TCG Storage Security Subsystem Class: Ruby

Version 1.00
Revision 1.00

January 07, 2020

Contact: admin@trustedcomputinggroup.org

PUBLISHED

Copyright © TCG 2020

TCG

Copyright © 2020 Trusted Computing Group, Incorporated.

Disclaimers, Notices, and License Terms

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.
Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.
This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on specification licensing through membership agreements.
Any marks and brands contained herein are the property of their respective owners.

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 DOCUMENT PURPOSE 1
1.2 SCOPE AND INTENDED AUDIENCE 1
1.3 KEY WORDS 1
1.4 DOCUMENT REFERENCES 1
1.5 DOCUMENT PRECEDENCE 2
1.6 SSC TERMINOLOGY 2
1.7 LEGEND 3

2 RUBY SSC OVERVIEW 4

2.1 RUBY SSC USE CASES AND THREATS 4
2.2 SECURITY PROVIDERS (SPS) 4
2.3 INTERFACE COMMUNICATION PROTOCOL 4
2.4 CRYPTOGRAPHIC FEATURES 4
2.5 AUTHENTICATION 5
2.6 TABLE MANAGEMENT 5
2.7 ACCESS CONTROL & PERSONALIZATION 5
2.8 ISSUANCE 5
2.9 SSC DISCOVERY 5
2.10 MBR SHADOWING 5
2.11 ADDITIONAL LOCKING RANGES 5
2.12 MANDATORY FEATURE SETS 5
2.13 OPTIONAL FEATURE SETS 6

3 RUBY SSC FEATURES 7

3.1 SECURITY PROTOCOL 1 SUPPORT 7
3.1.1 Level 0 Discovery (M) 7
3.1.1.1 Level 0 Discovery Header 7
3.1.1.2 TPer Feature (Feature Code = 0x0001) 8
3.1.1.3 Locking Feature (Feature Code = 0x0002) 8
3.1.1.3.1 MBR Done Bit Definition 9
3.1.1.3.2 MBR Enabled Bit Definition 9
3.1.1.3.3 Locking Enabled Bit Definition 9
3.1.1.4 Geometry Reporting Feature (Feature Code = 0x0003) 10
3.1.1.4.1 Overview 10
3.1.1.4.2 Align 10
3.1.1.4.3 LogicalBlockSize 10
3.1.1.4.4 AlignmentGranularity 11
3.1.1.4.5 LowestAlignedLBA 11
3.1.1.5 Ruby SSC V1.00 Feature (Feature Code = 0x0304) 12
3.2 SECURITY PROTOCOL 2 SUPPORT 13
3.2.1 ComID Management 13
3.2.2 Stack Protocol Reset (M) 13
3.2.3 TPER_RESET command (M) 13
3.3 COMMUNICATIONS 13
3.3.1 Communication Properties 13
3.3.2 Supported Security Protocols 14
3.3.3 ComIDs 14
3.3.4 Synchronous Protocol 15
3.3.4.1 Payload Encoding 15
3.3.4.1.1 Stream Encoding Modifications 15
3.3.4.1.2 TCG Packets 15

3.3.4.1.3 Payload Error Response 15
3.3.5 Storage Device Resets *16*
3.3.5.1 Interface Resets 16
3.3.5.2 TCG Reset Events 16
3.3.6 Protocol Stack Reset Commands (M) *16*

4 RUBY SSC-COMPLIANT FUNCTIONS AND SPS 17

4.1 SESSION MANAGER 17
4.1.1 Methods *17*
4.1.1.1 Properties (M) 17
4.1.1.2 StartSession (M) 18
4.1.1.3 SyncSession (M) 18
4.1.1.4 CloseSession (O) 18
4.2 ADMIN SP 18
4.2.1 Base Template Tables *18*
4.2.1.1 SPInfo (M) 18
4.2.1.2 SPTemplates (M) 18
4.2.1.3 Table (M) 19
4.2.1.4 MethodID (M) 20
4.2.1.5 AccessControl (M) 20
4.2.1.6 ACE (M) 26
4.2.1.7 Authority (M) 27
4.2.1.8 C_PIN (M) 28
4.2.2 Base Template Methods *28*
4.2.3 Admin Template Tables *29*
4.2.3.1 TPerInfo (M) 29
4.2.3.2 Template (M) 29
4.2.3.3 SP (M) 30
4.2.4 Admin Template Methods *30*
4.2.5 Crypto Template Tables *30*
4.2.6 Crypto Template Methods *30*
4.2.6.1 Random 30
4.3 LOCKING SP 30
4.3.1 Base Template Tables *30*
4.3.1.1 SPInfo (M) 30
4.3.1.2 SPTemplates (M) 31
4.3.1.3 Table (M) 31
4.3.1.4 Type (N) 32
4.3.1.5 MethodID (M) 32
4.3.1.6 AccessControl (M) 33
4.3.1.7 ACE (M) 56
4.3.1.8 Authority (M) 59
4.3.1.9 C_PIN (M) 60
4.3.1.10 SecretProtect (M) 61
4.3.2 Base Template Methods *61*
4.3.3 Crypto Template Tables *61*
4.3.4 Crypto Template Methods *61*
4.3.4.1 Random 61
4.3.5 Locking Template Tables *61*
4.3.5.1 LockingInfo (M) 61
4.3.5.2 Locking (M) 62
4.3.5.2.1 Geometry Reporting Feature Behavior 64
4.3.5.2.2 LockOnReset Restrictions 64
4.3.5.3 MBRControl (MBR_O) 64
4.3.5.3.1 DoneOnReset Restrictions 65
4.3.5.4 MBR (MBR_O) 65

4.3.5.5 K_AES_128 or K_AES_256 (M) 65
4.3.6 Locking Template Methods *66*
4.3.7 SD Read/Write Data Command Locking Behavior Interactions with Range Crossing *67*
4.3.8 Non Template Tables *67*
4.3.8.1 DataStore (M) 67
5 APPENDIX – SSC SPECIFIC FEATURES 68
5.1 RUBY SSC-SPECIFIC METHODS 68
5.1.1 Activate – Admin Template SP Object Method *68*
5.1.1.1 Activate Support 68
5.1.1.2 Side effects of Activate 68
5.1.2 Revert – Admin Template SP Object Method *68*
5.1.2.1 Revert Support 69
5.1.2.2 Side effects of Revert 69
5.1.2.2.1 Effects of Revert on the PIN Column Value of C_PIN_SID 70
5.1.3 RevertSP – Base Template SP Method *70*
5.1.3.1 RevertSP Support 70
5.1.3.2 KeepGlobalRangeKey parameter (Locking Template-specific) 71
5.1.3.3 Side effects of RevertSP 71
5.2 LIFE CYCLE 72
5.2.1 Issued vs. Manufactured SPs *72*
5.2.1.1 Issued SPs 72
5.2.1.2 Manufactured SPs 72
5.2.2 Manufactured SP Life Cycle States *72*
5.2.2.1 State definitions for Manufactured SPs 73
5.2.2.2 State transitions for Manufactured SPs 73
5.2.2.2.1 Manufactured-Inactive to Manufactured 73
5.2.2.2.2 ANY STATE to ORIGINAL FACTORY STATE 74
5.2.2.3 State behaviors for Manufactured SPs 74
5.2.2.3.1 Manufactured-Inactive 74
5.2.2.3.2 Manufactured 74
5.2.3 Type Table Modification *74*
5.3 BYTE TABLE ACCESS GRANULARITY 75
5.3.1 Table Table Modification *75*
5.3.1.1 MandatoryWriteGranularity 75
5.3.1.1.1 Object Tables 75
5.3.1.1.2 Byte Tables 75
5.3.1.2 RecommendedAccessGranularity 76
5.3.1.2.1 Object Tables 76
5.3.1.2.2 Byte Tables 76
5.4 EXAMPLES OF ALIGNMENT GEOMETRY REPORTING 77

TABLES

Table 1 Ruby SSC Terminology 2
Table 2 SP Table Legend 3
Table 3 Level 0 Discovery Header 7
Table 4 Level 0 Discovery - TPer Feature Descriptor 8
Table 5 Level 0 Discovery - Locking Feature Descriptor 8
Table 6 Level 0 Discovery - Geometry Reporting Feature Descriptor 10
Table 7 Level 0 Discovery - Ruby SSC V1.00 Feature Descriptor 12
Table 8 TPER_RESET Command 13
Table 9 ComID Assignments 14
Table 10 Supported Tokens 15

Table 11 reset_types 16
Table 12 Properties Requirements 17
Table 13 Admin SP - SPInfo Table Preconfiguration 18
Table 14 Admin SP - SPTemplates Table Preconfiguration 18
Table 15 Admin SP - Table Table Preconfiguration 19
Table 16 Admin SP - MethodID Table Preconfiguration 20
Table 17 Admin SP - AccessControl Table Preconfiguration 21
Table 18 Admin SP - ACE Table Preconfiguration 26
Table 19 Admin SP - Authority Table Preconfiguration 27
Table 20 Admin SP - C_PIN Table Preconfiguration 28
Table 21 Admin SP – TPerInfo Columns 29
Table 22 Admin SP - TPerInfo Table Preconfiguration 29
Table 23 Admin SP - Template Table Preconfiguration 29
Table 24 Admin SP - SP Table Preconfiguration 30
Table 25 Locking SP - SPInfo Table Preconfiguration 30
Table 26 Locking SP - SPTemplates Table Preconfiguration 31
Table 27 Locking SP - Table Table Preconfiguration 31
Table 28 Locking SP - MethodID Table Preconfiguration 32
Table 29 Locking SP - AccessControl Table Preconfiguration 33
Table 30 Locking SP - ACE Table Preconfiguration 56
Table 31 Locking SP - Authority Table Preconfiguration 59
Table 32 Locking SP - C_PIN Table Preconfiguration 60
Table 33 Locking SP - SecretProtect Table Preconfiguration 61
Table 34 Locking SP – LockingInfo Columns 62
Table 35 Locking SP - LockingInfo Table Preconfiguration 62
Table 36 Locking SP - Locking Table Preconfiguration 63
Table 37 Locking SP - MBRControl Table Preconfiguration 65
Table 38 Locking SP - K_AES_128 Table Preconfiguration 65
Table 39 Locking SP - K_AES_256 Table Preconfiguration 65
Table 40 LifeCycle Type Table Modification 74
Table 41 Table Table Additional Columns 75

TCG Copyright 2020

1 Introduction

1.1 Document Purpose

The Storage Workgroup specifications provide a comprehensive architecture for putting Storage Devices under policy control as determined by the trusted platform host, the capabilities of the Storage Device to conform to the policies of the trusted platform, and the lifecycle state of the Storage Device as a Trusted Peripheral.

1.2 Scope and Intended Audience

This specification defines the Ruby Security Subsystem Class (SSC). Any SD that claims Ruby SSC compatibility SHALL conform to this specification.

The intended audience for this specification is both trusted Storage Device manufacturers and developers that want to use these Storage Devices in their systems.

1.3 Key Words

Key words are used to signify SSC requirements.

The Key Words "**SHALL**", "**SHALL NOT**", "**SHOULD**," and "**MAY**" are used in this document. These words are a subset of the RFC 2119 key words used by TCG, and have been chosen since they map to key words used in T10/T13 specifications. These key words are to be interpreted as described in [1].

The following key words are used in this document to mark specific features as Optional

- (MBR O) Values in this specification marked with this identifier are components of the MBR Shadowing feature.
- (ALR O) Values in this specification with this identifier are components of the Additional Locking Ranges feature.

In addition to the above key words, the following are also used in this document to describe the requirements of particular features, including tables, methods, and usages thereof.

- **Mandatory (M):** When a feature is Mandatory, the feature SHALL be implemented. A Compliance test SHALL validate that the feature is operational.
- **Optional (O):** When a feature is Optional, the feature MAY be implemented. If implemented, a Compliance test SHALL validate that the feature is operational.
- **Excluded (X):** When a feature is Excluded, the feature SHALL NOT be implemented. A Compliance test SHALL validate that the feature is not operational.
- **Not Required (N)** When a feature is Not Required, the feature MAY be implemented. No Compliance test is required.

1.4 Document References

[1]. IETF RFC 2119, 1997, "Key words for use in RFCs to Indicate Requirement Levels"

[2]. Trusted Computing Group (TCG), "TCG Storage Architecture Core Specification", Version 2.01

[3]. NIST, FIPS-197, 2001, "Advanced Encryption Standard (AES)"

[4]. Trusted Computing Group (TCG), "TCG Storage Interface Interactions Specification", Version 1.08

[5]. Trusted Computing Group (TCG), "TCG Storage Opal SSC Feature Set: PSID", Version 1.00

[6]. Trusted Computing Group (TCG), "TCG Storage Feature Set: Block SID Authentication", Version 1.00

[7]. Trusted Computing Group (TCG), "TCG Storage Feature Set: Single_User_Mode", Version 1.00

[8]. Trusted Computing Group (TCG), "Configurable Namespace Locking Feature Set", Version 1.00

[9]. TCG Storage Opal SSC Feature Set: PSK Secure Messaging, Version 1.00, Revision 1.00

1.5 Document Precedence

In the event of conflicting information in this specification and other documents, the precedence for requirements is:

1. This specification
2. Storage Interface Interactions Specification [4]
3. TCG Storage Architecture Core Specification [2]

1.6 SSC Terminology

This section provides special definitions that are not defined in [2].

Table 1 Ruby SSC Terminology

Term	Definition
Manufactured SP	A Manufactured SP is an SP that was created and preconfigured during the SD manufacturing process
N/A	Not Applicable.
Original Factory State (OFS)	The original state of an SP when it was created in manufacturing, including its table data, access control settings, and life cycle state. Each Manufactured SP has its own Original Factory State. Original Factory State applies to Manufactured SPs only.
Vendor Unique (VU)	These values are unique to each SD manufacturer. Typically VU is used in table cells.
MM MM	The LSBs of a User Authority object's UID (hexadecimal) as well as the corresponding C_PIN credential object's UID (hexadecimal)
NN NN	The LSBs of a Locking object's UID (hexadecimal) as well as the corresponding K_AES_128/K_AES_256 object's UID (hexadecimal)
XX XX	The LSBs of an Admin Authority object's UID (hexadecimal) as well as the corresponding C_PIN credential object's UID (hexadecimal)

TCG Copyright 2020
Published

1.7 Legend

The following legend defines SP table cell coloring coding. This color coding is informative only. The table cell content is normative.

