Field Definition

1st Target TCP Port Number

This 1st target TCP port number is an optional address. This address is optional from several perspectives. First, it is used in cases where the 1st target IP address is defined via static/parameter push approach or via F1 BIOS panels. Second, in cases where the static/parameter push approach or F1 BIOS panel approach is used, the 1st target TCP port number can still be NULL. In this second scenario, a NULL value here indicates the default TCP port number of 3260 should be used for all iSCSI traffic to the 1st Target IP Address. In this second scenario, a non-NULL value here should be used as the TCP port number to be sued for all iSCSI traffic to the 1st target IP address. This 1st target TCP port number is an IETF compliant TCP port number associated with the 1st target IP addresses. It is used to aid in the management of a variety of traffic at the 1st target, where types of traffic flow through a defined TCP port. The default here is 3260.

Perspective	Definition	Description
Format	T1PT[15:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1PT[15:0]=0x0000	1st Target TCP port number not valid
	T1PT[15:0]> 0x0000 T1PT[15:0]< 0xFFFF	1st Target TCP port number is valid
Default	T1PT[15:0]=0x0000	1st Target TCP port number not valid (implies 3260d or 860d is to be used)
Examples	T1PT[15:0]=0x0100	1st Target TCP port number for iSCSI is 256
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 44: 1st Target TCP Port Field Definition

1st Target Name

This 1st target name is an optional name. This name is optional from several perspectives. First, it is used in cases where the 1st target IP name is defined via static/parameter push approach or via F1 BIOS panels. Second, in cases where the static/parameter push approach or F1 BIOS panel approach is used, the 1st target name can still be optional. In this second scenario, a NULL value here indicates that the initiator should establish a discovery session with the 1st target and send a SENDTARGET command to obtain the storage topology visible to this initiator. Note that while the SENDTARGET response can present several IQNs, it is assumed that the initiator can determine the appropriate LUN to boot off of by using hashing or string matching algorithms. In the second scenario, a non-NULL value here indicates the specific IQN to be accessed during boot. Note that while IQNs can represent a target node fronting several LUNs, it is assumed here that a defined IQN represents a single LUN.

This 1st target name is an iSCSI RFC compliant string in terms of formatting that represents the iSCSI target. The iSCSI RFC specifies that a given IQN can be 223 bytes in length. However, due to space limitations in the static/parameter push mode and reasonably screen space on the BIOS/F1 panels, this architecture limits size of the string to 72 bytes. It is used during an iSCSI sessions as the transaction responder iSCSI identity. Only one name can be associated with a given iSCSI session while multiple initiator IP addresses may be defined for that given iSCSI session.

The IQN format for the name is a 72 character string where the 1st 4 characters are 'iqn.' while the remaining 68 characters are alphanumeric using UTF-8 encoding. The EUI format for the name is a 20 character string where the 1st 4 characters are 'eui.' while the remaining 16 characters represent 16 characters representing 16 hex characters.

Note that any spaces must be presented in the string as 0x20 and that if 72 valid characters are used, the length of the field (being 8 bytes) defines the end of the string (ie no /n or trailing 0x00s beyond the 72nd character).

Note, as part of the 4.10 and later revision of the specification, a construct has been added to provide the full 223 characters for the iSCSI name. In essence, several EPIDs have been designated to provide the remaining characters in the name such that least significant 72 characters are defined in the field and the most significant characters are defined in the appropriate EPID. These fields are concatentated together in the form of EPID(0x0A)||T1IQN in the case of the initiator name. Below illustrates several examples.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
T1IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0A) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI 1st target name = EPID(0x0A)||T1IQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr099…chr000>” ie all 104 characters…”iqn.”+100 chars
T1IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0A) = 0x00…0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI 1st target name = EPID(0x0A)||T1IQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	T1IQN[71:0][7:0]	Up to a 72 byte string (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1IQN[71:0][7:0]=0x00	1st Target IQN is not valid
	T1IQN[71:0][7:0]!= 0x00	1st Target IQN is valid
Default	T1IQN[71:0][7:0]=0x00	1st Target IQN is not valid
Examples	T1IQN[71:0][7:0]=’iqn.date.blade1.tgt1’	IQN format – note 1st 4 chrs are iqn. (note string format)
	T1IQN[71:0][7:0]=’eui.F0E1D2C3B4A59678’	EUI format – note 1st 4 chrs are eui. (note string format0
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 45: 1st Target IQN Field Definition

1st Target Boot LUN

This boot disk LUN number is an optional parameter used in cases where the boot disk LUN number is defined via static/parameter push approach or via F1 BIOS panel.

For cases where 8 byte LUNs are required, EPID(0x07) provides the boot LUN definition in 8 byte formats.

The rules for how to interpret these fields are as follows. The least significant byte of the boot LUN is specified in this field and adheres to the boot LUN options bits in the OPT field. The full 8 byte format for the boot LUN is specified in EPID(0x07) and can ignore the boot LUN options bits in the OPT field. If the initiator supports the boot LUN EPs, it checks for them first.. If present, it uses them. If not present, it uses this field and assumes a 1 byte LUN is defined. Note for initiators that are unaware of or do not support the 8 byte LUN EPs, this field is used as governed by the option bits.

The configuration tool must ensure it picks the right least significant byte to place in this field

Below are 2 examples

Boot LUN = 0x000E000000000000….”0b00” format

EPID(0x08) = 0x000E000000000000
T2BL[7:0] = 0x0E

Boot LUN = 0x800000000000000E..."0b80" format
EPID(0x08) = 0x800000000000000E
T2BL[7:0] = 0x0E

Perspective	Definition	Description
Format	T1BL[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1BL[7:0]=0x00	1st Target Boot LUN is 0
	T1BL[7:0]!=0x00	1st Target Boot LUN is not 0
Default	T1BL[7:0]=0x00	1st Target Boot LUN is 0
Example	T1BL[7:0]=0x04	1st Target Boot LUN is 4
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 46: 1st Target Boot LUN Field Definition

1st Target CHAP ID

This 1st target CHAP ID is an optional string. It is optional from several perspectives. First, it is used in cases where the 1st target CHAP ID is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication.

This 1st target CHAP ID in an IETF compliant ID used to aid in the authentication of the target to the initiator. Specifically, the 1st target would encode this information during iSCSI session establishment and send it to the initiator where the initiator, running agreed to authentication algorithms, would validate the 1st target is actually valid. When the initiator and 1st target CHAP ID/PW is used, it is in the context of mutual authentication.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	T1SDI[15:0][7:0]	16 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1SDI[15:0][7:0]=0x00	1st Target security descriptor ID not valid
	T1SDI[15:0][7:0]!=0x00	1st Target security descriptor ID valid
Default	T1SDI[15:0][7:0]=0x00	1st Target security descriptor ID not valid
Example	T1SDI[15:0][7:0]='first_tgt_id'	Chap ID = "first_tgt_id"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 47: 1st Target CHAP ID Field Definition

Note, when larger more complex security constructs are required, there are two possible solutions available to the solution. First solution is use the option bit in the options field 1st Target Security Context to indicate that the given CHAP ID and PW fields be concatenated into one single field. For example, when the option is set for pre-share keys or X.509 certificates, 1st target CHAP ID field and 1st target CHAP PW field are concatenated into on 1st target security field. Refer to Table 48: 1st Target Complex Security Field Definition for format details. When pre-shared key option is selected, the concatenated field represents the pre-shared key. When the X.509 certificate option is selected, the concatenated field represents distinguished name (DN). Both are ascii strings terminated either by 0x00 or by using all 32 bytes of the concatenated field. The second solution is using the extended parameter construct to specify a larger more complex security parameter.

Perspective	Definition	Description
Format	T1SDX[39:0][7:0]	T1SDI ‖ T1SDP (note see option bit) (aka 40 bytes of space via 16 ‖ 24)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1SDX[39:0][7:0]=0x00	1st Target security descriptor not valid
	T1SDX[39:0][7:0]!=0x00	1st Target security descriptor valid
Default	T1SDX[39:0][7:0]=0x00	1st Target security descriptor not valid
Example	T1SDX[39:0][7:0]= 0xFEDCBA98 76543210 F0E1D2C3 B4A59687	32 byte security key Defined as pre-share key or distinguished name based on the option 1st Target Security Context settings. See **Error! Reference source not found.**.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 48: 1st Target Complex Security Field Definition

1st Target CHAP PW

This 1st target CHAP PW is an optional string. It is optional from several perspectives. First, it is used in cases where the 1st target CHAP PW is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication.

This 1st target CHAP PW in an IETF compliant PW used to aid in the authentication of the target to the initiator. Specifically, the 1st target would encode this information during iSCSI session establishment and send it to the initiator where the initiator, running agreed to authentication algorithms, would validate the 1st target is actually valid. When the initiator and target CHAP ID/PW is used, it is in the context of mutual authentication.

To enforce strong CHAP authentication, the password is 24 bytes in size which provides the minimum of 128 bits of randomness on the secrets. The format of the password is an ASCII field.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	T1SDP[23:0][7:0]	24 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1SDP[23:0][7:0]=0x00	1st Target security descriptor PW not valid
	T1SDP[23:0][7:0]!=0x00	1st Target security descriptor PW valid
Default	T1SDP[23:0][7:0]=0x00	1st Target security descriptor PW not valid
Example	T1SDP[23:0][7:0]='first_tgt_pw'	Chap PW = "first_tgt_pw"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 49: 1st Target CHAP Password Field Definition

Note, when larger more complex security constructs are required, there two possible solutions available to the solution. First solution is use the option bit in the options field xxx to indicate that the given CHAP ID and PW fields be concatenated into one single field. For example, with the option bit set, initiator CHAP ID field and initiator CHAP PW field are concatenated into on initiator security field. Refer to Table 48: 1st Target Complex Security Field Definition for format details. The second solution is using the extended parameter construct to specify a larger more complex security parameter.

2nd Target IP Address

This 2nd target IP address is an optional address is used in cases where the 2nd target IP address is defined via static/parameter push or via F1 BIOS panels. This IP address defines the IP address for the target to use for iSCSI traffic. In the case of multihomed systems, several IP addresses may be employed for a variety of reasons such as multiple classes of iSCSI traffic, iSCSI and mgmt traffic coexisting on a given NIC, or for failover scenarios. The initiator should use this address to access the 2nd target.

This 2nd target IP address is an IETF compliant IP address that represents the iSCSI target IP address. It is used during an iSCSI session as the transaction responder IP address. There may be more than one IP address associated with a given target through use of portal groups on the target. This one or more IP address is associated with an iSCSI IQN string that represents the identity of the iSCSI target node.

Note that in the context of IPv6, the 8 bytes below can support a link local IPv6 address. If a full IPv6 address is required, the associated EPID of EPID(0x05) provides the appropriate upper 8 bytes of a given IPv6 address.

Also note that IPv4 vs IPv6 interpretation of the 8 bytes below is governed by the OPT[31] bit in the parameter options field.

Perspective	Definition	Description
Format	T2IP[63:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2IP[63:0]=0x00	2nd Target IP address not valid
	T2IP[63:0]> 0x00 T2IP[63:0]< 0xFFFFFFFF FFFFFFFF	2nd Target IP address is valid
Default	T2IP[63:0]=0x00	2nd Target IP address is not valid
Examples	T2IP[63:0]=0x00000000 11223344	IPV4 address of 33.66.97.132
	T2IP[63:0]=0x11223344 11223344	IPV6 address of FE80::1122:3344:1122:3344 Note FE80:: = FE80:0000:0000:0000 (canned link local format for high 8 bytes)
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 50: 2nd Target IP Address Field Definition

2nd Target TCP Port Number

This 2nd target TCP port number is an optional address. This address is optional from several perspectives. First, it is used in cases where the 2nd target IP address is defined via static/parameter push approach or via F1 BIOS panels. Second, in cases where the static/parameter push approach or F1 BIOS panel approach is used, the 2nd target TCP port number can still be NULL. In this second scenario, a NULL value here indicates the default TCP port number of 3260 should be used for all iSCSI traffic to the 2nd Target IP Address. In this second scenario, a non-NULL value here should be used as the TCP port number to be sued for all iSCSI traffic to the 2nd target IP address.

This 2nd target TCP port number is an IETF compliant TCP port number associated with the 2nd target IP addresses. It is used to aid in the management of a variety of traffic at the 2nd target, where types of traffic flow through a defined TCP port. The default here is 3260.

Perspective	Definition	Description
Format	T1PT[15:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T1PT[15:0]=0x0000	2nd Target TCP port number not valid
	T1PT[15:0]> 0x0000 T1PT[15:0]< 0xFFFF	2nd Target TCP port number is valid
Default	T1PT[15:0]=0x0000	2nd Target TCP port number not valid (implies 3260d or 860d is to be used)
Examples	T1PT[15:0]=0x0100	1st Target TCP port number for iSCSI is 256
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 51: 2nd Target TCP Port Field Definition

2nd Target Name

This 2nd target IP address is an optional address. This address is optional from several perspectives. First, it is used in cases where the 2nd target IP address is defined via static/parameter push approach or via F1 BIOS panels. Second, in cases where the static/parameter push approach or F1 BIOS panel approach is used, the 2nd target IP address can still be optional. In this second scenario, a NULL value here indicates that the initiator should establish a discovery session with the 2nd target and send a SENDTARGET command to obtain the storage topology visible to this initiator. Note that while the SENDTARGET response can present several IQNs, it is assumed that the initiator can determine the appropriate LUN to boot off of by using hashing or string matching algorithms. In the second scenario, a non-NULL value here indicates the specific IQN to be accessed during boot. Note that while IQNs can represent a target node fronting several LUNs, it is assumed here that a defined IQN represents a single LUN.

This 2nd target IQN string is an iSCSI RFC compliant string in terms of formatting that represents the iSCSI target. The iSCSI RFC specifies that a given IQN can be 223 bytes in length. However, due to space limitations in the static/parameter push mode and reasonably screen space on the BIOS/F1 panels, this architecture limits size of the string to 72 bytes. It is used during an iSCSI sessions as the transaction responder iSCSI identity. Only one IQN can be associated with a given iSCSI session while multiple initiator IP addresses may be defined for that given iSCSI session.

The IQN format for the name is a 72 character string where the 1st 4 characters are 'iqn.' while the remaining 68 characters are alphanumeric using UTF-8 encoding. The EUI format for the name is a 20 character string where the 1st 4 characters are 'eui.' while the remaining 16 characters represent 16 characters representing 16 hex characters.

Note that any spaces must be presented in the string as 0x20 and that if 72 valid characters are used, the length of the field (being 8 bytes) defines the end of the string (ie no /n or trailing 0x00s beyond the 72nd character).

Note, as part of the 4.10 and later revision of the specification, a construct has been added to provide the full 223 characters for the iSCSI name. In essence, several EPIDs have been designated to provide the remaining characters in the name such that least significant 72 characters are defined in the field and the most significant characters are defined in the appropriate EPID. These fields are concatentated together in the form of EPID(0x0B)||T2IQN in the case of the initiator name. Below illustrates several examples.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
T2IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0B) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI 2nd target name = EPID(0x0B)||T2IQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr099…chr000>” ie all 104 characters…”iqn.”+100 chars
T2IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0B) = 0x00…0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI 2nd target name = EPID(0x0B)||T2IQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	T2IQN[71:0][7:0]	Up to a 72 byte string (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2IQN[71:0][7:0]=0x00	2nd Target IQN is not valid
	T2IQN[71:0][7:0]!= 0x00	2nd Target IQN is valid
Default	T2IQN[71:0][7:0]=0x00	2nd Target IQN is not valid
Examples	T2IQN[71:0][7:0]=’iqn.date.blade1.tgt1’	IQN format – note 1st 4 chrs are iqn. (note string format)
	T2IQN[71:0][7:0]=’eui.F0E1D2C3B4A59678’	EUI format – note 1st 4 chrs are eui. (note string format0
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 52: 2nd Target IQN Field Definition

2nd Target Boot LUN

This boot disk LUN number is an optional parameter used in cases where the boot disk LUN number is defined via static/parameter push approach or via F1 BIOS panel.

For cases where 8 byte LUNs are required, EPID(0x08) provides the boot LUN definition in 8 byte formats.

The rules for how to interpret these fields are as follows. The least significant byte of the boot LUN is specified in this field and adheres to the boot LUN options bits in the OPT field. The full 8 byte format for the boot LUN is specified in EPID(0x08) and ignores the boot LUN options bits in the OPT field. If the initiator supports the boot LUN EPs, it checks for them first. If present, it uses them. If not present, it uses this field and assumes a 1 byte LUN is defined. Note for initiators that are unaware of or do not support the 8 byte LUN EPs, this field is used as governed by the option bits.

The configuration tool must ensure it picks the right least significant byte to place in this field

Below are 2 examples

Boot LUN = 0x000E000000000000…."0b00" format
EPID(0x08) = 0x000E000000000000
T2BL[7:0] = 0x0E

Boot LUN = 0x800000000000000E..."0b80" format
EPID(0x08) = 0x800000000000000E
T2BL[7:0] = 0x0E

Perspective	Definition	Description
Format	T2BL[7:0]	
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2BL[7:0]=0x00	2nd Target Boot LUN is 0
	T2BL[7:0]!=0x00	2nd Target Boot LUN is not 0
Default	T2BL[7:0]=0x00	2nd Target Boot LUN is 0
Example	T2BL[7:0]=0x04	2nd Target Boot LUN is 4
Producer	Wizard: Set value, wrs via OOB or DHCP	Wizard determines options, wrs to PSA or DHCP svr
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem or acquires from DHCP	Initiator to use determine locale (mem or DHCP)

Table 53: 2nd Target Boot LUN Field Definition

2nd Target CHAP ID

This 2nd target CHAP ID is an optional string. It is optional from several perspectives. First, it is used in cases where the 2nd target CHAP ID is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication.

This 2nd target CHAP ID in an IETF compliant ID used to aid in the authentication of the target to the initiator. Specifically, the 2nd target would encode this information during iSCSI session establishment and send it to the initiator where the initiator, running agreed to authentication algorithms, would validate the 1st target is actually valid. When the initiator and 2nd target CHAP ID/PW is used, it is in the context of mutual authentication.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	T2SDI[15:0][7:0]	16 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2SDI[15:0][7:0]=0x00	2nd Target security descriptor ID not valid
	T2SDI[15:0][7:0]!=0x00	2nd Target security descriptor ID valid
Default	T2SDI[15:0][7:0]=0x00	2nd Target security descriptor ID not valid
Example	T2SDI[15:0][7:0]='second_tgt_id'	Chap PW = "second_tgt_id"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, user input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 54: 2nd Target CHAP ID Field Definition

Note, when larger more complex security constructs are required, there are two possible solutions available to the solution. First solution is use the option bit in the options field 2nd Target Security Context to indicate that the given CHAP ID and PW fields be concatenated into one single field. For example, when the option is set for pre-share keys or X.509 certificates 2nd target CHAP ID field and 2nd target CHAP PW field are concatenated into on 2nd target security field. Refer to Table 55: 2nd Target Complex Security Field Definition for format details. When pre-shared key option is selected, the concatenated field represents the pre-shared key. When the X.509 certificate option is selected, the concatenated field represents a distinguished name (DN). Both are ASCII strings terminated either by 0x00 or by using all 32 bytes of the concatenated field. The second solution is using the extended parameter construct to specify a larger more complex security parameter.

Perspective	Definition	Description
Format	T2SDX[39:0][7:0]	T2SDI || T2SDP (note see option bit) (aka 40 bytes of space via 16||24)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2SDX[39:0][7:0]=0x00	2nd Target security descriptor not valid
	T2SDX[39:0][7:0]!=0x00	2nd Target security descriptor valid
Default	T2SDX[39:0][7:0]=0x00	2nd Target security descriptor not valid
Example	T2SDX[39:0][7:0]= 0xFEDCBA98 76543210 F0E1D2C3 B4A59687	32 byte security key Defined as pre-share key or distinguished name based on the option 1st Target Security Context settings. See **Error! Reference source not found.**.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 55: 2nd Target Complex Security Field Definition

2nd Target CHAP PW

This 2nd target CHAP PW is an optional string. It is optional from several perspectives. First, it is used in cases where the 2nd target CHAP PW is defined via static/parameter push approach or via F1 BIOS panel. Second, even if static/parameter push or F1 BIOS panel approach is employed, this parameter may NULL if there is no requirement for authentication.

This 2nd target CHAP PW in an IETF compliant PW used to aid in the authentication of the target to the initiator. Specifically, the 2nd target would encode this information during iSCSI session establishment and send it to the initiator where the initiator, running agreed to authentication algorithms, would validate the 2nd target is actually valid. When the initiator and target CHAP ID/PW is used, it is in the context of mutual authentication.

To enforce strong CHAP authentication, the password is 24 bytes in size which provides the minimum of 128 bits of randomness on the secrets. The format of the password is an ASCII field.

Please refer to 0 Out of Band Security Context for iSCSI Login for a detailed discussion on the use of this parameter.

Perspective	Definition	Description
Format	T2SDP[23:0][7:0]	24 bytes of ASCII characters (no /n)
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	T2SDP[23:0][7:0]=0x00	2nd Target security descriptor PW not valid
	T2SDP[23:0][7:0]!=0x00	2nd Target security descriptor PW valid
Default	T2SDP[23:0][7:0]=0x00	2nd Target security descriptor PW not valid
Example	T2SDP[23:0][7:0]='second_tgt_pw'	Chap PW = "second_tgt_pw"
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	MM/RSA: Set value, wrs via OOB to PSA	MM/RSA uses user input, wrs to PSA
	BIOS: If local, use input, else PSA to mem	Local input overrules remote input
Consumer	Initiator: Rds from mem for security	Initiator to use for security

Table 56: 2nd Target CHAP Password Field Definition

Note, when larger more complex security constructs are required, there two possible solutions available to the solution. First solution is use the option bit in the options field xxx to indicate that the given CHAP ID and PW fields be concatenated into one single field. For example, with the option bit set, initiator CHAP ID field and initiator CHAP PW field are concatenated into on initiator security field. Refer to Table 55: 2nd Target Complex Security Field Definition for format details. The second solution is using the extended parameter construct to specify a larger more complex security parameter.

Extended Parameters

In the advent there is a need for some additional parameters that do not warrant a change in the data structure version, the following format can be used. All remaining space in the 432 byte iSCSI data structure can be used for these additional parameters. Moreover, any additional parameters must be blade right after the defined parameters or all other additional parameters defined. I.e. it is not allowed to skip space and place an additional parameter at the bottom of the 432 byte data structure space.

Note that the count includes the ID and count fields so size of the actual value involved is count-4. To clarify this statement, consider this example. Suppose a given deployment has 3 extended parameters where the first

is defined to have a length of 23 bytes, the second a length of 47 bytes, and a third to have a length of 7 bytes. The actual space for values for are; first = 19 bytes, second = 43 bytes, third = 3 bytes. Moreover, assuming the addressing started 0x00000000 for the first extended parameter, the second extended parameter would start at 0x00000013(19d) and the third would start at 0x0000003F(63d).

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x00	Valid Extended parameter if LEN indicates valid EPIDs
	EPID[7:0]!=0x00	Valid Extended parameter if LEN indicates valid EPIDs
	EPCT[15:0]=0x0000	Invalid if EPID if LEN indicates EPIDs present
	EPCT[15:0]!=0x0000	Length of option identified by EPID (if LEN indicates EPIDs are present) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0] = 0x00	Defined per EPID
	EPOPT[7:0] != 0x00	Defined per EPID
	EPVAL[EPCT-5:0][7:0] != 0x00	Valid value for given EPID
Default	EPID[7:0] = 0x00	Valid Extended parameter if LEN indicates valid EPIDs
Example	EPID[7:0] = 0x48, EPCT[15:0]=0x000A EPOPT[7:0] = 0x55 EPVAL[5:0][7:0] = 0x112233445566	Extended parameter ID defined as 0x48, length is 10 bytes, options is 0x55, and Extended parameter value define as 112233445566
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
	Potentially from MM/RSA via OOB	Need to understand if needed
	Potentially from F1/BIOS Panels	Need to understand if needed
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 57: Extended Parameter Field Definition

Currently Defined Extended Parameters

The extended parameters are positioned to evolve over time. They represent the capability to add new parameters to the iSCSI parameter data structure as warranted.

Extended Parameter Naming Authority

The current version of this document presents the defined extended parameters that must be supported. Any new extended parameters must be folded into this document in a timely fashion. The process for adding new extended parameters involves vetting the proposed new extended parameters with affected groups and, once consensus is reached, the new extended parameter definition must be folded into this document. Note that all vetting must consider if there is name space available and, if not, proposals for migrating the version field to incorporate the more mature and stable extended parameters into the base data structure.

Null (EPID = 0x00)

The null extended parameter can be used to managed unused or reserve space on the parameter memory region.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x00	NULL Extended Parameter
	EPCT[15:0]!=0x05	Length of NULL EPVAL field EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Security masking of the this EPVAL and EPOPT field
	EPVAL[EPCT:0][7..0] = 0x00	Value for EPVAL for NULL Extended parameter
Example	EPID[7:0] = 0x00, EPCT[15:0]=0x0005 EPOPT[7:0]=0x00 EPVAL[EPCT-5:0][7:0] = 0x00	Extended parm ID defined as 00 (NULL), length for this example is 5 bytes (4 for ID, len, opt and 1 for value), the value for this example is 0x00.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 58: NULL Extended Parameter Definition

Import PKCS#12 Data (EPID = 0x01) - Reserved

The Import PKCS#12 Data extended parameter can be used to indicated to the initiator the following value should be treated as an IPsec security string. This extended parameter is used to only import the data for the current usage and not to be saved to persistent storage. Note that value is presented as an ASCII string that is 0x00 terminated.

This EPID is no longer in use. The field is reserved to prevent implementation confusion.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x01	Import PKCS#12 Extended Parameter
	EPCT[15:0]!=0x00	Length of PKCS#12 EPVAL field (max length is 1336 bytes) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment
	EPVAL[EPCT-5:0][7..0] = PW[31:0][7:0] || PKCS#12File[up to 1302][7:0]	Value for EPVAL for Import EPID PW is 32 byte ascii string (0x00 terminated) or 32 bytes when used in full PKCS#12 range from 0 to 1302 bytes (format is implementation specific)
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 59: Import PKCS#12 Extended Parameter Definition

Import and store PKCS#12 Data (EPID = 0x02) - Reserved

The Import PKCS#12 Data extended parameter can be used to indicated to the initiator the following value should be treated as an IPsec security string. This extended parameter is used to import the data for the current usage and to store it in persistent storage for future usage. Note that value is presented as an ASCII string that is 0x00 terminated.

This EPID is no longer in used. It is reserved to prevent implementation confusion.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x02	Import PKCS#12 Extended Parameter
	EPCT[15:0]!=0x00	Length of PKCS#12 EPVAL field (max length is 1336 bytes) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[EPCT-5:0][7..0] = PW[31:0][7:0] || PKCS#12File[up to 1302][7:0]	Value for EPVAL for Import/store EPID PW is 32 byte ascii string (0x00 terminated) or 32 bytes when used in full PKCS#12 range from 0 to 1302 bytes (format is implementation specific)
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 60: Import and Store PKCS#12 Extended Parameter Definition

1st Target IPsec Secret Data (EPID = 0x03)

The 1st Target IPsec Secret Data extended parameter can be used to pass the Initiator the IPsec secret when a CHAP and IPsec are mutually required. This secret represents either the pre-shared key or a distinguished name depending on the setting defined in Table 21: 1st Target Security Context Option Definition. Note that value is presented as an ASCII string that is 0x00 terminated.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x03	1st Target IPsec secret Extended Parameter
	EPCT[15:0]!=0x00	Length of IPsec secret EPVAL field (max length is 36 bytes) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[EPCT-5:0][7..0] = PW[31:0][7:0]	Value for EPVAL for 1st Target IPsec secret Secret is 32 byte ASCII string (0x00 terminated) or 32 bytes when used in full
Example	EPID[7:0] = 0x03 EPCT[15:0] = 0x24 EPOPT[7:0] = 0x00 EPVAL[31:0][7:0] = “abcdefghijklmnopqrstuvwxyz12345”	How to use the EPID to define the IPSEC secret
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 61 1st Target IPsec Secret Data Extended Parameter Definition

2nd Target IPsec Secret Data (EPID = 0x04)

The 2nd Target IPsec Secret Data extended parameter can be used to pass to the Initiator the IPsec secret when a CHAP and IPsec are mutually required. This secret represents either the pre-shared key or a distinguished name depending on the setting defined in Table 23 2nd Target Security Context Option Definition. Note that value is presented as an ASCII string that is 0x00 terminated.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x04	2nd Target IPsec secret Extended Parameter
	EPCT[15:0]!=0x00	Length of IPsec secret EPVAL field (max length is 36 bytes) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[EPCT-5:0][7..0] = PW[31:0][7:0]	Value for EPVAL for 2nd Target IPsec secret Secret is 32 byte ASCII string (0x00 terminated) or 32 bytes when used in full
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 62 2nd Target IPsec Secret Data Extended Parameter Definition

IPV6 Global Address Bytes (EPID = 0x05)

The upper 8 bytes of required for full IPV6 support are defined in this EPID. Each IP address cited in the basic data structure has its upper 8 bytes defined here. These bytes are only required if the IPV6 addresses to be used are to define point beyond the local IPV6 network segment. If IPV6 addresses are to be used in the local network segment, then the canned FED0:0000:0000:000 is to be used. As a result, this EPID is not strictly dependent on the IPV4 / IPV6 option bit since IPV6 on a local network segment does not require these parameters.

