TCG Storage
Security Subsystem Class: Pyrite

Specification Version 1.00
Revision 1.00

August 5, 2015

Contact: admin@trustedcomputinggroup.org

TCG

PUBLISHED

Disclaimers, Notices, and License Terms

TABLE OF CONTENTS

Tables

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 Pyrite Security Subsystem Class (SSC). Any SD that claims Pyrite 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].

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]. Trusted Computing Group (TCG), "TCG Storage Interface Interactions Specification", Version 1.04

[4]. Trusted Computing Group (TCG), "TCG Storage Security Subsystem Class: Opal", Version 1.00

[5]. Trusted Computing Group (TCG), "TCG Storage Security Subsystem Class: Opal", Version 2.01

[6]. Trusted Computing Group (TCG), "TCG Storage Feature Set: Block SID Authentication", Version 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 [3]

3. TCG Storage Architecture Core Specification [2]

1.6 SSC Terminology

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

Table 1 Pyrite 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.
NN NN	The LSBs of a Locking object's UID (hexadecimal)
(MBR O)	Values in this specification marked with this identifier are components of the MBR Shadowing feature.

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	Pyrite SSC specified	Fixed	• Cell content is Read-Only. • Access control is fixed. • Value is specified by the Pyrite 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

2 Pyrite SSC Overview

2.1 Pyrite SSC Use Cases and Threats

Begin Informative Content

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

- Provide logical locking of the storage device interface
- Enable interoperability between multiple SD vendors

A Pyrite SSC compliant SD:

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

A SD that implements Pyrite SSC provides a mechanism to control access to user data. This SSC does not specify capabilities for cryptographic protection of user data at rest.

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: 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 and locking under host control via either an explicit lock or implicit lock triggered by a reset event.
- Repurpose & End-of-Life Preparation: reset of locking credential(s) for Storage Device repurposing or decommissioning.

This specification is defined as a limited subset of [5], with the intent to provide a smaller set of features and use cases while remaining command compatible, but without mandating support of media encryption, such that an application that can manage a Pyrite SSC-based SD can also manage a SD that is compatible with [5].

End Informative Content

2.2 Security Providers (SPs)

A Pyrite 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 Pyrite 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 Authentication

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

2.5 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.

2.6 Access Control & Personalization

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

2.7 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.

2.8 SSC Discovery

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

2.9 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 not absent, then all components of this feature as identified by “(MBR O)” SHALL be supported, and Level 0 discovery SHALL report that MBR Shadowing is not absent.

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

2.10 Mandatory Feature Sets

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

- Block SID Authentication ([6])

3 Pyrite SSC Features

3.1 Security Protocol 1 Support

3.1.1 Level 0 Discovery (M)

Refer to [2] for more details.

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

- Level 0 Discovery Header
- TPer Feature Descriptor
- Locking Feature Descriptor
- Pyrite 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)

** = the present current state of the respective feature

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 Absent	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 Absent (MBR O) = VU
  - If MBR Shadowing feature is not absent (i.e., is supported), this bit SHALL be 0.
  - If MBR Shadowing feature is absent (i.e., is not supported), this bit SHALL be 1.
- MBR Done (MBR O) = **
- MBR Enabled (MBR O) = **

- Media Encryption = 0
- Locked = **
- Locking Enabled = See 3.1.1.3.1
- Locking Supported = 1

3.1.1.3.1 LockingEnabled Definition

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

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

3.1.1.4 Pyrite SSC Feature (Feature Code = 0x0302)

Table 6 Level 0 Discovery - Pyrite SSC Feature Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			Feature Code (0x0302)				
1								(LSB)
2	Version				Reserved			
3	Length							
4	(MSB)			Base ComID				
5								(LSB)
6	(MSB)			Number of ComIDs				
7								(LSB)
8-12	Reserved for future common SSC parameters							
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 = 0x0302
- Version = 0x1 or any version that supports the defined features in this SSC
- Length = 0x10
- Base ComID = VU
- Number of ComIDs = 0x0001 (minimum value)
- Initial C_PIN_SID PIN Indicator = 0x00
  - 0x00 = The initial C_PIN_SID PIN value is equal to the C_PIN_MSID PIN value
- Behavior of C_PIN_SID PIN upon TPer Revert = 0x00
  - 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

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 Pyrite 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, it 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;

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 [3]). There is no IF-RECV response to the TPER_RESET command.

The TPER_RESET command is defined in Table 7.

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

Table 7 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. For details, reference “Invalid Transfer Length parameter on IF-SEND” in [3].

