TRUSTED® COMPUTING GROUP

REFERENCE

TCG Storage Application Note: Configurable Namespace Locking Examples

Version 1.00
Revision 1.00
January 7, 2020

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

CONTENTS

TABLES

1 INTRODUCTION

1.1 Document Purpose

This application note provides examples illustrating the operation of Configurable Namespace Locking (CNL) for the Opal Security Subsystem Class (SSC).

1.2 Intended Audience

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 Document References

[1] Trusted Computing Group (TCG), “TCG Storage Architecture Core Specification”, Version 2.01

[2] Trusted Computing Group (TCG), “Storage Interface Interactions Specification”, Version 1.08

[3] Trusted Computing Group (TCG), “TCG Storage Opal SSC Feature Set: Configurable Namespace Locking”, Version 1.00

[4] NVM Express, Inc., “NVM Express”, Version 1.4

1.4 Abbreviations and Notations

Kn	Key number n
LBA	Logical Block Address
LO	Locking Object
LOn	Locking Object number n
MEK	Media Encryption Key
NS	Namespace
NSn	Namespace number n
LBA[k..j]	A range of LBAs (from k through j, inclusive)
NSn.LBA[k..j]	A range of LBAs in namespace n

2 Namespace Manipulation Examples

2.1 Overview

These examples illustrate the effects of executing various TCG methods and NVMe commands upon the Locking SP. Each example builds upon the previous example. These examples can be viewed as a state transition diagram, with each table representing a state, and the method invocation representing a state transition from the preceding example (state) to the following example (state).

The tables that follow indicate the rows in the Locking SP, although the columns do not correspond (for the most part) to Locking SP columns. The meanings are:

1. Locking Object: An informal name for the Locking object.
2. Namespace: A list of namespaces controlled by that Locking object.
3. NS Global Range: The Namespace Global Range column value in that row, i.e., TRUE or FALSE. This is ignored for the Locking object for the global range.
4. Range in NS: A description of the LBA ranges controlled by that Locking object.

A blank row indicates a Locking Object that does not have a namespace or range assigned to it.

2.2 Initial State

Four namespaces [NS1, …, NS4] are associated with the Global Range Locking object.

Table 1 Example – Initial State

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS3 NS4	Ignored	All	K1 K2 K3 K4

2.3 Assign Namespace Global Range Locking object for NS1

Assign (NSID = NS1, RangeStart = 0, RangeLength = 0) returns { LO1, TRUE, SUCCESS }

Results:

a) LO1 is added
b) Control of NS1 is transferred to LO1

Table 2 Example – After Assign Namespace Global Range Locking object for NS1

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS3 NS4	Ignored	All	K2 K3 K4
LO1	NS1	TRUE	All	K1

2.4 Assign Namespace Global Range Locking object for NS3

Assign (NSID = NS3, RangeStart = 0, RangeLength = 0) returns { LO2, TRUE, SUCCESS }

Results:

a) LO2 is added
b) Control of NS3 is transferred to LO2

Table 3 Example – After Assign Namespace Global Range Locking object for NS3

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All	K1
LO2	NS3	TRUE	All	K3

2.5 Assign Namespace Non-Global Range Locking object for NS1.LBA[10..19]

Assign (NSID = NS1, RangeStart = 10, RangeLength = 10) returns { LO3, FALSE, SUCCESS }

Results:

a) LO3 is added
b) K5 is created
c) Control of NS1.LBA[10..19] is transferred from LO1 to LO3

Table 4 Example – After Assign Namespace Non-Global Range Locking object for NS1.LBA[10..19]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	ALL	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19]	K1
LO2	NS3	TRUE	All	K3
LO3	NS1	FALSE	LBA[10..19]	K5

2.6 Assign Namespace Non-Global Range Locking object for NS1.LBA[30..39]

Assign (NSID = NS1, RangeStart = 30, RangeLength = 10) returns { LO4, FALSE, SUCCESS }

Results:

a) LO4 is added
b) K6 is created
c) Control of NS1.LBA[30..39] is transferred from LO1 to LO4

Table 5 Example – After Assign Namespace Non-Global Range Locking object for NS1.LBA[30..39]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6

2.7 Assign of overlapping Namespace Non-Global Range Locking object does not work

Assign (NSID = NS1, RangeStart = 15, RangeLength = 10) returns INVALID_PARAMETER

Reason for failure: Range NS1.LBA[15..24] overlaps LO3 (NS1.LBA[10..19])

Results:

a) Locking SP is unchanged:

Table 6 Example – Assign of overlapping Namespace Non-Global Range Locking object does not work

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6

2.8 Assign Namespace Non-Global Range Locking object for NS3.LBA[15..24]

Assign (NSID = NS3, RangeStart = 15, RangeLength = 10) returns { LO5, FALSE, SUCCESS }

Results:

a) LO5 is added
b) K7 is created
c) Control of NS3.LBA[15..24] is transferred from LO2 to LO5

Note: This succeeds where the previous Assign() failed because they refer to different namespaces. The logical block address spaces are independent for each namespace.