The IPv6 prefix are ascii strings in order to make the processing of them simpler. Note that if any field is less than the full string value, such as subnet, that is permissible since the next field is demarcated by the 4 character prefix such as “TGT1”.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x05	2nd Target IPsec secret Extended Parameter
	EPCT[15:0]=0x94 (148d)	Length of IPsec secret EPVAL field (max length is 148 bytes) EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[EPCT-5:EPCT-28][7..0]= "DISC:DDDD:DDDD:DDDD:DDDD"	Upper 8 bytes for discovery address
	EPVAL[EPCT-29:EPCT-52][7:0] = "GATE:GGGG:GGGG:GGGG:GGGG"	Upper 8 bytes for gateway address
	EPVAL[EPCT-53:EPCT-76][7:0]= "SUBN:SSSS:SSSS:SSSS:SSSS"	Upper 8 bytes for Subnet address
	EPVAL[EPCT-77:EPCT-100][7..0]= "INIT:IIII:IIII:IIII:IIII"	Upper 8 bytes for Initiator address
	EPVAL[EPCT-101:EPCT-124][7:0] = "TGT1:T1T1:T1T1:T1T1:T1T1"	Upper 8 bytes for Target 1 (primary) address
	EPVAL[EPCT-125:EPCT-144][7:0]= "TGT2:T2T2:T2T2:T2T2:T2T2"	Upper 8 bytes for Target 2 (alternate) address
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 63 2nd Target IPsec Secret Data Extended Parameter Definition

Disk Image Identification (EPID = 0x06)

The Disk Image Identification can be used to identify and verify the disk image being accessed. There is no assumption about the format of the value or how a given component (BIOS, initiator, configuration or deployment tool) uses the value specified here.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x06	Disk image indentifier
	EPCT[15:0]!=0x00	Length of Disk Image Indentifier. It expectation is that the length of this EPVAL is less than 259 bytes EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPOPT[7:0]=0b1xxxxxxx	Security masking of the this EPVAL and EPOPT field
	EPVAL[EPCT:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x06, EPCT[15:0]=0x15 EPOPT[7:0]=0x00 EPVAL[EPCT-5:0][7:0] = “my disk image” Or EPVAL[EPCT-5:0][7:0] = 0x123456789abc	Extended parm ID defined as 06 (Image ID), length for this example is 21 bytes, there are no options, the value for this example can be an ascii string or hex value. The user defines the format.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 64: Disk Image Identification Extended Parameter Identification

1st Tgt Full LUN Definition (EPID = 0x07)

In special cases or advanced implementations, the full LUN definition may be desirable. The full 16 byte value, in hex, of the LUN identifier to use for booting that resides behind the 1st target (T1IP, T1IQN, etc).

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x07	Disk image indentifier
	EPCT[15:0]=0x14	Length of 1st Tgt full LUN definition. It expectation is that the length of this EPVAL is always 20 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[15:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x07 EPCT[15:0]=0x14 EPOPT[7:0]=0x00 EPVAL[15:0][7:0] = 0x0001000000000000	Extended parm ID defined as 07 (Tgt 1 Full LUN), length for this example is 20 bytes, there are no options, the value for this example are hex value.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 65: 1st Target Full LUN Definition Extended Parameter Identification

2nd Tgt Full LUN Definition (EPID = 0x08)

In special cases or advanced implementations, the full LUN definition may be desirable. The full 8 byte value, in hex, of the LUN identifier to use for booting that resides behind the 1st target (T1IP, T1IQN, etc).

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x08	Disk image indentifier
	EPCT[15:0]=0x14	Length of 2nd Tgt full LUN definition. It expectation is that the length of this EPVAL is always 20 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[15:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x08 EPCT[15:0]=0x14 EPOPT[7:0]=0x00 EPVAL[15:0][7:0] = 0x0001000000000000	Extended parm ID defined as 08 (Tgt 2 Full LUN), length for this example is 20 bytes, there are no options, the value for this example aare hex value.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 66: 2nd Target Full LUN Definition Extended Parameter Identification

Initiator Extended iSCSI Name (EPID = 0x09)

In special cases or advanced implementations, where the full 223 character iSCSI name is required, this EPID provides most significant 151 characters of the full name for the initiator.

Recall, the format for usage examples of this field is as follows.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
IIQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x09) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x09)||IIQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr099…chr000>” ie all 104 characters…”iqn.”+100 chars
IIQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x09) = 0x00…..0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x09)||IIQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x09	iSCSI initiator extended name
	EPCT[15:0]=0x9B	Length of iSCSI extended initiator name definition. It expectation is that the length of this EPVAL is always 155 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[96:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x09 EPCT[15:0]=0x9B EPOPT[7:0]=0x00 EPVAL[96:0][7:0] = 0x0001000000000000	Extended parm ID defined as 09, length for this example is 155 bytes, there are no options, the value for this example are hex value.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 67: ISCSI Initiator Extended Name

1st Tgt Extended iSCSI Name (EPID = 0x0A)

In special cases or advanced implementations, where the full 223 character iSCSI name is required, this EPID provides most significant 151 characters of the full name for the 1st target.

Recall, the format for usage examples of this field is as follows.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
T1IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0A) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x0A)||T1IQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr100…chr000>” ie all 104 characters…”iqn.”+100 chars
T1IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0A) = 0x00…..0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x0A)||T1IQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x0A	iSCSI initiator extended name
	EPCT[15:0]=0x9B	Length of iSCSI extended initiator name definition. It expectation is that the length of this EPVAL is always 155 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[96:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x0A EPCT[15:0]=0x9B EPOPT[7:0]=0x00 EPVAL[96:0][7:0] = 0x0001000000000000	Extended parm ID defined as 0A, length for this example is 155 bytes, there are no options, the value for this example are hex value.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 68: ISCSI 1st Target Extended Name

2nd Tgt Extended iSCSI Name (EPID = 0x0B)

In special cases or advanced implementations, where the full 223 character iSCSI name is required, this EPID provides most significant 151 characters of the full name for the 2nd target.

Recall, the format for usage examples of this field is as follows.

Example 1:
Iscsi name = “iqn.<chr219…chr000>” ie all 223 characters…”iqn.”+219 chars
T2IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0B) = “iqn.<chr219…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x0B)||T2IQN
= “iqn.<chr219…chr072>||<chr071…chr000>”
= “iqn.<chr219…chr000>” ie all characters

Example 2:
Iscsi name = “iqn.<chr099…chr000>” ie all 104 characters…”iqn.”+100 chars
T2IQN = “<chr071…chr000> ie least 72 characters of name
EPID(0x0B) = 0x00…..0x00||“iqn.<chr099…chr072> ie the most significant 151 characters
So
iSCSI Initiator name = EPID(0x0B)||T2IQN
= “iqn.<chr099…chr072>||<chr071…chr000>”
= “iqn.<chr099…chr000>” ie all 104 characters

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x0B	iSCSI initiator extended name
	EPCT[15:0]=0x9B	Length of iSCSI extended initiator name definition. It expectation is that the length of this EPVAL is always 155 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0bxxxxxxxx	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPVAL[96:0][7..0] = 0x00	Value for EPVAL for disk image Extended parameter
Example	EPID[7:0] = 0x0B EPCT[15:0]=0x9B EPOPT[7:0]=0x00 EPVAL[96:0][7:0] = 0x0001000000000000	Extended parm ID defined as 0B, length for this example is 155 bytes, there are no options, the value for this example are hex value.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 69: ISCSI 2nd Target Extended Name

Initiator Extended Secrets (EPID = 0x0C)

In special cases or advanced implementations, where the initiator would like 1 or 2 larger secrets, this EPID provides 2 64 byte secrets that can be used separately with each target. These extended secrets are

to be presented in ASCII hex and are left justified and either terminated by a 0x00 (NULL) or 64 characters in length.

Perspective	Definition	Description
Format	EPID[7:0]	Extended parameter identifier
	EPCT[15:0]	Extended parameter length
	EPOPT[7:0]	Extended parameter option
	EPVAL[EPCT:0][7:0]	Extended parameter value
DS Version	See Section 3	Applicable versions
DS Level	See Section 3	Applicable levels
Values	EPID[7:0]=0x0C	iSCSI initiator extended name
	EPCT[15:0]=0x84	Length of iSCSI extended initiator name definition. It expectation is that the length of this EPVAL is always 132 bytes, EPCT = EPID+EPCT+EPOPT+EPVAL
	EPOPT[7:0]=0b00000000	Options defined for this EPID. The definition of the option bits are beyond the context of parameter deployment.
	EPOPT[7:0]=0b00000001	Extended secret to be used with 1st target is present
	EPOPT[7:0]=0b00000010	Extended secret to be used with 2nd target is present
	EPOPT[7:0]=0b00000011	Both extended secret to be used with respective targets are present
	EPVAL[127:0][7..0] = IESC2[63..0][7:0] || IESC1[63..0][7:0]	Extended secret values as mapped into EPID
Example	EPID[7:0] = 0x0C EPCT[15:0]=0x84 EPOPT[7:0]=0x03 EPVAL[127:0][7:0] = Secret2 || secret1	Extended parm ID defined as 0C. length for this example is 132 bytes, option bits indicate presences of secrets, the value is the concatenation of secret 2 || secret 1.
Producer	Wizard: Set value, wrs via OOB	Wizard determines values, wrs to PSA
Consumer	Initiator: Rds from mem to determine parsing	Initiator to use for any extended parameters

Table 70: Initiator Extended Secrets

Chapter 3. iSCSI Configuration Parameter Evolution

The evolution of these iSCSI configuration parameters is driven by several influences including industry directions, implementation evolution, and a desire to prudently evolve the parameter set. As a result, the certain parameters delivered out of band via BMC/BIOS and inband via DHCP will evolve over time.

In the Appendix A, a detailed discussion of the specifics of evolution is presented as an aid in understanding the goals and context of the changes. Please refer to Appendix A for more details.

Chapter 4. iSCSI Parameter Inclusion Matrix

The following subsections present the iSCSI parameter usage versus the data structure version and level are presented. The character "X" signifies the parameter is present for the given version and level. The character " " signifies the parameter is absent for the given version and level. The character "C" signifies the parameter definition has changed in either the field size or the bit definitions in the field.

iSCSI Base Parameter Inclusion Matrix

The usage of the data structure base parameters versus the data structure version and level are presented.

Parameter	**ID**	**Val**	**Val**	**Val**	**Val**	**Val**	**Val**	**Val**	**Val**	**Val**
Data Structure Version	VER[7:0]	1	1	2	3	?				
Data Structure Level	LVL]7:0]	1	2	1	1	?				
Data Structure Length	LEN[15:0]	X	X	X	X	X				
Checksum	CHK[7:0]	X	X	X	X	X				
Static/Dynamic	S_D[7:0]	X	X	X	X	X				
Options (see below)	OPT[31:0]									
Retry Count	RTRY[7:0]	X	X	X	X	X				
Timeout Value	TOVAL[7:0]	X	X	X	X	X				
Scope/Vendor ID	SVID[63:0]	X	X	X	X	X				
Client Alt ID	CAID[63:0]	X	X	X	X	X				
Discovery IP address	DIP[63:0]	X	X	X	X	X				
Initiator IP address	IIP[63:0]	X	X	X	X	X				
Initiator name	IIQN[71:0][7:0]	X	X	X	X	X				
Initiator CHAP id	ISDI[15:0][7:0]	X	X	X	X	X				
Initiator CHAP pw	ISDP[23:0][7:0]	X	X	C	X	X				
Subnet Mask	MSK[63:0]	X	X	X	X	X				
VLAN Tag	VLAN[16:0]			X	X	X				
Gateway IP address	GRIP[63:0]	X	X	X	X	X				
1st Target IP address	T1IP[63:0]	X	X	X	X	X				
1st Target TCP port	T1PT[15:0]	X	X	X	X	X				
1st Target name	T1IQN[71:0][7:0]	X	X	X	X	X				
1st Target Boot LUN	T1BL[7:0]	X	X	X	X	X				
1st Target CHAP id	T1SDI[15:0][7:0]	X	X	X	X	X				
1st Target CHAP pw	T1SDP[23:0][7:0]	X	X	C	X	X				
2nd Target IP address	T2IP[63:0]	X	X	X	X	X				
2nd Target TCP port	T2PT[15:0]	X	X	X	X	X				
2nd Target name	T2IQN[71:0][7:0]	X	X	X	X	X				
2nd Target Boot LUN	T2BL[7:0]	X	X	X	X	X				
2nd Target CHAP id	T2SDI[15:0][7:0]	X	X	X	X	X				
2nd Target CHAP pw	T2SDP[23:0][7:0]	X	X	C	X	X				
Extended Parms (see below)										

Table 71: iSCSI Parameter Base Data Structure Usage

Base Option Parameter Inclusion Matrix

The options field in the base datastructure provides several options used by the initiator to guide in the acquisition of the iSCSI parameters. The option usage versus data structure version and level is presented below.

Parameter	ID	Val	Val	Val	Val	Val	Val	Val	Val	Val	Val
Data Structure Version	VER[7:0]	1	1	2	3	?					
Data Structure Level	LVL]7:0]	1	2	1	1	?					
IP address format	OPT[31]	X	X	X	X	X					
DHCP format	OPT[30]	X	X	X	X	X					
Boot LUN usage	OPT[29]	X	X	X	X	X					
Security Persistence	OPT[28]	X	X	X	X	X					
1st Target security context	OPT[27:24]	X	X	X	X	C					
1st Target security transport	OPT[23:22]	X	X	X	X						
2nd Target security context	OPT[21:18]	X	X	X	X	C					
2nd Target security transport	OPT[17:16]	X	X	X	X						
Target presence	OPT[15:14]	X	X	X	X	X					
Alt client ID usage for IP	OPT[13:12]	X	X	X	X	X					
Alt client ID usage for iSCSI	OPT[11:10]	X	X	X	X	X					
Discovery Address Usage	OPT[9:8]		X	X	X	X					
DHCP Option 60 format	OPT[7]				X	X					

Table 72: iSCSI Parameter Option Usage Matrix

iSCSI Extended Parameter Inclusion Matrix

The extended parameters used to augment the base data structure. While not strictly tied to the base data structure version or level, the following matrix shows where they surfaced in the evolution of the iSCSI parameters. Note that since each extended parameter is defined by a parameter ID, any extended parameter can be concatenated to the end of any basic parameter data structure.

Parameter	ID	Val	Val	Val	Val	Val	Val	Val	Val	Val
Version	VER[7:0]	1	1	2	3	?				
Level	LVL]7:0]	1	2	1	1	?				
NULL	EPID[7:0]=0x00	X	X	X						
Import PKCS#12	EPID[7:0]=0x01	X	X	X						
Import/store PKCS#12	EPID[7:0]=0x02	X	X	X	C	X				
1st Target IPSEC secret	EPID[7:0]=0x03	X	X	X	C	X				
2nd Target IPSEC secret	EPID[7:0]=0x04	X	X	X	C	X				
IPv6 global addr info	EPID[7:0]=0x05	X	X	X	C	X				
Disk Image Identification	EPID[7:0]=0x06			X	C	X				
Tgt1 Full LUN Definition	EPID[7:0]=0x07				X	X				
Tgt2 Full LUN Definition	EPID[7:0]=0x08				X	X				
Initiator extended name	EPID[7:0]=0x09					X				
1st Target extended name	EPID[7:0]=0x0A					X				
2nd Target extended name	EPID[7:0]=0x0B					X				
Initiator extended secrets	EPID[7:0]=0x0C					X				

Table 73: iSCSI Extended Parameter Usage Matrix

Chapter 5. iSCSI Parameter Precedence

Given that the definition and deployment of iSCSI boot parameters can be done at several levels, some rules of precedence must be defined. This section defines those rules of precedence and the resulting implications.

iSCSI Parameter Deployment Options and the Problem

The deployment of iSCSI parameters can take place at several levels in the architecture. At the lowest level, an iSCSI hardware initiator may have the ability to store iSCSI boot parameters locally in some non-volatile memory on the actual adapter. The next level up, the BIOS or pre-OS environment may have the ability to storage iSCSI boot parameters in some non-volatile memory on the actual server. The next level up, the system management entity such as the management module or RSAII can store iSCSI boot parameters in some non-volatile memory on the system mgmt entity. At the highest level, a deployment tool, running on some other server node can store the iSCSI boot parameters for many servers in its own non-volatile memory.

Moreover, if configured appropriate, the iSCSI initiator whether hardware or software based can access a DHCP server to retrieve iSCSI boot parameters from network services such as DHCP or SLP

The main problem with this arrangement is that parameters defined and stored at a low level may be used incorrectly depending on the behavior in the higher levels of the architecture. For example, if a iSCSI hardware initiator stores iSCSI boot parameters in its local non-volatile memory, how does it know, when the configuration changes, when to accept new parameters from upper levels or when to acquire iSCSI boot parameters from network services.

So, the precedence and behavior of iSCSI boot parameter management must be defined.

iSCSI Parameter Mgmt at the Initiator level

For any iSCSI initiator that can store parameters locally in some private non-volatile memory such as EEPROM on an adapter, there are a several capabilities that must be present to support the iSCSI boot architecture.

Capability	Required/Optional	Description
User/Admin Control	Required	Modes of defining and accepting iSCSI boot parameters must be visible and selectable by the user and admin when configuring the iSCSI inititor.
Manual Mode Selection	Optional	Manual mode selection, if supported, defines that the user at the local server wishes to configure the iSCSI parameters manually by entering in the iSCSI boot parameter fields locally through manual text input. It is up to the initiator vendor identify if their initiator will do any parameter consistency checks to ensure proper format or value. If manual mode configuration is re-entered at a later date, the values defined previous should be presented. When parameters are saved, a confirmation to the user/admin should be presented.
Manual Mode Deselection	Optional	Manual mode deselection must be supported if manual mode configuration is supported on the system. This requirement is needed to ensure that higher levels of parameter mgmt can be used at a later date. Upon deselection, a message should be presented to the user/admin confirming change in methodology and any implications, such as local values lost, that will be incurred.
DHCP Acquire Selection	Optional	DHCP Acquire selection, if supported, defines that the user/admin wants the initiator to acquire **ALL** iSCSI parameters from a DHCP server. To make this mode work, a DHCP server configured to provide the vendor options or root path option defined in 0 Chapter 7. iSCSI Initiator Config Parameters Acquisition From DHCP be configured in a DHCP server visible to the iSCSI boot initiator DHCP client. Note that DHCP Acquire Selection specifies the iSCSI initiator will, on its own actions, reach out to the DHCP server. It does NOT cover the scenario where upper levels of parameter mgmt have specified that the initiator should acquire iSCSI parameters from DHCP.
DHCP Acquire Deselection	Optional	DHCP Acquire deselection must be supported if DHCP acquire selection is supported. This requirement is needed to ensure that higher levels of parameter mgmt can be used at a later date. Upon deselection, a message should be presented to the user/admin confirming change in methodology and any implications, such any local values lost, that will be incurred.
BIOS Accept Selection	Required	BIOS accept selection must be supported to be compliant with the iSCSI boot architecture. This iSCSI parameter mgmt methodology indicates that the initiator should accept an iSCSI parameter block from system BIOS. Note that within this iSCSI block coming from system BIOS, there is the ability to acquire all or some of the iSCSI parameters from DHCP. As a result, the iSCSI parameter block can direct the initiator to acquire some or all of the iSCSI parameters from a DHCP server using the format defined in 0 Chapter 7. iSCSI Initiator Config Parameters Acquisition From DHCP.

Capability	Required/Optional	Description
BIOS Accept Deselection	Required	BIOS accept deselection must be supported in order to disengaged the BIOS accept iSCSI parameter methodology based on specific circumstances. For example, during problem determination, one may want to do some specific tests at the initiator itself.

Table 74: iSCSI Parm Mgmt at Initiator level

Below in Table 75: iSCSI Parameter Mgmt Precedence at Initiator level, the precedence of parameters are defined. Basically, if the iSCSI initiator is used for booting, then BIOS accept takes highest precedence while DHCP acquire (not specified by BIOS parameters) is next, and, finally, manual mode is specified. Note that a given iSCSI initiator vendor can implement this precedence as desired (negative active logic for example), but the precedence must be adhered to.

Configuration	Boot	BIOS	DHCP	Manual	Description
Non boot	0	x	x	x	iSCSI is not used for booting…so others are don't cares. Any values in BIOS/DHCP/Manual are meaningless
Boot using BIOS Accept	1	1	x	x	iSCSI initiator is used for booting. The BIOS mode is used. In this mode, the BIOS may indicate to the initiator to acquire some or all parameters from DHCP. Any values in DHCP/Manual fields are ignored.
Boot using DHCP	1	0	1	x	iSCSI initiator is used for booting. The BIOS mode is not used and ALL DHCP parameters are acquired from the DHCP server. Aka, the BIOS did NOT tell the initiator to acquire parameters from DHCP. Note it is up to the initiator vendor to decide if more granularity is needed in the DHCP configuration aka if IP or iSCSI or both parameters should be acquired.
Boot using Manual	1	0	0	1	iSCSI initiator is used for booting. The manual mode of configuration is used.

Table 75: iSCSI Parameter Mgmt Precedence at Initiator level

iSCSI Parameter Mgmt at the Server Level

For a given server, the iSCSI parameter configuration mgmt can be accomplished by two methodologies; manual and BIOS accept. Below in these methodologies are presented

Capability	Required/Optional	Description
User/Admin Control	Required	Modes of defining and accepting iSCSI boot parameters must be visible and selectable by the user and admin when configuring the BIOS to use iSCSI as a booting resource.
Manual Mode Selection	Optional	Manual mode selection, if supported, defines that the user at the local server wishes to configure the iSCSI parameters manually by entering in the iSCSI boot parameter fields locally through manual text input. The system BIOS updates the local server non volalile storage through commands to the service processor residing on the server. The system BIOS can optionally ensure consistency (format and value). If manual mode configuration is re-entered at a later date, the values defined previous should be presented. When parameters are saved, a confirmation to the user/admin should be presented.
Manual Mode Deselection	Optional	Manual mode deselection must be supported if manual mode configuration is supported on the system. This requirement is needed to ensure that higher levels of parameter mgmt can be used at a later date. Upon deselection, a message should be presented to the user/admin confirming change in methodology and any implications, such as local values lost, that will be incurred.
BIOS Accept Selection	Required	BIOS accept selection must be supported to be compliant with the iSCSI boot architecture. This iSCSI parameter mgmt methodology indicates that the initiator should accept an iSCSI parameter block from system BIOS. Note that within this iSCSI block coming from system BIOS, there is the ability to acquire all or some of the iSCSI parameters from DHCP. As a result, the iSCSI parameter block can direct the initiator to acquire some or all of the iSCSI parameters from a DHCP server using the format defined in 0 Chapter 7. iSCSI Initiator Config Parameters Acquisition From DHCP.
BIOS Accept Deselection	Required	BIOS accept deselection must be supported in order to disengaged the BIOS accept iSCSI parameter methodology based on specific circumstances. For example, during problem determination, one may want to do some specific tests at the initiator itself.

Table 76: iSCSI Parameter Mgmt at the Server Level

Below in Table 75: iSCSI Parameter Mgmt Precedence at Initiator level, the precedence of parameters are defined. Basically, if the iSCSI initiator is used for booting, then BIOS accept takes highest precedence while DHCP acquire (not specified by BIOS parameters) is next, and, finally, manual mode is specified. Note that a given iSCSI initiator vendor can implement this precedence as desired (negative active logic for example), but the precedence must be adhered to.

Configuration	Boot	BIOS	Manual	Description
Non boot	0	x	x	iSCSI is not used for booting…so others are don't cares. Any values in BIOS/DHCP/Manual are meaningless
Boot using BIOS Accept	1	1	x	iSCSI initiator is used for booting. The BIOS mode is used. In this mode, the BIOS may indicate to the initiator to acquire some or all parameters from DHCP. Any values in DHCP/Manual fields are ignored.
Boot using Manual	1	0	1	iSCSI initiator is used for booting. The manual mode of configuration is used. The given server should save the parameters in local non-volatile storage accessible through the service processor.

Table 77: iSCSI Parameter Mgmt Precedence at Server Level

iSCSI Parameter Mgmt at the Mgmt Entity Level

The management of iSCSI parameters at the mgmt entity level is similar to the server level. If parameters can be stored on a management entity such as the management module or RSAII, the both the BIOS accept and manual modes are supported and have the same precedence as at the server level.

As the iSCSI parameters mgmt move higher up in the mgmt hierarchy, it is the responsible of the user/admin to note that current parameters are changed at the given level but that the changes are not guaranteed to be fed up to higher levels.

iSCSI Parameter Mgmt at the Deployment Wizard Level

The deployment wizard is the ultimate paradigm to manage the iSCSI parameters. As a result, the deployment wizard should attempt to retrieve the current iSCSI parameters from the lowest levels possible to ensure an accurate view of the ecosystem. Specifically, the deployment wizard should access down to the server level to ensure that the iSCSI parameters passed ot a given initiator is accurate. It may not be possible to actually retrieve parameters defined within a given initiator. However, this whole is covered by the initiator level settings and precedence, so a correctly configured initiator will always have the current and correct iSCSI parameters.

Chapter 6. iSCSI Initiator Config Parameters Acceptance from BIOS

In the following sections, the actual network flows for several approaches to accepting iSCSI initiator configuration parameters from BIOS is discussed.

iSCSI Initiator Parameters in Initiator Memory Space

When the iSCSI parameters are to be processed and transferred via out of band paths, the parameters can occupy a region of space defined by server BIOS. The region can range from 432 bytes to 1728 bytes in size. The 432 byte BIOS data structure page provides the ability to manage 4 instances of the iSCSI base parameters. With some of the advanced constructs defined in the "ESW iSCSI Boot Initiator Support Design" authored by Scott Dunham, several of thes 432 byte pages can be assembled to represent a large data structure space. The specifics of the data structure constructs are beyond the scope of this document. Please refer to the "ESW" document for more details.

Acceptance from BIOS Decision Algorithm

Below is an updated version of the algorithm decision flow for accepting iSCSI parameters from BIOS services.

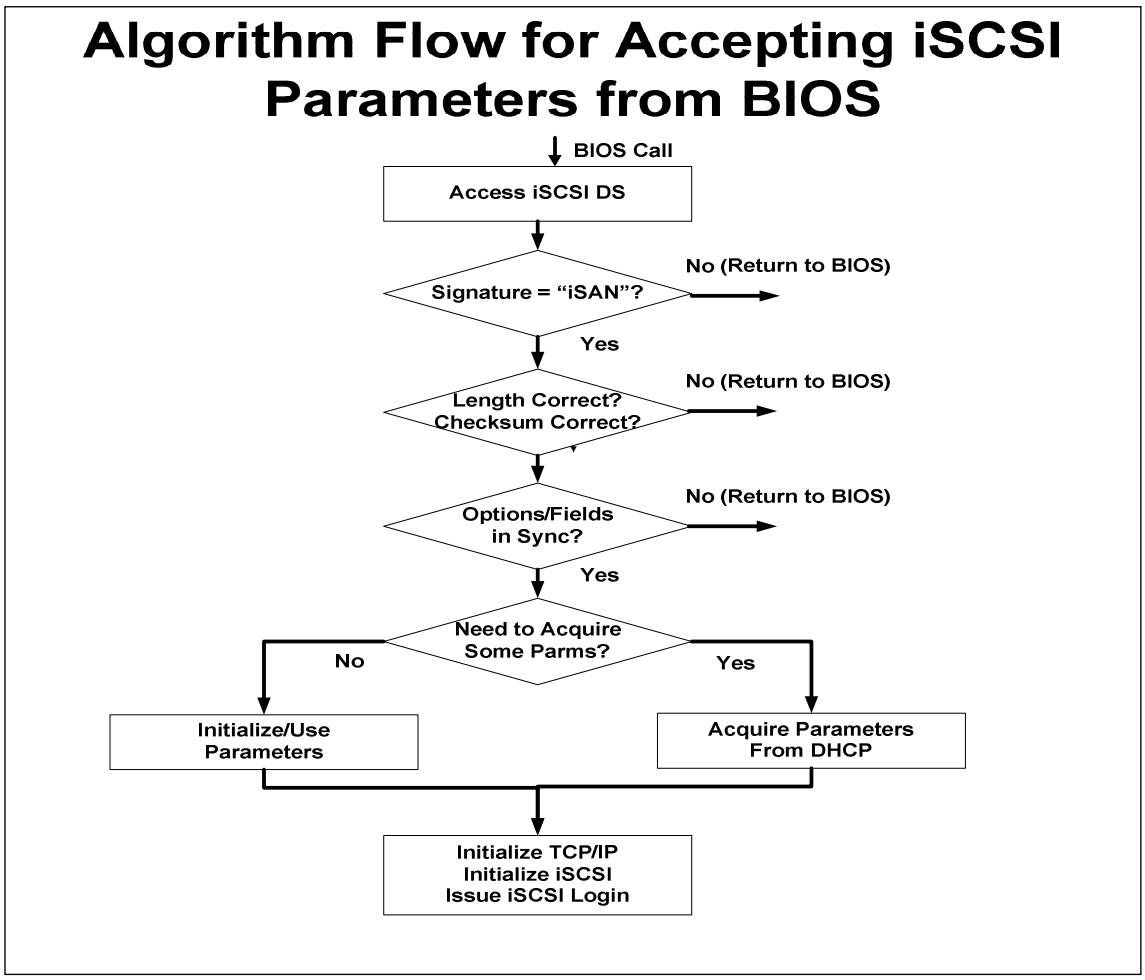

Figure 1: iSCSI Parameter Accept Algorithm Flow

Acceptance from BIOS Flows

This section outlines the BIOS⇔iSCSI boot initiator transactions for accepting all or part of the iSCSI configuration parameters from BIOS

Acceptance Flow 1: Full iSCSI DS from BIOS

In acceptance flow 1, all the iSCSI parameters are passed from BIOS to the initiator.

Acceptance Flow 1: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to use all iSCSI parameters from BIOS (part of the iSCSI DS)
Any security parameters are deployed out of band.
No Client_Alt ID or discovery IP address has been deployed apriori out of band.
No Client_Alt ID or discovery IP address is acquired from DHCP server via Option 204

Acceptance Flow 1: Network Flow

Below is an update network flow sequence from power on to iSCSI Login. This flow is focuses on accepting all the iSCSI parameters from BIOS. Note that BIOS retrieves these parameters from VPD space where deployment tools have placed the data structure originally.