Table 2 SP Table Legend

Table Cell Legend	R-W	Value	Access Control	Comment
Arial-Narrow	Read-only	Ruby SSC specified	Fixed	• Cell content is Read-Only. • Access control is fixed. • Value is specified by the Ruby SSC
Arial Narrow bold-under	Read-only	VU	Fixed	• Cell content is Read-Only. • Access Control is fixed. • Values are Vendor Unique (VU). A minimum or maximum value may be specified.
Arial-Narrow	Not Defined	(N)	Not Defined	• Cell content is (N). • Access control is not defined. • Any text in table cell is informative only. • A `Get` MAY omit this column from the method response.
Arial Narrow bold-under	Write	Preconfigured, user personalizable	Preconfigured, user personalizable	• Cell content is writable. • Access control is personalizable • `Get` Access Control is not described by this color coding
Arial-Narrow	Write	Preconfigured, user personalizable	Fixed	• Cell content is writable. • Access control is fixed. • `Get` Access Control is not described by this color coding

TCG Copyright 2020

2 Ruby SSC Overview

2.1 Ruby SSC Use Cases and Threats

Begin Informative Content

This specification is an implementation profile for trusted storage devices commonly deployed within, but not limited to, Data Center/Bulk Data/Enterprise class systems.

The Ruby SSC is an implementation profile for Storage Devices built to:

- Protect the confidentiality of stored user data against unauthorized access once it leaves the owner's control (involving a power cycle and subsequent deauthentication)
- Enable interoperability between multiple SD vendors

A Ruby SSC compliant SD:

- Facilitates feature discoverability
- Provides some user definable features (e.g. access control, locking ranges, user passwords, etc.)
- Supports Ruby SSC unique behaviors (e.g. communication, table management)

This specification addresses a limited set of use cases. They are:

- Deploy Storage Device & Take Ownership: the Storage Device is integrated into its target system and ownership transferred by setting or changing the Storage Device's owner credential.
- Activate or Enroll Storage Device: LBA ranges are configured and data encryption and access control credentials (re)generated and/or set on the Storage Device. Access control is configured for LBA range unlocking.
- Lock & Unlock Storage Device: unlocking of one or more LBA ranges by the host and locking of those ranges under host control via either an explicit lock or implicit lock triggered by a reset event.
- MBR Shadowing (if supported) provides a mechanism to boot into a secure pre-boot authentication environment to handle device unlocking.
- Repurpose & End-of-Life: erasure of data within one or more LBA ranges and reset of locking credential(s) for Storage Device repurposing or decommissioning.

End Informative Content

2.2 Security Providers (SPs)

A Ruby SSC compliant SD SHALL support at least two Security Providers (SPs):

1) Admin SP
2) Locking SP

The Locking SP MAY be created by the SD manufacturer.

2.3 Interface Communication Protocol

A Ruby SSC compliant SD SHALL implement the synchronous communications protocol as defined in Section 3.3.4.

This communication protocol operates based upon configuration information defined by:

1) The values reported via Level 0 Discovery (Section 3.1.1);
2) The combination of the host's communication properties and the TPer's communication properties (see Properties Method Section 4.1.1.1).

2.4 Cryptographic Features

A Ruby SSC compliant SD SHALL implement Full Disk Encryption for all host accessible user data stored on media. AES-128 or AES-256 SHALL be supported (see [3]).

TCG Copyright 2020

2.5 Authentication

A Ruby SSC compliant SD SHALL support password authorities and authentication.

2.6 Table Management

This specification defines the mandatory tables and mandatory/optional table rows delivered by the SD manufacturer. The creation or deletion of tables after manufacturing is outside the scope of this specification. The creation or deletion of table rows post-manufacturing is outside the scope of this specification.

2.7 Access Control & Personalization

Initial access control policies are preconfigured at SD manufacturing time on manufacturer created SPs. A Ruby SSC compliant SD SHALL support personalization of certain Access Control Elements of the Locking SP.

2.8 Issuance

The Locking SP MAY be present in the SD when the SD leaves the manufacturer. The issuance of SPs is outside the scope of this specification.

2.9 SSC Discovery

Discovery is a process for the Host to examine the storage device's TPer configurations, capabilities, and class.

Refer to [2] for details (see Section 3.1.1).

2.10 MBR Shadowing

"MBR Shadowing" is a feature that allows loading and execution of pre(OS)-boot code that is used to unlock LBA ranges. This feature and its components are Optional in this specification.

MBR Shadowing components are marked with the identifier "(MBR O)".

If the MBR Shadowing feature is supported, then all components of this feature as identified by "(MBR O)" SHALL be supported, and Level 0 discovery SHALL report that MBR Shadowing is supported.

If the MBR Shadowing feature is not supported, then components identified by "(MBR O)" MAY be absent, and Level 0 discovery SHALL report that MBR Shadowing is not supported.

2.11 Additional Locking Ranges

"Additional Locking Ranges" is a feature that provides support for separate media encryption and locking support for locking ranges in addition to the Global Range. This feature and its components are Optional in this specification.

Additional Locking Range components are marked with the identifier "(ALR O)".

If the Additional Locking Ranges feature is supported, then all components of this feature as identified by "(ALR O)" SHALL be supported, and the MaxRanges column of the LockingInfo table shall indicate the number of ranges that are supported.

If the Additional Locking Ranges feature is not supported, then components identified by "(ALR O)" MAY be not supported, and the MaxRanges column value of the LockingInfo table shall be 0.

2.12 Mandatory Feature Sets

A Ruby SSC compliant SD SHALL support the following TCG Storage Feature Sets:

1) PSID, Opal SSC Feature Set (refer to [5]).
2) Block SID Authentication (refer to [6])

TCG Copyright 2020
Published

2.13 Optional Feature Sets

A Ruby SSC compliant SD MAY support the following TCG Feature Sets:

1) Single User Mode Feature Set (refer to [7])
2) Configurable Namespace Locking Feature Set (refer to [8])
3) PSK Secure Msg Feature Set (refer to [9])

TCG Copyright 2020
Published

3 Ruby SSC Features

3.1 Security Protocol 1 Support

3.1.1 Level 0 Discovery (M)

Refer to [2] for more details.

A Ruby SSC compliant SD SHALL return the following Level 0 response:

- Level 0 Discovery Header
- TPer Feature Descriptor
- Locking Feature Descriptor
- Ruby SSC Feature Descriptor

3.1.1.1 Level 0 Discovery Header

Table 3 Level 0 Discovery Header

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				Length of Parameter Data				
2								
3								(LSB)
4	(MSB)							
5				Data structure revision				
6								
7								(LSB)
8	(MSB)							
…				Reserved				
15								(LSB)
16	(MSB)							
…				Vendor Specific				
47								(LSB)

- Length of parameter data = VU
- Data structure revision = 0x00000001 or any version that supports the defined features in this SSC
- Vendor Specific = VU

3.1.1.2 TPer Feature (Feature Code = 0x0001)

Table 4 Level 0 Discovery - TPer Feature Descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	Feature Code (0x0001)							(LSB)
2	Version				Reserved			
3	Length							
4	Reserved	ComID Mgmt Supported	Reserved	Streaming Supported	Buffer Mgmt Supported	ACK/NAK Supported	Async Supported	Sync Supported
5 - 15	Reserved							

- Feature Code = 0x0001
- Version = 0x1 or any version that supports the defined features in this SSC
- Length = 0x0C
- ComID Mgmt Supported = VU
- Streaming Supported = 1
- Buffer Mgmt Supported = VU
- ACK/NACK Supported = VU
- Async Supported = VU
- Sync Supported = 1

3.1.1.3 Locking Feature (Feature Code = 0x0002)

Table 5 Level 0 Discovery - Locking Feature Descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	Feature Code (0x0002)							(LSB)
2	Version				Reserved			
3	Length							
4	Reserved	MBR Shadowing Not Supported	MBR Done	MBR Enabled	Media Encryption	Locked	Locking Enabled	Locking Supported
5 - 15	Reserved							

- Feature Code = 0x0002
- Version = 0x2 or any version that supports the defined features in this SSC
- Length = 0x0C
- MBR Shadowing Not Supported (MBR O) = VU
  - If MBR Shadowing feature is supported, this bit SHALL be set to 0.
  - If MBR Shadowing feature is not supported, this bit SHALL be set to 1.
- MBR Done (MBR O) = See 3.1.1.3.1
- MBR Enabled (MBR O) = See 3.1.1.3.2
- Media Encryption = 1

- Locked = see [2]
- Locking Enabled = See 3.1.1.3.1
- Locking Supported = 1

3.1.1.3.1 MBR Done Bit Definition

The definition of the MBR Done bit is changed from [2] as follows:

If

a) MBR Shadowing Not Supported bit is set to zero;

b) MBR Enabled bit is set to one; and

c) MBRControl table's Done column has a value of "True",

then the MBR Done bit SHALL be set to one.

Else, the MBR Done bit SHALL be set to zero.

3.1.1.3.2 MBR Enabled Bit Definition

The definition of the MBR Enabled bit is changed from [2] as follows:

if

a) MBR Shadowing Not Supported bit is set to zero;

b) Locking Enabled bit is set to one; and

c) MBRControl table's Enabled column has a value of "True",

then the MBR Enabled bit SHALL be set to one.

Else the MBR Enabled bit SHALL be set to zero.

3.1.1.3.3 Locking Enabled Bit Definition

The definition of the Locking Enabled bit is changed from [2] as follows:

The Locking Enabled bit SHALL be set to one if an SP that incorporates the Locking template is in any state other than Nonexistent or "Manufactured-Inactive" state; otherwise, the Locking Enabled bit SHALL be set to zero.

3.1.1.4 Geometry Reporting Feature (Feature Code = 0x0003)

3.1.1.4.1 Overview

This information indicates support for logical block and physical block geometry. This feature MAY be returned in the Level 0 Discovery response. See [2] for additional information.

Table 6 Level 0 Discovery - Geometry Reporting Feature Descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	Feature Code (0x0003)						
1								(LSB)
2	Version				Reserved			
3	Length							
4	Reserved							ALIGN
5		Reserved						
6								
7								
8								
9								
10								
11								
12	(MSB)	LogicalBlockSize						
13								
14								
15								(LSB)
16	(MSB)	AlignmentGranularity						
17								
18								
19								
20								
21								
22								
23								(LSB)
24	(MSB)	LowestAlignedLBA						
25								
26								
27								
28								
29								
30								
31								(LSB)

- The Feature Code field SHALL be set to 0x0003.
- The Version field SHALL be set to 0x01.
- The Length field SHALL be set to 0x1C.

3.1.1.4.2 Align

If the value of the AlignmentRequired column of the LockingInfo table is TRUE, then the ALIGN bit shall be set to one. If the value of the AlignmentRequired column of the LockingInfo table is FALSE, then the ALIGN bit shall be set to zero.

3.1.1.4.3 LogicalBlockSize

LogicalBlockSize SHALL be set to the value of the LogicalBlockSize column in the LockingInfo table.

3.1.1.4.4 AlignmentGranularity

AlignmentGranularity SHALL be set to the value of the AlignmentGranularity column in the LockingInfo table.

3.1.1.4.5 LowestAlignedLBA

LowestAlignedLBA SHALL be set to the value of the LowestAlignedLBA column in the LockingInfo table.

3.1.1.5 Ruby SSC V1.00 Feature (Feature Code = 0x0304)

Table 7 Level 0 Discovery - Ruby SSC V1.00 Feature Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			Feature Code (0x0304)				
1								(LSB)
2	Version				Reserved			
3	Length							
4	(MSB)			Base ComID				
5								(LSB)
6	(MSB)			Number of ComIDs				
7								(LSB)
8	Reserved for future common SSC parameters							Range Crossing Behavior
9	(MSB)			Number of Locking SP Admin Authorities Supported				
10								(LSB)
11	(MSB)			Number of Locking SP User Authorities Supported				
12								(LSB)
13	Initial C_PIN_SID PIN Indicator							
14	Behavior of C_PIN_SID PIN upon TPer Revert							
15-19	Reserved for future common SSC parameters							

- Feature Code = 0x0304
- Version = 0x1 or any version that supports the defined features in this SSC
- Length = 0x10
- Base ComID = VU
- Number of ComIDs >= 1
- Range Crossing Behavior = VU
  - If set to 0, then the SD supports commands addressing consecutive LBAs in more than one LBA range if all the LBA ranges addressed are unlocked. See Section 4.3.7
  - If set to 1, then the SD terminates commands addressing consecutive LBAs in more than one LBA range. Termination is described in Section 4.3.7
- Number of Locking SP Admin Authorities >= 1
- Number of Locking SP User Authorities >= 2
- Initial C_PIN_SID PIN Indicator = VU
  - 0x00 = The initial C_PIN_SID PIN value is equal to the C_PIN_MSID PIN value
  - 0xFF = The initial C_PIN_SID PIN value is VU, and MAY not be equal to the C_PIN_MSID PIN value
  - 0x02 – 0x0F = Reserved
- Behavior of C_PIN_SID PIN upon TPer Revert = VU
  - 0x00 = The C_PIN_SID PIN value becomes the value of the C_PIN_MSID PIN column after successful invocation of Revert on the Admin SP's object in the SP table
  - 0xFF = The C_PIN_SID PIN value changes to a VU value after successful invocation of Revert on the Admin SP's object in the SP table, and MAY not be equal to the C_PIN_MSID PIN value

3.2 Security Protocol 2 Support

3.2.1 ComID Management

ComID management support is reported in Level 0 Discovery. Statically allocated ComIDs are also discoverable via the Level 0 Discovery response.

3.2.2 Stack Protocol Reset (M)

A Ruby SSC compliant SD SHALL support the Stack Protocol Reset command. Refer to [2] for details.

3.2.3 TPER_RESET command (M)

If the TPER_RESET command is enabled, the TPER_RESET command SHALL cause the following before the TPer accepts the next IF-SEND or IF-RECV command:

a) all dynamically allocated ComIDs SHALL return to the Inactive state;
b) all open sessions SHALL be aborted on all ComIDs;
c) all uncommitted transactions SHALL be aborted on all ComIDs;
d) the synchronous protocol stack for all ComIDs SHALL be reset to its initial state;
e) all TCG command and response buffers SHALL be invalidated for all ComIDs;
f) all related method processing occurring on all ComIDs SHALL be aborted;
g) TPer's knowledge of the host's communications capabilities, on all ComIDs, SHALL be reset to the initial minimum assumptions defined in [2] or the TPer's SSC definition;
h) the values of the ReadLocked and WriteLocked columns SHALL be set to True for all Locking SP's Locking objects that contain the Programmatic enumeration value in the LockedOnReset column;
i) if the MBR Shadowing feature is supported (e.g. MBR Shadowing Not Supported bit in the Locking Feature Descriptor is set to 0), then the value of the Done column of the Locking SP's MBRControl table SHALL be set to False, if the DoneOnReset column contains the Programmatic enumeration value.