Table 7 Example – After Assign Namespace Non-Global Range Locking object for NS3.LBA[15..24]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All **except** LBA[15..24]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6
LO5	NS3	FALSE	LBA[15..24]	K7

2.9 Modify Namespace Non-Global Range Locking object for LO5

Set (UID = LO5, RangeStart = 20, RangeLength = 10) returns SUCCESS

Results:

a) LO5 is modified
b) Control of NS3.LBA[15..19] is transferred from LO5 to LO2 (Namespace Global Range Locking object)
c) Control of NS3.LBA[25..29] is transferred from LO2 to LO5.

Table 8 Example – After Modify Namespace Non-Global Range Locking object for LO5

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All **except** LBA[20..29]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6
LO5	NS3	FALSE	LBA[20..29]	K7

2.10 Assign Namespace Non-Global Range Locking object with zero length

Assign (NSID = NS1, RangeStart = 30, RangeLength = 0) returns { LO6, FALSE, SUCCESS }

Results:

a) LO6 is created. It is associated with no LBAs.
b) K8 is created.
c) Because RangeLength is 0, the use of RangeStart = 30 does not cause an overlap with NS1.LBA[30..39], which is associated with LO4.

A later Set() may modify RangeStart and RangeLength, after which the specified Locking object will control the specified range.

Table 9 Example – After Assign Namespace Non-Global Range Locking object with zero length

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All **except** LBA[20..29]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	None	K8

2.11 Set Namespace Non-Global Range Locking object to non-zero length

Set (LO6, RangeStart = 0, RangeLength = 10) returns SUCCESS

Results:

a) LO6 is modified
b) Control of NS3.LBA[0..9] is transferred from LO2 to LO6:

Table 10 Example – After Set Namespace Non-Global Range Locking object non-zero length

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19] and LBA[30..39]	K1
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO4	NS1	FALSE	LBA[30..39]	K6
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.12 Deassign Namespace Non-Global Range Locking object for NS1.[30..39]

Deassign (UID = LO4, KeepNamespaceGlobalRangeKey = FALSE) returns SUCCESS

Results:

a) LO4 is removed
b) K6 is eradicated
c) LBA[30..39] is returned to control of LO1:

Table 11 Example – After Deassign Namespace Non-Global Range Locking object for NS1.[30..39]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19]	K1
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.13 Deassign Namespace Non-Global Range Locking object with KeepNamespaceGlobalRangeKey = TRUE does not work

Deassign (UID = LO3, KeepNamespaceGlobalRangeKey = TRUE) returns INVALID_PARAMETER

Results:

a) No change to Locking SP

Table 12 Example –Deassign Namespace Non-Global Range Locking object with KeepNamespaceGlobalRangeKey = TRUE does not work

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All **except** LBA[10..19]	K1
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS1	FALSE	LBA[10..19]	K5
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.14 Deassign Namespace Non-Global Range Locking object for NS1.[10..19]

Deassign (UID = LO3, KeepNamespaceGlobalRangeKey = FALSE) returns SUCCESS

Results:

a) LO3 is removed
b) K5 is eradicated
c) LBA[10..19] is returned to control of LO1:

Table 13 Example – After Deassign Namespace Non-Global Range Locking object for NS1.[10..19]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS2 NS4	Ignored	All	K2 K4
LO1	NS1	TRUE	All	K1
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.15 Deassign Namespace Global Range Locking object for NS1 with KeepNamespaceGlobalRangeKey = TRUE

Deassign (UID = LO1, KeepNamespaceGlobalRangeKey = TRUE) returns SUCCESS

Results:

a) LO1 is removed
b) K1 is transferred to control of Global Range Locking object
c) All LBAs of NS1 return to control of Global Range Locking object:

Table 14 Example – After Deassign Namespace Global Range Locking object for NS1 with KeepNamespaceGlobalRangeKey = TRUE

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.16 Assign Namespace Global Range Locking object for NS2

Assign (NSID = NS2, RangeStart = 0, RangeLength = 0) returns { LO1, TRUE, SUCCESS }

Results:

a) LO1 is added
b) Control of NS2 is transferred to LO1:

Table 15 Example – After Assign Namespace Global Range Locking object for Namespace 2