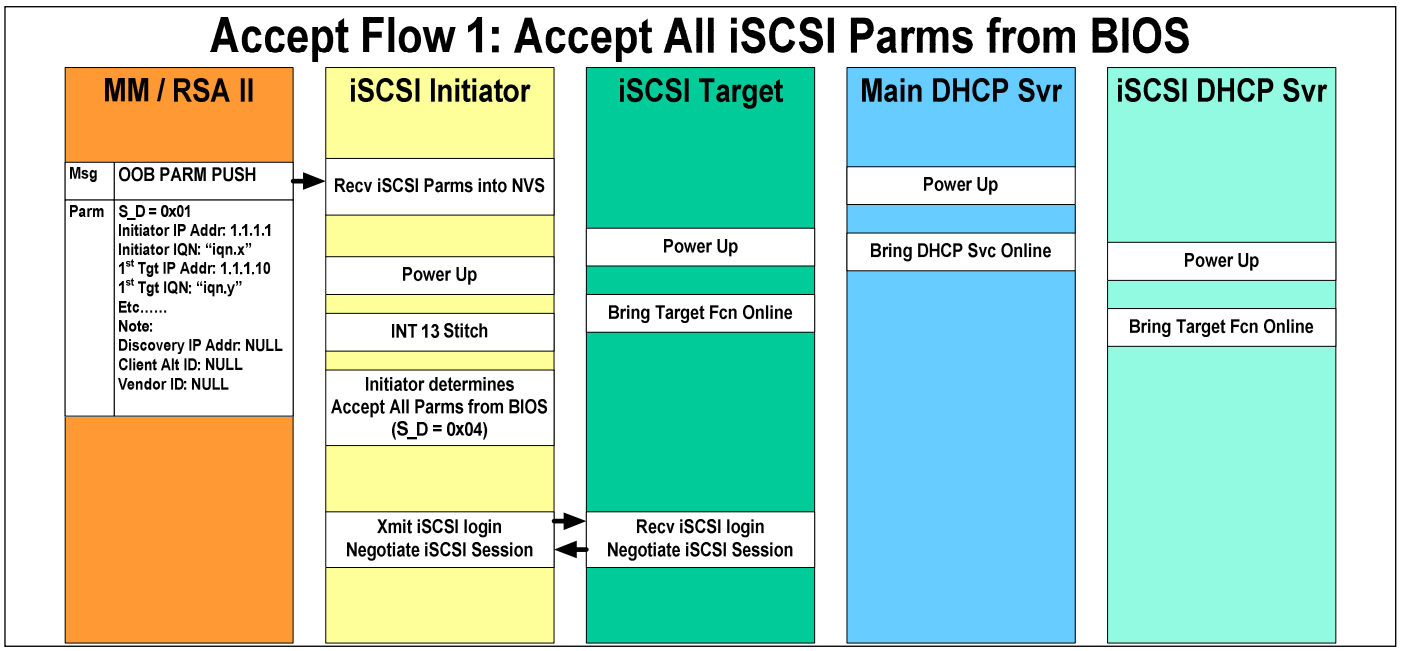

Figure 2: Acceptance Flow 1: Accepting All the Parameters from BIOS

Acceptance Flow 2: Accepting Some Parameters from BIOS

This flow is more fully covered in 0.

Chapter 7. iSCSI Initiator Config Parameters Acquisition From DHCP

In the following sections, the actual network flows for several approaches to acquiring iSCSI Initiator Parameters from DHCP is discussed.

DHCP Overview

Dynamic Host Configuration Protocol (DHCP) provides a means for a DHCP client such as a server to be configured at boot time be "reaching" out a DHCP server acquire the necessary parameters. In turn, the DHCP server provides those configuration parameters to the DHCP client. The iSCSI solution utilizes this protocol to acquire iSCSI parameters at boot time so that a given server can acquire the necessary iSCSI parameters to establish an iSCSI session with the appropriate target.

Examining the DHCP protocol in more detail, there are several phases to the protocol. First, the discovery phase involves the DHCP client broadcasting to the entire network that its looking for some configuration parameters. The second phase involves all the DHCP servers offering to help configuration the DHCP client. The third phase involves the DHCP client selecting the DHCP server it would like to acquire the configuration parameters from. The fourth phase involves the selected DHCP server responding with the configuration parameters that enable the DHCP client to configure itself.

An extension of the DHCP basic flow involves the DHCP client to have some of the configuration information on hand and, thus, not need to search for and select a DHCP server to use. In this extension, the DHCP client can ask the selected DHCP server to inform it of new (non IP related) configuration information.

Both the basic DHCP protocol and the inform extension is leveraged in the iSCSI deployment to provide a high degree of flexibility for the customer.

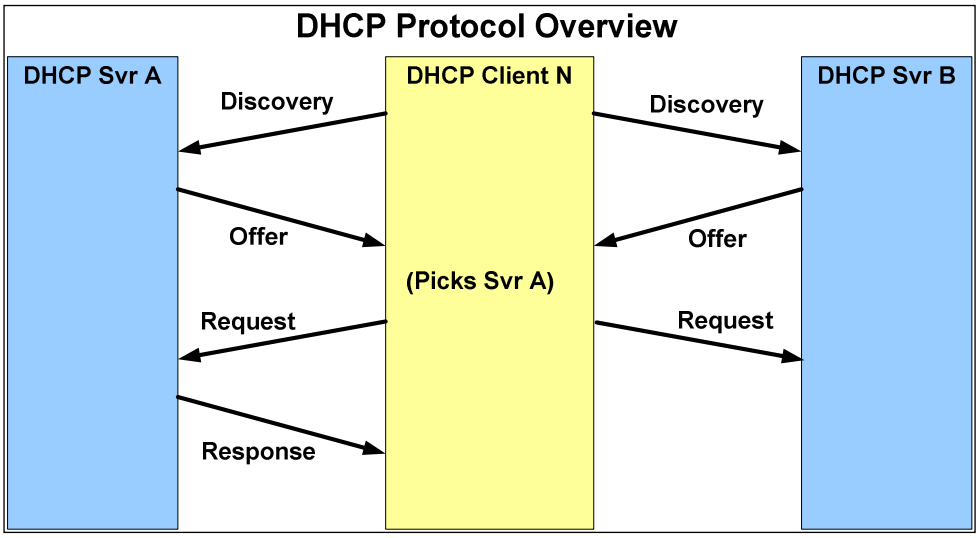

Figure 3: DHCP Introduction

DHCP Transaction Phases

The phases and roles of DHCP transactions are defined below.

DHCP Phase	Definition	Description
DHCPDISCOVER	Client discovering Server	This message sent from the DHCP client to all stations on the subnet is used to discover the DHCP servers in the subnet. A broadcast packet containing the basic identity info (no IP addresses) is sent with options.
DHCPOFFER	Server offer to Client	This message is sent from the DHCP server to all stations on the subnet is used to offer configuration information to whoever issued the DHCPDISCOVER. Since the DHCP server does not know who actually sent the DHCPDISCOVER, a broadcast packet with IP information is sent along with pertinent option information..
DHCPREQUEST	Client requesting Server	This message is sent from the DHCP client to the DHCP server indicating among other things that the DHCP client has selected this DHCP server as the server to provide any additional configuration information. The DHCPREQUEST message is a broadcast packet used to inform all other DHCP servers in the subnet the Server⇔Client relationship. Pertinent configuration options are included.
DHCPACK	Server acknowledges Client	This message is sent from the DHCP server to the DHCP client informing the client the server is aware of the Client⇔server relationship and provides any additional configuration information. This is a unicast packet between the DHCP client and DHCP Server.
DHCPINFORM	Client inquires Server	This message is sent from the DHCP Client to the DHCP server to inquire information updates such as lease extensions. This message is a unicast packet since the Client already owns an IP address.

Table 78: DHCP Message Phases

DHCP Transaction Implementation Aspects

Each of the 3 basic DHCP transaction phases warrant discussion from an implementation perspective

DHCPDISCOVER/DHCPOFFER Implementation Aspects

During the initial discovering of DHCP servers, 3 cases need to be supported. First, the case where no DHCP server responds with an offer with valid iSCSI credentials. Second, when one and only one DHCP server responds with the appropriate credentials in the context of iSCSI. Third, when multiple DHCP servers respond with the appropriate credentials.

When no DHCP server responds with iSCSI credentials, the DHCP client should rebroadcast the DHCPDISCOVER packet and wait a significantly longer period of time to respond. The increased time allows the DHCP server a larger window to respond. The increased time should be on the order of at least 4 times longer than the first DHCPDISCOVER timeout value. It is at the discretion of the initiator implementation whether more retries should be attempted or whether to post an error.

When one and only one DHCP server responds with the appropriate credentials, the initiator implementer can use the iSCSI parameters contained in the response. Using the offered parameters and not sending a DHCPREQUEST assumes that the network has only on iSCSI capable DHCP server in the network since the offering server does not receive a DHCPREQUEST to explicitly indicate selection. Moreover, the iSCSI parameters used, if in the offer, are static and identical across any iSCSI capable DHCP servers. Therefore, it

is HIGHLY recommended that a DHCPREQUST/DHCPACK transaction take place to ensure the server and client are aware of each other.

When more than on DHCP server responds with iSCSI credentials, the initiator implementor must respond with a DHCPREQUEST packet to inform the selected server of ownership.

DHCPREQUEST/DHCPACK Implementation Aspects

The key implementation aspect of DHCPREQUEST/DHCPACK is the usage of unicast vs broadcast transactions. In the DHCP RFCs, there is ambiguity on whether DHCPREQUEST can use a unicast mode. To be conservative, the initiator implementer should use broadcast for all DHCPREQUEST/DHCPACK transactions.

DHCPINFORM/DHCPACK Implementation Aspects

The key implementation aspect of DHCPINFORM/DHCPACK is the usage of unicast vs broadcast. The DHCP RFCs are more clear on the behavior here. In order to support a DHCPINFORM/DHCPACK transaction via a unicast transmission, several parameters must be available.

The DHCP client IP address is needed to provide a valid source address on the DHCPINFORM packet. This source address allows the DHCP server to target the DHCPACK to the requesting client. The standard publicized UDP ports should be used as well. The DHCP client IP can be obtained out of band via the iSCSI parameter block passed down to the initiator or, optionally, the initiator implementer wishes to retrieve an IP address from a DHCP server outside the context of iSCSI.

For unicast transmission, the DHCP server IP address is needed to provide a valid destination address on the DHCPINFORM packet. With the destination address in place, the flow through the network is identical to any other node to node IP packet. The standard publicized UDP ports should be used as well. The DHCP server IP address can be obtained out of band via the iSCSI parameter block passed down the initiator or can be retrieved via the DHCP vendor option 204 (Discovery IP address) or, optionally, the initiator implementer wishes to retrieve the designated server IP address from DHCP outside the context of iSCSI. This last means can be done, for example, by desiring and identifying a SLP server in the network.

For broadcast tramission, the DHCP server IP address does not have to be defined. The IP address to use for broadcast transactions is defined as 255.255.255.255. Using the broadcast address for the destination address and client IP address for the source address, the DHCP server is contacted in the broadcast context while the DHCP client, receiving the response, is contacted in a unicast context.

DHCP Message Fields of Interest

Several fields in a DHCP message are of interest. Below these fields are outlined.

DHCP Field	Definition	Description
YIADDR	Client IP address assigned	Defines the IP address assigned by the DHCP server to the DHCP client. Used if IP address has not been assigned to the DHCP client or the DHCP client does not have an IP address already owned.
CIADDR	Client IP address owned	Defines the IP address the DHCP client owns. Used during transactions such as DHCPINFORM transactions to inform the DHCP server that the DHCP client already has an IP address
CHADDR	Client HW address owned	Defines the hardware address the DHCP client owns. Typically, the EN MAC is used as the CHADDR address. This address enables the DHCP server to determine what set of parameters should be passed to the DHCP client. Note there are several ways to identify the client prior to having the IP address set, CHADDR using EN MAC is the most prevalent
SIADDR	Server IP address owned	Defines the DHCP server IP address to be used during transactions such as DHCPREQUEST and DHCPINFORM. Note that during DHCPDISCOVER, no knowledge of the DHCP server IP address is available so a broadcast transmission covers this transaction.

Table 79: DHCP Fields of Interest

DHCP Option Usage

DHCP allows one to specify options to aid in the DHCP client configuration flexibility. Some of these options are basic to the protocol while others are solution or implementation dependent. This section covers DHCP options most directly involved with the iSCSI deployment.

DHCP Option	Definition	Description
1	IP Subnet Mask	Defines the subnet mask to use for determining if an IP address is on the local subnet or needs to be routered to another subnet.
3	IP Router Address	Defines the router or gateway address to be used when accessing a device on another subnet
43	Vendor Option Response	Defines the encapsulation of all options associated with this vendor identifier. The DHCP client can assume that if a 43 response is provided, then this DHCP server is iSCSI aware. The format of this option is 43:len:optA:lenA:valueA:optB:lenB:valueB:…. Given the maximum length of any option including 43 is 255 bytes, RFC 3396 defines a means of sending larger responses to the DHCP client. Basically, 3396 defines an orderly concatentiation of multiple option responses.
51	IP address lease Time	Defines the duration of time that the client can use the IP address assigned to it. The DHCP server provides the option to the DHCP client to indicate the duration the client can used the IP address assigned during the DHCPOFFER transaction
54	DHCP Server Identifier	Defines the IP address of the DHCP server to providing the DHCP information. This option is used during the DHCPOFFER phase.
55	DHCP Parameter List	Defines the list of options the DHCP client expects to receive from the DHCP server. As defined in RFC2132, there is ambiguity on the usage of DHCP option 55 with some servers ignoring it. To better manage the ambiguity, a DHCP client can either include option 55 with ALL desired DHCP options or no include option 55 at all. As a result, a given implementer has the choice of implementing DHCP option 55. In the context of this specification, it is assumed DHCP option 55 is not present nor enumerated.
57	DHCP message size	Defines the message size of the DHCP messages that should be used between the DHCP server and client. The default is 576 bytes of which 504 is the true DHCP message.
60	Vendor Class Identifier	Defines the class of vendor requesting server. The DHCP server uses this option to scope the class of option responsese that should be used. In the context of iSCSI solutions, this option uses the following format to aid in determine which DHCP server to use and to aid the DHCP server to determine which options to use. If SVID field is defined, the format of option 60 is SVID field (if defined):VERS field:LVL field:Initiator mfg:Initiator type:initiator level. If the SVID is not defined, the format of option 60 is “IBM ISAN”:VER field:LVL field:initiator mfg: initiator type:initiator revision. An example of the latter format would be “IBM ISAN:29:58:IBM :01:01” where the VER[7:0]=29, LVL[7:0]=58, mfg=”IBM “,type=01,rev 01. See Option 60 discussion below for more details. Note that the 29 and 58 must be translated to a string form representing the value. The vendor class identifier is analogous to identifying the seating section in a stadium

DHCP Option	Definition	Description
61	Client Identifier	Defines the id of this specific DHCP client. The DHCP server uses this option to determine the exact values to respond with. The default is to use the EN MAC address of the port used but in the future, option should use the CAID field if defined. The client identify is analogous to identifying the seat in the identified section in a stadium. Note when using Microsoft DHCP server, the option 61 cannot be used. Instead, the EN MAC, preceded by a byte with the value 0x01 is used to distinquish one client from another.
255	Msg END	Defines the end of the variable length options space in the DHCP message.

Table 80: RFC 2132 Options of Interest

DHCP Option	Definition	Description
201	1st Target identification	Defines IP and iSCSI parameters set for the 1st target. When the scope (DHCP option 60) is “IBM ISAN” The format is “iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>” Note: for IPv4 addresses, the format is “ddd.ddd.ddd.ddd” Note: for IPv6 addresses, the format is /xxxx:xxxx::xxxx:::xxxx/ (since IPv6 uses colons to delimit substrings, “/” marks address) Note: Boot LUN format is “xx” When the scope (DHCP option 60) is “ISAN” The format (as defined in RFC 4173) is “iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>” Note: for IPv4 addresses, the format is “ddd.ddd.ddd.ddd” Note: for IPv6 addresses, the format is [xxxx:xxxx::xxxx:::xxxx] (since IPv6 uses colons to delimit substrings, “[“ ”]” marks address) Note: Boot LUN format is “xxxx-xxxx-xxxx-xxxx” per RFC 4173
202	2nd Target identification	Defines IP and iSCSI parameters set for the 1st target. When the scope (DHCP option 60) is “IBM ISAN” The format is “iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>” Note: for IPv4 addresses, the format is “ddd.ddd.ddd.ddd” Note: for IPv6 addresses, the format is /xxxx:xxxx::xxxx:::xxxx/ (since IPv6 uses colons to delimit substrings, “/” marks address) Note: Boot LUN format is “xx” When the scope (DHCP option 60) is “ISAN” The format (as defined in RFC 4173) is “iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>” Note: for IPv4 addresses, the format is “ddd.ddd.ddd.ddd” Note: for IPv6 addresses, the format is [xxxx:xxxx::xxxx:::xxxx] (since IPv6 uses colons to delimit substring, “[“ “]” marks address) Note: Boot LUN format is “xxxx-xxxx-xxxx-xxxx” per RFC 4173
203	Initiator identification	Defines the iSCSI information of the initiator. Note that the IP information is defined in the DHCP message elsewhere The format is Initiator iSCSI name in IQN or EUI format

DHCP Option	Definition	Description
204	Discovery identification	Defines the parameters needed for additional discovery. Basically, this option allows the customer to use the current DHCP server to tell the DHCP client to find the iSCSI ONLY parameters on another DHCP server. The format is "scope:<scope>:disc:<dhcp>:id:<client alt id>:flags:<flag bits>" Note: to be backward compatible, if the last character is NOT a colon, then there are no flag bits coming and the following rules apply: Use the scope as defined (ie scope:<IBM ISAN or other>:) If disc = NULL (ie disc:255.255.255.255:), then dhcp broadcast is used If id = NULL (ie id:null"), then client ID is EN MAC address In the future, flag bits (not to exceed 12 presented as 3 ascii hex chrs) will be defined for advanced functions. Note: for IPv4 addresses, the format is "ddd.ddd.ddd.ddd" Note: for IPv6 addresses, the format is [xxxx:xxxx::xxxx:::xxxx] (since IPv6 uses colons to delimit substrings, "[" "]" marks address)
205	iSCSI option space	Defines a DHCP option to pass iSCSI option bits to the DHCP client. This is used only for future or advanced configuration approaches. The intent is that this response matches the OPT field defined in the iSCSI parameter data structure. The format is "B0:<LSB>:B1:<LMSB>:B2:<HMSB>:B3:<MSB>" Note that this DHCP option is optional in the DHCP exchange
206	iSCSI Link Behavior	Defines the behavior iSCSI session establishment in terms of timeouts and retries to the 1st target and to the 2nd target. This option is for future advanced configurations where the specific link behavior needs to be specified. The format is "R1:<retry1>:R2:<retry2>:T1:<timeout1>:T2:<timeout2>"

Table 81: DHCP Vendor Options Defined for iSCSI

DHCP Option	Definition	Description
17	Root path	Defines the DHCP method of using the root path option to define the iSCSI target to use during boot. If a solution is going to use the root path option to define a single target, then the root path definition in the DHCP server configuration file must be scoped within the vendor ID scope. This is necessary to ensure that a DHCP client using the Ethernet MAC address as the identifier does not received the root path associated with a PXE boot or some other boot methodology. The decision on whether to use the old style format or the RFC 4173 format is based on the initiator knowing the capabilities of the initiator used. This is as a result of the fact that the admin may not define the scope to use that could indicate the type of format requested. If the admin does choose to use scope with root path, then the style decision is based on whether "IBM ISAN" or "ISAN" is used.
12 (host name)	Host name	Defines the DHCP method to define the initiator. If a solution going to use DHCP option 12 in conjunction with option 17, the initiator name used to login into the target is a string built up from the MAC address and the option 12 name. Note that DHCP servers can provide the host name via the "use-host-decle-names" flag

Table 82: DHCP Root Path Options Defined for iSCSI

DHCP Implementation Considerations

Vendor ID Usage

All requests from iSCSI boot initiator DHCP clients to DHCP servers **must use** option 60 to signal appropriate vendor scope. If the SVID is defined in the iSCSI parameter data structure passed from BIOS, then that value must be used. If the SVID is not defined, then the string "IBM ISAN" must be used.

Client ID Usage

All requests from iSCSI boot initiator DHCP clients to DHCP servers **must use** option 61 to signal the identity of this given client. If the client Alt ID is not defined, then the type field should be set to 0x01 and use the EN MAC address to define client identity. If the client Alt Id is defined, then the type field should be set to 0x00 and use the CAID field.

DHCP CHADDR Usage

All requests from iSCSI boot initiator DHCP clients to DHCP servers **must use** the CHADDR field containing the EN MAC address of the DHCP client.

DHCP CIADDR Usage

The use of CIADDR in the iSCSI boot sequence **must** conform to the DHCP usage of CIADDR. Namely, it should be used by the iSCSI boot initiator DHCP client to signal to the DHCP server any IP address it currently thinks it owns.

DHCP YIADDR Usage

The use of YIADDR in the iSCSI boot sequence **must** conform to the DHCP usage of YIADDR. Namely, the iSCSI boot initiator DHCP client **must** accept the YIADDR provided by the DHCP server during the DHCPREQUEST⇔DHCPACK transaction sequence.

DHCP SIADDR Usage

The use of SIADDR in the iSCSI boot sequence must conform to the DHCP usage of SIADDR. Namely, the iSCSI boot initiator DHCP client must use this address to access the DHCP server during DHCPREQUEST⇔DHCPACK or DHCPINFORM⇔DHCPACK transactions.

DHCP Option 1 Usage

DHCP option 1 allows the DHCP server to define the subnet mask to used to identify IP addresses on the current network segment. The subnet mask provided in the DHCPOFFER response provides the subnet mask to used if this DHCP server is selected.

DHCP Option 3 Usage

DHCP option 3 allows the DHCP server to define the gateway or router IP address to be used when the iSCSI boot initiator DHCP client desires to access an iSCSI target residing on a different network segment.

DHCP Option 255 Usage

DHCP option 255 allows the DHCP server to indicated to the iSCSI boot initiator DHCP client where the end of the variable option is, and in turn where the end of the DHCP msg is.

Single Frame Responses from DHCP Server

All transactions between iSCSI boot initiator DHCP clients and DHCP servers **must be** a single frame transactions. For transactions requiring larger frames than the default size, the option 57 allows the client to indicate to the server that larger frames are to be used. In general, the size of the frame should be set to something like 900 bytes or so to provide ample space for transactions.

DHCP Option 52 Usage

DHCP Option 52 allows the spac consumed by SNAME and BOOt options to be used for DHCP Option space. Since there is not need to use this space as outlined here and since a given DHCP server may also be used to set up a PXE boot, the iSCSI boot initiator **must not** use Option 52.

Option 43 Encapsulation

With the Option 43 encapsulation, there are several cases to consider.

Single Vendor Option Encapsulation

Any single vendor option encapsulated into the Option 43 response **must not** be larger than 252 bytes to ensure the entire vendor option is encapsulated in the option 43 response.

Multiple Vendor Option Encapsulation

Any group of vendor options encapsulated into the Option 43 response **must not** be larger than 252 bytes to ensure the complete set of options are contained in the option 43 response

Vendor Option Parsing

In examining the vendor option behavior and the vendor options that are used for iSCSI, it is quite likely that the vendor options will span multiple instances of the Option 43 response. RFC 3396 defines how multiple option responses can be concatenated into a single response context. As a result, the DHCP client must support RFC 3396.

Clearly, where possible, encapsulating complete iSCSI vendor options (Options 201, 202, etc) into several Option 43 responses such that no single iSCSI vendor option straddles two Option 43 response, is highly desirable. If a given DHCP server can enforce this arbitrary encapsulation, then that should be the preferred path. However, no assumption should be made that the special/customized DHCP server is available to support this streamlined approach.

In the event that Option 43 concatenation is required where an iSCSI vendor option does, indeed, span two Option 43 responses, the DHCP client can use the marker key strings present in the iSCSI vendor options to aid the parsing process. Additional marker strings can be added subject to negotiation and consensus.

Option 43 Concatentation

To encompass a large set of vendor options, RFC3396 supports multiple occurrences of options in a DHCP message. This allows multiple Option 43 responses to be present to provide all the vendor options needed.

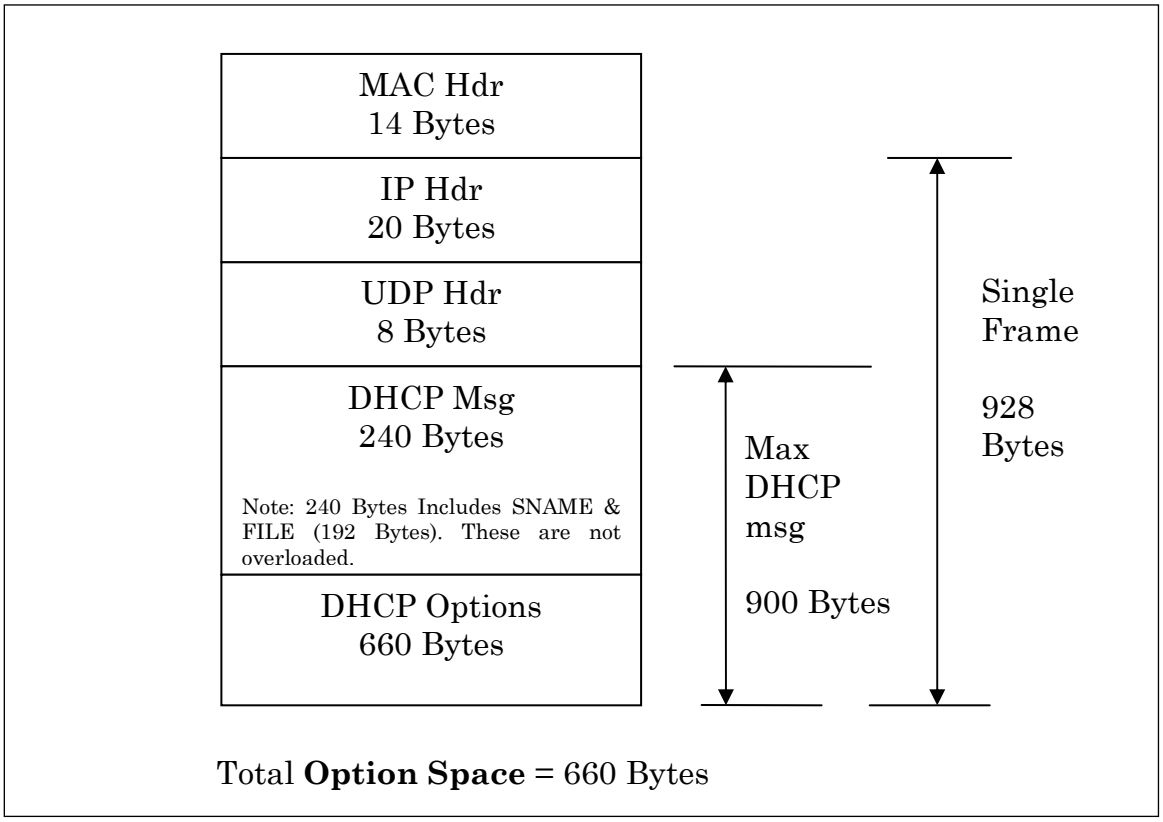

Figure 4: DHCP Frame Layout

DHCP Option 60 Usage

DHCP Option 60 allows the client to indicate to the DHCP the various levels of iSCSI components in the client. The client builds the value of option 60 on the fly as it builds the DHCP message. Ie there is no specific iscsi parameter values defined in non volatile storage defining the entire option 60 string. An implementation may choose to build the option 60 string and store in some volatile memory to be reused during the current session.

The format of the option 60 string can be one of 3 formats cited below

DHCP Option 60	Description
Opt 60 (SVID[63:0]=0x00)	"IBM ISAN:<VER[7:0]>:<LVL[7:0]>:<MFG[31:0]>:<TYPE[7:0]>:<REV[7:0]>" Indicates the DHCP server should use the "old" style DHCP option 201 and 202 format. Note that VER, LVL, MFG, TYPE, REV are optional components of Option 60
Opt 60 (SVID[63:0]=0x00)	"ISAN:<VER[7:0]>:<LVL[7:0]>:<MFG[31:0]>:<TYPE[7:0]>:<REV[7:0]>" Indicates the DHCP server should use RFC 4173 style DHCP option 201 and 202 format. Note that VER, LVL, MFG, TYPE, REV are optional components of Option 60
Opt 60 (SVID[63:0]!=0x00)	"<SVID[63:0]>: <VER[7:0]>:<LVL[7:0]>:<MFG[31:0]>:<TYPE[7:0]>:<REV[7:0]>" Indicates the DHCP server should use RFC 4173 style DHCP option 201 and 202 format. Note that VER, LVL, MFG, TYPE, REV are optional components of Option 60

Table 83: DHCP Option 60 Format

DHCP Option 60	Description
“IBM ISAN:00:00:NULL:00:00	Scope = “IBM ISAN” Parm version = 00 (meaning the initiator cant provide) Parm level = 00 (meaning the intiator cant provide MFG = “NULL” (meaning the initiator cant provide) Initiator type = 00 (meaning the initiator cant provide) Initiator revision = 00 (meaning the initiator cant provide) Note: “old” style format for Options 201 and 202 Note: Parm Option bit 7=1 indicates initiator should use this long form (all the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.
“ISAN:03:01:IBM :14:29	Scope = “ISAN” Parm version = 03 Parm level = 01 Mfg = “IBM “ (note the space) Initiator type = 14 Initiator revision = 29 Note: RFC 4173 style format for Options 201 and 202 Note: Parm Option bit 7=1 indicates initiator should use this long form (all the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.
“XYZ ISAN:03:01:IBM :14:29	Scope = “XYZ ISAN” Parm version = 03 Parm level = 01 Mfg = “IBM “ (note the space) Initiator type = 14 Initiator revision = 29 Note: RFC 4173 style format for Options 201 and 202 Note: Parm Option bit 7=1 indicates initiator should use this long form (all the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.
“IBM ISAN”	Scope = “IBM ISAN Note: “old” style format for Options 201 and 202 Note: Parm Option bit 7=0 indicates initiator should use this short form (none of the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.
“ISAN”	Scope = “ISAN Note: RFC 4173 style format for Options 201 and 202 Note: Parm Option bit 7=0 indicates initiator should use this short form (none of the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.
“XYZ ISAN”	Scope = “XYZ ISAN Note: RFC 4173 style format for Options 201 and 202 Note: Parm Option bit 7=0 indicates initiator should use this short form (none of the VER, LVL, MFG, TYPE, REV fields used) and the DHCP Server can substring if not interested.

Table 84: Examples of DHCP Option 60 Format

In either case, the numeric values in the option 60 are the ASCII string equivalents of the numeric values.