The TPER_RESET command is delivered by the transport IF-SEND command. If the TPER_RESET command is enabled, the TPer SHALL accept and acknowledge it at the interface level. If the TPER_RESET command is disabled, the TPer SHALL abort it at the interface level with the "Other Invalid Command Parameter" status (see [4]). There is no IF-RECV response to the TPER_RESET command.

The TPER_RESET command is defined in Table 8.

The Transfer Length SHALL be non-zero. All data transferred SHALL be ignored.

Table 8 TPER_RESET Command

FIELD	VALUE
Command	IF-SEND
Protocol ID	0x02
Transfer Length	Non-zero
ComID	0x0004

3.3 Communications

3.3.1 Communication Properties

The TPer SHALL support the minimum communication buffer size as defined in Section 4.1.1.1. For each ComID, the physical buffer size SHALL be reported to the host via the `Properties` method.

The TPer SHALL terminate any IF-SEND command whose transfer length is greater than the reported MaxComPacketSize size for the corresponding ComID. Refer to [4] for details about Invalid Transfer Length parameter on IF-SEND".

Data generated in response to methods contained within an IF-SEND command payload subpacket (including the required ComPacket / Packet / Subpacket overhead data) SHALL fit entirely within the response buffer. If the method response and its associated protocol overhead do not fit completely within the response buffer, the TPer:

1) SHALL terminate processing of the IF-SEND command payload,
2) SHALL NOT return any part of the method response if the Synchronous Protocol is being used, and
3) SHALL return an empty response list with a TCG status code of RESPONSE_OVERFLOW in that method's response status list.

3.3.2 Supported Security Protocols

The TPer SHALL support:

- IF-RECV commands with a Security Protocol values of 0x00, 0x01, 0x02.
- IF-SEND commands with a Security Protocol values of 0x01, 0x02.

3.3.3 ComIDs

For the purpose of communication using Security Protocol value 0x01, the TPer SHALL:

- support at least one statically allocated ComID for Synchronous Protocol communication.
- have the ComID Extension values set to 0x0000 for all statically allocated ComIDs.
- keep all statically allocated ComIDs in the Active state.

When the TPer receives an IF-SEND or IF-RECV command with an inactive or unsupported ComID, the TPer SHALL either:

- terminate the command as defined in [4] with "Other Invalid Command parameter", or
- follow the requirements defined in [2] for "IF-SEND to Inactive or Unsupported Reserved ComID" or "IF-RECV to Inactive or Unsupported Reserved ComID".

ComIDs SHALL be assigned based on the allocation shown in Table 9.

Table 9 ComID Assignments

ComID	Description
0x0000	Reserved
0x0001	Level 0 Device Discovery
0x0002-0x0003	Reserved for TCG
0x0004	TPER_RESET command
0x0005-0x07FF	Reserved for TCG
0x0800-0x0FFF	Vendor Unique
0x1000-0xFFFF	ComID management (Protocol ID=0x01 and 0x02)

TCG Copyright 2020

3.3.4 Synchronous Protocol

The TPer SHALL support the Synchronous Protocol. Refer to [2] for details.

3.3.4.1 Payload Encoding

3.3.4.1.1 Stream Encoding Modifications

The TPer SHALL support tokens listed in Table 10. If an unsupported token is encountered, the TPer SHALL treat this as a streaming protocol violation and return an error per the definition in Section 3.3.4.1.3.

Table 10 Supported Tokens

Acronym	Meaning
	Tiny atom
	Short atom
	Medium atom
	Long atom
SL	Start List
EL	End List
SN	Start Name
EN	End Name
CALL	Call
EOD	End of Data
EOS	End of session
ST	Start transaction
ET	End of transaction
MT	Empty atom

The TPer SHALL support the above token atoms with the B bit set to 0 or 1 and the S bit set to 0.

3.3.4.1.2 TCG Packets

Within a single IF-SEND/IF-RECV command, the TPer SHALL support a ComPacket containing one Packet, which contains one Subpacket. The Host MAY discover TPer support of capabilities beyond this requirement in the parameters returned in response to a `Properties` method.

The TPer MAY ignore Credit Control Subpackets sent by the host. The host MAY discover TPer support of Credit Management with Level 0 Discovery. For more details refer to Section 3.1.1 Level 0 Discovery (M).

The TPer MAY ignore the AckType and Acknowledgement fields in the Packet header on commands from the host and set these fields to zero in its responses to the host. The host MAY discover TPer support of the TCG packet acknowledgement/retry mechanism with Level 0 Discovery. For more details refer to Section 3.1.1 Level 0 Discovery (M).

The TPer MAY ignore packet sequence numbering and not enforce any sequencing behavior. Refer to [2] for details on discovery of packet sequence numbering support.

3.3.4.1.3 Payload Error Response

The TPer SHALL respond according to the following rules if it encounters a streaming protocol violation:

- If the error is on Session Manager or is such that the TPer cannot resolve a valid session ID from the payload (i.e. errors in the ComPacket header or Packet header), then the TPer SHALL discard the payload and immediately transition to the “Awaiting IF-SEND” state.
- If the error occurs after the TPer has resolved the session ID, then the TPer SHALL abort the session and MAY prepare a `CloseSession` method for retrieval by the host.

TCG Copyright 2020

3.3.5 Storage Device Resets

3.3.5.1 Interface Resets

Interface resets that generate TCG reset events are defined in [4].
Interface initiated TCG reset events SHALL result in:

1. All open sessions SHALL be aborted;
2. All uncommitted transactions SHALL be aborted;
3. All pending session startup activities SHALL be aborted;
4. All TCG command and response buffers SHALL be invalidated;
5. All related method processing SHALL be aborted;
6. For each ComID, the state of the synchronous protocol stack SHALL transition to “Awaiting IF-SEND” state;
7. No notification of these events SHALL be sent to the host.

3.3.5.2 TCG Reset Events

Table 11 replaces the definition of TCG reset_types that are defined in [2]:

Table 11 reset_types

Enumeration value	Associated Value
0	Power Cycle
1	Hardware
2	HotPlug
3	Programmatic
4-15	Reserved
16-31	Vendor Unique

3.3.6 Protocol Stack Reset Commands (M)

An IF-SEND containing a Protocol Stack Reset Command SHALL be supported.

Refer to [2] for details.

4 Ruby SSC-compliant Functions and SPs

4.1 Session Manager

4.1.1 Methods

4.1.1.1 Properties (M)

A Ruby compliant SD SHALL support the `Properties` method. The requirements for support of the various TPer and Host properties, and the requirements for their values, are shown in Table 12.

Table 12 Properties Requirements

Property Name	TPer Property Requirements and Values Reported	Host Property Requirements and Values Accepted
MaxComPacketSize	(M) 2048 minimum	(M) Initial Assumption: 2048 Minimum allowed: 2048 Maximum allowed: VU
MaxResponseComPacketSize	(M) 2048 minimum	(N) Although this is a legal host property, there is no requirement for the TPer to use it. The TPer MAY ignore this host property and not list it in the HostProperties result of the `Properties` method response.
MaxPacketSize	(M) 2028 minimum	(M) Initial Assumption: 2028 Minimum allowed: 2028 Maximum allowed: VU
MaxIndTokenSize	(M) 1992 minimum	(M) Initial Assumption: 1992 Minimum allowed: 1992 Maximum allowed: VU
MaxPackets	(M) 1 minimum	(M) Initial Assumption: 1 Minimum allowed: 1 Maximum allowed: VU
MaxSubpackets	(M) 1 minimum	(M) Initial Assumption: 1 Minimum allowed: 1 Maximum allowed: VU
MaxMethods	(M) 1 minimum	(M) Initial Assumption: 1 Minimum allowed: 1 Maximum allowed: VU
MaxSessions	(M) 1 minimum	N/A – not a host property
MaxAuthentications	(M) 2 minimum	N/A – not a host property
MaxTransactionLimit	(M) 1 minimum	N/A – not a host property
DefSessionTimeout	(M) VU	N/A – not a host property

4.1.1.2 StartSession (M)

A Ruby compliant SD SHALL support the following parameters for the `StartSession` method:

- HostSessionID
- SPID
- Write = support for “True" is (M), support for "False" is (N)
- HostChallenge
- HostSigningAuthority

4.1.1.3 SyncSession (M)

A Ruby compliant SD SHALL support the following parameters for the `SyncSession` method:

- HostSessionID
- SPSessionID

4.1.1.4 CloseSession (O)

A Ruby compliant SD MAY support the `CloseSession` method.

4.2 Admin SP

The Admin SP includes the Base Template and the Admin Template.

4.2.1 Base Template Tables

All tables included in the following subsections are mandatory.

4.2.1.1 SPInfo (M)

Table 13 Admin SP - SPInfo Table Preconfiguration

UID	SPID	Name	Size	SizeInUse	SPSessionTimeout	Enabled
00 00 00 02 00 00 00 01	00 00 02 05 00 00 00 01	“Admin”				T

4.2.1.2 SPTemplates (M)

*ST1 = this version number or any version number that complies with this SSC.

Table 14 Admin SP - SPTemplates Table Preconfiguration

UID	TemplateID	Name	Version
00 00 00 03 00 00 00 01	00 00 02 04 00 00 00 01	“Base”	00 00 00 02 *ST1
00 00 00 03 00 00 00 02	00 00 02 04 00 00 00 02	“Admin”	00 00 00 02 *ST1

4.2.1.3 Table (M)

Refer to Section 5.30 for a description and requirements of the MandatoryWriteGranularity and RecommendedAccessGranularity columns.

Table 15 Admin SP - Table Table Preconfiguration

UID	Name	CommonName	TemplateID	Kind	Column	NumColumns	Rows	RowsFree	RowBytes	LastID	MinSize	MaxSize	MandatoryWrite Granularity	RecommendedAccess Granularity
00 00 00 01 00 00 00 01	“Table”			Object									0	0
00 00 00 01 00 00 00 02	“SPInfo”			Object									0	0
00 00 00 01 00 00 00 03	“SPTemplates”			Object									0	0
00 00 00 01 00 00 00 06	"MethodID"			Object									0	0
00 00 00 01 00 00 00 07	"AccessControl"			Object									0	0
00 00 00 01 00 00 00 08	"ACE"			Object									0	0
00 00 00 01 00 00 00 09	"Authority"			Object									0	0
00 00 00 01 00 00 00 0B	"C_PIN"			Object									0	0
00 00 00 01 00 00 02 01	"TPerInfo"			Object									0	0
00 00 00 01 00 00 02 04	"Template"			Object									0	0
00 00 00 01 00 00 02 05	"SP"			Object									0	0

Begin Informative Content

[2] states, “The `Table` table in the Admin SP includes a row for each table that the TPer supports, in addition to a row for each table that exists in the Admin SP.” However, the Ruby SSC requires only the tables from the Admin SP to be included in the Admin SP’s `Table` table, as indicated in Table 15.

End Informative Content

4.2.1.4 MethodID (M)

*MT1 = refer to Section 5.1.2 for details on the requirements for supporting `Revert`.
*MT2 = refer to Section 5.1.1 for details on the requirements for supporting `Activate`.
.

Table 16 Admin SP - MethodID Table Preconfiguration

UID	Name	CommonName	TemplateID
00 00 00 06 00 00 00 08	"Next"		
00 00 00 06 00 00 00 0D	"GetACL"		
00 00 00 06 00 00 00 16	"Get"		
00 00 00 06 00 00 00 17	"Set"		
00 00 00 06 00 00 00 1C	"Authenticate"		
00 00 00 06 00 00 02 02 *MT1	"Revert"		
00 00 00 06 00 00 02 03 *MT2	"Activate"		
00 00 00 06 00 00 06 01	“Random”		

4.2.1.5 AccessControl (M)

The following Table 17contains Optional rows identified by (O)

*AC1 = TT TT TT TT is a shorthand for the LSBs of the Table object UIDs

*AC2 = TT TT TT TT is a shorthand for the LSBs of the SPTemplates object UIDs

*AC3 = TT TT TT TT is a shorthand for the LSBs of the MethodID object UIDs

*AC4 = TT TT TT TT is a shorthand for the LSBs of the ACE object UIDs

*AC5 = TT TT TT TT is a shorthand for the LSBs of the Authority object UIDs

*AC6 = TT TT TT TT is a shorthand for the LSBs of the Template object UIDs

*AC7 = TT TT TT TT is a shorthand for the LSBs of the SP object UIDs

*AC8 = refer to Section 5.1.2 for details on the requirements for supporting `Revert`

*AC9 = refer to Section 5.1.1 for details on the requirements for supporting `Activate`

Notes:

- The `InvokingID`, `MethodID` and `GetACLACL` columns are a special case. Although they are marked as Read-Only with fixed access control, the access control for invocation of the `Get` method is (N).
- The `ACL` column is readable only via the `GetACL` method.