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 Sync 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 0x01, the TPer SHALL:

- support at least one statically allocated ComID for Synchronous Protocol communication.
- have the ComID Extension values = 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 with an inactive or unsupported ComID, the TPer SHALL either:

- terminate the command as defined in [3] with “Other Invalid Command parameter”, or
- follow the requirements defined in [2] for “Inactive or Unsupported ComID parameter on IF-SEND” or “Inactive or Unsupported ComID parameter on IF-RECV”.

ComIDs SHALL be assigned based on the allocation presented in Table 8

Table 8 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)

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 9. 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 9 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.

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.

3.3.5 Storage Device Resets

3.3.5.1 Interface Resets

Interface resets that generate TCG reset events are defined in [3].
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 10 replaces the definition of TCG reset_types that are defined in [2]:

Table 10 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 Pyrite SSC-compliant Functions and SPs

4.1 Session Manager

4.1.1 Methods

4.1.1.1 Properties (M)

A Pyrite 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 11.

Table 11 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 Pyrite-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 Pyrite-compliant SD SHALL support the following parameters for the `SyncSession` method:

- HostSessionID
- SPSessionID

4.1.1.4 CloseSession (O)

A Pyrite-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 12 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 13 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 5.3 for a description and requirements of the MandatoryWriteGranularity and RecommendedAccessGranularity columns.

Table 14 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 Pyrite SSC requires only the tables from the Admin SP to be included in the Admin SP’s `Table` table, as indicated in Table 14.

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 15 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 contains 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 16 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						
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						

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 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						
TPerInfo																
		00 00 02 01 00 03 00 01	TPerInfoObj	Get		ACE_Anybody				ACE_Anybody						
		00 00 02 01 00 03 00 01	TPerInfoObj	Set		ACE_TPerInfo_Set_ProgrammaticResetEnable				ACE_Anybody						
Template																

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 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						
		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_Anybody						
*AC9		00 00 02 05 TT TT TT TT *AC7	SPObj	Activate		ACE_SP_SID				ACE_Anybody						

4.2.1.6 ACE (M)

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

Table 17 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
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
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)

Table 18 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 06	"SID"		F	Null	T	None	None	F	Password	C_PIN_SID	Null	Null						

4.2.1.8 C_PIN (M)

Table 19 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"		MSID	Null	VU	VU	FALSE
00 00 00 0B 00 00 84 02	"C_PIN_MSID"		MSID				

The PIN column value of C_PIN_SID is set to the PIN column value of C_PIN_MSID in OFS.

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 20 Admin SP – TPerInfo Columns

Column Number	Column Name	IsUnique	Colum 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 “Pyrite” as one of the list elements.

Table 21 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		[“Pyrite”] *TP3	FALSE

4.2.3.2 Template (M)

Table 22 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 23 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 Pyrite 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 24 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 25 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)

Refer to section 5.3 for a description and requirements of the MandatoryWriteGranularity and RecommendedAccessGranularity columns.

Refer to 2.9 for details on the requirements for supporting tables identified with “(MBR O)”.

Table 26 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 08 01	"LockingInfo"			Object									0	0
00 00 00 01 00 00 08 02	"Locking"			Object									0	0
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 10 01	"DataStore"			Byte			0x00020000 min						VU	VU

4.3.1.4 Type (N)

The Type table is (N) by Pyrite. 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 3 entries (1 User authority, 1 Admin authority, and 1 Boolean operator).

4.3.1.5 MethodID (M)

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

Table 27 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 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)

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

(MBR O) = Refer to 2.9 for requirements for supporting rows marked by this identifier.

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 28 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						

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 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						
		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						

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 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						
		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						

Table Association - informative only															
	UID														
	InvokingID	00 00 00 08 00 03 A8 01	00 00 00 08 00 03 A8 02	00 00 00 08 00 03 D0 00											
	InvokingID Name - informative only	ACE_C_PIN_User1_Set_PIN	ACE_C_PIN_User2_Set_PIN	ACE_Locking_GlobalRange_Get_RangeStartToActiveKey											
	MethodID	Set	Set	Set											
	CommonName														
	ACL	ACE_ACE_Set_BooleanExpression	ACE_ACE_Set_BooleanExpression	ACE_ACE_Set_BooleanExpression											
	Log														
	AddACEACL														
	RemoveACEACL														
	GetACLACL	ACE_Anybody	ACE_Anybody	ACE_Anybody											
	DeleteMethodACL														
	AddACELog														
	RemoveACELog														
	GetACLLog														
	DeleteMethodLog														
	LogTo														