The “IBM ISAN”, SVID[63:0], VER[7:0], and LVL[7:0] have been explicitly defined elsewhere in the document. The MFG[31:0], TYPE[7:0] and REV[7:0] are defined here to address various initiator architectures and revision levels. Below is the MFG, TYPE and REV definitions.

DHCP Option 60 Manufacturer ID

The manufacturer ID allows a given vendor do identify them selves to the DHCP server in the context of iSCSI. This 4 byte field is ASCII based, allowing for 4 characters to define the vendor. Any shorter values should have trailing spaces. Note that this field is optional on the option 60 string.

MFG[31:0]	**Description**
MFG[31:0]=0x00000000	Default value of all “0”. Indicates nothing has been specified
MFG[31:0]!=0x00000000	Some vendor ID is specified
MFG[31:0]=”NULL“	An example of no manufacture ID present
MFG[31:0]=”IBM “	An example of manufacturer ID for an IBM intiator (note the trailing space)
MFG[31:0]=”QLGC”	An example of manufacturer ID for a Qlogic HBA
MFG[31:0]=”ADPT “	An example of manufacturer ID for an Adaptec HBA intiator
MFG[31:0]=”BRCM”	An example of manufacturer ID for a Broadcom HBA
MFG[31:0]=”MSFT “	An example of manufacturer ID for a Microsoft intiator
MFG[31:0]=”LNUX”	An example of manufacturer ID for a Linux initiator
MFG[31:0]=”UNIX”	An example of manufacturer ID for an Unix initiator
MFG[31:0]=”<xxxx>”	An example of manufacturer ID not current defined

Table 85: DHCP Option 60 MFG Definition

DHCP Option 60 Type ID

The type ID allows a given iSCSI initiator vendor to specify the type of initiator supported. The value is numeric. This field can be used for general type categorization or can be used to highlight specific features in a given initiator. For example, a TYPE[7:0]=0x29 may mean a 2 port, non ipsec capable, initiator while TYPE[7:0]=0x2A may mean a 2 port, ipsec capable initiator.

Note this field is optional on the DHCP option 60 string.

TYPE[7:0]	Description
TYPE[7:0]=0x00	Type not specified because the initiator does not have this construct available
TYPE[7:0]>= 0x10 **TYPE[7:0]< 0x2F**	Family of software initiators running on xSeries (X86) platforms
TYPE[7:0]>= 0x30 **TYPE[7:0]< 0x4F**	Family of software initiators running on pSeries (POWER/PPC) platforms
TYPE[7:0]>= 0x50 **TYPE[7:0]< 0x6F**	Family of software initiators running on iSeries (POWER) platforms
TYPE[7:0]>= 0x70 **TYPE[7:0]< 0xFF**	Reserved for future use. One potential application for these values is if more than 239 initiator types are available within a given family. A value in this region could be used to alter the string format in the future.
TYPE[7:0]= 0xFF	Family of initiators of unknown type. This value makes no claims or assumptions that given initiator conforms to iSCSI behavior defined in this document

Table 86: DHCP Option 60 TYPE Definition

DHCP Option 60 Revision ID

The REV ID allows a given manufacture to specify the revision level of a given initiator type. It can be used for global revision levels or can be used for precision level. It is beyond the context of this definition if a given manufacturer wishes to use the 8 bits different. For example, if a given manufacture wants to use some sort of major/minor approach, then all that is required is that the format stay in the context defined below (aka all values are between 0x01 and 0xF0). It is the manufacturers responsibility to articulate to all users any differences in the format stated below so that the users can correctly program the DHCP server to respond.

Note that this field is option on the DHCP option 60 string.

REV[7:0]	Description
REV[7:0]=0x00	Revision not specified because the initiator does not have this construct available
REV[7:0]>= 0x01 **REV[7:0]< 0xF0**	Valid revision numbers for given initiator type
REV[7:0]>= 0xF0 **REV[7:0]< 0xFF**	Reserved for future use. One potential application for these values is if more than 239 initiator revisions are available for a given initiator type. A value in this region could be used to alter the string format in the future.
REV[7:0]= 0xFF	Family of initiators of unknown revision. This value makes no claims or assumptions that given initiator conforms to iSCSI behavior defined in this document

Table 87: DHCP Option 60 REV Definition

Typically, a given DHCP server configuration file tends to subset the option 60 string. This mechanism provides a way to control the resolution of the scope field. For example, if an early adopter initiator does not create the full option string and, thus, the only information provided is the "IBM ISAN" or SVID[63:0] to determine the proper scope. On the other hand, if explicit knowledge of the DHCP client implementation is required, the entire DHCP option 60 string can be used with a simple test to determine if character 9 is a ":" as the indicator that the four additional fields are present. Note that processing the VER, LVL, MFG, TYPE, REV can place a significant burden on the DHCP server so caution should be exercised.

Below is an example of how to subset the DHCP option 60 at the DHCP server. Note that windows DHCP servers, by default, match the leading characters of option 60 to the scope. As a result, no explicit substring operation is needed. Recall that in PXE implementations, the actual option 60 string is defined as "PXEClient:Arch:xxxxx:UNDI:yyyzzz" while the definition in windows DHCP servers is "PXEClient"

```
#DHCP Config file
ddns-update-style ad-hoc;
#other global parameters for DHCP

#define option space
option space iSCSI;
option iSCSI.initiator code 203 = string;
option iSCSI.target1 code 201 = string;
option iSCSI.target2 code 202 = string;
#more iscsi options

#the actual vendor-class-identifier sent by the client = "IBM ISAN:3:1:IBM :29:56"

class "iscsi" {
    Match if substring(option vendor-class-identifier,0,8) = "IBM ISAN";      #increase 8 to catch more of string
    #Global class options as needed
    group {
                subclass "vendor classes" "IBM ISAN" {
                    vendor-option-space iSCSI;
                }
                host blade_a {
                    hardware Ethernet = 11:11:11:11:11:11;
                    fixed address = 123.123.123.1;
                    option iSCSI.initiator = "iqn.2005-03.com.ibm.blade_a";
                    option iSCSI.target1 = "iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1";
                    option iSCSI.target2 = "iscsi:123.123.123.20:6:3260:0:iqn.2005.03.com.ibm.lun_a2";
                    #more iSCSI and DHCP options as needed
                } #end blade_a
                host blade_b {
                    hardware Ethernet = 22:22:22:22:22:22;
                    fixed address = 123.123.123.2;
                    option iSCSI.initiator = "iqn.2005-03.com.ibm.blade_b";
                    option iSCSI.target1 = "iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_b1";
                    option iSCSI.target2 = "iscsi:123.123.123.20:6:3260:0:iqn.2005.03.com.ibm.lun_b2";
                    #more iSCSI and DHCP options as needed
                } #end blade_b
                #more blades as needed
    } #end group

    #other DHCP classes
    # other DHCP options
    # other vendor options
    #etc
}
```

Figure 5: DHCP Server Substring Parsing Example

DHCP Option 17 Usage

DHCP Option 17 allows the root path option in DHCP to be used to provide a single iSCSI target. The Option 17 root path must be defined within the vendor ID scope to ensure it is actually defined for iSCSI targets vs other boot methodologies.

A DHCP client would issue a DHCPDISCOVER, DHCPREQUEST, etc with the vendor ID defined. Depending on how the admin configures the DHCP server, the DHCP server must respond with the vendor options defined for iSCSI or respond with Option 17 defined within the Vendor ID scope. As a result, the DHCP client **must** see a response containing the **either** Option 43 response **or** Option 17 response but not both. The DHCP client can then use whichever option included in the response. As a result, the admin **must not** define both in the vendor ID scope.

Supporting the Option 17 approach to iSCSI parameter deployment provides a backward compatibility to the less sophisticated, less evolved methodology for deploying iSCSI parameters.

VLAN and DHCP Implications

Recall, that VLAN tags used on 802.1Q VLANs provide a way to logically isolate a subset of stations on a network segment from other stations on that same segment. In the context of DHCP, VLANs introduce an conundrum in that the DHCP client and DHCP server need to be on the same VLAN so the DHCP client cannot obtain the VLAN tag from the DHCP server. As a result, VLAN tag configuration can only be done using the out of band parameter deployment or via a manual configuration operation on a GUI interface.

iSCSI Initiator Parameter Mapping into DHCP Space

When acquiring iSCSI parameters from DHCP space, the representation is different due to DHCP to the DHCP architecture. Below is the parameter representation in DHCP space. Note that representation of each of the option 20x and option 17 is in ASCII format consistent with string constructs used in DHCP servers.

DHCP Option	Usage	Format
201	1st Target	"iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>"
		1st target = IP address of 1st target
		Protocol = transport (defaults to TCP) = 6
		Port = TCP port (defaults to 3260)
		LUN = boot LUN (defaults to 0)
		Iscsi name = IQN or EUI format of iSCSI name
		E.g., Option 201("old") = iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1"
		For examples of new style, please refer to RFC 4173
202	2nd target	"iscsi:<1st target>:<protocol>:<port>:<LUN>:<iscsi name>"
		1st target = IP address of 1st target
		Protocol = transport (defaults to TCP) = 6
		Port = TCP port (defaults to 3260)
		LUN = boot LUN (defaults to 0)
		Iscsi name = IQN or EUI format of iSCSI name
		E.g., Option 202()0ld") ="iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1"
		For examples of new style, please refer to RFC 4173
203	Initiator	Initiator iSCSI name in IQN or EUI format as specified by RFC 3720 (iscsi)
		E.g., Option 203 = "iqn.2005-03.com.ibm.blade_a"
204	Scope	"scope:<scope>:disc:<dhcp>:id:<client alt id>:flags:<flag bits>"
		Scope = scope/vendor id (if specified use, else use from parm block)
		Disc = discovery IP address (if specified use, else use from parm block)
		Id = Client Alt Id (if specified use, else use from parm block)
		Flag bits = (future/optional) bits to aid in the usage of 204…max 12 bits (3 ascii hex)
		E.g., Option 204 = "scope:XYZ iSAN:disc:123.123.123.123:id:svr 0001:flags:1a9"
205	Options	"B0:<LSB>:B1:<LMSB>:B2:<HMSB>:B3:<MSB>"
		LSB = least significant byte as a 2 character ascii string
		LMSB = low middle significant byte as 2 character ascii string
		HMSB = high middle significant byte as 2 character ascii string
		MSB = most significant byte as 2 character ascii string
		E.g., Option 205 = "B0:AF:B1:CC:B2:44:B3:21" (Note this equals 0x2144CCAF)
		Note: Option 205 is an optional DHCP parameter.
206	Link	"R1:<retry1>:R2:<retry2>:T1:<timeout1>:T2:<timeout2>"
		Retry1 = link retry cnt for accessing target 1 as ascii number Not specified…use values in parm block
		Retry2 = link retry cnt for accessing target 2 as ascii number Not specified…use values in parm block
		Timeout1 = timeout period for accessing target 1 as ascii number Not specified…use values in parm block
		Tmeout2 = timeout period for accessing target 2 as ascii number Not specified…use values in parm block
		E.g., Option 206 = "R1:3:R2:4:T1:8:T2:8"

Table 88: iSCSI Parameters in DHCP Space (Option 20x)

DHCP Option	Usage	Format
17	Only Target	“iscsi:< target>:<protocol>:<port>:<LUN>:<iscsi name>”
		target = IP address of 1st target
		Protocol = transport (defaults to TCP)
		Port = TCP port (defaults to 3260)
		LUN = boot LUN (defaults to 0)
		Iscsi name = IQN or EUI format of iSCSI name
		E.g., Option 17 = “iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1”
12	Initiator name	<host name>
		Host name = host name received from the DHCP server or provided otherwise
		Initiator name = “iqn.1986-03.ibm.com.<MAC address>.<host name>
		e.g. , “iqn.1986-03.ibm.com.11:22:33:44:55:66.blade_a”

Table 89: iSCSI Parameters in DHCP space (option 17 and 12)

Below is the mapping between the iSCSI parameters and representation in DHCP space.

Parameter	Name	DHCP Option 17	DHCP Vendor Option	Definition
Signature	"ISAN"	NA	NA	FYI only, not part of data structure
Version	VER[7:0]	NA	NA	Version of the parm data structure
Level	LVL]7:0]	NA	NA	Level of data structure within version
Data Structure Length	LEN[15:0]	NA	NA	Length in bytes of the total data
Checksum	CHK[7:0]	NA	NA	1 byte checksum of data structure total
Static/Dynamic	S_D[7:0]	NA	NA	Flavors of static and/or dynamic
Options	OPT[31:0]	NA	205(optional)	Parameter structure options
Retry Count	RTRY[7:0]	NA	206	Retries to attempt
Timeout Value	TOVAL[7:0]	NA	206	Time out durations per attempt
Scope/Vendor ID	SVID[63:0]	NA	204	Scope ID to aid in DHCP scope/usage
Client Alt ID	CAID[63:0]	NA	204	Support Further DHCP refinement
Discovery IP address	DIP[63:0]	NA	204	IP address of DHCP server for unicast
Initiator IP address	IIP[63:0]	DHCP	DHCP	iSCSI initiator IP address
Initiator name	IIQN[71:0][7:0]	NA	203	iSCSI iqn tag
Initiator CHAP id	ISDI[15:0][7:0]	NA	NA	CHAP ID or 1st half of security key
Initiator CHAP pw	ISDP[23:0][7:0]	NA	NA	CHAP PW or 2nd half of security key
Subnet Mask	MSK[63:0]	001	001	IP network subnet mask
VLAN tag	VLAN[15:0]	NA	NA	VLAN tag for iSCSI traffic
Gateway IP address	GRIP[63:0]	003	003	IP network gateway mask
1st Target IP address	T1IP[63:0]	017	201	iSCSI target IP address of storage
1st Target TCP port	T1PT[15:0]	017	201	iSCSI TCP port on target IP address
1st Target name	T1IQN[71:0][7:0]	017	201	iSCSI iqn tag
1st Target Boot LUN	T1BL[7:0]	017	201	Boot LUN residing on 1st Target - option
1st Target CHAP id	T1SDI[15:0][7:0]	NA	NA	CHAP ID or 1st half of security key
1st Target CHAP pw	T1SDP[23:0][7:0]	NA	NA	CHAP PW or 2nd half of security key
2nd Target IP address	T2IP[63:0]	NA	202	iSCSI target IP address of storage
2nd Target TCP port	T2PT[15:0]	NA	202	iSCSI TCP port on target IP address
2nd Target name	T2IQN[71:0][7:0]	NA	202	iSCSI iqn tag
2nd Target Boot LUN	T2BL[7:0]	NA	202	Boot LUN residing on 2nd target
2nd Target CHAP id	T2SDI[15:0][7:0]	NA	NA	CHAP ID or 1st half of security key
2nd Target CHAP pw	T2SDP[23:0][7:0]	NA	NA	CHAP PW or 2nd half of security key
Extended parameters	EPID[7:0] EPCT[15:0] EPOPT[7:0] EPVAL[EPCT:0][7:0]	NA	NA	for ID+CT+values (4 byte min per)

Table 90: iSCSI Parameter Mapping to DHCP Space

Acquire from DHCP Decision Algorithm

Below is an updated version of the algorithm decision flow for acquiring iSCSI parameters from DHCP services. In Figure 6: iSCSI Parameter Acquire Algorithm Flow, a pictorial flow is presented while in 0 Inband Error Scenarios presents a pseudo code flow.

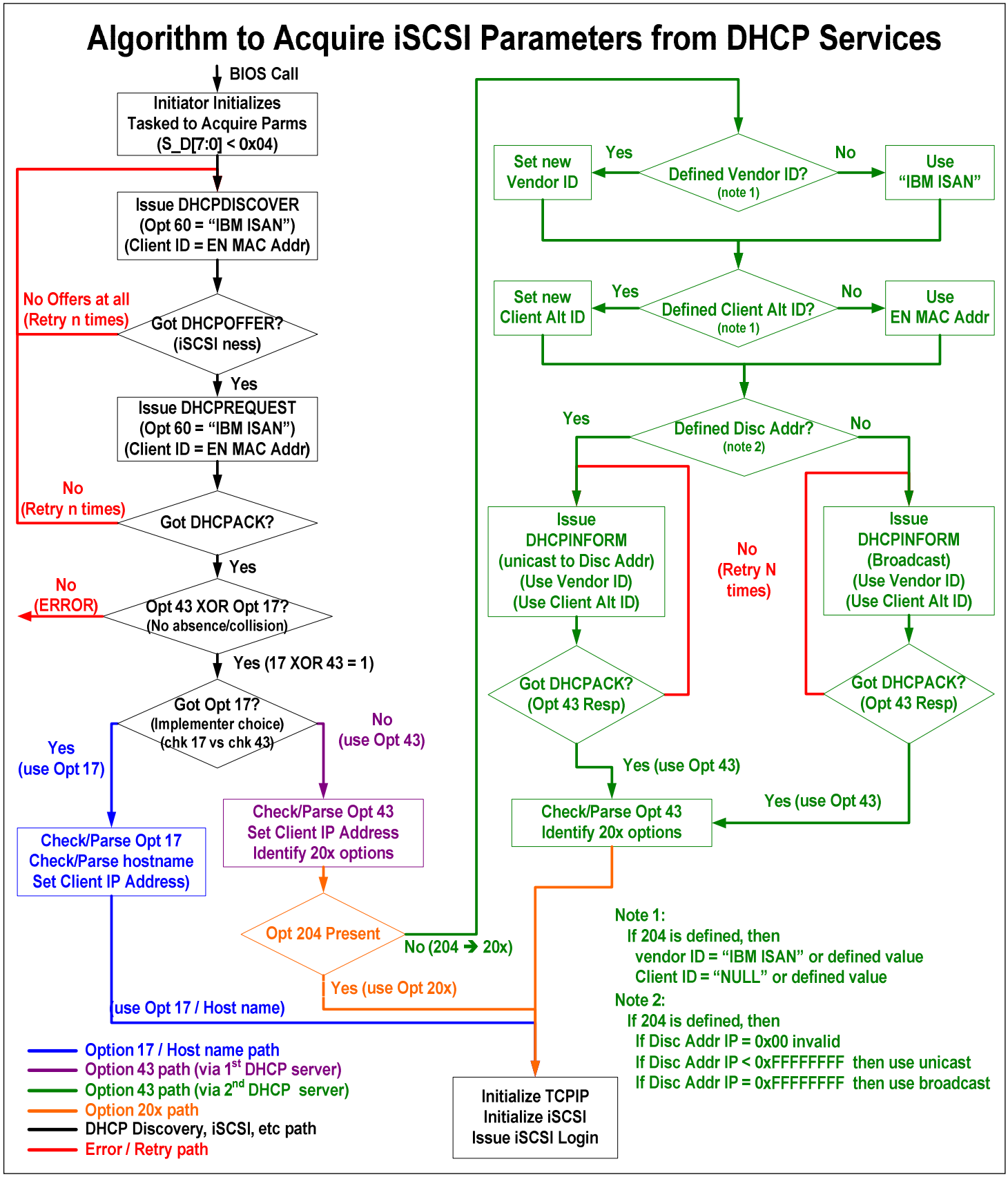

Figure 6: iSCSI Parameter Acquire Algorithm Flow

DHCP Acquisition Flows

In this section, a detailed discussion of the DHCP network flows for acquiring the iSCSI initiator configuration parameters is presented.

Acquire Flow 1: Acquire all IP/iSCSI Parms from one DHCP Svr

In acquire flow 1, the dedicated iSCSI DHCP server manages and serves up all IP, boot, and iSCSI parameters. This turns out to be the simplest flow to discuss.

Acquire Flow 1: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to acquire iSCSI parameters from DHCP service.
DHCP service sets IP information to "static" so that it does not change if the network changes.
Dedicated iSCSI DHCP Server on the same subnet with the initiator.
iSCSI DHCP Server serves up all IP, boot, iSCSI parameters to the Initiator based on Option 60
All other DHCP servers ignore DHCP transactions due to vendor ID (no awareness of Option 60)
Any security parameters are deployed out of band.
No Client_Alt ID or discovery IP address has been deployed apriori out of band.
No Client_Alt ID or discovery IP address is acquired from DHCP server via Option 204

Acquire Flow 1: Network Flow

Below is the network flow sequence from power on to iSCSI Login. This flow is intended to be completely transparent to any existing DHCP service in the network. The existing DHCP services will not respond because they do not have any vendor ID scope defined in their existing DHCP config file. As a result, the existing DHCP service can remain untouched!!

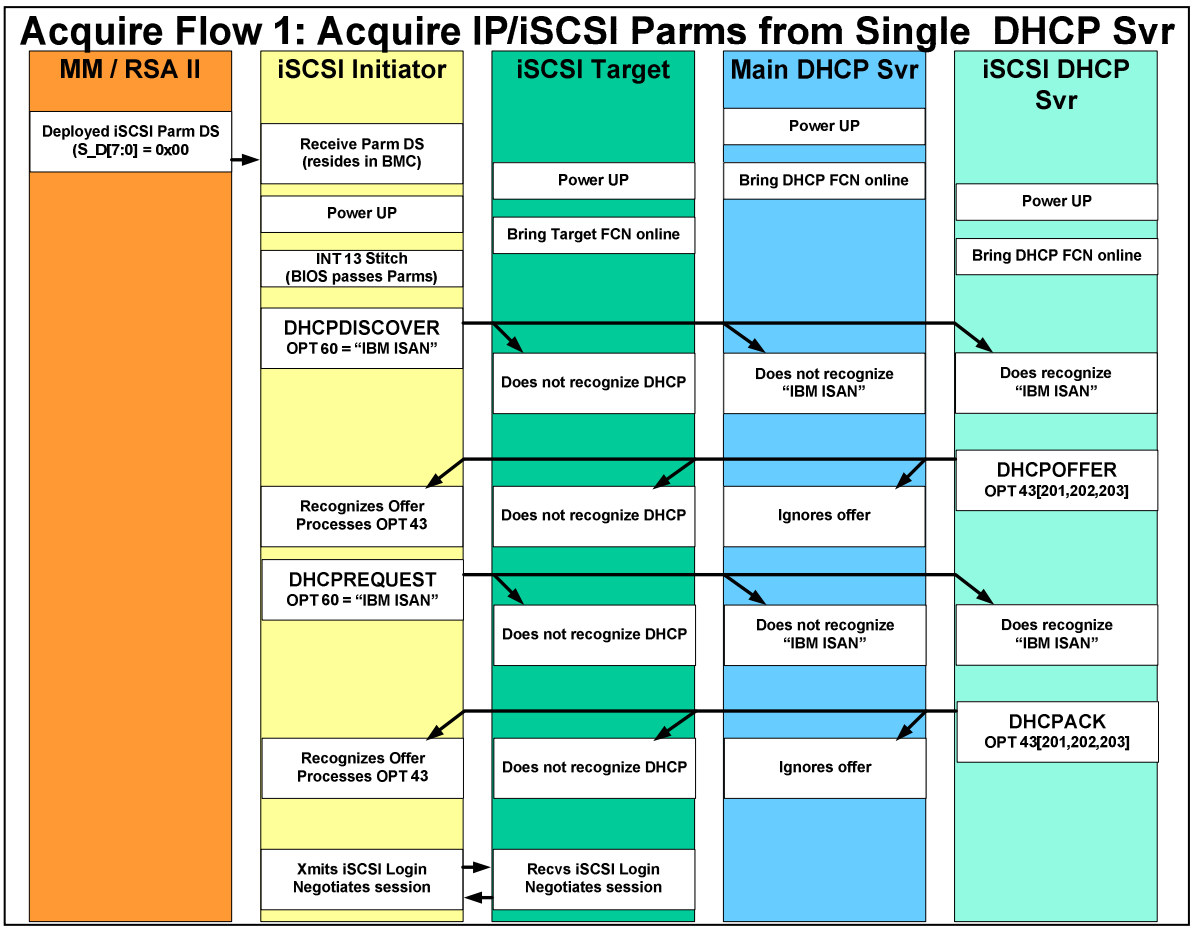

Figure 7: Acquire Flow 1: All Parameters from Dedicated DHCP Server

Acquire Flow 2: Acquire IP/iSCSI Parms from Two DHCP Svrs

In acquire flow 2, the dedicated iSCSI DHCP server manages and serves up only iSCSI parameters. This turns out to be the simplest flow to discuss.

Acquire Flow 2: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to acquire information from DHCP service.
DHCP service sets IP information to "static" so that it does not change due to network changes.
Dedicated iSCSI DHCP Server on the same subnet with the initiator.
iSCSI DHCP Server serves up all iSCSI parameters to the Initiator based on vendor ID (Option 60)
Other DHCP Server is programmed to respond based on vendor ID (Option 60)
Unique and Different (customer specific) vendor ID is used to aid appropriate DHCP svc response
Any security settings are deployed apriori out of band

Acquire Flow 2: Network Flow

Below is the network flow sequence from power on to iSCSI Login. This flow focuses on resolving the problem of coexistence with current DHCP servers. In this flow, the existing DHCP service is updated to accept the vendor ID and reply back with the option 204 parameters which define where to do a broadcast DHCPINFORM transaction to acquire the remaining parameters.

Since the existing DHCP service is being updated to accept the vendor ID of "IBM iSAN", the admin may choose to add IP information to the config file so that all IP information is centralized on the existing DHCP service while all the iSCSI specific information is on a new dedicated iSCSI DHCP service.

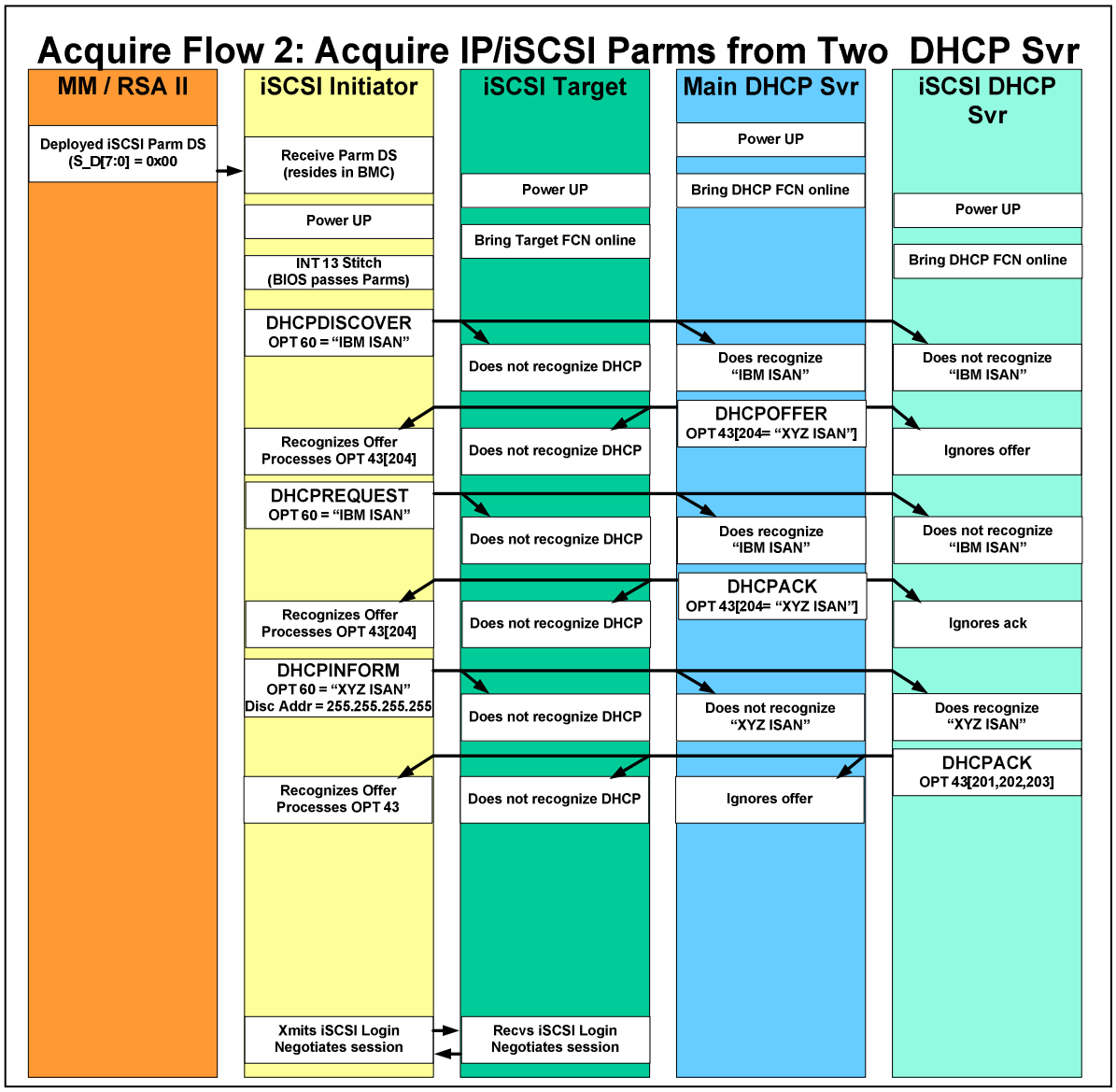

Figure 8: Acquire Flow 2: iSCSI Parameters from Dedicated DHCP Service

Acquire Flow 3: Accept iSCSI Parameters from iSCSI DHCP Server

Acquire flow 3 is similar to acquire flow 2 except a DHCPINFORM unicast is used to acquire iSCSI parameters from the 2nd DHCP server.

Acquire Flow 3: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to acquire information from DHCP service.
DHCP service sets IP information to "static" so that it does not change due to network changes.
Dedicated iSCSI DHCP Server on the same subnet with the initiator.
iSCSI DHCP Server serves up all iSCSI parameters to the Initiator based on vendor ID (Option 60)

Other DHCP Server is programmed to respond based on vendor ID (Option 60)
Unique and Different (customer specific) vendor ID is used to aid appropriate DHCP svc response
A Discovery IP address is defined to support a unicast to the 2nd DHCP server
Any security settings are deployed apriori out of band

Acquire Flow 3: Network Flow

Below is the network flow sequence from power on to iSCSI Login. This flow focuses on resolving the problem of coexistence with current DHCP servers. In this flow, the existing DHCP service is updated to accept the vendor ID and reply back with the option 204 parameters which define where to do a DHCPINFORM unicast transaction to acquire the remaining parameters.