Table 17 Admin SP - AccessControl Table Preconfiguration

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
Table																
		00 00 00 01 00 00 00 00	Table	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 01 TT TT TT TT *AC1	TableObj	Get		ACE_Anybody				ACE_Anybody						
SPInfo																
		00 00 00 02 00 00 00 01	SPInfoObj	Get		ACE_Anybody				ACE_Anybody						
SPTemplates																
		00 00 00 03 00 00 00 00	SPTemplates	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 03 TT TT TT TT *AC2	SPTemplatesObj	Get		ACE_Anybody				ACE_Anybody						
MethodID																
		00 00 00 06 00 00 00 00	MethodID	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 06 TT TT TT TT *AC3	MethodIDObj	Get		ACE_Anybody				ACE_Anybody						

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
ACE																
		00 00 00 08 00 00 00 00	ACE	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 08 TT TT TT TT *AC4	ACEObj	Get		ACE_Anybody				ACE_Anybody						
Authority																
		00 00 00 09 00 00 00 00	Authority	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 09 TT TT TT TT *AC5	AuthorityObj	Get		ACE_Anybody				ACE_Anybody						
(0)		00 00 00 09 00 00 00 03	Makers	Set		ACE_Set_Enabled				ACE_Anybody						
(0)		00 00 00 09 00 00 02 01	Admin1	Set		ACE_Set_Enabled				ACE_Anybody						
(0)		00 00 00 09 00 00 02 00 (+XX)	AdminXX	Set		ACE_Set_Enabled				ACE_Anybody						

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
C_PIN																
		00 00 00 0B 00 00 00 00	C_PIN	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 0B 00 00 00 01	C_PIN_SID	Get		ACE_C_PIN_SID_Get_NOPIN				ACE_Anybody						
		00 00 00 0B 00 00 00 01	C_PIN_SID	Set		ACE_C_PIN_SID_Set_PIN				ACE_Anybody						
		00 00 00 0B 00 00 84 02	C_PIN_MSID	Get		ACE_C_PIN_MSID_Get_PIN				ACE_Anybody						
(0)		00 00 00 0B 00 00 02 01	C_PIN_Admin1	Get		ACE_C_PIN_SID_Get_NOPIN				ACE_Anybody						

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(0)		00 00 00 0B 00 00 02 00 (+XX)	C_PIN_AdminXX	Get		ACE_C_PIN_SID_Get_NOPIN				ACE_Anybody						
(0)		00 00 00 0B 00 00 02 01	C_PIN_Admin1	Set		ACE_C_PIN_Admins_Set_PIN				ACE_Anybody						
(0)		00 00 00 0B 00 00 02 00 (+XX)	C_PIN_AdminXX	Set		ACE_C_PIN_Admins_Set_PIN				ACE_Anybody						
TPerInfo																
		00 00 02 01 00 03 00 01	TPerInfoObj	Get		ACE_Anybody				ACE_Anybody						

TCG Copyright 2020
Published

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 02 01 00 03 00 01	TPerInfoObj	Set		ACE_TPerInfo_Set_ProgrammaticResetEnable				ACE_Anybody						
Template																
		00 00 02 04 00 00 00 00	Template	Next		ACE_Anybody				ACE_Anybody						
		00 00 02 04 TT TT TT TT *AC6	TemplateObj	Get		ACE_Anybody				ACE_Anybody						
SP																
		00 00 00 00 00 00 00 01	ThisSP	Authenticate		ACE_Anybody				ACE_Anybody						
		00 00 00 00 00 00 00 01	ThisSP	Random		ACE_Anybody				ACE_Anybody						
		00 00 02 05 00 00 00 00	SP	Next		ACE_Anybody				ACE_Anybody						

Table association - Informative text	UID	InvokingID	InvokingID Name - informative text	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 02 05 TT TT TT TT *AC7	SPObj	Get		ACE_Anybody				ACE_Anybody						
*AC8		00 00 02 05 TT TT TT TT *AC7	SPObj	Revert		ACE_SP_SID, ACE_Admin				ACE_Anybody						
*AC9		00 00 02 05 TT TT TT TT *AC7	SPObj	Activate		ACE_SP_SID				ACE_Anybody						

4.2.1.6 ACE (M)

The following Table 18 contains Optional rows designated with (O).

*ACE1 = This row is (M) if the TPer supports either `Activate` or `Revert`, and (N) otherwise.

Table 18 Admin SP - ACE Table Preconfiguration

Table Association - Informative text	UID	Name	CommonName	BooleanExpr	Columns
BaseACEs					
	00 00 00 08 00 00 00 01	"ACE_Anybody"		Anybody	All
	00 00 00 08 00 00 00 02	"ACE_Admin"		Admins	All
Authority					
	00 00 00 08 00 03 00 01	"ACE_Set_Enabled"		SID	Enabled
C_PIN					
	00 00 00 08 00 00 8C 02	"ACE_C_PIN_SID_Get_NOPIN"		Admins OR SID	UID, CharSet, TryLimit, Tries, Persistence
	00 00 00 08 00 00 8C 03	"ACE_C_PIN_SID_Set_PIN"		SID	PIN
	00 00 00 08 00 00 8C 04	"ACE_C_PIN_MSID_Get_PIN"		Anybody	UID, PIN

Table Association - Informative text	UID	Name	CommonName	BooleanExpr	Columns
(O)	00 00 00 08 00 03 A0 01	"ACE_C_PIN_Admins_Set_PIN"		Admins OR SID	PIN
TPerInfo					
	00 00 00 08 00 03 00 03	"ACE_TPerInfo_Set_ProgrammaticResetEnable"		SID	ProgrammaticResetEnable
SP					
*ACE1	00 00 00 08 00 03 00 02	"ACE_SP_SID"		SID	All

4.2.1.7 Authority (M)

Notes:

- All Admin SP Admin Authorities are Optional

Table 19 Admin SP - Authority Table Preconfiguration

UID	Name	CommonName	IsClass	Class	Enabled	Secure	HashAndSign	PresentCertificate	Operation	Credential	ResponseSign	ResponseExch	ClockStart	ClockEnd	Limit	Uses	Log	LogTo
00 00 00 09 00 00 00 01	"Anybody"		F	Null	T	None	None	F	None	Null	Null	Null						
00 00 00 09 00 00 00 02	"Admins"		T	Null	T	None	None	F	None	Null	Null	Null						
00 00 00 09 00 00 00 03 (O)	"Makers"		T	Null	T	None	None	F	None	Null	Null	Null						
00 00 00 09 00 00 00 06	"SID"		F	Null	T	None	None	F	Password	C_PIN_SID	Null	Null						
00 00 00 09 00 00 02 01 (O)	"Admin1"		F	Admins	F	None	None	F	Password	C_PIN_Admin1	Null	Null						

UID	Name	CommonName	IsClass	Class	Enabled	Secure	HashAndSign	PresentCertificate	Operation	Credential	ResponseSign	ResponseExch	ClockStart	ClockEnd	Limit	Uses	Log	LogTo
00 00 00 09 00 00 02 00 (+XX) (O)	"AdminXX"		F	Admins	F	None	None	F	Password	C_PIN_AdminXX	Null	Null						

4.2.1.8 C_PIN (M)

Table 20 Admin SP - C_PIN Table Preconfiguration

UID	Name	CommonName	PIN	CharSet	TryLimit	Tries	Persistence
00 00 00 0B 00 00 00 01	"C_PIN_SID"		**VU**	Null	**VU**	**VU**	FALSE
00 00 00 0B 00 00 84 02	"C_PIN_MSID"		**MSID**				
00 00 00 0B 00 00 02 01 (O)	"C_PIN_Admin1"		**“”**	Null	**0**	**0**	FALSE
00 00 00 0B 00 00 02 00 (+XX) (O)	"C_PIN_AdminXX"		**“”**	Null	**0**	**0**	FALSE

For devices that will be used in environments where an automated take ownership process is required, the initial PIN column value of C_PIN_SID SHALL be set to the PIN column value of C_PIN_MSID. In order to allow for alternative take ownership processes, the initial PIN column value of C_PIN_SID MAY be Vendor Unique (VU).

Begin Informative Content

Several activation / take ownership models are possible. The simplest model is a process whereby the host discovers the initial C_PIN_SID PIN value by performing a `Get` operation on the C_PIN_MSID object. This model requires that the initial C_PIN_SID PIN be the value of the C_PIN_MSID PIN.

Before a device vendor chooses to implement an activation / take ownership model based on a vendor unique SID PIN, the device vendor must undertake due diligence to ensure that the ecosystem exists to support such an activation / take ownership model. Having a C_PIN_SID PIN value that is different from the C_PIN_MSID PIN value may have serious ramifications, such as the inability to take ownership of the device.

See Section 5.1.2.2.1 for an explanation of how `Revert` affects the value of the C_PIN_SID PIN column.

End Informative Content

4.2.2 Base Template Methods

Refer to Section 4.2.1.4 for supported methods.

4.2.3 Admin Template Tables

4.2.3.1 TPerInfo (M)

The TPerInfo table has the following columns, in addition to those defined in [2]:

Table 21 Admin SP – TPerInfo Columns

Column Number	Column Name	IsUnique	Column Type
0x08	ProgrammaticResetEnable		boolean

- **ProgrammaticResetEnable**
  This column indicates whether support for programmatic resets is enabled or not. If ProgrammaticResetEnable is TRUE, then the TPER_RESET command is enabled. If ProgrammaticResetEnable is FALSE, then the TPER_RESET command is not enabled. This column is readable by Anybody and modifiable by the SID authority.

*TP1 = the value in the GUDID column SHALL comply with the format defined in [2].

*TP2 = this version or any version that supports the defined features in this SSC.

*TP3 = the SSC column is a list of names and SHALL have “Ruby” as one of the list elements.

Table 22 Admin SP - TPerInfo Table Preconfiguration

UID	Bytes	GUDID	Generation	Firmware Version	ProtocolVersion	SpaceForIssuance	SSC	ProgrammaticResetEnable
00 00 02 01 00 03 00 01		**VU** *TP1			1 *TP2		[“Ruby”] *TP3	FALSE

4.2.3.2 Template (M)

Table 23 Admin SP - Template Table Preconfiguration

UID	Name	Revision Number	Instances	MaxInstances
00 00 02 04 00 00 00 01	"Base"	1	**VU**	**VU**
00 00 02 04 00 00 00 02	"Admin"	1	1	1
00 00 02 04 00 00 00 06	"Locking"	1	1	1

4.2.3.3 SP (M)

*SP1 = this row only exists in the Admin SP's OFS when the Locking SP is created by the manufacturer.

Table 24 Admin SP - SP Table Preconfiguration

UID	Name	ORG	EffectiveAuth	DateOfIssue	Bytes	LifeCycle	Frozen
00 00 02 05 00 00 00 01	"Admin"					Manufactured	FALSE
00 00 02 05 00 00 00 02 *SP1	"Locking"					Manufactured-Inactive	FALSE

4.2.4 Admin Template Methods

Refer to Section 4.2.1.4 for supported methods.

4.2.5 Crypto Template Tables

A Ruby SSC compliant SD is not required to support any Crypto template tables.

4.2.6 Crypto Template Methods

Refer to Section 4.2.1.4 for supported methods.

4.2.6.1 Random

The TPer SHALL implement the `Random` method with the constraints stated in this subsection. TPer support of the following parameters is mandatory:

- `Count`

Attempts to use unsupported parameters SHALL result in a method failure response with TCG status INVALID_PARAMETER. The TPer SHALL support `Count` parameter values less than or equal to 32.

4.3 Locking SP

4.3.1 Base Template Tables

All tables defined with (M) in section titles are mandatory.

4.3.1.1 SPInfo (M)

Table 25 Locking SP - SPInfo Table Preconfiguration

UID	SPID	Name	Size	SizeInUse	SPSessionTimeout	Enabled
00 00 00 02 00 00 00 01	00 00 02 05 00 00 00 02	"Locking"				T

4.3.1.2 SPTemplates (M)

*SP1 = this version number or any number that supports the defined features in this SSC

Table 26 Locking SP - SPTemplates Table Preconfiguration

UID	TemplateID	Name	Version
00 00 00 03 00 00 00 01	00 00 02 04 00 00 00 01	"Base"	00 00 00 02 *SP1
00 00 00 03 00 00 00 02	00 00 02 04 00 00 00 06	"Locking"	00 00 00 02 *SP1

4.3.1.3 Table (M)

The following Table 27 contains Optional rows designated with (O).

TT1 = only one of the two K_AES table is required

Refer to Section 0 for a description and requirements of the MandatoryWriteGranularity and RecommendedAccessGranularity columns.

Table 27 Locking SP - Table Table Preconfiguration

UID	Name	CommonName	TemplateID	Kind	Column	NumColumns	Rows	RowsFree	RowBytes	LastID	MinSize	MaxSize	MandatoryWrite Granularity	RecommendedAccess Granularity
00 00 00 01 00 00 00 01	"Table"			Object									0	0
00 00 00 01 00 00 00 02	"SPInfo"			Object									0	0
00 00 00 01 00 00 00 03	"SPTemplates"			Object									0	0
00 00 00 01 00 00 00 06	"MethodID"			Object									0	0
00 00 00 01 00 00 00 07	"AccessControl"			Object									0	0
00 00 00 01 00 00 00 08	"ACE"			Object									0	0
00 00 00 01 00 00 00 09	"Authority"			Object									0	0
00 00 00 01 00 00 00 0B	"C_PIN"			Object									0	0
00 00 00 01 00 00 00 1D	"SecretProtect"			Object									0	0
00 00 00 01 00 00 08 01	"LockingInfo"			Object									0	0
00 00 00 01 00 00 08 02	"Locking"			Object									0	0

TCG Copyright 2020
Published

UID	Name	CommonName	TemplateID	Kind	Column	NumColumns	Rows	RowsFree	RowBytes	LastID	MinSize	MaxSize	MandatoryWrite Granularity	RecommendedAccess Granularity
00 00 00 01 00 00 08 03 (MBR O)	"MBRControl"			Object									0	0
00 00 00 01 00 00 08 04 (MBR O)	"MBR"			Byte			**0x08000000 min**						**VU**	**VU**
00 00 00 01 00 00 08 05 *TT1	"K_AES_128"			Object									0	0
00 00 00 01 00 00 08 06 *TT1	"K_AES_256"			Object									0	0
00 00 00 01 00 00 10 01	"DataStore"			Byte			**0x00020000 min**						**VU**	**VU**

4.3.1.4 Type (N)

The `Type` table is (N) for Ruby. The following types as defined by [2] SHALL meet the following requirements:

- The "boolean_ACE" type (00000005 000040E) SHALL include the OR Boolean operator.
- The "AC_element" type (00000005 00000801) SHALL support at least 5 entries (2 User authorities, 1 Admin authorities, and 2 Boolean operators).

4.3.1.5 MethodID (M)

**MT1 = refer to Section 5.1.3 for details on the requirements for supporting RevertSP.*

Table 28 Locking SP - MethodID Table Preconfiguration

UID	Name	CommonName	TemplateID
00 00 00 06 00 00 00 08	"Next"		
00 00 00 06 00 00 00 0D	"GetACL"		
00 00 00 06 00 00 00 10	"GenKey"		
00 00 00 06 00 00 00 11 *MT1	"RevertSP"		
00 00 00 06 00 00 00 16	"Get"		
00 00 00 06 00 00 00 17	"Set"		
00 00 00 06 00 00 00 1C	"Authenticate"		
00 00 00 06 00 00 06 01	“Random”		

4.3.1.6 AccessControl (M)

The following Table 29 contains Optional rows designated with (O).

*AC1 = refer to Section 5.1.3 for details on the requirements for supporting RevertSP

*AC2 = TT TT TT TT is a shorthand for the LSBs of the Table object UIDs

*AC3 = TT TT TT TT is a shorthand for the LSBs of the SPTemplates object UIDs

*AC4 = TT TT TT TT is a shorthand for the LSBs of the MethodID object UIDs

*AC5 = TT TT TT TT is a shorthand for the LSBs of the ACE object UIDs

*AC6 = only K_AES_128 or K_AES_256 related rows mandatory

*AC7 = TT TT TT TT is a shorthand for the LSB of the Authority object UIDs

*AC8 = TT TT TT TT is a shorthand for the LSBs of the SecretProtect object UIDs

Notes:

- The `InvokingID`, `MethodID` and `GetACLACL` columns are a special case. Although they are marked as Read-Only with fixed access control, the access control for invocation of the `Get` method is (N).
- The `ACL` column is readable only via the `GetACL` method.