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 E0 00	ACE_Locking_GlobalRange_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						
(MBR O)		00 00 00 08 00 03 F8 01	ACE_MBRControl_Set_DoneToDOR	Set		ACE_ACE_Set_BooleanExpression				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 00 08 00 03 FC 00	ACE_DataStore_Get_All	Set		ACE_ACE_Set_BooleanExpression				ACE_Anybody						
		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 *AC6	AuthorityObj	Get		ACE_Authority_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 00 09 00 03 00 01	User1	Set		ACE_Authority_Set_Enabled				ACE_Anybody						
		00 00 00 09 00 03 00 02	User2	Set		ACE_Authority_Set_Enabled				ACE_Anybody						
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						

Table Association - informative only														
UID														
InvokingID	00 00 00 0B 00 03 00 01	00 00 00 0B 00 03 00 02	00 00 00 0B 00 01 00 01	00 00 00 0B 00 03 00 01										
InvokingID Name - informative only	C_PIN_User1	C_PIN_User2	C_PIN_Admin1	C_PIN_User1										
MethodID	Get	Get	Set	Set										
CommonName														
ACL	ACE_C_PIN_Admins_Get_All_NOPIN	ACE_C_PIN_Admins_Get_All_NOPIN	ACE_C_PIN_Admins_Set_PIN	ACE_C_PIN_User1_Set_PIN										
Log														
AddACEACL														
RemoveACEACL														
GetACLACL	ACE_Anybody	ACE_Anybody	ACE_Anybody	ACE_Anybody										
DeleteMethodACL														
AddACELog														
RemoveACELog														
GetACLLog														
DeleteMethodLog														
LogTo														

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	Set		ACE_C_PIN_User2_Set_PIN				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						

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 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_Anybody						
MBRControl																
(MBR O)		00 00 08 03 00 00 00 01	MBRControlObj	Get		ACE_Anybody				ACE_Anybody						
(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						

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 04 00 00 00 00	MBR	Set		ACE_Admin				ACE_Anybody						
DataStore																
		00 00 10 01 00 00 00 00	DataStore	Get		ACE_DataStore_Get_All				ACE_Anybody						
		00 00 10 01 00 00 00 00	DataStore	Set		ACE_DataStore_Set_All				ACE_Anybody						

4.3.1.7 ACE (M)

*ACE1 = The TPer SHALL support the values of “Admins” and “Admins OR User1” in the BooleanExpr column of the ACE_C_PIN_User1_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 = The TPer SHALL support the values of “Admins” and “Admins OR User2” in the BooleanExpr column of the ACE_C_PIN_User2_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.

(MBR O) = Refer to 2.9 for requirements for supporting rows marked by this identifier.

Table 29 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
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					
	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
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 *ACE2	PIN
Locking					
	00 00 00 08 00 03 D0 00	"ACE_Locking_GlobalRange_Get_ RangeStartToActiveKey"		Admins	RangeStart, RangeLength, ReadLockEnabled, WriteLockEnabled, ReadLocked, WriteLocked, LockOnReset, ActiveKey

Table Association –Informative Column	UID	Name	CommonName	BooleanExpr	Columns
	00 00 00 08 00 03 E0 00	"ACE_Locking_GlobalRange_Set_RdLocked"		Admins	ReadLocked
	00 00 00 08 00 03 E8 00	"ACE_Locking_GlobalRange_Set_WrLocked"		Admins	WriteLocked
	00 00 00 08 00 03 F0 00	"ACE_Locking_GlblRng_Admins_Set"		Admins	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)

Table 30 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						
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 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						

4.3.1.9 C_PIN (M)

Table 31 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	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

4.3.2 Base Template Methods

Refer to section 4.3.1.5 for supported methods.

4.3.3 Crypto Template Tables

A Pyrite 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]:

Table 32 Locking SP – LockingInfo Columns

Column Number	Column Name	IsUnique	Colum 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] for a detailed explanation).
  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] for a detailed explanation).
  This column SHALL NOT be modifiable by the host and may be retrieved by Anybody.

Table 33 Locking SP - LockingInfo Table Preconfiguration

UID	Name	Version	EncryptSupport	MaxRanges	MaxReEncryptions	KeysAvailableCfg	AlignmentRequired	LogicalBlockSize	AlignmentGranularity	LowestAlignedLBA
00 00 08 01 00 00 00 01			None	0						

4.3.5.2 Locking (M)

*LT1 = Only a limited set of LockOnReset values is required to be supported by Pyrite SSC SDs. Refer to section 4.3.5.2.1 for details.