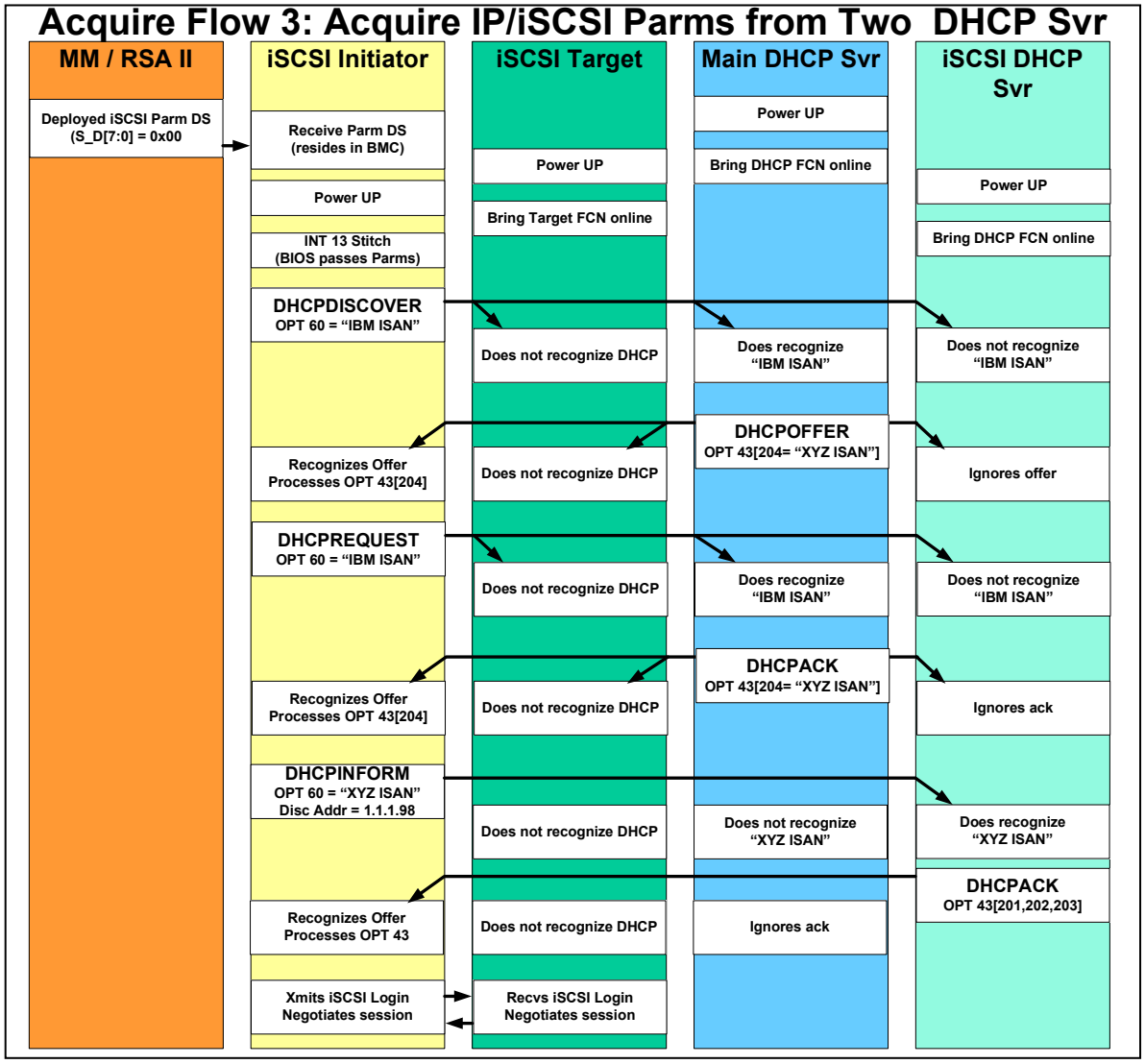

Figure 9: Acquire Flow 3: iSCSI Parameters from Dedicated DHCP Service

Acquire Flow 4: Accept iSCSI Parameters from iSCSI DHCP Server

In acquire flow 4, the dedicated iSCSI DHCP server manages and serves up only iSCSI parameters. This turns out to be the simplest flow to discuss. The IP and discovery information is deployed out of band apriori.

Acquire Flow 4: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to with basic IP and discovery parameters.
Standard vendor ID ("IBM ISAN") or customer specific vendor ID ("My ISAN") can be used
Dedicated and targeted DHCP server has remaining iSCSI parameters
Dedicated and targeted iSCSI DHCP Server on the same subnet with the initiator.
No actual DHCP discovery is done since IP information deployed apriori
Dedicated and targeted iSCSI DHCP Server responds to appropriate vendor ID (Option 60)
Client_Alt ID may have been deployed apriori out of band or via Option 204.
Any security settings are deployed apriori out of band

Acquire Flow 4: Network Flow

Below is the network flow sequence from power on to iSCSI Login. This flow focuses on resolving the problem of broadcast storms when DHCP clients try to acquire information. In this flow, the initiator is configured apriori to have some basic information such IP address and discovery IP address. The targeted DHCP server is updated to accept the vendor ID and reply back with the remaining iSCSI parameters. As a result, there is no DHCP discovery/offer sequence executed. Note, this scenario can be extended to SLP or iSNS servers if the initiator, target, and SLP/iSNS services can support the operations and are in place.

Since the transaction is a unicast transaction to a targeted DHCP service, the targeted DHCP service does not have to be publicized on the network.

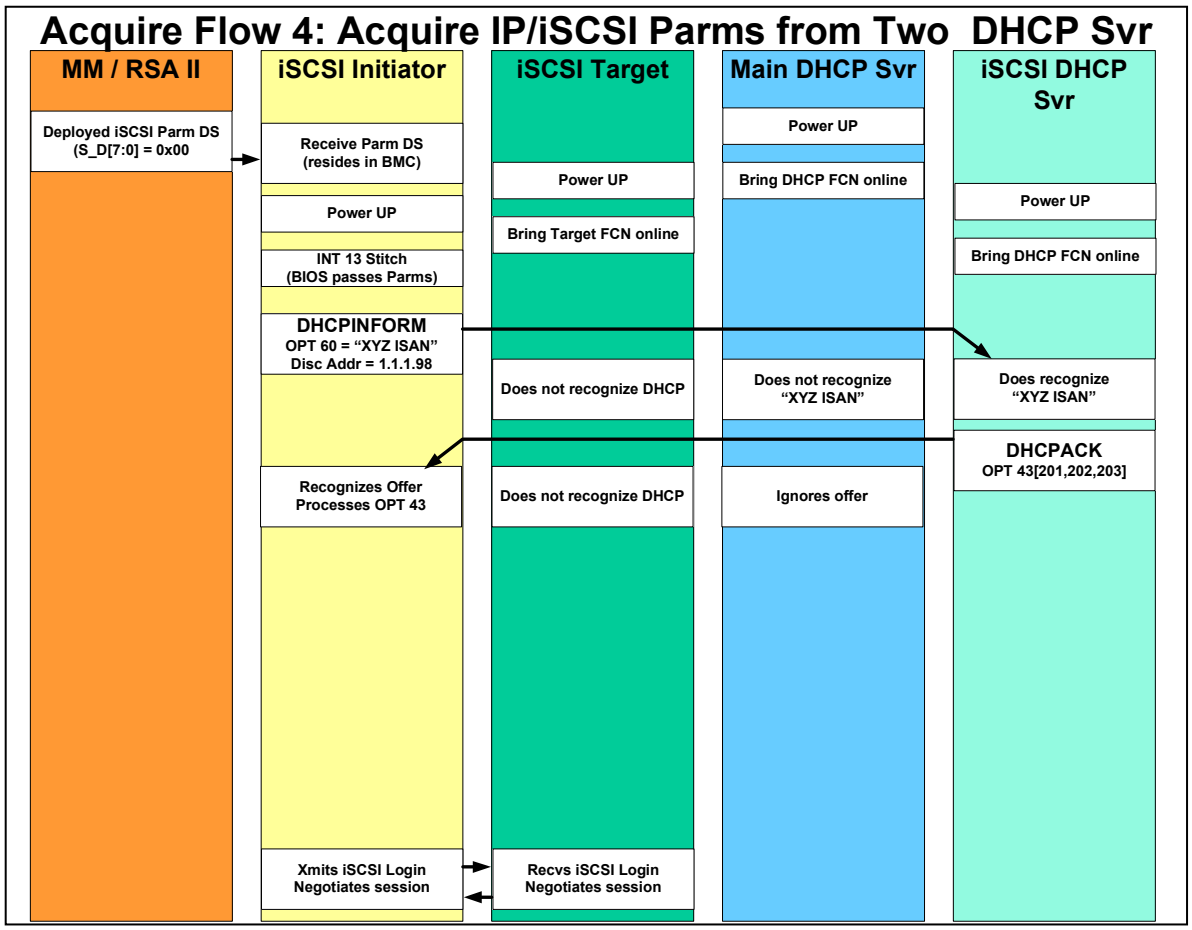

Figure 10: Acquire Flow 4: Partial Acquire of iSCSI Parameters from DHCP

Acquire Flow 5: Accept iSCSI Parameters from iSCSI DHCP Server

In acquire flow 5, the multiple DHCP servers recognize and serves up the iSCSI parameters. In this case, a DHCPREQUEST is required so that the designated server is understands it's the server of choice.

Acquire Flow 5: Assumptions and Definitions

Below are some assumptions and definitions to note.

Initiator is set to acquire information from DHCP service.
DHCP service sets IP information to "static" so that it does not change due to network changes.
Dedicated iSCSI DHCP Server on the same subnet with the initiator.
2 DHCP servers can serve up all iSCSI parameters to the Initiator based on vendor ID (Option 60)
"IBM ISAN" is the vendor ID is used to aid appropriate DHCP svc response
Any security settings are deployed apriori out of band

Acquire Flow 5: Network Flow

Below is the network flow sequence from power on to iSCSI Login. This flow focuses on resolving the which DHCP server to use in acquiring iSCSI parameters. In this flow, Both DHCP servers can respond appropriately. The initiator decides which server to use via the DHCPREQUEST transaction.

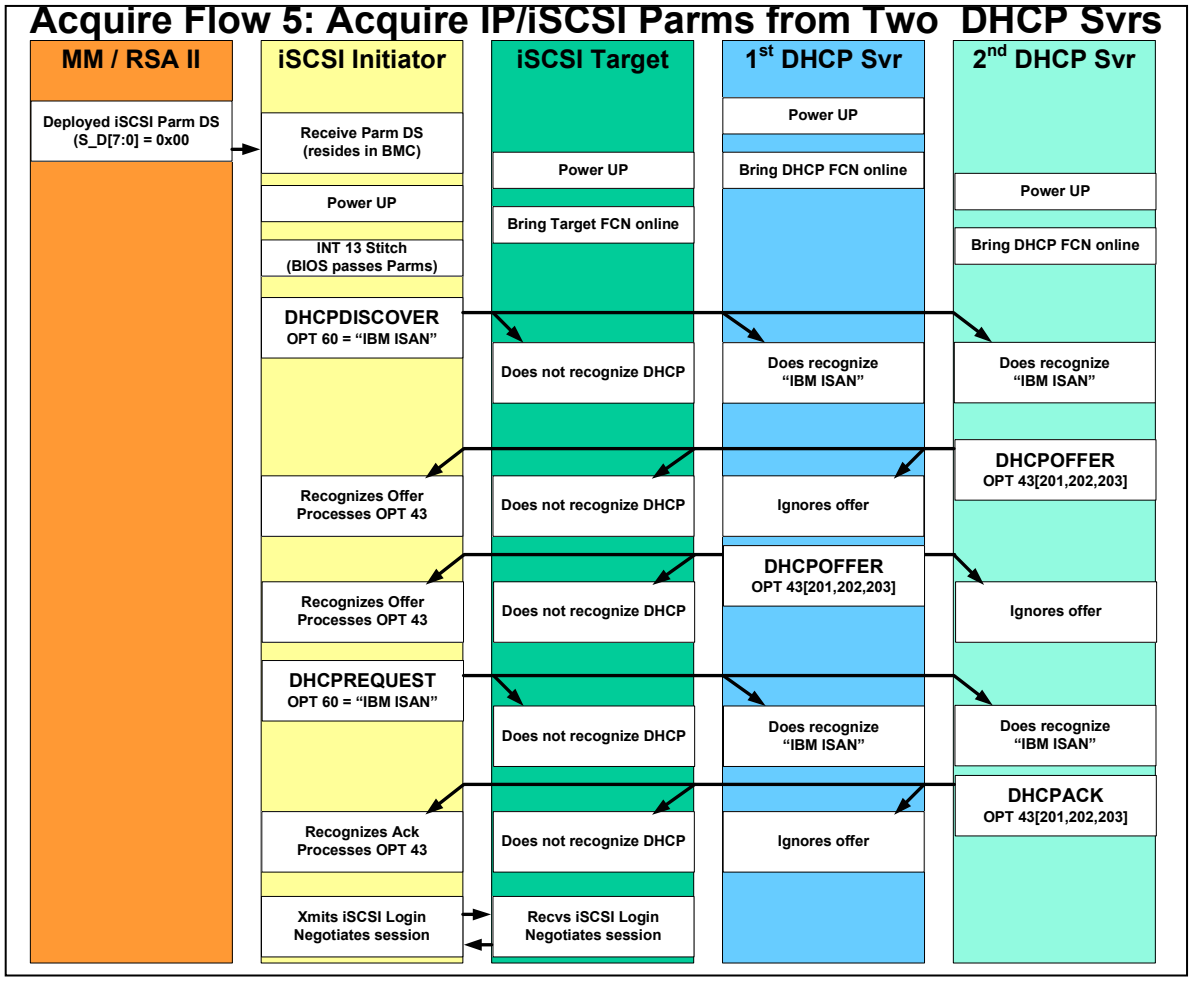

Figure 11: Acquire Flow 5: Multiple DHCP Offers to Acquire iSCSI Parameters

Chapter 8. iSCSI Security Contexts

To aid the reader in understanding the security implications of iSCSI parameters, this section pulls the security aspects into a single section.

Out of Band Security Context for Parameter Deployment

The security model for iSCSI parameters is comprised of the ideal case, supporting full encryption via SSL, and the minimal case, supporting authenticated logins. Below is the security model for iSCSI parameters deployed out of band.

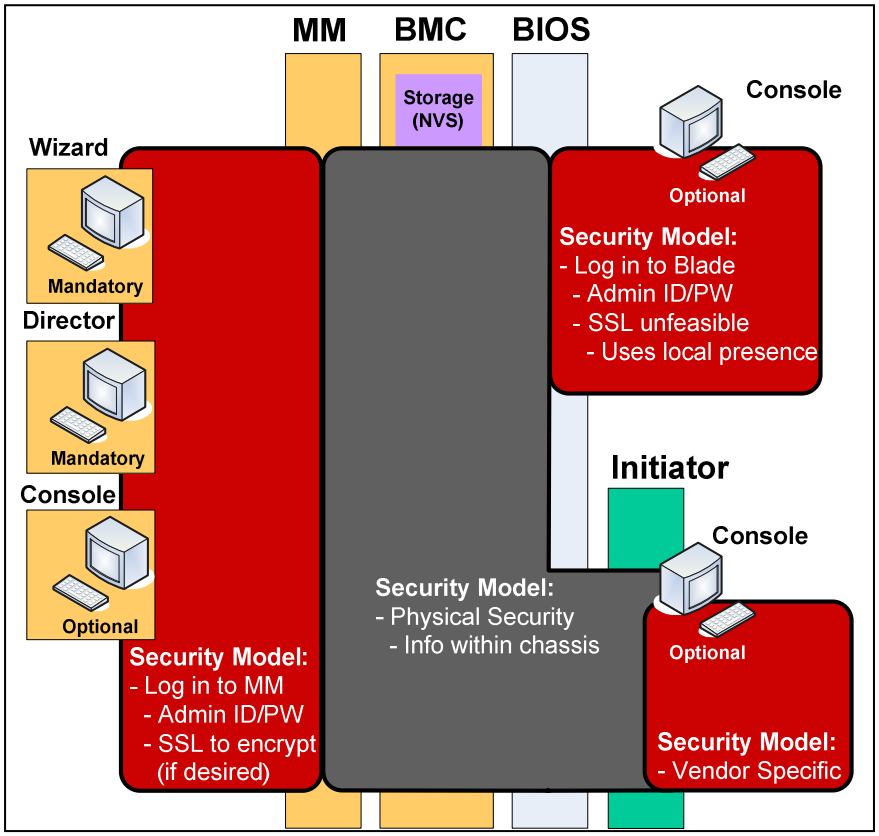

Figure 12: iSCSI Parameter Security Model

Ideal Security Processing

In the ideal case, the establishment of a SSL connection between the management module or RSA II adapter and external configuration tools such as Director or storage wizard provides an encrypted path on any network. With encryption, no nefarious or rogue users can "snoop" iSCSI parameters and use them to gain access to the iSCSI target. Moreover, any "power" users with access to the network can still be prevented from seeing iSCSI parameters due to explicitly control of the SSL keys used to establish the SSL connection.

Minimal Security Processing

In the minimal case, an authentication scheme on a physically secure network may be sufficient in some deployments. In this minimal case, the security model relies on the management module or RSA II access being on a network that is secure in terms of visibility to the users or external world. With that in place, encryption is not required since its assume any user with access to this secure network as the rights to see and use the iSCSI parameters.

BIOS Security Processing

SSL support for BIOS is not required due to the fact that the only way to gain access via the BIOS path is through local interface (via direct F1/BIOS panels or via KVM redirect through the management module or RSA II). In the direct access path, the physical locality removes the SSL requirement since any unauthorized access would have to be done physically local. In the KVM redirect, the Management module or RSA II can provide the SSL support outside the physical chassis.

Preview of Emerging BMC Security Processing

While the architecture is not completed yet, there is an effort to introduce authentication into the BMC service processor subsystem. The outlook is for the BMC to support CHAP authentication against specific regions of non volatile space. Moreover, one ID and PW combination will be managed for the blade internal access from such services at BIOS or applications and a second ID and PW combination will be managed for blade external access from such services as MM or director. Later revisions of this document will detail the BMC security behavior in the context of iSCSI parameter configuration.

Initiator Security Processing

For initiators providing configuration tools to configure the initiator, it is the responsibility of the initiator implementor to define the security model. For all physically local accesses, this can be a simple as relying on physical security of the system (aka panels local or redirected through the management module or RSA II SSL pipe). For any remote access via an initiator vendor network tool suite, the appropriate security should be introduced in a prudent fashion. Given the variety of iSCSI initiator vendors and their respective rollout schedules, it is beyond the scope of this document to define the exact approach to be taken. In general, the initiator vendors should pursue a secure model for network tool suites that provide similar capabilities to SSL.

Inband Security Context for Parameter Deployment

Given that inband deployment is on a open network, there are no security parameters intended to be used that are from the DHCP server. If there are security parameters only deployed out of band then the out of band security model implementation should be used.

Security Implications for Parameter Deployment

It is important to highlight several implications of this security model.

First, a fully secure configuration of iSCSI is dependent on the management module and/or RSAII as well as the appropriate iSCSI configuration tools to have SSL support. Moreover, the appropriate key management must be in place to ensure keys are accurate and secure. It should be noted that a SSL capability on these components is desirable for more than just iSCSI.

Second, since it is assumed that any physical tapping to networks completely contained within BladeCetner or server, the physical security of the “chassis walls” allows for unsecure transactions completely contained within the given chassis.

Third, it is beyond the scope of this document to define the security model for a given iSCSI target. Given that given solution may have multiple targets from different vendors on different security models or have different timeframes for introducing robust security models, it would be difficult in this document to state the security models for all the flavors of iSCSI targets available. It should be noted that if a robust security model is desired, the appropriate targets should be used.

Out of Band Security Context for iSCSI Login

Since iSCSI initiators and targets can reside on large networks and, thus, be visible to all network nodes, the iSCSI standard provides several security contexts to ensure levels of security. The most basic level, called unsecure iSCSI session, entails not using any security information either at log in or during packet transfer. The intermediate level, called authenticated iSCSI session, entails using authentication via CHAP to validate the iSCSI initiator and iSCSI target are verified at log in with packet transfers in the form. The advanced level, called encrypted iSCSI session, entails using encryption via IPSEC to verify the iSCSI initiator and iSCSI target as well as to encrypt each packet transfer between them.

Unsecure iSCSI Session

In an unsecure iSCSI session, no effort is made to validate the session parties on either end nor is there any effort to conceal or obscure the actual data in the packet transfers. Clearly, there is a security exposure with an unsecure iSCSI session. However, during initial solution development, an unsecure iSCSI session may make development and debug easier.

Authenticated iSCSI Session

In an authenticated iSCSI session, the iSCSI initiator and iSCSI target are validated. While several industry standard approaches such as SRP or Kerberos are available, Challenge Handshake Authentication Protocol (CHAP) is most prevalent and, in turn, implemented in this document. There are two forms of CHAP authentication supported here. One way CHAP involves the iSCSI target authenticating the iSCSI initiator at session establishment. Mutual CHAP involve the iSCSI target authenticating the iSCSI initiator as well as the iSCSI initiator authenticating the iSCSI target at session establishment.

One Way CHAP

For one way CHAP, during the iSCSI session negotiation, the iSCSI initiator and iSCSI target use several parameters to prove thevalidity of the iSCSI initiator to the iSCSI target. In essence, there are three values required for one way CHAP: Initiator ID (ISDI[15:0][7:0]), Initiator password (ISDP[23:0][7:0], and a random value (some random number of at least 96 bits). These three values are one way hashed via the MD5 algorithm to develop a remainder. Note that a one way hash such as MD5 indicates a function that generates a unique value such that switching the unique value with any single input will generate yet another unique value and thus, hide or obscure the at least one of the original inputs from any hacking attacks. The figure below depicts the flow for one way CHAP.

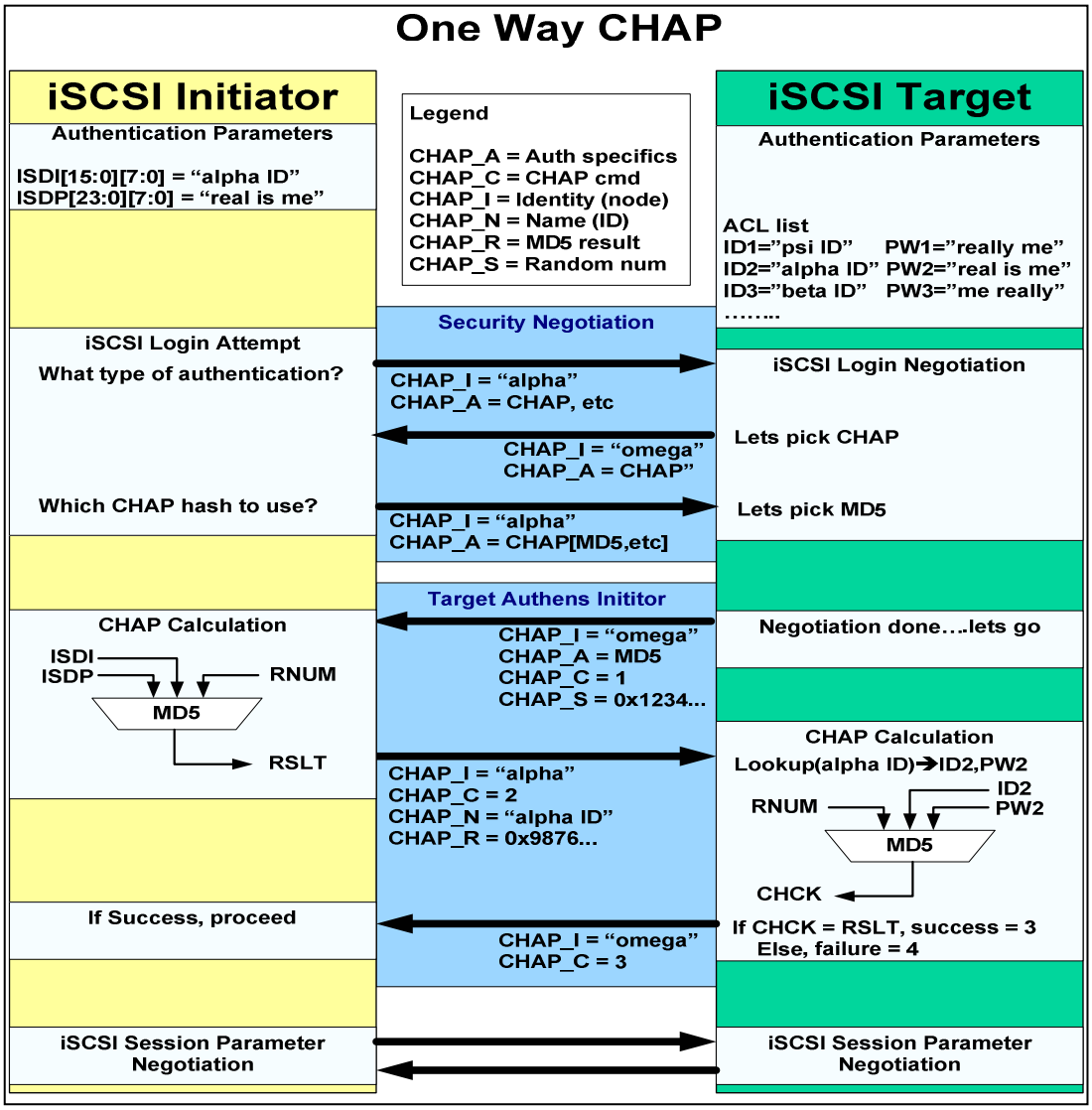

Figure 13: One Way CHAP Flow

Mutual CHAP

For Mutual CHAP, during the iSCSI session negotiation, the iSCSI initiator and iSCSI target use several parameters to prove validity of the iSCSI initiator to the iSCSI target and to prove the validity of the

iSCSI target to the iSCSI Initiator. To prove the validation in both directions, there are two CHAP flows. The initiator to target flow is similar to one way CHAP. The target to initiator flow a reflection of the initiator to target flow. In essence, there are three values required to validate the initiator to the target: Initiator ID (ISDI[15:0][7:0]), Initiator password (ISDP[23:0][7:0], and a random value (some random number of at least 96 bits). These three values are one way hashed via the MD5 algorithm to develop a remainder. Similarly, to prove the validity of the target to the initiator, there are three values required: Target ID for first target (T1SDI[15:0][7:0]), Target PW for first target (T1SDP[23:0][7:0]), and a random value (some random number of at least 96 bits). If the iSCSI session is destined for the second target, then T2SDI, T2SDP, and random value is used. Note that a one way hash such as MD5 indicates a function that generates a unique value such that switching the unique value with any single input will generate yet another unique value and thus, hide or obscure the at least one of the original inputs from any hacking attacks. The figure below depicts the flow for mutual CHAP where the initiator to target and target to initiator flows are segmented for clarity.

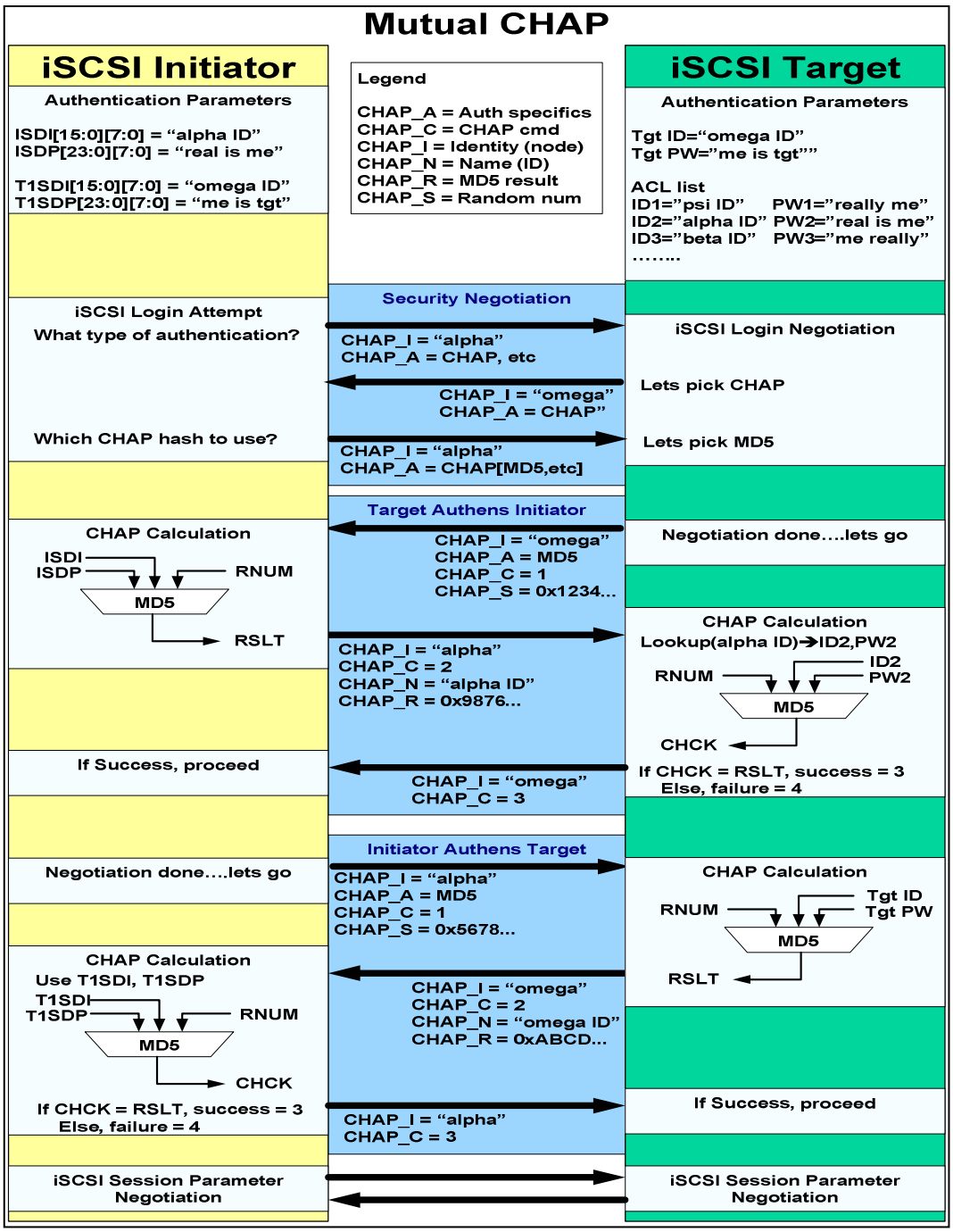

Figure 14: Mutual CHAP Flow

Encrypted iSCSI Session

In future versions of this document, a detailed look at iSCSI session encryption will be presented.