Table 29 Locking SP - AccessControl Table Preconfiguration

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
SP																
		00 00 00 00 00 00 00 01	ThisSP	Authenticate		ACE_Anybody				ACE_Anybody						
		00 00 00 00 00 00 00 01	ThisSP	Random		ACE_Anybody				ACE_Anybody						
*AC1		00 00 00 00 00 00 00 01	ThisSP	RevertSP		ACE_Admin				ACE_Anybody						
Table																
		00 00 00 01 00 00 00 00	Table	Next		ACE_Anybody				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 01 TT TT TT TT *AC2	TableObj	Get		ACE_Anybody				ACE_Anybody						
SPInfo																
		00 00 00 02 00 00 00 01	SPInfoObj	Get		ACE_Anybody				ACE_Anybody						
SPTemplates																
		00 00 00 03 00 00 00 00	SPTemplates	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 03 TT TT TT TT *AC3	SPTemplatesObj	Get		ACE_Anybody				ACE_Anybody						
MethodID																
		00 00 00 06 00 00 00 00	MethodID	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 06 TT TT TT TT *AC4	MethodIDObj	Get		ACE_Anybody				ACE_Anybody						
ACE																
		00 00 00 08 00 00 00 00	ACE	Next		ACE_Anybody				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 08 TT TT TT TT *AC5	ACEObj	Get		ACE_ACE_Get_All				ACE_Anybody						
		00 00 00 08 00 03 80 00	ACE_ACE_Get_All	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 90 00	ACE_Authority_Get_All	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 A8 01	ACE_C_PIN_User1_Set_PIN	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 08 00 03 A8 02	ACE_C_PIN_User2_Set_PIN	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
(0)		00 00 00 08 00 03 A8 00 (+MM MM)	ACE_C_PIN_UserMMMM_Set_PIN	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 04 40 01	ACE_User1_Set_CommonName	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 04 40 02	ACE_User2_Set_CommonName	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(O)		00 00 00 08 00 04 40 00 (+MM MM)	ACE_UserMMMM_Set_CommonName	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
*AC6		00 00 00 08 00 03 B0 00	ACE_K_AES_128_GlobalRange_GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
*AC6 (ALR O)		00 00 00 08 00 03 B0 01	ACE_K_AES_128_Range1_GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 00 08 00 03 B0 00 (+NN NN)	ACE_K_AES_128_RangeNNNN _GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
*AC6 (ALR O)		00 00 00 08 00 03 B8 00	ACE_K_AES_256_GlobalRange_GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
*AC6 (ALR O)		00 00 00 08 00 03 B8 01	ACE_K_AES_256_Range1_GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 00 08 00 03 B8 00 (+NN NN)	ACE_K_AES_256_RangeNNNN _GenKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 D0 00	ACE_Locking_GlobalRange_Get_RangeStartToActiveKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
(ALR O)		00 00 00 08 00 03 D0 01	ACE_Locking_Range1_Get_RangeStartToActiveKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(ALR O)		00 00 00 08 00 03 D0 00 (+NN NN)	ACE_Locking_RangeNNNN_Get_RangeStartToActiveKey	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 E0 00	ACE_Locking_GlobalRange_Set_RdLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
(ALR O)		00 00 00 08 00 03 E0 01	ACE_Locking_Range1_Set_RdLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(ALR O)		00 00 00 08 00 03 E0 00 (+NN NN)	ACE_Locking_RangeNNNN_Set_RdLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 E8 00	ACE_Locking_GlobalRange_Set_WrLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
(ALR O)		00 00 00 08 00 03 E8 01	ACE_Locking_Range1_Set_WrLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(ALR O)		00 00 00 08 00 03 E8 00 (+NN NN)	ACE_Locking_RangeNNNN_Set_WrLocked	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
(MBR O)		00 00 00 08 00 03 F8 01	ACE_MBRControl_Set_DoneToDOR	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		00 00 00 08 00 03 FC 00	ACE_DataStore_Get_All	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 08 00 03 FC 01	ACE_DataStore_Set_All	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
Authority																
		00 00 00 09 00 00 00 00	Authority	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 09 TT TT TT TT *AC7	AuthorityObj	Get		ACE_Authority_Get_All, ACE_Anybody_Get_CommonName				ACE_Anybody						
		00 00 00 09 00 01 00 01	Admin1	Set		ACE_Admins_Set_CommonName				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(0)		00 00 00 09 00 01 00 00 (+XX XX)	AdminXXXX	Set		ACE_Authority_Set_Enabled, ACE_Admins_Set_CommonName				ACE_Anybody						
		00 00 00 09 00 03 00 01	User1	Set		ACE_Authority_Set_Enabled, ACE_User1_Set_CommonName				ACE_Anybody						
		00 00 00 09 00 03 00 02	User2	Set		ACE_Authority_Set_Enabled, ACE_User2_Set_CommonName				ACE_Anybody						
(0)		00 00 00 09 00 03 00 00 (+MM MM)	UserMMMM	Set		ACE_Authority_Set_Enabled, ACE_UserMMMM_Set_CommonName				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
C_PIN																
		00 00 00 0B 00 00 00 00	C_PIN	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 0B 00 01 00 01	C_PIN_Admin1	Get		ACE_C_PIN_Admins_Get_All_NOPIN				ACE_Anybody						
(0)		00 00 00 0B 00 01 00 00 (+ XX XX)	C_PIN_AdminXXXX	Get		ACE_C_PIN_Admins_Get_All_NOPIN				ACE_Anybody						
		00 00 00 0B 00 03 00 01	C_PIN_User1	Get		ACE_C_PIN_Admins_Get_All_NOPIN				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 0B 00 03 00 02	C_PIN_User2	Get		ACE_C_PIN_Admins_Get_All_NOPIN				ACE_Anybody						
(0)		00 00 00 0B 00 03 00 00 (+MM MM)	C_PIN_UserMMMM	Get		ACE_C_PIN_Admins_Get_All_NOPIN				ACE_Anybody						
		00 00 00 0B 00 01 00 01	C_PIN_Admin1	Set		ACE_C_PIN_Admins_Set_PIN				ACE_Anybody						
(0)		00 00 00 0B 00 01 00 00 (+XX XX)	C_PIN_AdminXXXX	Set		ACE_C_PIN_Admins_Set_PIN				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 00 0B 00 03 00 01	C_PIN_User1	Set		ACE_C_PIN_User1_Set_PIN				ACE_Anybody						
		00 00 00 0B 00 03 00 02	C_PIN_User2	Set		ACE_C_PIN_User2_Set_PIN				ACE_Anybody						
(0)		00 00 00 0B 00 03 00 00 (+MM MM)	C_PIN_UserMMMM	Set		ACE_C_PIN_UserMMMM_Set_PIN				ACE_Anybody						

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
SecretProtect																
		00 00 00 1D 00 00 00 00	SecretProtect	Next		ACE_Anybody				ACE_Anybody						
		00 00 00 1D TT TT TT TT *AC8	SecretProtectObj	Get		ACE_Anybody				ACE_Anybody						
LockingInfo																
		00 00 08 01 00 00 00 01	LockingInfoObj	Get		ACE_Anybody				ACE_Anybody						
Locking																
		00 00 08 02 00 00 00 00	Locking	Next		ACE_Anybody				ACE_Anybody						
		00 00 08 02 00 00 00 01	Locking_GlobalRange	Get		ACE_Locking_GlobalRange_Get_RangeStartToActiveKey, ACE_Anybody_Get_CommonName				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(ALR O)		00 00 08 02 00 03 00 01	Locking_Range1	Get		ACE_Locking_Range1_Get_RangeStartToActiveKey, ACE_Anybody_Get_CommonName				ACE_Anybody						
(ALR O)		00 00 08 02 00 03 00 00 (+NN NN)	Locking_RangeNNNN	Get		ACE_Locking_RangeNNNN_Get_RangeStartToActiveKey, ACE_Anybody_Get_CommonName				ACE_Anybody						
		00 00 08 02 00 00 00 01	Locking_GlobalRange	Set		ACE_Locking_GlblRng_Admins_Set, ACE_Locking_GlobalRange_Set_RdLocked, ACE_Locking_GlobalRange_Set_WrLocked, ACE_Admins_Set_CommonName				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(ALR O)		00 00 08 02 00 03 00 01	Locking_Range1	Set		ACE_Locking_Admins_RangeStartToLOR, ACE_Locking_Range1_Set_RdLocked, ACE_Locking_Range1_Set_WrLocked, ACE_Admins_Set_CommonName				ACE_Anybody						
(ALR O)		00 00 08 02 00 03 00 00 (+NN NN)	Locking_RangeNNNN	Set		ACE_Locking_Admins_RangeStartToLOR, ACE_Locking_RangeNNNN_Set_RdLocked, ACE_Locking_RangeNNNN_Set_WrLocked, ACE_Admins_Set_CommonName				ACE_Anybody						
MBRControl																
(MBR O)		00 00 08 03 00 00 00 01	MBRControlObj	Get		ACE_Anybody				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
(MBR O)		00 00 08 03 00 00 00 01	MBRControlObj	Set		ACE_MBRControl_Admins_Set, ACE_MBRControl_Set_DoneToDOR				ACE_Anybody						
MBR																
(MBR O)		00 00 08 04 00 00 00 00	MBR	Get		ACE_Anybody				ACE_Anybody						
(MBR O)		00 00 08 04 00 00 00 00	MBR	Set		ACE_Admin				ACE_Anybody						
K_AES_128																
*AC6		00 00 08 05 00 00 00 01	K_AES_128_GlobalRange_Key	Get		ACE_K_AES_Mode				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 08 05 00 03 00 01	K_AES_128_Range1_Key	Get		ACE_K_AES_Mode				ACE_Anybody						
*AC6 (ALR O)		00 00 08 05 00 03 00 00 (+NN NN)	K_AES_128_RangeNNNN_Key	Get		ACE_K_AES_Mode				ACE_Anybody						
*AC6		00 00 08 05 00 00 00 01	K_AES_128_GlobalRange_Key	GenKey		ACE_K_AES_128_GlobalRange_GenKey				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 08 05 00 03 00 01	K_AES_128_Range1_Key	GenKey		ACE_K_AES_128_Range1_GenKey				ACE_Anybody						
*AC6 (ALR O)		00 00 08 05 00 03 00 00 (+NN NN)	K_AES_128_RangeNNNN_Key	GenKey		ACE_K_AES_128_RangeNNNN_GenKey				ACE_Anybody						
K_AES_256																
*AC6		00 00 08 06 00 00 00 01	K_AES_256_GlobalRange_Key	Get		ACE_K_AES_Mode				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 08 06 00 03 00 01	K_AES_256_Range1_Key	Get		ACE_K_AES_Mode				ACE_Anybody						
*AC6 (ALR O)		00 00 08 06 00 03 00 00 (+NN NN)	K_AES_256_RangeNNNN_Key	Get		ACE_K_AES_Mode				ACE_Anybody						
*AC6		00 00 08 06 00 00 00 01	K_AES_256_GlobalRange_Key	GenKey		ACE_K_AES_256_GlobalRange_GenKey				ACE_Anybody						

TCG Copyright 2020
Published

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
*AC6 (ALR O)		00 00 08 06 00 03 00 01	K_AES_256_Range1_Key	GenKey		ACE_K_AES_256_Range1_GenKey				ACE_Anybody						
*AC6 (ALR O)		00 00 08 06 00 03 00 00 (+NN NN)	K_AES_256_RangeNNNN_Key	GenKey		ACE_K_AES_256_RangeNNNN_GenKey				ACE_Anybody						
DataStore																
		00 00 10 01 00 00 00 00	DataStore	Get		ACE_DataStore_Get_All				ACE_Anybody						

Table Association - informative only	UID	InvokingID	InvokingID Name - informative only	MethodID	CommonName	ACL	Log	AddACEACL	RemoveACEACL	GetACLACL	DeleteMethodACL	AddACELog	RemoveACELog	GetACLLog	DeleteMethodLog	LogTo
		00 00 10 01 00 00 00 00	DataStore	Set		ACE_DataStore_Set_All				ACE_Anybody						

4.3.1.7 ACE (M)

The following Table 30 contains Optional rows designated with (O).

*ACE1 = The TPer SHALL support the values of “Admins” and “Admins OR UserMMMM” in the BooleanExpr column of each ACE_C_PIN_UserMMMM_Set_PIN ACE. The TPer SHALL fail the Set method invocation with status INVALID_PARAMETER if the host attempts to set a value not supported by the TPer.

*ACE2 = Only K_AES_128 or K_AES_256 related rows mandatory.

(MBR O) = Refer to 2.10 for requirements for supporting rows marked by this identifier.

(ALR O) = Refer to 2.11 for requirements for supporting rows marked by this identifier.