Table 34 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 *LT1	00 00 00 00 00 00 00 00									

4.3.5.2.1 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 = If MBR Shadowing is supported (i.e., not absent – see 2.9 and 3.1.1.3), only a limited set of DoneOnReset values is required to be supported by Pyrite SSC SDs. Refer to section 4.3.5.3.1 for details.

Table 35 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 MBR Shadowing is supported (i.e., not absent – see 2.9 and 3.1.1.3), the MBR minimum size SHALL be 128 MB (0x08000000).

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

4.3.6 Locking Template Methods

Refer to Section 4.3.1.5 for supported methods.

4.3.7 Non Template Tables

4.3.7.1 DataStore (M)

The DataStore is a byte table. It can be used by the host for generic secure data storage. 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 Pyrite SSC-Specific Methods

5.1.1 Activate – Admin Template SP Object Method

`Activate` is a Pyrite 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 [3]) `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 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 Pyrite 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:

- The row in the Admin SP's SP table that represents this 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 revert the personalized values to their original factory values. The mechanism for erasure is implementation-specific. This operation has the side effect of disabling the locking interface. Informative note: Revert as defined in this SSC does not initiate any user data erasure. If user data erasure is desired, it is recommended to use the native erase function defined for the underlying interface after execution of Revert .
- 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 personalization of the Admin SP itself, with the exception of the PIN column value of the C_PIN_SID object. See section **Error! Reference source not found.** 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 that were in the Manufactured-Inactive life cycle state SHALL be unaffected.
- 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.

5.1.3 RevertSP – Base Template SP Method

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

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

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 Side effects of RevertSP

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

- The SP's object in the Admin SP's SP table 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 revert the personalized values to their original factory values. The mechanism for erasure is implementation-specific. This operation has the side effect of disabling the locking interface. Informative note: RevertSP as defined in this SSC does not initiate any user data erasure. If user data erasure is desired, it is recommended to use the native erase function defined for the underlying interface after execution of RevertSP.
- 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 Pyrite SSC-compliant TPers that support issuance, refer to [2] for the life cycle states and life cycle management.

5.2.1.2 Manufactured SPs

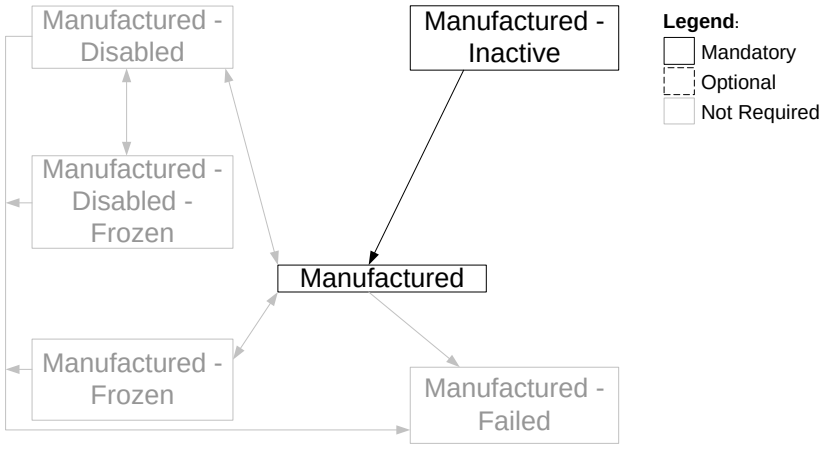
[2] defines the life cycle and life cycle management of Manufactured SPs as implementation-specific.

Pyrite SSC-compliant SPs that are created in manufacturing (Manufactured SPs) SHALL NOT have 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 1.

Figure 1 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 can 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 Pyrite SSC-compliant Manufactured SPs.

For the Admin SP, the only transition for which support is mandatory is "ANY STATE to ORIGINAL FACTORY STATE" (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 (5.2.2.2.2) is mandatory. Specifically:

- Support for the transition from Manufactured to Manufactured-Inactive 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 (5.2.2.2.1) 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 features 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 by [2].

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

5.2.3 Type Table Modification

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

Table 36 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 37.

Table 37 Table Table Additional Columns

Column Number	Column Name	IsUnique	Colum 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 this row 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 2) 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.

Figure 2 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 this row 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 3) is False, then the performance of the TPer MAY be reduced when processing the method.

Figure 3 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)

When the host invokes the Get method on a byte table, if ValidRecommendedGranularity (see Figure 4) is False, then the performance of the TPer MAY be reduced when processing the method.

Figure 4 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)