Chapter 9. Solution Usage of Parameters and Error Scenarios

The solution usage and error behaviors help to enumerate the various operating behaviors. Given the security issues, the out of band and inband usage and error scenarios are discussed separately. The intent of this section is to aid implementers on the various modes operation that should be supported as well as the various error behaviors that should be supported.

Note next revision will fold in error codes for xSeries BIOS

Error Code Definition

With a variety of initiator implementations and a variety of platforms, it is difficult to specify all possible ways a given initiator can respond to the calling platform with completely common set of error codes using a singularly universal mechanism to respond. However, it is possible to specify generalized set of codes that can be used to in the actual initiator/platform error processing. Below is the list of error codes that a given initiator should use to inform the calling platform that some sort of error occurred during operation.

Error Code	Definition
00d	Successful (or not attempted)
80d	Invalid parameter structure
81d	Unsupported parameter version
82d	No network connection
83d	No target response (timeout)
84d	Target found, but not ready
85d	No DHCP server response
86d	No valid DHCP data found
87d	Target rejected initiator IQN
88d	Reserved
89d	CHAP login failure
90d	Target CHAP identification failure
91d	Other communication error
100d	Unsupported DHCP Option combination
101d	DHCP Option parameter format error
102d	Incompatible Option 20x and Parameter Option bits
103d	General DHCP Option error
FFh	Not used – adapter BIOS override used

Table 91: iSCSI Initiator Error Code Definition

Note that it is beyond the scope of this document to describe, in detail, how a given initiator passes the error code up to the given platform. In general, for xSeries platforms, the error code should be returned in the AX register where AH represents the primary or 1st target status while the AL represents the secondary or 2nd target status. For example, if a given initiator incurs CHAP login failures on both targets, AH=89d and AL=89d.

Out of Band Usage

In general, the out of band deployment scenario is well controlled and the control of option bits versus actual parameters present is well defined. Recall that there is a spectrum of usage models to consider covering, at the one extreme, all the iSCSI parameters are supplied out of band and, at the other extreme, some if not none of the iSCSI parameters are supplied out of band.

It is envisioned that the early implementers will accept ALL the parameters from BIOS via out of band deployment or ALL the parameters from a DHCP server via inband deployment. As implementations evolve, the various hybrid usage models will be added to this section (they are stated elsewhere already – See the definition of the S_D field).

Out of Band Error Scenarios

There are 4 classes of checks that are performed with some required and other optional. The four classes consist of structure checks, basic checks, security checks, and advanced checks. Structure checks are focused on making sure the data structure received from BIOS is correct (BIOS may or may not do its own set of checks on the structure). Basic checks are focused on the minimum needed to set up connections to the targets (no security or advanced discovery are check at this point). Security checks are focused on ensuring that the option bits settings defined for security agree and support the actual parameters present. Advanced checks are focused for any advance discovery implementations such as a unicast to a DHCP server to reduce 'broadcast storms" or the inclusion of iSNS support.

The structure checks and the basic checks are required since they are needed regardless of the implementation or mode of usage. Security and advance checks are only if security or advanced discovery is used. If neither of this functions are used, the default settings for each parameter and option bit should be used to ensure default correctness. As a signal to implementer on whether security and advanced discovery checks must be done or an error posted, refer to the table below where several initial checks will indicate if additional checks are required or an error posted because the implementation does not support additional checks involved. For example, consider advance checks involving the discovery IP address. Here if a check of the S_D field indicates S_D[7:0]=0x00, then no further advanced checks are needed since all parameters come from DHCP server

```
If (signature != "ISAN" at designated location) error return error code = 80d
```

Figure 15: Signature Error Checking

```
If (VER[7:0] = N and initiator supports N-1 or less) error return error code 81d
If (VER[7:0] = N and initiator supports N) continue processing
If (VER[7:0] = N-1 and initiator supports N) continue processing      //Initinator should support N and N-1
If (VER[7:0] = N-2 or less and initiator supports N) error return error code 81d
```

Figure 16: Version Error Checking

```
If (LVL[7:0] = N and initiator supports N-1 or less) error return error code 81d
If (LVL[7:0] = N and initiator supports N) continue processing
If (LVL[7:0] = N-1 and initiator supports N) continue processing      //Initinator should support N and N-1
If (LVL[7:0] = N-2 or less and initiator supports N) error return error code 81d
```

Figure 17: Level Error Checking

```
If(LEN[7:0] = M & Byte Offset [M-1] = 0x00) continue processing
If(LEN[7:0] = M & Byte Offset [M-1] != 0x00) error return error code 80d          //length does not match data
```

Figure 18: Length Error Checking

```
If((CHK[7:0] xor (xor of bytes[LEN[7:0])) = 0x00) continue processing
If((CHK[7:0] xor (xor of bytes[LEN[7:0])) != 0x00) error return error code 80d          //data not valid
```

Figure 19: Checksum Error Checking

```
If(S_D[7:0]=0x04 & IIP[63:0] = 0x00) error return error code 80d          //hybrids come later
If(S_D[7:0]=0x04 & IQN[71:0] = 0x00) error return error code 80d          //hybrids come later
If(S_D[7:0]=0x04 & MSK[63:0] = 0x00) error return error code 80d          //hybrids come later
If(S_D[7:0]=0x04 & GRIP[63:0] = 0x00) error return error code 80d         //hybrids come later
If(S_D[7:0]=0x04 & T1IP[63:0] = 0x00) error return error code 80d         //hybrids come later
If(S_D[7:0]=0x04 & T1IQN[71:0] = 0x00) error return error code 80d        //hybrids come later
If(S_D[7:0]=0x04 & T1PT[15:0] = 0x00) error return error code 80d         //hybrids come later
If(S_D[7:0]=0x04 & T1BL[7:0] = 0x00 & OPT[29] = 0b0) error return error code 80d          //hybrids come later
If(S_D[7:0]=0x04 & T2IP[63:0] = 0x00) error return error code 80d         //hybrids come later
If(S_D[7:0]=0x04 & T2IQN[71:0] = 0x00) error return error code 80d        //hybrids come later
If(S_D[7:0]=0x04 & T2PT[15:0] = 0x00) error return error code 80d         //hybrids come later
If(S_D[7:0]=0x04 & T2BL[7:0] = 0x00 & OPT[29] = 0b0) error return error code 80d          //hybrids come later
```

Figure 20: Parameter Presence Error Checking

```
If(OPT[27:24] = 0b0101 & S_D[7:0] = 0x04 & T1xxx missing) error return error code 80d    //no target chap
If(OPT[27:24] = 0b0111 & S_D[7:0] = 0x04 & T1xxx missing | Ixxx missing ) error return error code 80d
                                                                                          //no target or init
If(OPT[27:24] = 0b1xxx & T1 EPIDs missing) error return error code 80d    //no target for IPSEC
```

Figure 21: Option bits 27:24 Error Checking

```
If(OPT[21:18] = 0b0101 & T2xxx missing) error return error code 80d    //no target for one way chap
If(OPT[21:18] = 0b0111 & T2xxx missing | Ixxx missing) error return error code 80d    //no target or init
If(OPT[21:18] = 0b1xxx & T2 EPIDs missing) error return error code 80d    //no target for IPSEC
```

Figure 22: Option bits 21:18 Error Checking

```
If(OPT[15:14] = 0b1x & T1xxx missing) error return error code 80d
If(OPT[15:14] = 0bx1 & T2xxx missing) error return error code 80d
If(OPT[15:14] = 0b00 & S_D[7:0] = 0x04) error return error code 80d      /no tgts present and no dhcp
```

Figure 23: Option bits 15:14 Error Checking

```
//advanced check
If (OPT[31] = 0b0 & IIP[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b0 & MSK[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b0 & GRIP[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b0 & T1IP[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b0 & T2IP[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b0 & DIP[63:32] != 0x00) error return error code 80d
If (OPT[31] = 0b1 & initiator does not support IPv6) error return error code 80d
```

Figure 24: Option bit 31 Error Checking

```
//advanced check
If(OPT[13:12] = 0b01 & S_D[7:0]=0x04 & SVID[63:0] = 0x00) error return code 80d
If(OPT[13:12] = 0b10 & S_D[7:0]=0x04 & CAID[63:0] = 0x00) error return code 80d
```

Figure 25: Option bits 13:12 Error Checking

```
//advanced check
If(OPT[11:10] = 0b01 & S_D[7:0]=0x04 & SVID[63:0] = 0x00) error return code 80d
If(OPT[11:10] = 0b10 & S_D[7:0]=0x04 & CAID[63:0] = 0x00) error return code 80d
```

Figure 26: Option bits 11:10 Error Checking

```
//advanced check
If(OPT[9:8] = 0b00 & S_D[7:0]=0x04 & DIP[63:0] = 0x00) error return code 80d
If(OPT[9:8] = 0b01 & S_D[7:0]=0x04 & DIP[63:0] = 0x00 | init does not support SLP) error return code 80d
If(OPT[9:8] = 0b10 & S_D[7:0]=0x04 & DIP[63:0] = 0x00)| init does not support iSNS)error return code 80d
```

Figure 27: Option bits 9:8 Error Checking

Inband Usage

From a usage perspective, the inband deployment can be viewed as three approaches. First, due to legacy needs, a hostname / Option 17 approach can be used to provide the bare minimum support (common hostname for client, one targe only connectivity, no security, etc). Second, due to the need of more robustness, the options 203 and options 202 and/or 201 can be used to provide better support (initiator name unique to session, 2 targets, etc). Note, it is assumed that either hostname/option 17 OR options 203,202,201 are used…but not both for iSCSI. This either or approach should ease the parsing and checking involved. Third, in advanced solutions, options 204 through 206 as well as option 203 and 201 and/or 202 can be used. Note that it is assumed that if a solution deployment is implementing either option 204 or 205 or 206, then any of the three can be configured and used again to ease the parsing and checking burden by the initiators and to allow the admin to evolve the solution over time.

Inband Error Scenarios

In general, to aid the implementers of inband deployment, a hierarchy of checks can ease the determination of good DHCP parameters. In general, either the hostname/option 17 OR option 203 should be present and well formed. Assuming the 203 case further, either 201 OR 202 should be present and well formed with 203. Lastly, if an implementer wishes to use the 204 through 206, then ANY of three should be present and well formed. This approach allows the implementer and admin to evolve the solution as their understanding grows.

Note that well formed simply means the actual format of the option are correct. For example, in 201 or 202, a well formed option would have the markers, such as “iSCSI” or “:”s, correctly in place. Similarly, for example, option 205 would have its markers, such as “B0:” or “:B2:” correctly in place. Note DHCP option 205 is truly optional since it currently conveys no information.

Also, note that a given implementation may inform the DHCP server if it can process the options specified. This can be done in the DHCP Option 60 string as defined earlier. While the DHCP server may substring the ‘vendor identication ID”, any debug or network sniff efforts can see the entire string (or one could have the DHCP serve print the entire string to a log file). Of course, in advanced DHCP configuration file implementations, the entire Option 60 string can be used to provide all or some of the 20x options.

Inband Psuedo Code for DHCP Behavior

Since nominal flow and error conditions are tightly interlocked, this section presents pseudo code describing, in detail, the flows, checks, and error conditions that may be encountered while acquiring iSCSI parameters from DHCP services. There are a total of 12 snippets of pseudo code presented. The first snippet defines some structures used to make the remaining snippets more readable as well as presenting the overall flow of the acquisition and usage of the iSCSI parameters.. Snippet two presents the flow for DHCPDISCOVER/DHCPREQUEST and can be viewed as the “main” function snippet pulling together the most of the remaining snippets. Snippet three presents the flow of DHCPINFORM. Snippets four through eleven present the flows for checking the proper syntax of iSCSI DHCP options (17/hostname as well as 201 through 206). The last snippet presents the flow, using the acquired iSCSI parameters, for establishing the iSCSI session with the designated target.

Note that the code snippets below are intended to provide a conceptual flow of the decision/assessment process. These snippets are not intended to be extracted as is and used in actual code. In the future, an appendix will provide actual code that can be used if a given iSCSI initiator implementer wishes to.

```
// Conceptual data structures for ease of reading

struct DHCP_REPLY{
    opt43;                                  //encapsulated vendor parms in 20x context
    opt17;                                  //target in 17/hname context
    hname;                                  //initiator in 17/hname context
    client_ip;                              //client IP address
    subnet;                                 //network subnet
    gway_ip;                                //gateway or router address
    //other stuff
}

struct DHCP_REPLY reply;                    //working reply


struct ISCSI_PARMS {                        //data structure for the iscsi parms
    //option 20x
    opt201;                                 //first target
    opt202;                                 //second target
    opt203;                                 // initiator
    opt204;                                 //discovery
    opt205;                                 //option bits – value not currently specified
    opt206;                                 //retry/timout
    //option 17 and host name
    opt 17;                                 //target
    hname;                                  //initiator
    //…….other stuff
}

Struct ISCSI_PARMS parms;                   //individual iscsi parms

STATUS UseiSCSI(){
    GetParametersfromDHCP();                //Start the DHCP activities to acquire parms
    StartiSCSISession();                    //establish actual iSCSI with acquired parms
    If(iSCSISessionGood()) {
       return SUCCESS;                      //session establish…congrats
    }
    else {
       return ERROR;                        //session not established error
    }
}
```

Figure 28: Conceptual Data Structures for DHCP/iSCSI Parameters

```
//Conceptual function for acquiring iSCSI parameters from DHCP using DHCPDISCOVER/DHCPREQUEST

STATUS GetParametersFromDhcp ()
{
   for (j=0 ; j < DHCP_DISCOVER_RETRIES + 1; j++) {                    //Iterate to find DHCP server
       SendDhcpDiscover();                                             //Send DHCPDISCOVER to solicit DHCP servers
       GoodOffer = FALSE;
       while (!DiscoveryTimeout()) {                                   //Bound the waiting for DHCPOFFER
            reply = GetPacketFromNIC();                                //Get DHCPOFFER from NIC layer
            if (reply == 0) continue;                                  //no pkt yet

            if (IsValidDHCPOffer(reply) &&
                ( HasOpt43(reply) || HasOpt17andHname(reply) )) {      //Got a good offer with 43 or 17/hname
                  GoodOffer = TRUE;                                    //Good offer…jump out of loop
                  break;
            }
       }

     if (GoodOffer == FALSE) continue;                                 //Tried several times…no good offers..punt

     /* Got a good offer, sending the request */
     SendDhcpRequest();                                                //send DHCPREQUEST to signal selected DHCP svr
     while (!RequestTimeout()) {
         reply = GetPacketFromNIC();                                   //Get pkt…chk if a DHCP packet
         if (reply == 0) continue;                                     //no reply yet

         if (IsValidDHCPAck(reply)) {                                  //check DHCP is DHCPACK
             if (!(HasOpt43(reply) && !(HasOpt17andHname(reply))) {    //Neither opt 43 nor opt 17/hname present
                  return ERROR;                                        //With no parms..need to exit
             }
             If ((HasOpt43(reply) && (HasOpt17asiSCSI(reply))) {       //Both opt 43 and iSCSI form of 17 present
                  return ERROR;                                        //With parm conflict…need to exit
             }
             //////////////////////
             // ruled out 17 vs 43 invalid combos…so now lets proceed
             //////////////////////
             SetIPMaskGway(reply);                                     //Set client IP addr, subnet, gateway, etc
             if ((HasOpt43(reply) && !(HasOpt17asiSCSI(reply))) {      //Ensure opt 43 and not opt 17/hname present
                  CopyFromREPLYtoPARMs();                              //extract options out of reply and copy to parms
                  if (HasOpt204(parms)) {                              //take 204 if present
                       if (check_204_syntax(parms.opt204)) {
                            HandleDHCPInform();                        //use DHCPINFORM to get 201/202/203
                                                                       //parms is updated per options present in inform
                       }
                  }
                  ////////////////////////////////////////////////////////////////////////////////////////////////////
                  //now have the most current set of parameters either from OFFER or from INFORM
                  ////////////////////////////////////////////////////////////////////////////////////////////////////
                  If (HasOpt206(parms)){
                     If(check_206_syntax(parms.opt206))                //good 206 syntax
                        UseOption206IfExist(parms);                    //get 206 if present and well formed
                  }
                  If (HasOpt205(parms)){
                     If(check_205_syntax(parms.opt205))                //good 205 syntax – note values unspecified
                        UseOption205IfExist(parms);                    //take 205 if present    - note values unspecified
                  }
                  If (HasOpt203(parms)){
                     If(check_203_syntax(parms.opt203))                //good 203 syntax
                        UseOptions203(parms);                          //get 203 current reply and well formed
                  }
                  If (HasOpt201(parms)){
                     If(check_201_syntax(parms.opt201))                //good 201 syntax
                        UseOption201IfExist(parms);                    //get 201 if exist and well formed
                  }
```

```
                If (HasOpt202(parms)){
                   If(check_202_syntax(parms.opt202))                    //good 202 syntax
                      UseOption202IfExist(parms);                        //get 202 if exist and well formed
                }
                if (HasOpt203(parms) && ( HasOpt201(parms) || HasOpt202(parms) ) {
                    return SUCCESS;                                      //if (( 203 & (201 | 202)) then got parms proceed
                }
                else {
                    return ERROR;                                        //don't have enough parms…punt
                }
            if (!(HasOpt43(reply)) && (HasOpt17asiSCSI(reply))) {        //Ensure not opt 43 and opt 17/hname present
                CopyFromREPLYtoPARMs();                                  //extract options out of reply and copy to parms
                if (check_17_syntax(parms.opt17))
                    TakeOption17(parms);   //get 17 if exist and well formed
                if (check_hname_syntax(parms.hname))                     /get hname if exist and well formed
                    TakeHname(parms);
                If (HasOpt17(parms) && HasHname(parms)){
                    return SUCCESS;                                      //got 17 and hname…go ahead
                else {
                    return ERROR;                                        //don't have enough parms…punt
                }
            }
        }                                                                // while - wait for ACK
    }                                                                    // for - DHCP DISCOVER

   return ERROR;
}
```

Figure 29: Pseudo Code for Using DHCPDISCOVER/DHCPREQUEST

```
// conceptual function for acquiring iSCSI parameters from DHCP using DHCPINFORM


STATUS HandleDHCPInform()
{
      ////////////////////////////////
      // check of the vendor scope definition has changed
      ////////////////////////////////
      scope = parms.opt204.scope;
      if (scope != "IBM ISAN") dhcp_option_60 = scope                    // if not "IBM ISAN"…customer has changed it
      else dhcp_option_60 = "IBM ISAN";


      ////////////////////////////////
      // check if the client id definition has been changed…note that windows really does not support
      ////////////////////////////////
      client_id = parms.opt204.id
      if (client id != "null"){                                          // if not "null"….customer has changed it
          dhcp_option_61 = Load61with0PlusIDstr();                       // its 0 + id…aka…0x00+"svr 0001"
      }
      else{
          dhcp_option_61 = Load61with1PlusMAChex();                      //else it's 1+ MAC …aka…0x01aabbccddeeff
      }


      ////////////////////////////////
      // check if discovery ip addr is defined
      ////////////////////////////////
      discoveryIP = parms.opt204.disc_ip;
      if (DiscoveryIP != 0.0.0.0){                                       //test for valid discovery ip addr
         if (DiscoveryIP == 255.255.255.255) {
            SendDhcpInformByBroadcast();                                 //255.255.255.255 signals broadcast
       } else {
            SendDhcpInformByUnicast();                                   //other signals unicast
       }
     Else return ERROR;                                                  /0.0.0.0 not valid

for (retry = 1 + DHCP_INFORM_RETRIES; retry > 0; retry--) {              //try inform several times
     while (!InformTimeout()) {
         reply = GetPacketFromNIC();
         if (reply == 0) continue;

         if (IsValidDHCPInformReply() && HasOption43() {
             ParsenPlaceOpt206IfExist(reply, parms);                     //place 206 – if present – into parms
             ParsenPlaceOpt205IfExist(reply, parms);                     //place 205 – if present – into parms
             ParsenPlaceOpt203IfExist(reply, parms);                     //place initiator into parms
             ParsenPlaceOpt201IfExist(reply, parms);                     //place first target – if present – into parms
             ParsenPlaceOpt202IfExist(reply, parms);                     //place second target – if present – into parms
             return SUCCESS;                                             // 43 is in place, parms updated…return to use values
         }
     }                                                                   // wait for ack
  }                                                                      // for retry DHCPINFORM
     return ERROR;
}
```

Figure 30: Pseudo Code for Using DHCPINFORM

```
bool check_17_syntax (dhcp_option_17)
{
    //checking option 17 syntax
    // note that the format of option 17 is "iscsi:< target>:<protocol>:<port>:<lun>:<iscsi name>"
    // for example….“iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1”
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 17;
    char tmp_delim = ":";
    char *token_type;
    char *token_target;
    char *token_protocol;
    char *token_port;
    char *token_lun;
    char *token_name;

    int check_17_syntax_format_index = 0;

    token_type = strtok(tmp_option, tmp_delim);                    //peel off type or class…e.g. iscsi
    token_target = strtok(null,tmp_delim);                         //peel off “aaa.bbb.ccc.ddd” ..e.g. 123.123.123.123
    token_protocol = strtok(null, tmp_delim);                      //peel off protocol…e.g.6
    token_port = strtok(null,tmp_delim);                           //peel off tcp port….e.g. 3260
    token_lun = strtok(null, tmp_delim);                           //peel off lun number…e.g. 0
    token_name = strtok(null,tmp_delim);                           //peel off iscsi name…e.g.iqn.2005-03.ibm.com.lun_a1

    if (token_type == “iscsi”) check_17_syntax_format_index++;
    if (token_target is a valid ip address format) check_17_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the ip address
        ///////////////
    if (token_protocol == 6) check_17_syntax_format_index++;
    if (token_port is a valid tcp port format) check_17_syntax_format_index++;
    if (token_lun is less than 255”) check_17_syntax_format_index++;
    if (token_name is a valid iscsi iqn format) check_17_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the iqn name specifically
        ///////////////

    if (check_17_syntax_format_index == 6)                         //6 means all parts are well formed
      {
        return(true);                                              //valid option 17 format…proceed
      }
    else
      {
        return(false);                                             //error invalid format…defer to caller on go or no go
      }
}
```

Figure 31: DHCP Option 17 Syntax Checking

```
Bool check_12_syntax (DHCP option 12)
{
    //checking option 12 syntax
    // note that the format of option 12 is "<host name>"
    // for example…."blade_a"
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    If (dhcp_option 12 is an ascii string)                    //host name is well formed
      {
         return(true);                                        //valid option 12 format…proceed
      }
    else
      {
         return(false);                                       //error invalid format…defer to caller on go or no go
      }
}
```

Figure 32: DHCP Option 12 Syntax Checking

```
bool check_201_syntax (dhcp_option_201)
{
    //checking option 201 syntax
    // note that the format of option 201 is "iscsi:< target>:<protocol>:<port>:<lun>:<iscsi name>"
    // for example….“iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1”
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 201;
    char tmp_delim = ":";
    char *token_type;
    char *token_target;
    char *token_protocol;
    char *token_port;
    char *token_lun;
    char *token_name;

    int check_201_syntax_format_index = 0;

    token_type = strtok(tmp_option, tmp_delim);                    //peel off type or class…e.g. iscsi
    token_target = strtok(null,tmp_delim);                         //peel off “aaa.bbb.ccc.ddd” ..e.g. 123.123.123.123
    token_protocol = strtok(null, tmp_delim);                      //peel off protocol…e.g.6
    token_port = strtok(null,tmp_delim);                           //peel off tcp port….e.g. 3260
    token_lun = strtok(null, tmp_delim);                           //peel off lun number…e.g. 0
    token_name = strtok(null,tmp_delim);                           //peel off iscsi name…e.g.iqn.2005-03.ibm.com.lun_a1

    if (token_type == “iscsi”) check_201_syntax_format_index++;
    if (token_target is a valid ip address format) check_201_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the ip address
        ///////////////
    if (token_protocol == 6) check_201_syntax_format_index++;
    if (token_port is a valid tcp port format) check_201_syntax_format_index++;
    if (token_lun is less than 255”) check_201_syntax_format_index++;
    if (token_name is a valid iscsi iqn format) check_201_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the iqn name specifically
        ///////////////

    if (check_201_syntax_format_index == 6)                        //6 means 201 is well formed
      {
        return(true);                                              //valid option 201 format…proceed
      }
    else
      {
        return(false);                                             //error invalid format…defer to caller on go or no go
      }
}
```

Figure 33: DHCP Option 201 Syntax Checking

```
bool check_202_syntax (dhcp_option_202)
{
    //checking option 202 syntax
    // note that the format of option 202 is "iscsi:< target>:<protocol>:<port>:<lun>:<iscsi name>"
    // for example…."iscsi:123.123.123.20:6:3260:0:iqn.2005-03.com.ibm.lun_a1"
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 202;
    char tmp_delim = ":";
    char *token_type;
    char *token_target;
    char *token_protocol;
    char *token_port;
    char *token_lun;
    char *token_name;

    int check_202_syntax_format_index = 0;

    token_type = strtok(tmp_option, tmp_delim);                    //peel off type or class…e.g. iscsi
    token_target = strtok(null,tmp_delim);                         //peel off "aaa.bbb.ccc.ddd" ..e.g. 123.123.123.123
    token_protocol = strtok(null, tmp_delim);                      //peel off protocol…e.g.6
    token_port = strtok(null,tmp_delim);                           //peel off tcp port….e.g. 3260
    token_lun = strtok(null, tmp_delim);                           //peel off lun number…e.g. 0
    token_name = strtok(null,tmp_delim);                           //peel off iscsi name…e.g.iqn.2005-03.ibm.com.lun_a1

    if (token_type == "iscsi") check_202_syntax_format_index++;
    if (token_target is a valid ip address format) check_202_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the ip address
        ///////////////
    if (token_protocol == 6) check_202_syntax_format_index++;
    if (token_port is a valid tcp port format) check_202_syntax_format_index++;
    if (token_lun is less than 255") check_202_syntax_format_index++;
    if (token_name is a valid iscsi iqn format) check_202_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the iqn name specifically
        ///////////////

    if (check_202_syntax_format_index == 6)                        //6 means 202 is well formed
      {
        return(true);                                              //valid option 202 format…proceed
      }
    else
      {
        return(false);                                             //error invalid format…defer to caller on go or no go
      }
}
```

Figure 34: DHCP Option 202 Syntax Checking

```
bool check_203_syntax (dhcp_option_203)
{
    //checking option 203 syntax
    // note that the format of option 203 is " :<iscsi name>"
    // for example…."iqn.2005-03.com.ibm.blade_a"
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 203;
    char tmp_delim = ":";
    char *token_name;

    int check_203_syntax_format_index = 0;

    if (token_option is a valid iscsi iqn format) check_203_syntax_format_index++;
        ///////////////
        //note here is where another strtok sequence could be used to parse/build the iqn name specifically
        ////////////

    if (check_203_syntax_format_index == 1)                    //1 means 203 is well formed
      {
        return(true);                                          //valid option 203 format…proceed
      }
    else
      {
        return(false);                                         //error invalid format…defer to caller on go or no go
      }
}
```

Figure 35: DHCP Option 203 Syntax Checking

```
bool check_204_syntax (dhcp_option_204)
{
    //checking option 204 syntax
    // note that the format of option 204 is “scope:<scope>:disc:<dhcp>:id:<client alt id>:flags:<flag bits>”
    // for example….“scope:xyz isan:disc:123.123.123.123:id:svr 0001:flags:1a9”
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 204;
    char tmp_delim = “:”;
    char *token_scope;
    char *token_scope_name;
    char *token_disc;
    char *token_disc_value;
    char *token_id;
    char *token_id_name;
    char *token_flag;
    char *token_flag_bits;

    int check_204_syntax_format_index = 0;

    token_scope = strtok(tmp_option, tmp_delim);                    //peel off type or class…e.g. scope
    token_scope_name = strtok(null,tmp_delim);                      //peel off scope value ..e.g. xyz_isan
    token_disc = strtok(null, tmp_delim);                           //peel off disc…e.g. disc
    token_disc_value = strtok(null,tmp_delim);                      //peel off disc value…e.g 123.123.123.123
    token_id = strtok(null, tmp_delim);                             //peel off id…e.g. id
    token_id_name = strtok(null,tmp_delim);                         //peel off id_name…e.g. svr 0001
    token_flag = strtok(null, tmp_delim);                           //peel off flags     (optional/future)
    token_flag_bits = strtok(null, tmp_delim);                      //peel off flag bits…e.g 1a9     (optional/future)

    if (token_scope == “scope”) check_204_syntax_format_index++;
    if (token_scope_name is ascii and up to 8 characters) check_204_syntax_format_index++;
    if (token_disc == “disc”) check_204_syntax_format_index = check_204_syntax_format_index + 1;
    if (token_disc_value is a valid ip address format) check_204_syntax_format_index++;
        ////////////////////
        //note here is where another strtok sequence could be used to parse/build the ip address
        //its worth noting the value of disc_value since it determines unicast vs broadcast.
        ////////////////////
    if (token_id == ”id”) check_204_syntax_format_index++;
    if (token_id_name is ascii and up to 8 characgters) check_204_syntax_format_index++;

    ////////////////////////////////////////////////////////
    //flag and flag bits are option..they don’t have to be there…but if there, they must have the valid format
    ////////////////////////////////////////////////////////
    if (token_flag == “flags” || token_flag == null)                // remember optional/future
          check_204_syntax_format_index++;
    if ((token_flag_bits is 3 characters of ascii) | token_flag_bits == null)   // remember optional/future
          check_204_syntax_format_index++;
          ////////////////
          //note here is where another strtok sequence could be used to parse/build the iqn name specifically
          //////////////

   if (check_204_syntax_format_index == 8)                          //8 means 204 is well formed
       {
          return(true);                                             //valid option 204 format…proceed
       }
    else
```

```
        {
          return(false);                                          //error invalid format…defer to caller on go or no go
        }
}
```