Table 30 Locking SP - ACE Table Preconfiguration

Table Association -Informative Column	UID	Name	CommonName	BooleanExpr	Columns
Base ACEs					
	00 00 00 08 00 00 00 01	"ACE_Anybody"		Anybody	All
	00 00 00 08 00 00 00 02	"ACE_Admin"		Admins	All
	00 00 00 08 00 00 00 03	"ACE_Anybody_Get_CommonName"		Anybody	UID, CommonName
	00 00 00 08 00 00 00 04	"ACE_Admins_Set_CommonName"		Admins	CommonName
ACE					
	00 00 00 08 00 03 80 00	"ACE_ACE_Get_All"		Admins	All
	00 00 00 08 00 03 80 01	"ACE_ACE_Set_BooleanExpression"		Admins	BooleanExpr
Authority					

Table Association -Informative Column	UID	Name	CommonName	BooleanExpr	Columns
	00 00 00 08 00 03 90 00	"ACE_Authority_Get_All"		Admins	All
	00 00 00 08 00 03 90 01	"ACE_Authority_Set_Enabled"		Admins	Enabled
	00 00 00 08 00 04 40 01	"ACE_User1_Set_CommonName"		Admins	CommonName
	00 00 00 08 00 04 40 02	"ACE_User2_Set_CommonName"		Admins	CommonName
(O)	00 00 00 08 00 04 40 00 (+MM MM)	"ACE_UserMMMM_Set_CommonName"		Admins	CommonName
C_PIN					
	00 00 00 08 00 03 A0 00	"ACE_C_PIN_Admins_Get_All_NOPIN"		Admins	UID, CharSet, TryLimit, Tries, Persistence
	00 00 00 08 00 03 A0 01	"ACE_C_PIN_Admins_Set_PIN"		Admins	PIN
	00 00 00 08 00 03 A8 01	"ACE_C_PIN_User1_Set_PIN"		Admins OR User1 *ACE1	PIN
	00 00 00 08 00 03 A8 02	"ACE_C_PIN_User2_Set_PIN"		Admins OR User2 *ACE1	PIN
(O)	00 00 00 08 00 03 A8 00 (+MM MM)	"ACE_C_PIN_UserMMMM_Set_PIN"		Admins OR UserMMMM *ACE1	PIN
K_AES					
	00 00 00 08 00 03 BF FF	"ACE_K_AES_Mode"		Anybody	Mode
K_AES_128					
*ACE2	00 00 00 08 00 03 B0 00	"ACE_K_AES_128_GlobalRange_ GenKey"		Admins	All
* ACE2 (ALR O)	00 00 00 08 00 03 B0 01	"ACE_K_AES_128_Range1_ GenKey"		Admins	All
* ACE2 (ALR O)	00 00 00 08 00 03 B0 00 (+NN NN)	"ACE_K_AES_128_RangeNNNN_ GenKey"		Admins	All
K_AES_256					
* ACE2	00 00 00 08 00 03 B8 00	"ACE_K_AES_256_GlobalRange_ GenKey"		Admins	All
* ACE2 (ALR O)	00 00 00 08 00 03 B8 01	"ACE_K_AES_256_Range1_ GenKey"		Admins	All
* ACE2 (ALR O)	00 00 00 08 00 03 B8 00 (+NN NN)	"ACE_K_AES_256_RangeNNNN_ GenKey"		Admins	All
Locking					

Table Association -Informative Column	UID	Name	CommonName	BooleanExpr	Columns
	00 00 00 08 00 03 D0 00	"ACE_Locking_GlobalRange_Get_ RangeStartToActiveKey"		Admins	RangeStart, RangeLength, ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset, ActiveKey
(ALR O)	00 00 00 08 00 03 D0 01	"ACE_Locking_Range1_Get_ RangeStartToActiveKey"		Admins	RangeStart, RangeLength, ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset, ActiveKey
(ALR O)	00 00 00 08 00 03 D0 00 (+NN NN)	"ACE_Locking_RangeNNNN_Get_ RangeStartToActiveKey"		Admins	RangeStart, RangeLength, ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset, ActiveKey
	00 00 00 08 00 03 E0 00	"ACE_Locking_GlobalRange_Set_RdLocked"		Admins	ReadLocked
(ALR O)	00 00 00 08 00 03 E0 01	"ACE_Locking_Range1_Set_RdLocked"		Admins	ReadLocked
(ALR O)	00 00 00 08 00 03 E0 00 (+NN NN)	"ACE_Locking_RangeNNNN_Set_RdLocked"		Admins	ReadLocked
	00 00 00 08 00 03 E8 00	"ACE_Locking_GlobalRange_Set_WrLocked"		Admins	WriteLocked
(ALR O)	00 00 00 08 00 03 E8 01	"ACE_Locking_Range1_Set_WrLocked"		Admins	WriteLocked
(ALR O)	00 00 00 08 00 03 E8 00 (+NN NN)	"ACE_Locking_RangeNNNN_Set_WrLocked"		Admins	WriteLocked
	00 00 00 08 00 03 F0 00	"ACE_Locking_GlblRng_Admins_Set"		Admins	ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset

Table Association -Informative Column	UID	Name	CommonName	BooleanExpr	Columns
(ALR O)	00 00 00 08 00 03 F0 01	"ACE_Locking_Admins_RangeStartToLOR"		Admins	RangeStart, RangeLength, ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset
MBRControl					
(MBR O)	00 00 00 08 00 03 F8 00	"ACE_MBRControl_Admins_Set"		Admins	Enable, Done, DoneOnReset
(MBR O)	00 00 00 08 00 03 F8 01	"ACE_MBRControl_Set_DoneToDOR"		Admins	Done, DoneOnReset
DataStore					
	00 00 00 08 00 03 FC 00	"ACE_DataStore_Get_All"		Admins	All
	00 00 00 08 00 03 FC 01	"ACE_DataStore_Set_All"		Admins	All

4.3.1.8 Authority (M)

The following Table 31 contains Optional rows designated with (O).
Notes:

1. Admin1 is required.
   Any additional Admin authorities are (O).

2. User1 and User2 are required.
   Any additional User authorities are (O).

Table 31 Locking SP - Authority Table Preconfiguration

UID	Name	CommonName	IsClass	Class	Enabled	Secure	HashAndSign	PresentCertificate	Operation	Credential	ResponseSign	ResponseExch	ClockStart	ClockEnd	Limit	Uses	Log	LogTo
00 00 00 09 00 00 00 01	"Anybody"	""	F	Null	T	None	None	F	None	Null	Null	Null						
00 00 00 09 00 00 00 02	"Admins"	""	T	Null	T	None	None	F	None	Null	Null	Null						

UID	Name	CommonName	IsClass	Class	Enabled	Secure	HashAndSign	PresentCertificate	Operation	Credential	ResponseSign	ResponseExch	ClockStart	ClockEnd	Limit	Uses	Log	LogTo
00 00 00 09 00 01 00 01	"Admin1"	""	F	Admins	T	None	None	F	Password	C_PIN_Admin1	Null	Null						
00 00 00 09 00 01 00 00 (+XX XX) (O)	"AdminXXXX"	""	F	Admins	F	None	None	F	Password	C_PIN_Admin XXX	Null	Null						
00 00 00 09 00 03 00 00	"Users"	""	T	Null	T	None	None	F	None	Null	Null	Null						
00 00 00 09 00 03 00 01	"User1"	""	F	Users	F	None	None	F	Password	C_PIN_User1	Null	Null						
00 00 00 09 00 03 00 02	" User2"	""	F	Users	F	None	None	F	Password	C_PIN_User2	Null	Null						
00 00 00 09 00 03 00 00 (+MM MM) (O)	"UserMMMM"	""	F	Users	F	None	None	F	Password	C_PIN_UserMMMM	Null	Null						

4.3.1.9 C_PIN (M)

The following Table 32 includes Optional rows designated with (O)

Notes:

1. If the Locking SP's original life cycle state is “Manufactured-Inactive”, see Section 5.1.1.2 for the initial value of C_PIN_Admin1.PIN. If the Locking SP's original life cycle state is Manufactured, then the initial value of C_PIN_Admin1.PIN is the same as the Admin SP's C_PIN_MSID.PIN value.

Table 32 Locking SP - C_PIN Table Preconfiguration

UID	Name	CommonName	PIN	CharSet	TryLimit	Tries	Persistence
00 00 00 0B 00 01 00 01	"C_PIN_Admin1"		SID or MSID[1]	Null	**0**	**0**	FALSE

UID	Name	CommonName	PIN	CharSet	TryLimit	Tries	Persistence
00 00 00 0B 00 01 00 00 (+XX XX) (O)	"C_PIN_AdminXXXX"		""	Null	**0**	**0**	FALSE
00 00 00 0B 00 03 00 01	"C_PIN_User1"		""	Null	**0**	**0**	FALSE
00 00 00 0B 00 03 00 02	"C_PIN_User2"		""	Null	**0**	**0**	FALSE
00 00 00 0B 00 03 00 00 (+MM MM) (O)	"C_PIN_UserMMMM"		""	Null	**0**	**0**	FALSE

4.3.1.10 SecretProtect (M)

At least one of the objects shown in Table 33 SHALL be supported

Table 33 Locking SP - SecretProtect Table Preconfiguration

UID	Table	ColumnNumber	ProtectMechanisms
00 00 00 1D 00 00 00 1D	00 00 00 01 00 00 08 05 (K_AES_128)	0x03	**VU**
00 00 00 1D 00 00 00 1E	00 00 00 01 00 00 08 06 (K_AES_256)	0x03	**VU**

Note: The "VU" entries in Table 33 indicate that there is no requirement set by this specification as to the value reported. It is NOT a requirement to report the "Vendor Unique" `protection_types` value.

4.3.2 Base Template Methods

Refer to Section 4.3.1.5 for supported methods.

4.3.3 Crypto Template Tables

A Ruby SSC compliant SD is not required to support any Crypto template tables.

4.3.4 Crypto Template Methods

Refer to Section 4.3.1.5 for supported methods.

4.3.4.1 Random

Refer to Section 4.2.6.1 for additional constraints imposed on the `Random` method.

4.3.5 Locking Template Tables

4.3.5.1 LockingInfo (M)

The LockingInfo table has the following columns, in addition to those defined in [2]:

TCG Copyright 2020
Published

Table 34 Locking SP – LockingInfo Columns

Column Number	Column Name	IsUnique	Column Type
0x07	AlignmentRequired		boolean
0x08	LogicalBlockSize		uinteger_4
0x09	AlignmentGranularity		uinteger_8
0x0A	LowestAlignedLBA		uniteger_8

- **AlignmentRequired**
  This column indicates whether the TPer requires ranges in the Locking table to be aligned. If AlignmentRequired is TRUE, then the TPer requires ranges to be aligned. If AlignmentRequired is FALSE, then the TPer does not require ranges to be aligned.
  This column SHALL NOT be modifiable by the host and may be retrieved by Anybody.
- **LogicalBlockSize**
  This column indicates the number of bytes in a logical block.
  This column SHALL NOT be modifiable by the host and may be retrieved by Anybody.
- **AlignmentGranularity**
  This column indicates the number of logical blocks in a group, for alignment purposes (see 5.4).
  This column SHALL NOT be modifiable by the host and may be retrieved by Anybody.
- **LowestAlignedLBA**
  This column indicates the lowest logical block address that is located at the beginning of an alignment granularity group (see 5.4).
  This column SHALL NOT be modifiable by the host and may be retrieved by Anybody.

Table 35 Locking SP - LockingInfo Table Preconfiguration

UID	Name	Version	EncryptSupport	MaxRanges	MaxReEncryptions	KeysAvailableCfg	AlignmentRequired	LogicalBlockSize	AlignmentGranularity	LowestAlignedLBA
00 00 08 01 00 00 00 01			Media Encryption	**0**[1]						

Note:
1. The MaxRanges column specifies the number of additional locking ranges supported. Refer to Section 2.11 for requirements needed to support additional locking ranges.

4.3.5.2 Locking (M)

The following Table 36 contains Optional rows designated with (O).
*LT1 = The ActiveKey can be a K_AES_128 object reference (UID) or a K_AES_256 object reference (UID)
*LT2 = Only a limited set of LockOnReset values is required to be supported by Ruby SSC SDs. Refer to Section 4.3.5.2.2 for details.

Table 36 Locking SP - Locking Table Preconfiguration

UID	Name	CommonName	RangeStart	RangeLength	ReadLockEnabled	WriteLockEnabled	ReadLocked	WriteLocked	LockOnReset	ActiveKey	NextKey	ReEncryptState	ReEncyptRequest	AdvKeyMode	VerifyMode	ContOnReset	LastReEncryptLBA	LastReEncState	GeneralStatus
00 00 08 02 00 00 00 01	"Locking_GlobalRange"	""	0	0	F	F	**F**	**F**	Power Cycle *LT2	K_AES_128[256]_GlobalRange_Key *LT1									
00 00 08 02 00 03 00 01 (ALR O)	"Locking_Range1"	""	0	0	F	F	**F**	**F**	Power Cycle *LT2	K_AES_128[256]_Range1_Key *LT1									
00 00 08 02 00 03 00 00 (+NN NN) (ALR O)	"Locking_RangeNNNN"	""	0	0	F	F	**F**	**F**	Power Cycle *LT2	K_AES_128[256]_RangeNNNN_Key *LT1									

4.3.5.2.1 Geometry Reporting Feature Behavior

The following RangeStart and RangeLength behaviors SHALL be implemented.

4.3.5.2.1.1 RangeStart Behavior

This column value defines the starting LBA value for this range. In non-Global Range rows, this column MAY be modifiable based on access control settings. Changes to this column are subject to the same constraints and checks defined for this column when rows of the Locking table are created (see [2]).

When processing a Set method or CreateRow method on the Locking table for a non-Global Range row, if:

a) the AlignmentRequired column in the LockingInfo table is TRUE;
b) RangeStart is non-zero; and
c) StartAlignment (see Figure 1) is non-zero,

then the method SHALL fail and return an error status code INVALID_PARAMETER.

Figure 1 - StartAlignment

StartAlignment = (RangeStart modulo AlignmentGranularity) - LowestAlignedLBA

where: LowestAlignedLBA and AlignmentGranularity are columns in the LockingInfo table (see 4.3.5.1)

4.3.5.2.1.2 RangeLength Behavior

This column value defines the quantity of contiguous LBAs for this LBA range (starting with the value defined in the RangeStart column). In non-Global Range rows, this column MAY be modifiable based on access control settings. Changes to this column are subject to the same constraints and checks defined for this column when rows of the Locking table are created (see [2]).

When processing a Set method or CreateRow method on the Locking table for a non-Global Range row, if:

a) the AlignmentRequired column in the LockingInfo table is TRUE;
b) RangeLength is non-zero; and
c) LengthAlignment (see Figure 2) is non-zero,

then the method SHALL fail and return an error status code INVALID_PARAMETER.

Figure 2 - LengthAlignment

If RangeStart is zero, then
LengthAlignment = (RangeLength modulo AlignmentGranularity) - LowestAlignedLBA

If RangeStart is non-zero, then
LengthAlignment = (RangeLength modulo AlignmentGranularity)

where:
LowestAlignedLBA and AlignmentGranularity are columns in the LockingInfo table (see 4.3.5.1)

4.3.5.2.2 LockOnReset Restrictions

The TPer SHALL support the following LockOnReset column values:

a) { 0 } (i.e. Power Cycle); and
b) { 0, 3 } (i.e. Power Cycle and Programmatic).

4.3.5.3 MBRControl (MBR_O)

*MC1 = Only a limited set of DoneOnReset values is required to be supported by Ruby SSC SDs. Refer to Section 4.3.5.3.1 for details.

Table 37 Locking SP - MBRControl Table Preconfiguration

UID	Enable	Done	DoneOnReset
00 00 08 03 00 00 00 01	False	**False**	**Power Cycle** *MC1

4.3.5.3.1 DoneOnReset Restrictions

The TPer SHALL support the following DoneOnReset column values:

a) { 0 } (i.e. Power Cycle); and
b) { 0, 3 } (i.e. Power Cycle and Programmatic).

4.3.5.4 MBR (MBR_O)

If supported, than the MBR minimum size SHALL be 128 MB (0x08000000).

The initial contents of the MBR table SHALL be vendor unique.

4.3.5.5 K_AES_128 or K_AES_256 (M)

At least one of the following Table 38 or Table 39 SHALL be supported.

The following Table 38 contains Optional rows designated with (O).
*K1 = indirectly writable using the `GenKey` Method.

Table 38 Locking SP - K_AES_128 Table Preconfiguration

UID	Name	CommonName	Key	Mode
00 00 08 05 00 00 00 01	"K_AES_128_GlobalRange_Key"		**VU** *K1	**VU**
00 00 08 05 00 03 00 01 (ALR O)	"K_AES_128_Range1_Key"		**VU** *K1	**VU**
00 00 08 05 00 03 NN NN (ALR O)	"K_AES_128_RangeNNNN_Key"		**VU** *K1	**VU**

Table 39 Locking SP - K_AES_256 Table Preconfiguration

UID	Name	CommonName	Key	Mode
00 00 08 06 00 00 00 01	"K_AES_256_GlobalRange_Key"		**VU** *K1	**VU**
00 00 08 06 00 03 00 01 (ALR O)	"K_AES_256_Range1_Key"		**VU** *K1	**VU**

UID	Name	CommonName	Key	Mode
00 00 08 06 00 03 NN NN (ALR O)	"K_AES_256_RangeNNNN_Key"		**VU** *K1	**VU**

4.3.6 Locking Template Methods

Refer to Section 4.3.1.5 for supported methods.

4.3.7 SD Read/Write Data Command Locking Behavior Interactions with Range Crossing

If the SD receives a read or write command that spans multiple Locking ranges and the Locking ranges are not locked, the SD SHALL either:

- Process the data transfer as specified in [2], if Range Crossing Behavior bit is set to 0 (in Level 0 Discovery Ruby SSC Feature, see 3.1.1.5)

  OR

- Terminate the command with “Other Invalid Command Parameter” as defined in [4], if Range Crossing Behavior bit is set to 1 (in Level 0 Discovery Ruby SSC Feature).

4.3.8 Non Template Tables

4.3.8.1 DataStore (M)

The DataStore is a byte table. It can be used by the host for generic secure data storage. The DataStore table SHALL be at least 128KB in size (the Table table object that represents the DataStore table SHALL have a Rows column value of at least 0x00020000).

The access control for modification or retrieval of data in the table initially requires a member of the Admins class authority. These access control settings are personalizable. Initial DataStore content value is VU.

5 Appendix – SSC Specific Features

5.1 Ruby SSC-Specific Methods

5.1.1 Activate – Admin Template SP Object Method

`Activate` is a Ruby SSC-specific method for managing the life cycle of SPs created in manufacturing (Manufactured SP), whose initial life cycle state is “Manufactured-Inactive”.

```
SPObjectUID.Activate[ ]
=>
[ ]
```

`Activate` is an object method that operates on objects in the Admin SP’s `SP` table. The TPer SHALL NOT permit `Activate` to be invoked on the SP objects of issued SPs.

Invocation of `Activate` on an SP object that is in the “Manufactured-Inactive” state causes the SP to transition to the “Manufactured” state. Invocation of `Activate` on an SP in any other life cycle state SHALL complete successfully provided access control is satisfied, and have no effect. The `Activate` method allows the TPer owner to “turn on” an SP that was created in manufacturing.

This method operates within a Read-Write session to the Admin SP. The SP SHALL be activated immediately after the method returns success if its invocation is not contained within a transaction.

In case of an “Activate Error” (see [4]) `Activate` SHALL fail with a status of FAIL.

The MethodID for `Activate` SHALL be 00 00 00 06 00 00 02 03.

5.1.1.1 Activate Support

Support for `Activate` within transactions is (N), and the behavior of Activate within transactions is out of the scope of this document.

If the Locking SP was created in manufacturing, and its Original Factory State is “Manufactured-Inactive” (see Section 5.2.2), support for `Activate` on the Locking SP’s object in the `SP` Table is mandatory.

5.1.1.2 Side effects of Activate

Upon successful activation of an SP that was in the “Manufactured-Inactive” state, the following changes SHALL be made:

- The `LifeCycleState` column of SP’s object in the Admin SP’s `SP` table SHALL change to “Manufactured”.
- The current SID PIN (C_PIN_SID) in the Admin SP is copied into the `PIN` column of Admin1’s `C_PIN` credential (C_PIN_Admin1) in the activated SP. This allows for taking ownership of the SP with a known PIN credential.
- Any TPer functionality affected by the life cycle state of the SP based on the templates incorporated into it is modified as defined in the appropriate Template reference Section of [2], and as defined in the “State transitions for Manufactured SPs” section (Section 5.2.2.2) and “State behaviors for Manufactured SPs” section (Section 5.2.2.3) of this specification.

5.1.2 Revert – Admin Template SP Object Method

`Revert` is a Ruby SSC-specific method for managing the life cycle of SPs created in manufacturing (Manufactured SP).

```
SPObjectUID.Revert[ ]
=>
[ ]
```

Revert is an object method that operates on objects in the Admin SP's SP table. The TPer SHALL NOT permit Revert to be invoked on the SP objects of issued SPs.

Invoking Revert on an SP object causes the SP to revert to its Original Factory State. This method allows the TPer owner (or TPer manufacturer, if access control permits and the Maker authorities are enabled) to remove the SP owner's ownership of the SP and revert the SP to its Original Factory State.

Invocation of Revert is permitted on Manufactured SPs that are in any life cycle state. Successful invocation of Revert on a Manufactured SP that is in the "Manufactured-Inactive" life cycle state SHALL have no effect on the SP.

This method operates within a Read-Write session to the Admin SP. The TPer SHALL revert the SP immediately after the method is successfully invoked outside of a transaction. If Revert is invoked on the Admin SP's object in the SP table, the TPer SHALL abort the session immediately after reporting status of the method invocation if invoked outside of a transaction. The TPer MAY prepare a CloseSession method for retrieval by the host to indicate that the session has been aborted.

The MethodID for Revert SHALL be 00 00 00 06 00 00 02 02.

5.1.2.1 Revert Support

Support for Revert within transactions is (N), and the behavior is out of the scope of this document.

Support for Revert on the Admin SP's object in the SP table is mandatory. (Note that the OFS of the Admin SP is Manufactured, see 5.2.2).

If the Locking SP was created in manufacturing, support for Revert on the Locking SP's object in the SP Table is mandatory.

5.1.2.2 Side effects of Revert

Upon successful invocation of the Revert method, the following changes SHALL be made:

- If the Locking SP is not in the "Manufactured-Inactive" life cycle state, then successful invocation of the Revert method on the Locking SP or Admin SP SHALL cause the media encryption keys to be eradicated, which has the side effect of securely erasing all data in the User LBA portion of the SD.
- If the Locking SP is in the "Manufactured-Inactive" life cycle state, then successful invocation of the Revert method on the Locking SP SHALL NOT cause the erasure of any user data in the SD.

Interactions with interface commands during the processing of the Revert method are defined in [4].

If any TCG reset occurs prior to completing the eradication of all media encryption keys in the SD, then the Revert operation SHALL be aborted and the Locking SP SHALL NOT revert to its Original Factory State.

Begin Informative Content

If any TCG reset occurs during the processing of the Revert method, the result of erasing user data is undefined.

In this case, it may be that some encryption keys have been eradicated and some have not been.

End Informative Content

Upon completion of the eradication of all media encryption keys in the SD, or if the Locking SP is in the "Manufactured-Inactive" life cycle state, the following changes SHALL be made:

- The row in the Admin SP's SP table that represents the invoked SP SHALL revert to its original factory values.
- The SP itself SHALL revert to its Original Factory State. While reverting to its Original Factory State, the TPer SHALL erase all personalization of the SP, and return personalized values to their Original Factory State values. The mechanism for erasure of personalization is implementation-specific.
- When Revert is successfully invoked on the SP object for the Admin SP (UID = 00 00 02 05 00 00 00 01), the ***entire TPer*** SHALL revert to its Original Factory State, including;

- All Admin SP personalization with the exception of the PIN column value of the C_PIN_SID object. See Section 5.1.2.2.1 for the effects of `Revert` upon the PIN column value of the C_PIN_SID object.
- All issued SPs SHALL be deleted, and all Manufactured SPs SHALL revert to Original Factory State. Manufactured SPs in the "Manufactured-Inactive" life cycle state SHALL NOT be affected.

- Any TPer functionality affected by the life cycle state of the SP based on the templates incorporated into it is modified as defined in the appropriate Template reference section of [2], and as defined in the "State transitions for Manufactured SPs" section (Section 5.2.2.2) and "State behaviors for Manufactured SPs" section (Section 5.2.2.3) of this specification.

5.1.2.2.1 Effects of Revert on the PIN Column Value of C_PIN_SID

When Revert is successfully invoked on the SP object for the Admin SP (UID = 00 00 02 05 00 00 00 01), the PIN column value of the C_PIN_SID object SHALL be affected as follows:

1. If the SID authority has never been successfully authenticated, then the C_PIN_SID PIN column SHALL remain at its current value.
2. If the SID authority has previously been successfully authenticated, then:
   a) If the value of the "Behavior of C_PIN_SID PIN upon TPer Revert" field in the Ruby SSC V1.00 Level 0 Feature Descriptor is 0x00, then the C_PIN_SID PIN column SHALL be set to the PIN column value of the C_PIN_MSID object. Additionally, the "Initial C_PIN_SID PIN Indicator" field SHALL be set to 0x00 upon completion of the Revert.
   b) If the value of the "Behavior of C_PIN_SID PIN upon TPer Revert" field in the Ruby SSC V1.00 Level 0 Feature Descriptor is not 0x00, then the C_PIN_SID PIN column SHALL be set to a vendor unique (VU) value.

5.1.3 RevertSP – Base Template SP Method

`RevertSP` is a Ruby SSC-specific method for managing the life cycle of an SP, if it was created in manufacturing (Manufactured SP).

```
ThisSP.RevertSP[ KeepGlobalRangeKey = boolean ]
=>
[ ]
```

`RevertSP` is an SP method in the Base Template.

Invoking `RevertSP` on an SP SHALL cause it to revert to its Original Factory State. This method allows the SP owner to relinquish control of the SP and revert the SP to its Original Factory State.

This method operates within a Read-Write session to an SP. The TPer SHALL revert the SP immediately after the method is successfully invoked outside of a transaction. Upon completion of reverting the SP, the TPer SHALL report status of the method invocation if invoked outside of a transaction, and then immediately abort the session. The TPer MAY prepare a `CloseSession` method for retrieval by the host to indicate that the session has been aborted.

The MethodID for `RevertSP` SHALL be 00 00 00 06 00 00 00 11.

5.1.3.1 RevertSP Support

Support for `RevertSP` within transactions is (N), and the behavior is out of the scope of this document.

If the Locking SP was created in manufacturing, support for `RevertSP` on the Locking SP is mandatory.

5.1.3.2 KeepGlobalRangeKey parameter (Locking Template-specific)

The optional **KeepGlobalRangeKey** parameter is a Locking Template-specific optional parameter. This parameter provides a mechanism for the Locking SP to be "turned off" without eradicating the media encryption key for the Global locking range. This allows the TCG management of the SD's locking and media encryption features to be disabled without causing a cryptographic erase of the user data associated with the Global locking range.

When this parameter is present and set to True, the TPer SHALL continue to use the media encryption key associated with the Global locking range after the Locking SP transitions to the "Manufactured-Inactive" state.

If the Global Range is either Read Unlocked or Write Unlocked at the time of invocation of `RevertSP`, then the TPer shall comply with the request to keep the Global Range's media encryption key.

If the Global Range is Read Locked and Write Locked then invocation of the RevertSP method with the **KeepGlobalRangeKey** parameter set to True SHALL fail with status FAIL, and the SP SHALL NOT change life cycle states.

If the Locking SP was created in manufacturing, support for the **KeepGlobalRangeKey** parameter is mandatory for the Locking SP.

The parameter number for **KeepGlobalRangeKey** SHALL be 0x060000.

5.1.3.3 Side effects of RevertSP

Upon successful invocation of the `RevertSP` method, the following changes SHALL be made:

- If the **KeepGlobalRangeKey** parameter is not present or set to False, then successful invocation of the `RevertSP` method on the Locking SP or Admin SP SHALL cause the media encryption keys to be eradicated, which has the side effect of securely erasing all data in the User LBA portion of the SD.
- If the **KeepGlobalRangeKey** parameter is set to True, then successful invocation of the `RevertSP` method on the Locking SP SHALL cause all media encryption keys to be eradicated except for the Global Range's media encryption key (K_AES_{128,256}_GlobalRange_Key).

Interactions with interface commands during the processing of the `RevertSP` method are defined in [4].

If any TCG reset occurs prior to completing the eradication of media encryption keys in the SD, then the operation SHALL be aborted and the Locking SP SHALL NOT revert to its Original Factory State.

Begin Informative Content

If any TCG reset occurs during the processing of the `RevertSP` method, the result of erasing user data is undefined.

In this case, it may be that some encryption keys have been eradicated and some have not been.

End Informative Content

Upon completion of the eradication of media encryption keys in the SD, the following changes SHALL be made:

- The row in the Admin SP's `SP` table that represents the Locking SP SHALL revert to its original factory value.
- The Locking SP itself SHALL revert to its Original Factory State. While reverting to its Original Factory State, the TPer SHALL erase all personalization of the SP, and return the personalized values to their Original Factory State values. The mechanism for erasure of personalization is implementation-specific.
- Any TPer functionality affected by the life cycle state of the SP based on the templates incorporated into it is modified as defined in the appropriate Template reference Section of [2], and as defined in the "State transitions for Manufactured SPs" section (Section 5.2.2.2) and "State behaviors for Manufactured SPs" section (Section 5.2.2.3) of this specification.

5.2 Life Cycle

5.2.1 Issued vs. Manufactured SPs

5.2.1.1 Issued SPs

[2] describes the life cycle states for SPs that are created through the issuance process.

For Ruby SSC-compliant TPers that support issuance, refer to [2] for the life cycle states and life cycle management.