Figure 36: DHCP Option 204 Syntax Checking

```
bool check_205_syntax (dhcp_option_205)
{
    //checking option 205 syntax
    // note that the format of option 205 is "B0:<LSB>:B1:<LMSB>:B2:<HMSB>:B3:<MSB>"
    // for example…."B0:AF:B1:CC:B2:44:B3:21"    (Note this equals 0x2144CCAF)
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 205;
    char tmp_delim = ":";
    char *token_b0;
    char *token_b0_value;
    char *token_b1;
    char *token_b1_value;
    char *token_b2;
    char *token_b2_value;
    char *token_b3;
    char *token_b3_value;


    int check_205_syntax_format_index = 0;

    token_b0 = strtok(tmp_option, tmp_delim);                     //peel off B0    e.g. B0
    token_b0_value = strtok(null,tmp_delim);                      //peel off B0 value…e.g. AF
    token_b1 = strtok(null, tmp_delim);                           //peel off B1…e.g. B1
    token_b1_value = strtok(null,tmp_delim);                      //peel off B1 value…e.g CC
    token_b2 = strtok(null, tmp_delim);                           //peel off B2…e.g. B2
    token_b2_value = strtok(null,tmp_delim);                      //peel off B2 value…e.g. 44
    token_b3 = strtok(null, tmp_delim);                           //peel off B3…e.g. B3
    token_b3_value = strtok(null, tmp_delim);                     //peel off B3 value…e.g. 21

    if (token_b0 == "B0") check_205_syntax_format_index++;
    if (token_b0_value is 2 characters of ascii hex) check_205_syntax_format_index++;
    if (token_b1 == "B1") check_205_syntax_format_index++;
    if (token_b1_value is 2 characters of ascii hex) check_205_syntax_format_index++;
    if (token_b2 == "B2") check_205_syntax_format_index++;
    if (token_b2_value is 2 characters of ascii hex) check_205_syntax_format_index++;
    if (token_b3 == "B3") check_205_syntax_format_index++;
    if (token_b3_value is 2 characters of ascii hex) check_205_syntax_format_index++;


    if (check_205_syntax_format_index = 8)
      {
        return(true);                                             //valid option 205 format…proceed
      }
    else
      {
        return(false);                                            //error invalid format…defer to caller on go or no go
      }
}
```

Figure 37: DHCP Option 205 Syntax Checking

```
bool check_206_syntax (dhcp_option_206)
{
    //checking option 206 syntax
    // note that the format of option 206 is "R1:<retry1>:R2:<retry2>:T1:<timeout1>:T2:<timeout2>"
    // for example…."R1:3:R2:4:T1:8:T2:8"
    // parsing is done by using the strtok function found in the ansi string library…please refer to understand behavior

    char tmp_option = dhcp option 206;
    char tmp_delim = ":";
    char *token_r1;
    char *token_r1_value;
    char *token_r2;
    char *token_r2_value;
    char *token_t1;
    char *token_t1_value;
    char *token_t2;
    char *token_t2_value;


    int check_206_syntax_format_index = 0;

    token_r1 = strtok(tmp_option, tmp_delim);                    //peel off R1    e.g. R1
    token_r1_value = strtok(null,tmp_delim);                     //peel off R1 value…e.g. 3
    token_r2 = strtok(null, tmp_delim);                          //peel off R2…e.g. R2
    token_r2_value = strtok(null,tmp_delim);                     //peel off R2 value…e.g 4
    token_t1 = strtok(null, tmp_delim);                          //peel off T1…e.g. T1
    token_t1_value = strtok(null,tmp_delim);                     //peel off T1 value…e.g. 8
    token_t2 = strtok(null, tmp_delim);                          //peel off T2…e.g. T2
    token_t2_value = strtok(null, tmp_delim);                    //peel off T2 value…e.g. 8

    if (token_r1 == "R1") check_206_syntax_format_index++;
    if (token_r1_value is 1 character of ascii hex) check_206_syntax_format_index++;
    if (token_r2 == "R2") check_206_syntax_format_index++;
    if (token_r2_value is 1 character of ascii hex) check_206_syntax_format_index++;
    if (token_t1 == "T1") check_206_syntax_format_index++;
    if (token_t1_value is 1 character of ascii hex) check_206_syntax_format_index++;
    if (token_t2 == "T2") check_206_syntax_format_index++;
    if (token_t2_value is 1 character of ascii hex) check_206_syntax_format_index++;


   if (check_206_syntax_format_index = 8)
       {
          return(true);                                          //valid option 206 format…proceed
       }
     else
       {
          return(false);                                         //error invalid format…defer to caller on go or no go
       }
}
```

Figure 38: DHCP Option 206 Syntax Checking

```
// detemine when to proceed to set up an iscsi session

STATUS StartiSCSISession()
{
//is there enough in place to proceed
if (HasOpt203(parms))
   {
     if (HasOpt201(parms) && HasOpt203(parms))                    //initiator and first target good
      {
         /////////
         //initiator = DHCP option 203
         //1st target = DHCP option 201
         ////////
         BuildiSCSISession(parms.opt203, parms.opt201);           // iscsi session between initiator and 1st target
         return SUCCESS;
      }
     else
      {
       if (HasOpt202(parms) && HasOpt203(parms))                  //initiator and second target good
         {
         /////////
         //initiator = DHCP option 203
         //2nd target = DHCP option 202
         ////////
         BuildiSCSISession(parms.opt203, parms.opt202);           // iscsi session between initiator and 2nd target
         return SUCCESS;
         }
      }
   }
else
   {
    if (HasOpt17(parms) && HasHname(parms)                        //hostname and option 17 good
     {
         /////////
         //initiator = DHCP option 12 (host name)
         // target = DHCP option 17
         ////////
         BuildiSCSISession(parms.opt17, parms.hname);    // iscsi session between initiator and target
         return SUCCESS;
     }
    else
     {
     return ERROR;                                                //no iscsi values to use for session
     }
    }
}
```

Figure 39: Decision Tree to Start iSCSI Session

Since the DHCP Option 205 is optional, there is no need to check its actual contents for valid combinations at this time. In the future, if 205 is used for some key information, then its actual values and value combinations would need to be verified.

Chapter 10. Infrastructure High Level Design Reference

The infrastructure for iSCSI parameter deployment, external deployment tools such as Director or storage wizard access the non volatile storage (NVS) via management module/RSA II access to BMC. The management module / RSA II provides the plumbing while the BMC provides a 'gatekeeper" function on the NVS. In addition, user panels at various points may be provided depending on solution requirements.

NVS storage of iSCSI Parameters

The non-volatile storage for the iSCSI parameters consists of 4 instances of the iSCSI parameter data structures plus some overhead space to help managing the iSCSI parameter space. This section details the NVS architecture.

NVS Storage Layout

NVS consists of 2KB of non volatile space managed by the BMC processor. The space consists overhead plus 4 instances of the iSCSI parameter block. With the overhead and the 4 instances, each iSCSI parameter paragraph is 434B. The iSCSI parameter paragraphs can also be chained together to form larger parameter blocks for such things is IPSEC certificate. Below, graphically presents the arrangement of parameter blocks

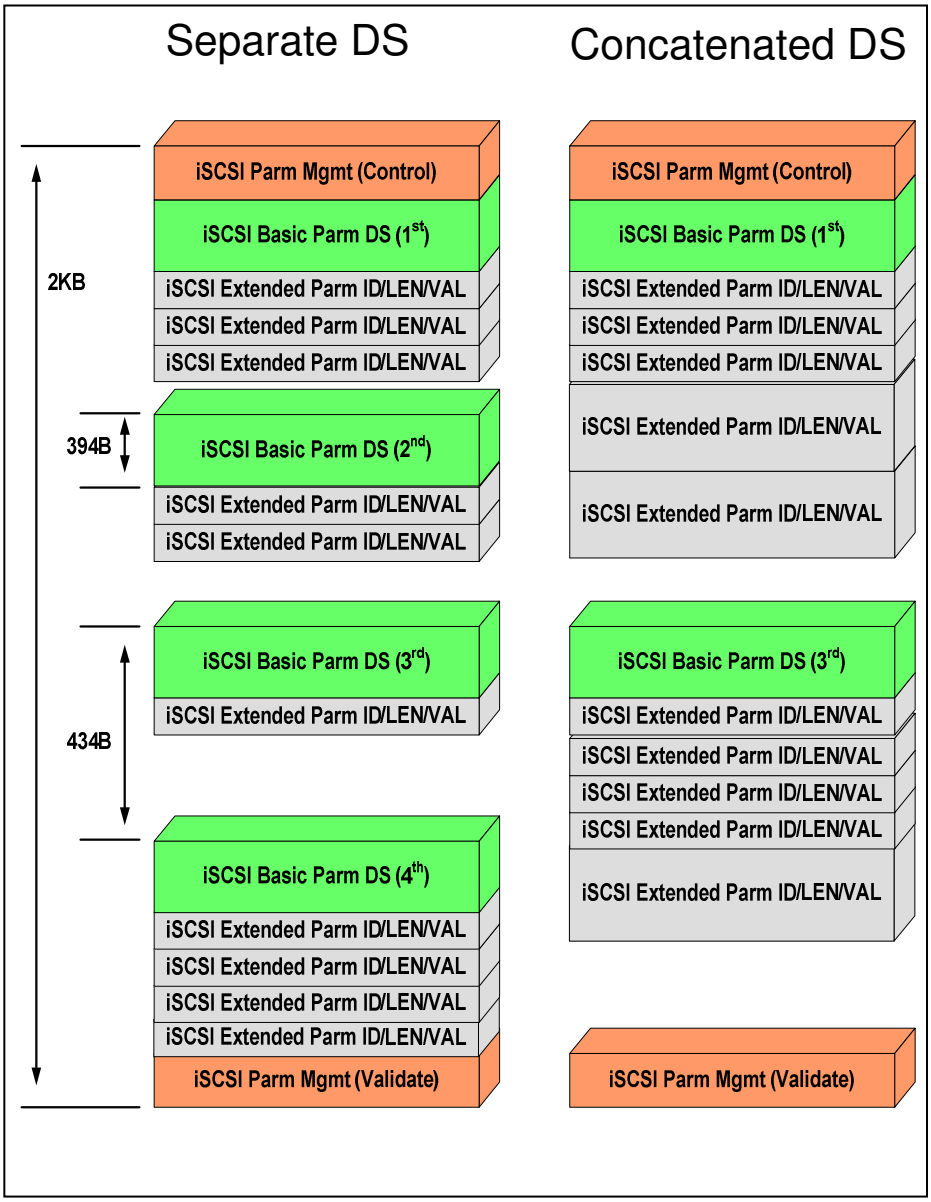

Figure 40: NVS Space Layout

NVS Overhead Parameters

There several overhead parameters used to manage the actual iSCSI data structures. Only a brief summary of these parameters are presented here for convenience. For more detail descriptions and specific offsets, please refer to “ESW iSCSI Boot Initiator Support Design” (ESWISCI101.07.pdf) authored by Scott Dunham.

In general, the management header enumerates the mapping of iSCSI parameters to physical ports. Up to 4 ports can be mapped. During the boot sequence, the BIOS accesses the mgmt header to determine the appropriate parsing needed to be done to retrieve the actual iSCSI parameters.

Parameter	Name	Bits	Definition
Mgmt Header Version	MHVER[7:0]		Version of the Mgmt header structure
Mgmt Header Checksum	MHCHK]7:0]		Checksum of the Mgmt header structure
Initiator 1 Layout	I1LOUT[7:0]	[7]	Enable/disable boot from this port
Initiator 1 Layout	I1LOUT[7:0]	[6]	Hardware or software initiator
Initiator 1 Layout	I1LOUT[7:0]	[5]	Reserved
Initiator 1 Layout	I1LOUT[7:0]	[4]	Full inband parms…aka fast path for full inband
Initiator 1 Layout	I1LOUT[7:0]	[3:2]	Port ID method
Initiator 1 Layout	I1LOUT[7:0]	[1:0]	Parm block chaining
Initiator 1 Usage	I1USE[7:0]	[7:6]	4th Parm block in a sequence
Initiator 1 Usage	I1USE[7:0]	[5:4]	3rd Parm block in a sequence
Initiator 1 Usage	I1USE[7:0]	[3:2]	2nd Parm block in a sequence
Initiator 1 Usage	I1USE[7:0]	[1:0]	1st Parm block in a sequence
Initiator 1 Map A	I1MAPA[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 1 Map B	I1MAPB[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 1 Map C	I1MAPC[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 2 Layout	I2LOUT[7:0]	[7]	Enable/disable boot from this port
Initiator 2 Layout	I2LOUT[7:0]	[6]	Hardware or software initiator
Initiator 2 Layout	I2LOUT[7:0]	[5]	Reserved
Initiator 2 Layout	I2LOUT[7:0]	[4]	Full inband parms…aka fast path for full inband
Initiator 2 Layout	I2LOUT[7:0]	[3:2]	Port ID method
Initiator 2 Layout	I2LOUT[7:0]	[1:0]	Parm block chaining
Initiator 2 Usage	I2USE[7:0]	[7:6]	4th Parm block in a sequence
Initiator 2 Usage	I2USE[7:0]	[5:4]	3rd Parm block in a sequence
Initiator 2 Usage	I2USE[7:0]	[3:2]	2nd Parm block in a sequence
Initiator 2 Usage	I2USE[7:0]	[1:0]	1st Parm block in a sequence
Initiator 2 Map A	I2MAPA[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 2 Map B	I2MAPB[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 2 Map C	I2MAPC[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 3 Layout	I3LOUT[7:0]	[7]	Enable/disable boot from this port
Initiator 3 Layout	I3LOUT[7:0]	[6]	Hardware or software initiator
Initiator 3 Layout	I3LOUT[7:0]	[5]	Reserved
Initiator 3 Layout	I3LOUT[7:0]	[4]	Full inband parms…aka fast path for full inband
Initiator 3 Layout	I3LOUT[7:0]	[3:2]	Port ID method
Initiator 3 Layout	I3LOUT[7:0]	[1:0]	Parm block chaining
Initiator 3 Usage	I3USE[7:0]	[7:6]	4th Parm block in a sequence
Initiator 3 Usage	I3USE[7:0]	[5:4]	3rd Parm block in a sequence
Initiator 3 Usage	I3USE[7:0]	[3:2]	2nd Parm block in a sequence
Initiator 3 Usage	I3USE[7:0]	[1:0]	1st Parm block in a sequence
Initiator 3 Map A	I3MAPA[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 3 Map B	I3MAPB[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 3 Map C	I3MAPC[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 4 Layout	I4LOUT[7:0]	[7]	Enable/disable boot from this port
Initiator 4 Layout	I4LOUT[7:0]	[6]	Hardware or software initiator
Initiator 4 Layout	I4LOUT[7:0]	[5]	Reserved
Initiator 4 Layout	I4LOUT[7:0]	[4]	Full inband parms…aka fast path for full inband

Parameter	Name	Bits	Definition
Initiator 4 Layout	I4LOUT[7:0]	[3:2]	Port ID method
Initiator 4 Layout	I4LOUT[7:0]	[1:0]	Parm block chaining
Initiator 4 Usage	I4USE[7:0]	[7:6]	4th Parm block in a sequence
Initiator 4 Usage	I4USE[7:0]	[5:4]	3rd Parm block in a sequence
Initiator 4 Usage	I4USE[7:0]	[3:2]	2nd Parm block in a sequence
Initiator 4 Usage	I4USE[7:0]	[1:0]	1st Parm block in a sequence
Initiator 4 Map A	I4MAPA[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 4 Map B	I4MAPB[7:0]		Port mapping – 3 bytes to support different mappings
Initiator 4 Map C	I4MAPC[7:0]		Port mapping – 3 bytes to support different mappings

Table 92: NVS Management Header Parameter Summary

In general, the capabilities header enumerates all the iSCSI Initiators in the given server. Given this is a list construct, there can be many initiators listed. During the boot process, the BIOS updates this header so that any external deployment tool can retrieve the capabilities the given server in terms of iSCSI..

Parameter	Name	Bits	Definition
Cap Header Version	CHVER[7:0]		Version of the Capabilities header structure
Cap Header Lenght	CHLEN[7:0]		Length of the Capabilities header structure
Cap Header Checksum	CHCHK]7:0]		Checksum of the Capabilities header structure
iSCSI Parm Version	CHPVER[7:0]		iSCSI Parm Version. Note: it should equal VER[7:0]
iSCSI Parm Length	CHPLEN[15:0]		Max Length of an iSCSI Parm page
Software Initiator Map	SWMAP[7:0]		Mapping of software initiators to physical ports
Software Initiator Count	SWCNT[7:0]		Number of SW Initiators in this server
Hardware Initiator Map	HWMAP[7:0]		Mapping of Hardware initiators to physical ports
Hardware Initiator Count	HWCNT[7:0]		Number of HW Initiators in this server
SW 1 Slot	SW1SLT[7:0]		Slot number for software initiator 1
SW 1 PFA	SW1PFA[15:0]		Slot PFA for software initiator 1
●			
●			
SW N Slot	SWNSLT[7:0]		Slot number for software initiator N
SW N PFA	SWNPFA[15:0]		Slot PFA for software initiator N
HW 1 Slot	HW1SLT[7:0]		Slot number for Hardware initiator 1
HW 1 PFA	HW1PFA[15:0]		Slot PFA for Hardware initiator 1
or			Or depending on the mapping format
HW 1 MAC	HW1MAC[47:0]		MAC address for the Hardware initiator 1
●			
●			
HW N Slot	HWNSLT[7:0]		Slot number for Hardware initiator N
HW N PFA	HWNPFA[15:0]		Slot PFA for Hardware initiator N
or			Or depending on the mapping format
HW N1 MAC	HWNMAC[47:0]		MAC address for the Hardware initiator N

Table 93:NVS Capabilities Header Parameter Summary

BMC iSCSI Parameters Operations

The BMC service processor manages the NVS storage for iSCSI configuration parameters. Specifically, all reads, and writes to the NVS iSCSI space goes through the BMC interface.

READ Operations to Extract iSCSI Parameters from NVS

Read operations involve the BIOS or deployment tools (MM/RSAII, Wizards, Director, etc) requesting the BMC to provide the current iSCSI parameters from NVS. .

READ operations involve a read command to the BMC where the results are destined to the tools and Initiator s via BIOS. The BMC should respond with the actual values for all fields and options.

WRITE Operations to Inject iSCSI Parameters into NVS

WRITE operations involve the BMC receiving an iSCSI parameter block from BIOS or deployment tools and update the NVS space, regardless of the present values in NVS.

Lock Operation to Guarantee Correct Operation

Since reads and writes can come from external tools via management module or RSAII or can come from BIOS, the resource lock should be used prior to reading or writing iSCSI parameters to the NVS space. This ensures that reads are fully accurate and writes are fully completed before another service tries to access the iSCSI parameters. Obviously, prior to reading or writing, the requestor should poll until the iSCSI parameter space is available and should released in a timely manner (after the last transaction for example).

BMC Operation Command Format

Below is the command format for the operations accessing the IPMI compliant BMC for iSCSI parameters

The new iSCSI field access commands are extensions of the current VPD access commands.

GetVPD(DeviceID, Offset, Length)

SetVPD(DeviceID, Offset, Length, Data)

DeviceID:

A 1 byte field indicating which "section" of “VPD” is to be accessed.

The following values for iSCSI fields are currently defined:

20h = Boot sequence data

21h = System iSCSI capabilities data

22h = iSCSI initiator parameter storage area header and initiator boot information

23h = iSCSI initiator parameter storage area parameter block 1 (READ: BIOS intended)

24h = iSCSI initiator parameter storage area parameter block 2 (READ: BIOS intended)

25h = iSCSI initiator parameter storage area parameter block 3 (READ: BIOS intended)

26h = iSCSI initiator parameter storage area parameter block 4 (READ: BIOS intended)

Offset:

A 2 byte field (little endian format) indicating the offset into the DeviceID "section" to start reading from.

Length:

A 2 byte field (little endian) indicating the amount of data (in bytes) to read.

Data:

"Length" number of bytes of the actual data to be written or verified.

iSCSIsemaphoreGet()

iSCSIsemaphoreClear()

BIOS iSCSI Parameter Operations

The BIOS operations consist of two classes of operations. First, the BIOS must retrieve the parameters to NVS iSCSI space via the BMC access. Second, the BIOS must present the menus for users to enter the parameters locally on the F1/BIOS menus.

BIOS Processing

During the boot process, the BIOS reads the NVS iSCSI space via BMC to retrieve the iSCSI parameters using the initiator intended read operation. Once retrieved, the BIOS validates the iSCSI data structure in terms of length an checksum. Assuming all is in order, the BIOS places the header "ISAN" on the data structure (1st 4 bytes) to aid the initiator in finding the data structure into the EBDA space. If the data structure retrieved is corrupted, the BIOS post an error.

In the event that there is no valid data structure available in NVS iSCSI space, the BIOS creates a "dummy" structure in EBDA space that indicate that the initiator should acquire iSCSI configuration parameters from DHCP or other discovery services.

During the boot process when the user or admin chooses to configuration the iSCSI parameters locally, the BIOS will write the results to the NVS iSCSI space with the appropriate warnings to the user/admin about overwriting existing iSCSI parameters.

BIOS Pseudo Code for iSCSI

Below is the BIOS pseudo code for boot up. This view is provide the reader with the boot flow structure

```
During ROM scan:
Read selectable boot sequence
IF iSCSI boot is one of the selectable boot entries
   Obtain iSCSI data area semaphore (wait up to 15 seconds)
   Read iSCSI parameter storage area to determine initiator types
   (HW or SW) and locations
   Error exit if Initiator Parameter Storage area invalid
   IF PXE boot enabled (as selected in BIOS Setup)
      Run PXE ROMs for PXE bootable devices and
      Run PXE ROMs for additional SW iSCSI initiator selected devices
   ELSE
      Run PXE ROMs for SW iSCSI initiator selected devices
```

```
   Call all "non-HW-initiator" adapter ROMs as usual keeping track of
   HW initiator locations and Ethernet locations (for capabilities data)
   For each ordered initiator slot/PFA enabled in initiator boot info
      Write signature header to memory location
      Append initiator's parameter instance data in memory
      Copy initiator image to adapter ROM area
      IF SW initiator
         Put PFA of selected Ethernet port into register AX
      ELSE (HW initiator)
         Put PFA of HW initiator into register AX
      Put memory address of signature in register ESI
      Display message that iSCSI connection is being attempted
      Call entry point of initiator (initiator attempts target connection)
      Clear parameter memory area
      IF successful connection
         Set iSCSI boot indicator
         Exit
      ELSE
         Write error log
         HW initiator ROM image 0s its size
         IF SW initiator
            Clear SW initiator ROM area
   IF no initiator was successful AND iSCSI Critical Boot indication set
   (Boot sequence option)
      Disable SP POST watchdog timer
      Loop on retrying initiators which did not fail for an unrecoverable
      reason (unsupported/invalid parameter structure, invalid PW/ID,etc)
      Do NOT relog errors
      Exit when a successful initiator connection occurs or all have
      failed for unrecoverable reasons
ELSE
   Do adapter ROM scan but do not call HW initiator ROMs or
   SW initiator    ROM, keeping track of HW initiator and
   Ethernet locations
   Clear iSCSI boot indicator
Write iSCSI capabilities data area

During Boot:

IF iSCSI boot indicator set
   IF current boot sequence entry is iSCSI
      Disable drive remapping
      Use Int 13h DL=80h (Drive 80h) to read Master Boot Record (MBR)
      IF MBR is OK
         Pass control to MBR, which will use Drive 80h to boot
   ELSE
      Enable drive remapping to add desired drive number plus 1 to
      Int 13h DL=80h calls
      (Initiator will subtract 1 from DL if DL > 80h )
ELSE
   Boot normally
```

Figure 41: BIOS Pseudo Code for Boot

F1/BIOS Menus

The user and admin can enter iSCSI configuration parameters manually locally at the server. The iSCSI configuration options presented include the main iSCSI parameters and some of the more obscure parameters are set to default values. This configuration path is provided for user or admin convenience. Writes are written to the NVS iSCSI space per some sort of "commit" button so that the entire data structure is written once not on a parm by parm basis.

The document “ESW iSCSI Boot Initiator Support Design” (ESWISCI101.07.pdf) authored by Scott Dunham provides a detailed description of BIOS ⇔ Boot initiator interaction.

This document covers the details of the BIOS ⇔ Boot initiator handoff as well as the functionally placed on the BMC and non volatile storage (NVS) containing the iSCSI parameters.

Management Module / RSA II

The operations for the management consist of three sets of operations. First, the management module (or RSAII) must provide the plumbing to allow external deployment tools to pass the iSCSI parameters back and forth to the BMC. Second, optionally, the management module/RSA II may want to present panels to allow the user to enter the parameters manually. Third, for external access by tools, the management module/RSA II should provide SSL capabilities to build an encrypted connection with external tools.

MM / RSA II Plumbing

The Management module or RSA II provides the plumbing to pass traffic between external deployment tools and BMC. The plumbing can be provided via IPMI pass thru commands. In the plumbing operations does not make any checks on the format or validity of the parameters (ie no security masks or format checking).

Configuration Panels (optional)

Optionally, the management module or RSA II can present panels for the user to configure the initiator. The management module would retrieve the parameters using the user intended read operation and present those current values to panels. Once the parameters are updated and committed to the BMC via some sort of “commit” button. There is no assumption

SSL Support

As discussed in the security discussion earlier, the management module/RSAII should provide SSL capabilities that can be used to provide a secure connection between the management module/RSA II and external tools. It is beyond the scope of this document to articulate the specifics of this SSL service since its provided by the management module/RSAII outside the context of iSCSI.

Deployment tools

Deployment tools such as Director or storage wizards provide a more global / data center perspective. A given user uses these tools to configure the iSCSI nexus (ie the relationship and definition of the initiators and targets). These tools present user panels to an admin to configure many initiators and such. These tools use IPMI pass through commands to access management module or RSA II.

It is beyond the scope of this document to describe how the panels are defined and presented to the admin.

Chapter 11. Initiator General Design Considerations

The implementation of a given initiator must assess the following design considerations and react appropriately. The design considerations are broader than specific parameter considerations and provide some insight into evolution of the iSCSI parameter deployment.

Allocation of iSCSI Parameter Space

Noting, due to space constraints, some of the iSCSI parameter fields defined in this document consume less space than is defined in the RFC documentation, the initiator implementation **should** make every effort to define and allocate space to accept fully RFC 3720 compliant parameters. For example, RFC 3720 defines an iSCSI name to be up to 223 bytes in size. As a result, an initiator should strive to define its internal structure space to support an iSCSI name of 223 bytes in size even if the data structure passed to it by BIOS only provides 72 bytes of space.

Some fields, such as the iSCSI name, make parsing and processing values smaller than the space fairly simple (ie iSCSI names have no spaces) while others such as LUN may make parsing and processing more difficult. Where ambiguity lies in the field inclusion in a field space, the initiator implementation must provide a way to determine the valid subset of the space that is valid. For example, a given initiator implementation may chose to add an internal (not visible outside the actual implementation) a 1 byte field to indicate valid length or a given initiator implementation chose to increase the RFC defined spaces by 1 byte and define an internal (not visible outside the actual implementation) termination character.

If the given initiator implementation, attempting to parse fields or RFC spaces, determine that an invalid format or some other ambiguity is present, it **should** return control to the BIOS with an error message (either using data structure error return code or a unique parsing error code).

iSCSI and Associated RFC Evolution

As the iSCSI and associated RFCs evolve over time, a given initiator **should** become RFC compliant in a timely manner. This evolution implies functionality enhancements as well as addition of any additional error processing required to signal any ambiguity or difficulties. For example, if the iSCSI RFC enhances the iSCSI name to accept 500 bytes, and the given initiator implementation has not yet incorporated it, the initiator should respond with an error code (either the data structure error code or some other error code) indicating it is not yet compliant with the new RFC.

iSCSI Parameter Data Structure Evolution

With the evolution of the iSCSI parameter data structure by version or level, the given initiator implementation **should** determine if it can accept the data structure based on the version and level numbers. If it cannot, it should return control back to BIOS with an appropriate error code.

Chapter 12. Reference Documentation

Below in Table 94: iSCSI Specific IETF Drafts, Table 95: Additional iSCSI and SCSI Resources, Table 96: BIOS Specific Documents, Table 97: System Management Documents, Table 98: Network Services Documents, and Table 99: BladeCenter Specific Documentation Provide background information

iSCSI and SCSI References

Document Name	Targeted Content
draft-ietf-ips-iscsi-20.txt	The definition for the iSCSI protocol (RFC 3720)
draft-ietf-ips-isns-17.txt	The definition for the iSNS discovery services
draft-ietf-ips-iscsi-slp-04.txt	The definition for the SLP discovery services
draft-ietf-ips-iscsi-mib-08.txt	The definition for the iSCSI MIB
draft-ietf-ips-iscsi-reqmts-06.txt	The definition of Requirements for iSCSI initiators and targets
draft-ietf-ips-iscsi-boot-08.txt	The definition of client boot sequence using iSCSI targets
draft-black-ips-iscsi-dhchap-01.txt	The definition of CHAP enhancements for iSCSI
draft-gilligan-iscsi-fault-tolerance-00.txt	The definition of iSCSI fault tolerance
RFC 1510	The definition of Kerberos V5 authentication
RFC 1994	The definition of CHAP authentication
RFC 2131	The definition of DHCP configuration protocol
RFC 2132	The definition of DHCP options
RFC 2025	The definition of SPKM authentication
RFC 2945	The definition of SRP authentication
RFC 2401-2409	The definitions for aspects of IPSEC
RFC 3396	The definition of DHCP option concatenation

Table 94: iSCSI Specific IETF Drafts

Proposed ietf drafts and ratified RFCs can be searched at http://search.ietf.org
In addition to these proposed RFCs, there are several books and resources that provide insight to iSCSI and SCSI in general.

Document Name	Targeted Content
iSCSI, The Universal Storage Connection	Discuss iSCSI in detail
The SCSI Bus and IDE Interface	Discusses SCSI
www.t10.org	Discusses all the SCSI standards

Table 95: Additional iSCSI and SCSI Resources

BIOS Documentation

Document Name	Targeted Content
Nefinity BIOS Architecture 0.6	Background information Netfinity BIOS architecture and behavior
IBM eServer xSeries POST/BIOS setup	Background info on xSeries POST/BIOS focusing on network devices
PXE 2.1 Specification	Network and UNDI services available at boot time
Inside the Boot process for Windows	Article focusing on the boot process for windows
Linux Boot Sequence	Charts on boot focusing on Linux
Network PC System Design Guidelines	System design guidelines for network PCs

Table 96: BIOS Specific Documents

System Management Documentation

Document Name	Targeted Content
Wired for Mgmt 2.0	WMI specification

Document Name	Targeted Content
System mgmt BIOS Specification 2.3	SMBIOS structures and interfaces
System mgmt for Bladecenter Redbook	Redbook focusing on system management for bladecenter
Blade environmental firmware 0.06	Detailed look at APIs for managing blades
Mgmt module firmware 0.7	Detailed look at mgmt module firmware

Table 97: System Management Documents

Network Services Documentation

Document Name	Targeted Content
Cisco Network Boot Guide 1.0	Focuses on Cisco's implementation of iSCSI boot
Linux Network Boot Guide	Linux slant on PXE booting
Configuration Cisco DHCP servers	Cisco perspective on DHCP servers
DHCP Handbook	In depth discussion on DHCP behavior

Table 98: Network Services Documents

BladeCenter Documentation

Document Name	Targeted Content
IA32 Software developers guide Vol 3	System design especially System mgmt mode operations for Intel
AMD 64 Arch Programmers guide	System design especially system mgmt mode operations for AMD
iSCSI boot Tutorial	Overview of Cisco iSCSI boot on blades
Fiber Deployment Guide 0.1	FC deployment guide for blade
Server Storage Provisioning Tool 0.9	Storage provisioning tools for blade
BladeBIOS Specification 0.08	BIOS focus on blades
FC Blade Boot with FastT	Guide on booting blades using FC and FastT
Base Spec for processor blades 0.98	Processor blade design guide
Base Spec for Daughter cards 1.01	Daughter card for blade design guide
Base Spec for Switch Modules 1.02	Switch module design guide

Table 99: BladeCenter Specific Documentation

Appendix A: iSCSI Parameter Evolution 3.7 to 4.1

Below are the detailed discussions and conclusions on parameter evolution that occurred in 1Q06.

iSCSI V2 Configuration Parameters Changes Under Discussion

Backward Compatibility – Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: accepted

[01/10/05]: There is a view to ensure the config wizard and initiator implementations be able to Support V1 and V2 versions. This centers on how these components would evolve to Support V2 or just abandon V1. In an effort to facilitate this, the V2 additions will Coded as EPs so that the current wizard and initiators can continue to be supported while Being able to asynchronously evolve to V2. This requirement needs to be vetted for Other issues

[01/31/06]: Given the parm rules cited for each parm as of this date, the behavior of V1 wizard with V1 or V2 initiators (and BIOS) as well V2 wizard with V1 or V2 initiator (and BIOS) is Defined.

Specifically:

V1 wizard V1 Initiators and BIOS:

- the current rules so the wizard can retrieve capabilities and set mapping
- The current rules for the initiator applies (not aware of any V2 stuff at all)

V1 wizard, V1 BIOS, V2 initiators

- The V1 wizard uses the current parm mgmt DS and set parms as defined today (ie no extended parms for iscsi name or boot LUN)
- V1 BIOS would pass base parms as defined today
- The V2 initiators, using the rules for each parm, would use the base parm space As defined in V1

V1 wizard, V2 BIOS, V2 Initiators

- V1 wizard uses the current parm mgmt DS and set parms as defined today
- V2 BIOS would see parm mgmt DS for V1 (V2 parm mgmt DS not set) so Would parse/use as defined in V1
- V2 initiator would use the parm rules to figure out V1 parm format in play

V1 wizard, V2 BIOS, V1 initiators

- V1 wizard as today
- V2 BIoS would see parm mgmt DS for V1 (V2 set to 0), so would parse/use as V1 defined
- V1 initiators would use V1 rules for each parm

V2 wizard, V1 BIOS, V1 initiators

- V2 wizard, seeing no V2 parm mgmt capabilities section, realizes dealing with With V1 BIOS and would set V1 parm mgmt DS (aka act like V1 wizard)
- V1 BIOS knowing nothing of V2 would not set capabilities or read mgmt to parse
- V1 initiators knowing nothing of V2 would use base parm block

V2 Wizard, V1 BIOS, V2 initiator

- V2 wizard would behave as V1
- V1 BIOS would behave as today
- V2 initiator, using the rules defined here would basically use V1 base block (To be explored is the idea of passing the extended parms down in V1 context)

V2 Wizard, V2 BIOS, V1 initiator

- V2 wizard can use V2 parm mgmt DS since BIOS is using V2 constructs
- V2 BIOS would use V2 constructs
- V1 initiator would use base parm block. Note that V2 wizard must follow Same parm rules as specified here

V2, V2, V2

- All should be happy.

Configuration Parameter Map – Tentatively Accepted

Discussion Thread	Leader:	Conclusion:
[12/19/05]: The suggestion is that given some of the suggestions below, size of a given base parameter block, we may want to rearchitect the layout to more efficiently use the non volatile space. Perhaps 2 initiator blocks and 1st target block / 2nd target block approach. [01/10/05]: The BMC has allocated 4KB for all the parameter data structure. As a result, ramifications include: - The NVS map needs to updated (to date assume 4 instances @ 900+ bytes, 400 bytes for Overhead / control) - The device ids for access to NVS remain the same - The parm instances will still be self contained (initiator and target together) - Given that there may be other remote boot devices using this infrastructure, only 2 instances should be set for full DHCP default and have them targeted the 2 base EN ports The current constructs of parm page concatenation and mapping is maintained. [01/31/06]: proposal: Given that 4KB space is available, the parm mgmt DS (not the parms themselves but the DS to manage the structures – capabilities, mappings, etc) could be implemented to provide V1 in current locations and V2 in following appended locations. This would allow a V2 wizard (tool or director) to know if V2 is in play at the BIOS level be the fact the second version of the mgmt overhead parms are visible. Correspondingly, the V1 wizard would use the current version at the current offset and, thus, leave the V2 form of mapping blank so the BIOS would know if needed to do anything special for backward compatible. While this is a bit inefficient in terms of bytes used, it does allow V1 of BIOS and wizard to Remain unchanged and allow the evolution of the wizard and BIOS to proceed Asynchronously.		

Table 100: Configuration Parameter Map Discussion

Configuration Parameter Handoff - Accepted

Discussion Thread	Leader: Dunham	Conclusion: Accepted
[12/19/05]: The suggestion is to enhance the iSCSI parameter handoff in and out of band mode. Namely, there are three points to consider (and may be broken out uniquely as discussion Evolves): First, a scheme to control usage boot enablement from block such that remote tools can actually invoke iSCSI boot usage remotely/simplistically vs F1/BIOS settings updates. Second, a scheme to define boot device order from parm block such a remote tool can actually define the boot device order remotely/simplistically vs F1/BIOS settings updates. Third, provide a default construct to indicate to the initiator to use inband configuration WITHOUT invoking any external tool to enable. Realizing that in certain scenarios, there may be conflicts between this enhancements, we all need to reach consensus on the best way to accommodate as much of this as possible. [01/10/05]: Here are the items to date - The mfg programming of NVS will have defaults to do full DHCP so no need to invoke The wizard for the first time use / out of the box experience - The enablement of iSCSI would done remotely through MM interface. Aka from an External console the admin would access the blade to enable the function. Need to Explore if this is scriptable. A possible implication is that the BIOS team may have To more seriously develop a EXP paging mechanism to page in a variety of EXP roms. - The boot order is NOT going to be set via the parm block since there are only 4 entries in Boot order menu and to change that is deemed to intrusive on design and usage models (pxe boot, image deployment, etc).		

Table 101: Configuration Parameter Boot Control Discussion

iSCSI Name Length – Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted

[12/19/05]: The suggestion is to allow full length iSCSI names in the out of band deployments. The current format is to provide a 72 character string for iSCSI initiator name, 1st target name, and 2nd target name. RFC 3720 defines the iSCSI name to be as large as 223 bytes. As a result, the name space in the parameter block would have to be grown for all 3 names. Given the ground rules, especially on size in non volatile memory, careful attention must be must be paid to the implications involved. Regarding in band deployment, larger names have implications on option space size and requiring the DHCP server to support RFC 3396 to allow such potentially large strings (noting that RFC 4173 includes protocol, boot LUN, etc that can drive a given option beyond 255 bytes).

[01/10/05]: The name will be full length. The structure is the following:
- Full 223 characters will split between the 72 in the base parm and the remaining 152 characters will be in an extended parm that will be required.
- The end of the iSCSI name will be NULL terminated for decent parsing.

This approach maintains the backward compatibility of the V1 parms (with the at most The last char being NULL (0x00) in the base 72 characters), provide a EP header to recognize extended string to ensure well formed and deterministic Behavior on the length and checksum fields. The EP option field may use 1 bit To determine that the EP is required (still have to vet this).

[01/17/06]: The discussion last week centered on current behavior and corner cases. Investigated both Issues. On current behavior, the storage wizard does not explicitly NULL terminate the iSCSI name, instead when the string is updated, a new instance of the field is created with all 0x00s. For all scenarios except where the actual name is a full 72 characters, this is effectively NULL termination (resulting in a hole for the 72 character case). Corresponding, Qlogic behaves in a similar fashion. On the corner case scenario, this Implicit NULL termination could create issues. So to address this limitation, the following Proposal is put forward.

So here a proposal:
- The long format of the names are: Initiator = EP(0x09)||IIQN, 1st target = EP(0x0A)||T1IQN, and 2nd target EP(0x0B)||T2IQN. Note that this is the reverse of the concatentation discussed previously
- This implies that initiators have a deterministic behavior when new/unknown EPs are Are encountered.
- Note that this behavior does not require the version/level to be changed solely for this Field. Thus, V1 initiators can operating in this new V2 world for this parameter at least.

The behaviors are as follows:

V1 wizard and V1 Initiator
- No problems…very very similar to today

V1 wizard and V2 initiator
- Determine if EPs are present:
- if present then use the concatenated field (e.g. initiator = EP(0x09)||IIQN) If the name is less than 72 characters, the EPs need not be present
- if not present…then use use IIQN, T1IQN, T2IQN ONLY If the name is less than 72 characters, the EPs need not be present

V2 wizard and V1 initiator
- Given the DS length defined in the base spec includes any EP parameters, The V1 initiator, not recognizing these new EPs (or in some cases any EPs) Parameters, they must post an error indicating “unknown” EPs are present. Note

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted

This is a consistent behavior since somebody could issue an unknown EP today
And the initiator should reject it. This ensures that if the names are larger than 72 characters, there is feedback via Error codes displayed no later than at boot time to the admin and, thus, eliminates The need for sniffers to determine the names have been truncated.
- The V2 wizard must put the least significant 72 characters of name in the base field (IIQN, T1IQN, T2IQN) the remaining most significant characters in EP fields, if Required due to the name length. Ie a 100 character name just be set up as the most significant characters of the EP = 0x00 and the least significant characters of the EP be most significant 28 characters of the iscsi name and the least significant 72 chars of the iscsi name be in the xxIQN fields.
This provides backward compatibility
- The V1 initiator then uses ONLY IIQN, T1IQN, and T2IQN
V2 wizard and V2 initiator
- With both wizard and initiator at V2, then use the full fields:
Initiator = EP(0x09)||IIQN, 1st target = EP(0x0A)||T1IQN,
And 2nd target = EP(0x0B)||T2IQN.
- Same parsing as defined in V2 wizard/V1 initiator scenario.

So lets look at some examples:
Example 1:
Iscsi name = "iqn.<a0..u9>" ie all 223 characters
xxIQN = "<n8..n9><o0..u9> ie least 72 characters of name
EP = "iqn.<a0..m9><n0..n7> ie the most significant 151 characters
So
iSCSI name = EP||xxIQN = "iqn.<a0..u9>" ie all characters
Example 2:
iSCSI name = "iqn.<a0..h9>" ie 80 character name
xxIQN = <a8..a9><b0..h9>" ie least 72 characters of name
EP = "iqn.<a0..a7>" ie the most significant 8 characters of name
iSCSI name = EP||xxIQN = "iqn.<a0..h9>"
Example 3:
iSCSI name = "iqn.<a0..h1>" ie 72 character name
xxIQN = "iqn.<a0..h1>" ie all 72 characters
EP not present
iSCSI name = EP||xxIQN = NULL||"iqn.<a0..h1>"
Example 4:
iSCSI name = "iqn.<a0..b9> ie 20 character name
xxIQN = "iqn.<a0..b9>" ie all 20 characters. Note, to be consistent with with todays formatting, the 20 characters are left justified in the 72 byte field
EP not present
iSCSI name = EP||xxIQN = NULL||"iqn.<a0..b9>"
Question to all…the RFC seems to imply that the "iqn." Is part of the 223…fair enough, But seems odd to have 199 user definable characters. So we all need to agree where the parsing break is at. It may make most sense to work back from the end the name String and dump all characters over 72 into the EP field.

[02/22/06]: It has been brought to my attention there is a minor error in example 1 above. Basically, Actually have 223 characters, everywhere in example 1 that has a u9 needs to be replaced A v8.

Table 102: iSCSI Name Length Discussion

IPv6 Address Representation In Parameter Block - Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted
[12/19/05]: The suggestion is to provide a full 16 bytes for each IP address and such defined in in the configuration parameter block. This may be a more compact and forward looking approach to IPv6 than the current "link local" in base parms + "global prefix" in extended parms space. [01/10/05]: The EP parm for the header of IPv6 will be used. The EP will be optional		

Table 103: IPv6 Representation in Parm Block Discussion

Boot LUN Representation In Parameter Block - Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: accepted

[12/19/05]: The suggestion is to comply with RFC 3720 in presenting the Boot LUN as an 8 byte field when using out of band configuration mode.

[12/19/05]: Per RFC 3720, LUN[7:0][7:0] is presented in packets in reverse order. So the proposal would be to use the same convention. For example

[01/10/05]: Instead of changing the boot LUN space in the base parm, the EP parm defined for 8 byte LUNs will be used. The EP will be required (an subject to decision on EP option bit Discussion concerning required). The boot LUN byte in the base parm will no longer be used. To aid in parsing, the default value for this byte will be 0xFF
Updated the examples per SAM using the LUN format

LUN 0 ➔ EP 07/08 value of[63:0]=0x8000000000000000 (or just 0)
LUN 1 ➔ EP 07/08 value of[63:0]=0x8000000000000001
LUN 12 ➔ EP 07/08 value of[63:0]=0x800000000000000C
LUN 16 ➔ EP 07/08 value of[63:0]=0x8000000000000010
LUN 258 ➔ EP 07/08 value of[63:0]=0x8000000000000102
LUN 1036 ➔ EP 07/08 value of[63:0]=0x800000000000040C

To be honest, I believe I interpreted the SAM spec correctly…but without good examples to Track the mapping, I could be wrong. Feedback is welcomed.

[01/17/06]: There was discussion on this field centering around V1 vs V2 usage. Namely, whether There should be option bits for this or there should be a marker in the boot LUN space To indicate behavior.
So here is a proposal:
- Initiators requiring full 8 byte LUNs must use the EP(0x07) and EP(0x08).
- Initiators must scan for these EPs before chosing where the boot LUN definition is at.
- As a result, the version/level do not need to be changed for solely for this field.

The behaviors are as follows:

V1 wizard and V1 initiator:
- No problem…use the byte LUN and option bits as is.

V1 wizard and V2 initiator:
- Assess if EPs are present.
- If present, then use them exclusively (ignore option bits)
- If not present, use 1 byte LUN and option bits as defined in V1

V2 wizard and V1 initiator
- Program both the 1 byte LUN with least significant byte of LUN and use And set the option bits as desired
- Program EP(0x07) and EP(0x08) with the entire 8 byte LUN
- V1 initiator will use the 1 byte LUN and option bits if no EP support in place or Not present for some reason.
Otherwise, with proper EP support, will assess and use EP

V2 wizard and V2 initiator
- Assess if EPs are present and make the correct decisions (same as V1 wizard and V2 initiator above.

Table 104: Boot LUN Representation in Parm Block Discussion

IPv6 Address Representation In DHCP Server - Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted

[12/19/05]: The suggestion is to comply with RFC 4173 in presenting IPv6 addresses when using inband configuration mode.

[12/19/05]: From various RFCs, it appears IPv6 addresses are delimited by "[" "]". So the current use of "/" "/" should be replaced with "[" "]". The brevity representations are applicable inside the "[" "]". Here are several examples

IPv6 addr of "[3ffe:3700:0200:00ff:0000:0000:0000:0001]"
➔"[3ffe:3700:0200:00ff::1]

IPv6 Link local "[FE80:0000:0000:0000:192.0.0.1]" ➔ "[FE80::192.0.0.1]"

[01/10/06]: Since IPv4 is most prevalent, the demarcation approach vs IPv4 encapsulation approach Will be pursued to minimize changes to all. As a result, the initiator teams need to change the "/" "/" usage needs to be changed to the "[" "]" marker usage.

[02/16/06]: Even though the DHCP scope may be reopened due to 4173 LUN behavior (see below for Thread discussions), the lack of any IPv6 usage to date provides the opportunity change The demarcation markers independent of any DHCP scope issues or any V1/V2 initiator Issues. So regardless of the DHCP scope outcome, the demarcation markers need to be "[" and "]". Note, the demarker approach provides backward compatibility for Current V1 initiators (vs using the encapsulation approach and forcing V1 initiators to Change. If the DHCP scope discussion is approved (for boot LUN definition), then The demarcation vs encapsulation can be revisited since the new scope can force The IP address to be defined as we all agree to.

02/22/06]: FYI…the RFC that establishes "[""]" is RFC 2732

Table 105: IPv6 Representation in DHCP Server Discussion

Boot LUN Representation In DHCP Server - Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted

[12/19/05]: The suggestion is to comply with RFC 4173 in presenting boot LUN using inband configuration mode.

[12/19/05]: In looking at RFC 4173, the LUN presentation follow the following rules:

LUN value of "0" can be omitted completely
LUN is represented as 4 groups of hexadecimal digits separated "-"
Digits above 9 can be upper or lower case characters
Leading "0"s of a given quartet can be omitted
Trailing "0"s and "-" of a given quartet can be omitted

So some examples

LUN 0 ➔ blank or "0" or "0000-0000-0000-0000" or "8000-0000-0000-0000"
LUN 1 ➔ "1" or "0000-0000-0000-0001" or "8000-0000-0000-0001"
LUN 12 ➔ "C" or "0000-0000-0000-000C" or "8000-0000-0000-000C"
LUN 258 ➔ "102" or "0000-0000-0000-0102" or "8000-0000-0000-0102"
LUN 1036 ➔ "40C" or "0000-0000-0000-040C" or "8000-0000-0000-040C"

In general…all the brevity components of RFC 4173 are confusing…so need Prasenjt to concur with examples or expand on the brevity rules!!! For example, SAM uses the leading 0b10 || 0b000000 to defining the LUN addressing format, where would the "80" go?

[01/31/06]: I talked with Prasenjit, he indicated my order is backwards and that the 0x80 is the Most significant bits of the string. As a result, above is updated.

[02/16/06]: Given the size and "-" in the 4173 string, backward compatibility may be a bit dicey. So here Is are a couple of options:

Option 1:

Basically use a new DHCP scope id to demark using 4173 vs the old 2 chr field. The Solution would be something like this: The DHCP server would have the 20x fields Created for both the "IBM ISAN" scope and for a "ISAN" scope. See the DHCP Scope thread in this document for further details. For V1 initiators, The 20x fields would be set to the old format for LUN info. For V2 initiators, the 20x Would be set to the new 4173 compliant LUN info. This would allow V1 initiators And V2 initiators to coexist in the same LAN segment since the EN MAC or client id Would define the per client definition of each of the 20x parameters. The admin Would keep or phase out V1 initiators (via firmware updates or new cards) at his/her Discretion (ie removing V1 client specifies from the dhcp server when ready).

Option 2:

Since the 4173 LUN presentation is much larger, typically, than 2 characters, the V2 initiator could determine which format AS LONG AS the V1 initiator would post An error if the LUN definition is more than 2 characters and AS LONG AS any 4173 Abbreviations that reduce to down to 2 or less characters are identical to the simple 2 character format of today. So for example

- if the LUN is "8000-0000-0000-0001" that is abbreviated to "8000-1" then
  The V1 initiator should post an error (and some readme to help admin fix by Changing the LUN definition across the board to "1" or by changing the LUN To "0" …ie "0000-0000-0000-0000" abbreviated to "0" using the 4713 rules).
  The V2 initiator would be fine with either ""8000-1" or "0"
  NOTE: 4173 format for 0x0000000000000001 is "0-0-0-1"
- if the LUN is "0000-0000-0000-0023" then 4173 abbreviation would be "0-0-0-23". So there would never be simple 2 character format for the LUN.
  So V1 would post an error and need LUN changed throughout solution
  V2 would be fine.
- if the LUN is "69", then V1 initiator is fine. V2, realizing only 2 characters Are in play, would revert to the original 2 character format.

In essence, option 1 is cleaner but does put a burden on the DHCP server while option 2

Could end up being more error prone. [02/22/06]: Option 1 above (new DHCP scope for V2) is the accepted way to do this. The DHCP Section for examples

Table 106: Boot LUN Representation in DHCP Server Block Discussion

SNMP Address for Posting Boot Errors – Declined

Discussion Thread	Leader: Vojnovich	Conclusion: Tentatively Declined
[12/19/05]: The suggestion is to add an SNMP address to both the in band and out of band configuration modes to allow the initiator, in the case of error, to post information to SNMP server. [01/10/05]: No real activity on this front. It may prove difficult to include since the function may Be complex for initiators to implement in terms of space and operation as well as the The ramifications on the parms and DHCP behavior (new vendor option?) to support		

Table 107: SNMP Alert Address Discussion

IPSec Parameter Management – Accepted

Discussion Thread	Leader: Nelson	Conclusion: Accepted

[12/19/05]: The suggestion is to update/modify the current scheme for deploying IPSec configuration parameters. Given that IPSec is not widely implemented in all forms on a variety of iSCSI components, Randy Nelson was going to look at some better optimizations possible (including if the parm options bits need updated).

[01/17/06]: Waiting on Randy

[01/31/06]: Randy came back with a short description of where iSeries wants to take security settings. Basically, the plan is the following:

- OPT[27:24]= 0b0110 and 0b1101 will be changed to reserved
  Other values remain in play
- OPT[23:22] will be changed to reserved
- OPT[21:18]=0b0110 and 0b1101 will be changed to reserved
  Other values remain in play
- OPT[17:16] will be changed to reserved
- 1st target secret data extended parm stays in play
- 2nd target secret data extended parm stays in play
- The PKCS extended parms will be freed up since nobody actually coded to them
  (Will continue to use new extended IDs to make sure we "clear" any usage of
  These extended parms).

Since none of these have been used yet, there should be no implications on wizard, bios, Or initiators.

[02/16/06]: One note on this front. Even with the larger sizes, the usage of these extended parameters For security will likely span 2 blocks and thus need to leverage the block concatenation Scheme.

Table 108: IPSec Configuration Discussion

Generic DHCP Scope – Accepted

Discussion Thread	Leader: Vojnovich	Conclusion: Accepted
[12/19/05]: The suggestion is to update/modify the current scope of "IBM ISAN" used in the inband deployment (DHCP vendor scope) to a more acceptable, generic name. This has several implications. First, do we want to maintain the DHCP 60 long string format so that initiators can identify revision levels and such to the server during discovery (and thus provide a nice coexistence/migration path between old and new). Second, we would need to make sure that the scope is fairly unique so there are not scope collisions while making it general enough to be acceptable to industry vendors. [01/10/06]: Not a lot of work progressed on this front. The only movement is in the suggestion Of the generic scope name. Proposals include "ISAN" or "IP SAN" [01/17/06]: Since iSeries has built its environment around "IBM ISAN" and since this is an IBM Centric solution and since any admin can change the scope as desired (via wizard), it has Been decided to leave this parameter alone. If in the future, somebody wishes to push This as an industry standard, we can revisit this and assess options including using the DHCP vendor option 204 in an "IBM ISAN" scope to define another scope that provides The actual DHCP vendor options for the iSCSI information [02/16/06]: With the proposal options for 4173 compliant LUN format, the DHCP scope option may Need to be reopened. Basically, to cleanly help the admin manage backward compatibility And forward migration, reopening the DHCP scope can significantly assist. Basically The idea is the following - The DHCP server would have common 20x parameters created. The DHCP server Would have 2 scopes in play where one scope covers V1 initiators and the other Scope covers V2 initiators. - The scope option (DHCP option 60) used during DHCPDISCOVER or DHCPREQUEST would be the following: For V1 initiators, "IBM ISAN". For V2 initiators, "ISAN". Note, this still leaves the door open to using the Currently defined option 60 format to append vendor/version/etc fields on The option 60 if a particular implementation or vendor wishes to do so. Ie "IBM ISAN: 28:43:Qlogic:92:88" can be used as the option 60 value and The DHCP server can use all or just the leading x characters for the scope Identification. - V1 initiators would use "IBM ISAN" or "IBM ISAN:<blah>:<blah>:<etc>" - V2 initiators would user "ISAN" or "ISAN:<blah>:<blah>:<etc>" Of course, a given customer can override these defaults with their own scope either By passing down scope via out of band config or use DHCP Option 204 to define The new scope to use to get the actual parameters (for relay or proxy like behavior). [02/22/06]: using DHCP scope to demarcate V1 vs V2 solutions is simplest/least error prone. One note, if the Admin chooses unique DHCP scopes, the admin must make V1 scope Different from V2 scope.		

Table 109: Generic DHCP Scope Discussion

DHCP Scope Examples: Option 17

Here is a traditional DHCP config file example using option 17 to indicate iSCSI boot configuration between an V1 initiator and V2 initiator.

```
#############
# DHCP config File
############
#DHCP 3.0
use-host-decl-names on;
allow booting;
allow bootp;

ddns-update-style ad-hoc;

subnet 123.123.123.0    netmask 255.255.255.0 {
option dhcp-lease-time -1;
option routers 123.123.123.31;
option domain-name-servers 123.123.123.32,123.123.123.33;
option domain-name "haifa.ibm.com";

################
# Admin knows that blade A is using a V1 initiator
# for this example, assume boot LUN = 0x3
###############

        host blade_a {
        option root-path "iscsi:123.123.123.11:8:3260:3:iqn.2005-03.com.ibm.lun_a";
        hardware ethernet 11:11:11:11:11:11;
        fixed-address 123.123.123.1;
        }

##################
# Admin knows that blade B is using V2 initiator
# for this example, assume boot LUN = 0x8000000000000003…RFC 4173 LUN abbreviations not done for clarity
# Note iSCSI names remain small to be readible
#################

        host blade_b {
        option root-path "iscsi:123.123.123.11:8:3260:8000-0000-0000-0003:iqn.2005-03.com.ibm.lun_b";
        hardware ethernet 22:22:22:22:22:22;
        fixed-address 123.123.123.2;
        }
```

Figure 42: Root Path (Option 17) Usage in the Context of V1 and V2

DHCP Scope Examples: Option 201/202

Here is a traditional DHCP config file example using option 201/202 to indicate iSCSI boot configuration between an V1 initiator and V2 initiator.

```
#############
# DHCP config File
############
```

```
#DHCP 3.0
allow booting;
allow bootp;

ddns-update-style ad-hoc;

subnet 123.123.123.0    netmask 255.255.255.0 {
option dhcp-lease-time -1;
option routers 123.123.123.31;
option domain-name-servers 123.123.123.32,123.123.123.33;
option domain-name "haifa.ibm.com";

#define option space
option space iSCSI;
option iSCSI.initiator code 203 = string;
option iSCSI.target1 code 201 = string;
option iSCSI.target2 code 202 = string;
#more iscsi options

#the actual vendor-class-identifier sent by the client = “IBM ISAN:3:1:IBM :29:56”

########
# Vendor scope for V1 initiators…defaults used here
#######

class “iscsi_v1” {
    Match if substring(option vendor-class-identifier,0,8) = ”IBM ISAN”;       #increase 8 to catch more of string
    #Global class options as needed
    group {
                subclass “vendor classes” “IBM ISAN” {
                    vendor-option-space iSCSI;
                }

                host blade_a {
                    hardware Ethernet = 11:11:11:11:11:11;
                    fixed address = 123.123.123.1;
                    option iSCSI.initiator = “iqn.2005-03.com.ibm.blade_a”;
                    option iSCSI.target1 = “iscsi:123.123.123.11:6:3260:0:iqn.2005-03.com.ibm.lun_a1”;
                    option iSCSI.target2 = “iscsi:123.123.123.21:6:3260:0:iqn.2005.03.com.ibm.lun_a2”;
                    #more iSCSI and DHCP options as needed
                } #end blade_a
    } #end group
}

########
# Vendor scope for V2 initiators…defaults scopes used here
# For this example, the boot LUN is 0x8000000000000003…RFC 4173 abbreviations omitted for clarity
# note iSCSI names remain small to be readable
#######

class “iscsi_v2” {
    Match if substring(option vendor-class-identifier,0,8) = ”ISAN”;       #increase 4 to catch more of string
    #Global class options as needed
    group {
                subclass “vendor classes” “ISAN” {
                    vendor-option-space iSCSI;
                }

                host blade_b {
                    hardware Ethernet = 22:22:22:22:22:22;
                    fixed address = 123.123.123.2;
                    option iSCSI.initiator = “iqn.2005-03.com.ibm.blade_b”;
                    option iSCSI.target1 = “iscsi:123.123.123.11:6:3260:8000-0000-0000-0003:iqn.2005-03.com.ibm.lun_b1”;
                    option iSCSI.target2 = “iscsi:123.123.123.21:6:3260:8000-0000-0000-0003:iqn.2005.03.com.ibm.lun_b2”;
                    #more iSCSI and DHCP options as needed
                } #end blade_b
    } #end group
}
```

Figure 43: Vendor Options 201/202 in the Context of V1 and V2

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATIONS ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publications. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Trademarks

The following are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
IBM
IBM logo
Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.