5.2.1.2 Manufactured SPs

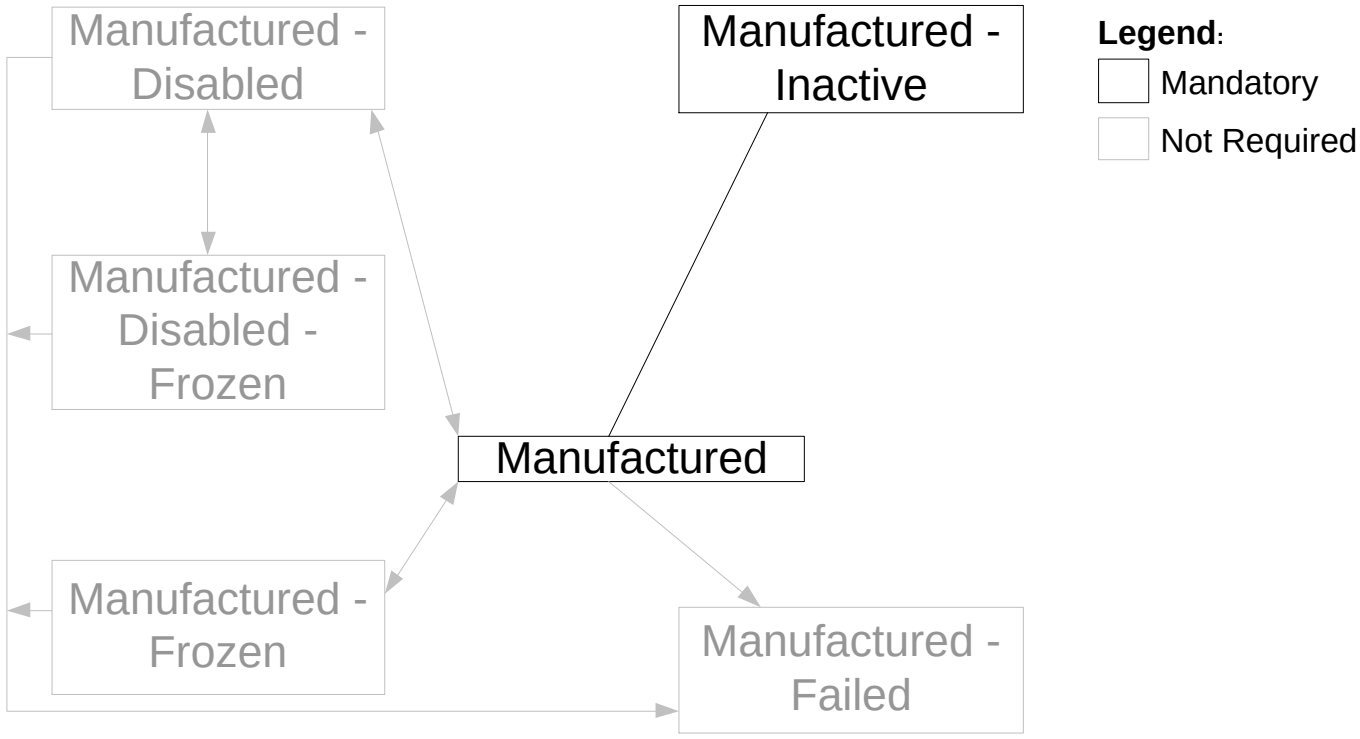
[2] defines the life cycle and life cycle management of Manufactured SPs as implementation-specific.

Ruby SSC-compliant SPs that are created in manufacturing (Manufactured SPs) SHALL NOT have an implementation-specific life cycle, and SHALL conform to the life cycle defined in Section 5.2.2.

5.2.2 Manufactured SP Life Cycle States

The state diagram for Manufactured SPs is shown in Figure 3.

Figure 3 Life Cycle State Diagram for Manufactured SPs



Additional state transitions may exist depending on the states supported by the SD and the SP's Original Factory State. Invoking `Revert` or `RevertSP` (see Sections 5.1.2 and 5.1.3) on the SP will cause the SP to transition back to its Original Factory State.

The Original Factory State of the Admin SP SHALL be Manufactured. The only state that is mandatory for the Admin SP is Manufactured.

If the Locking SP is a Manufactured SP, its Original Factory State SHALL be Manufactured-Inactive. Support for Locking SP states of Manufactured and Manufactured-Inactive are mandatory.

The other states in the state diagram are beyond the scope of this document.

5.2.2.1 State definitions for Manufactured SPs

1. **Manufactured-Inactive**: This is the Original Factory State for SPs that are created in manufacturing, where it is not desirable for the functionality of that SP to be active when the TPer is shipped. All templates that exist in an SP that is in the "Manufactured-Inactive" state SHALL be counted in the `Instances` column of the appropriate objects in the Admin SP's `Template` table. Sessions cannot be opened to SPs in the "Manufactured-Inactive" state. Only SPs whose Original Factory State was "Manufactured-Inactive" are allowed to return to the "Manufactured-Inactive" state.

   If the Locking SP is a Manufactured SP, support for the "Manufactured-Inactive" state is mandatory for the Locking SP.

2. **Manufactured**: This is the standard operational state of a Manufactured SP, and defines the initial required access control settings of an SP based on the Templates incorporated into the SP, prior to personalization.

   The Manufactured state is mandatory for the Admin SP.

   If the Locking SP is a Manufactured SP, support for the Manufactured state is mandatory for the Locking SP.

5.2.2.2 State transitions for Manufactured SPs

The following sections describe the mandatory and optional state transitions for Ruby SSC-compliant Manufactured SPs.

For the Admin SP, the only transition for which support is mandatory is "ANY STATE to ORIGINAL FACTORY STATE" (see 5.2.2.2.2). As the only mandatory state for the Admin SP is Manufactured, the only mandatory transition is from Manufactured to Manufactured with the side effect of reverting the entire TPer to its Original Factory State. See Section 5.1.2 for details.

If the Locking SP is a Manufactured SP, support for the "ANY STATE to ORIGINAL FACTORY STATE" transition (see 5.2.2.2.2) is mandatory. Specifically:

- Support for the "Manufactured to Manufactured-Inactive" transition is mandatory. This transition is accomplished via the Revert or RevertSP method (see Sections 5.1.2 and 5.1.3).

- Support for the "Manufactured-Inactive to Manufactured" transition (see5.2.2.2.1) is mandatory. This transition is accomplished via the Activate method (see Section 5.1).

5.2.2.2.1 Manufactured-Inactive to Manufactured

Triggers:

- The Activate method (see Section 5.1) is successfully invoked on the SP's object in the Admin SP's `SP` table.

Side effects:

- The value in the `LifeCycleState` column of the SP's object in the Admin SP's `SP` table changes to Manufactured.
- The current SID PIN (C_PIN_SID) in the Admin SP is copied into the `PIN` column of Admin1's `C_PIN` credential (C_PIN_Admin1) in the activated SP. This allows for taking ownership of the SP with a known PIN credential.
- Any functionality enabled by the templates incorporated into the SP becomes active.

When the Locking SP transitions from the "Manufactured-Inactive" state to the Manufactured state (via invocation of the Activate method), the SD SHALL NOT destroy any user data.

5.2.2.2.2 ANY STATE to ORIGINAL FACTORY STATE

Triggers:

- `Revert` or `RevertSP` is successfully invoked on the SP.

Side effects:

- The value in the `LifeCycleState` column of the SP's object in the Admin SP's `SP` table changes to the value of the SP's Original Factory State.
- The SP itself reverts to its Original Factory State, as described in the Sections 5.1.2 and 5.1.3.
- If the SP's Original Factory State was Manufactured-Inactive, any functionality enabled by the templates incorporated into the SP becomes inactive.

5.2.2.3 State behaviors for Manufactured SPs

5.2.2.3.1 Manufactured-Inactive

Any functionality enabled by the templates incorporated into the SP is inactive in this state. Sessions cannot be opened to SPs in this state.

When the Locking SP is in the Manufactured-Inactive state, the TCG management of the SD's locking SHALL be disabled.

5.2.2.3.2 Manufactured

Behavior of an SP in the Manufactured state is identical to the behavior of an SP in the Issued state, as described in [2].

When the Locking SP is in the Manufactured state, the TCG management of the SD's locking SHALL be enabled.

5.2.3 Type Table Modification

In order to accommodate the additional life cycle states defined in Ruby, the `life_cycle_state` type SHALL be defined as follows for Ruby:

Table 40 LifeCycle Type Table Modification

UID	Name	Format	Size	Description
00 00 00 05 00 00 04 05	life_cycle_state	Enumeration_Type, 0, 15		Used to represent the current life cycle state. The valid values are: 0 = issued, 1 = issued-disabled, 2 = issued-frozen, 3 = issued-disabled-frozen, 4 = issued-failed, 5-7 = reserved, 8 = manufactured-inactive, 9 = manufactured, 10 = manufactured-disabled, 11 = manufactured-frozen, 12 = manufactured-disabled-frozen, 13 = manufactured-failed, 14-15 = reserved

5.3 Byte Table Access Granularity

Begin Informative Content

While the general architecture defined in [2] allows data to be written into byte tables starting at any arbitrary byte boundary and with any arbitrary byte length, certain types of storage devices work more efficiently when data is written aligned to a larger block boundary. This section defines extensions to [2] that allow a device to report the restrictions that it enforces when the host invokes the `Set` method on byte tables.

End Informative Content

5.3.1 Table Table Modification

In order to allow a storage device to report its mandatory and recommended data alignment restrictions when accessing byte tables, the `Table` table SHALL contain the additional columns shown in Table 41.

Table 41 Table Table Additional Columns

Column Number	Column Name	IsUnique	Column Type
0x0D	MandatoryWriteGranularity		uinteger_4
0x0E	RecommendedAccessGranularity		uinteger_4

5.3.1.1 MandatoryWriteGranularity

This column is used to report the granularity that the storage device enforces when the host invokes the `Set` method on byte tables.

This column SHALL NOT be modifiable by the host.

5.3.1.1.1 Object Tables

For rows in the `Table` table that pertain to object tables, the value of this column SHALL be zero.

5.3.1.1.2 Byte Tables

For rows in the `Table` table that pertain to byte tables, this column indicates the mandatory access granularity (in bytes) for the `Set` method for the table described in these rows of the `Table` table. The MandatoryWriteGranularity column indicates the alignment requirement for both the access start offset (the Where parameter) and length (number of bytes in the Values parameter).

The value of this column SHALL be less than or equal to the value in the RecommendedAccessGranularity column in the same row of the `Table` table.

MandatoryWriteGranularity SHALL be less than or equal to 8192.

When the host invokes the `Set` method on a byte table, if ValidMandatoryGranularity (see Figure 4, which explains ValidMandatoryGranularity in pseudo-code representation) is False, then the method SHALL fail with status INVALID_PARAMETER.

If the TPer does not have a requirement on mandatory alignment for the byte table described in a row of the `Table` table, then its MandatoryWriteGranularity column SHALL be set to 1.

TCG Copyright 2020

Figure 4 ValidMandatoryGranularity

For the `Set` method:
ValidMandatoryGranularity is True if
a) (x modulo MandatoryWriteGranularity) = 0

and

b) (y modulo MandatoryWriteGranularity) = 0

where:
x = the start offset of the `Set` method
(i.e., the value of the Where parameter)
y = the number of data bytes being set
(i.e., the length of the Values parameter)

5.3.1.2 RecommendedAccessGranularity

This column is used to report the granularity that the storage device recommends when the host invokes the `Set` or `Get` method on byte tables.

This column SHALL NOT be modifiable by the host.

5.3.1.2.1 Object Tables

For rows in the `Table` table that pertain to object tables, the value of this column SHALL be zero.

5.3.1.2.2 Byte Tables

For rows in the `Table` table that pertain to byte tables, this column indicates the recommended access granularity (in bytes) for the `Set` and `Get` method for the table described in these rows of the `Table` table. The RecommendedAccessGranularity column indicates the alignment of data for the `Set` and `Get` method that allows for optimal `Set`/`Get` performance.

If the TPer does not have a recommended alignment for the byte table described in a row of the `Table` table, then its RecommendedAccessGranularity column SHALL be set to 1.

When the host invokes the `Set` method on a byte table, if ValidRecommendedGranularity (see Figure 5) is False, then the performance of the TPer MAY be reduced when processing the method.

Figure 5 ValidRecommendedGranularity for Set

For the `Set` method:
ValidRecommendedGranularity is True if
a) (x modulo RecommendedAccessGranularity) = 0

and

b) (y modulo RecommendedAccessGranularity) = 0

where:
x = the start offset of the `Set` method
(i.e., the value of the Where parameter)
y = the number of data bytes being set
(i.e., the length of the Values parameter)

TCG Copyright 2020

When the host invokes the Get method on a byte table, if ValidRecommendedGranularity (see Figure 6) is False, then the performance of the TPer MAY be reduced when processing the method.

Figure 6 ValidRecommendedGranularity for Get

For the Get method:
ValidRecommendedGranularity is True if
a) (x modulo RecommendedAccessGranularity) = 0

and

b) (y modulo RecommendedAccessGranularity) = 0

where:
x = the start offset of the Get method
(i.e., the value of the startRow component of the Cellblock parameter)
y = the number of data bytes being retrieved
(i.e., the difference of the endRow and startRow components of the Cellblock parameter, plus one)

5.4 Examples of Alignment Geometry Reporting

Figure 7 illustrates reporting for a typical legacy storage device where there is one logical block per physical block on the media.

Figure 7 - Example: AlignmentGranularity=1, Lowest Aligned LBA=0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Alignment Granularity																			

Figure 8 illustrates geometry for a storage device where there are 8 logical blocks per physical block (e.g., a 4K physical block) and the first logical block is aligned at the beginning of the first physical block.

Figure 8 - Example: AlignmentGranularity=8, Lowest Aligned LBA=0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
AlignmentGranularity								AlignmentGranularity								. . .			

Figure 9 illustrates geometry for a storage device where there are 8 logical blocks per physical block (e.g., a 4K physical block) and LBA=1 is the first logical block that is aligned at the beginning of a physical block

Figure 9 - Example: AlignmentGranularity=8, Lowest Aligned LBA=1

	0	**1**	2	3	4	5	6	7	8	9	10	11	12
AlignmentGranularity		AlignmentGranularity								. . .			

Figure 10 illustrates geometry for a storage device where there are 2000 logical blocks per physical block and LBA=1234 is the first logical block that is aligned at the beginning of a physical block.

Figure 10 - Example: AlignmentGranularity=2000, Lowest Aligned LBA=1234

	0	. . .	1230	1231	1232	1233	**1234**		. . .		3233	3234	. . .
AlignmentGranularity							AlignmentGranularity					. . .	