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS4	Ignored	All	K1 K4
LO1	NS2	TRUE	All	K2
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.17 Deassign Namespace Non-Global Range Locking object for Namespace 2 with KeepNamespaceGlobalRangeKey = FALSE

Deassign (UID = LO1, KeepNamespaceGlobalRangeKey = FALSE) returns SUCCESS

Results:

a) LO1 is removed
b) K2 is eradicated
c) K9 is created and associated with NS2 in place of K2
d) All LBAs in NS2 return to control of Global Range Locking object

Table 16 Example – After Deassign Namespace Non-Global Range Locking object for Namespace 2 with KeepNamespaceGlobalRangeKey = FALSE

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K9 K4
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.18 Use NVMe Namespace Management command to create NS5

Namespace Management (SEL = Create, NSID = NS5, NSZE, FLBS, DPS, NMIC) returns Successful Completion

Results:

a) NS5 is created and assigned to the Global Range Locking object
b) K10 is created

Table 17 Example – After Using NVMe Namespace Management command to create NS5

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4 NS5	Ignored	All	K1 K9 K4 K10
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.19 NVMe Namespace Management command to delete a namespace with an associated Locking object does not work

Namespace Management (SEL = Delete, NSID = NS3) returns Access Denied

Results:

a) No change to Locking SP

Table 18 Example – NVMe Namespace Management command to delete a namespace with an associated Locking object does not work

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4 NS5	Ignored	All	K1 K9 K4 K10
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.20 Use NVMe Namespace Management command to delete NS5

Namespace Management (SEL = Delete, NSID = NS5) returns Successful Completion

Results:

a) Because NS5 is assigned to the Global Range Locking object it can be deleted
b) K10 is eradicated

Table 19 Example – After Using NVMe Namespace Management command to delete NS5

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K9 K4
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K8

2.21 Use GenKey method for cryptographic erase of NS3.LBA[0..9]

LO6.GenKey () returns SUCCESS

Results:

a) K8 is eradicated
b) K11 is created

Table 20 Example – After Using GenKey method for cryptographic erase of NS3.LBA[0..9]

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K9 K4
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K11

2.22 Use NVMe Format NVM command for cryptographic erase of NS4

Format NVM (NSID = NS4, SES = Cryptographic Erase) returns Successful Completion

Results:

a) K4 is eradicated
b) K12 is created

Table 21 Example – After Using NVMe Format NVM command for cryptographic erase of NS4

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K9 K12
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO5	NS3	FALSE	LBA[20..29]	K7
LO6	NS3	FALSE	LBA[0..9]	K11

2.23 Use NVMe Format NVM command for cryptographic erase of NS3

Format NVM (NSID = NS3, SES = Cryptographic Erase) returns Successful Completion

Results:

a) K3, K7, and K11 are eradicated
b) K13, K14, and K15 are created

Table 22 Example – After Using NVMe Format NVM command for cryptographic erase of NS3

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K9 K12
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K13
LO5	NS3	FALSE	LBA[20..29]	K14
LO6	NS3	FALSE	LBA[0..9]	K15

2.24 Use GenKey method on the Global Range Locking Object

GlobalRange.GenKey () returns SUCCESS

Results:

a) K1, K9, and K12 are eradicated
b) K16, K17, and K18 are created

Table 23 Example – After Using GenKey method on the Global Range Locking Object

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K16 K17 K18
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K13

LO5	NS3	FALSE	LBA[20..29]	K14
LO6	NS3	FALSE	LBA[0..9]	K15

3 Locking SP Modes Examples

3.1 Overview

These examples illustrate the namespace assignments that map to the various Locking SP modes. Each example is independent of the prior example.

3.2 No namespaces have been created

The mode of the Locking SP is not defined when no namespaces exist in the NVM subsystem.

Table 24 Example – No namespaces exist

Locking Object	Namespace
Global	

3.3 Only one single unassigned namespace exists

The Locking SP is in the Global LO / Multiple NS mode when only one single unassigned namespace exists in the NVM subsystem before any manipulations.

Table 25 Example – Only one single unassigned namespace exists

Locking Object	Namespace
Global	NS1

3.4 Multiple unassigned namespaces exist

The Locking SP is in the Global LO / Multiple NS mode when multiple unassigned namespaces exist in the NVM subsystem before any manipulations.

Table 26 Example – Multiple unassigned namespaces exist

Locking Object	Namespace
Global	NS1 NS2 NS3

3.5 One single namespace with multiple locking ranges – case 1

The Locking SP is in the Multiple LO / Single NS mode when only one single namespace exists in the NVM subsystem with LBA ranges under the control of other locking objects by using the Set method (not by using the Assign method). Note that the NSID column value will be 0 for all locking objects in this case.

Table 27 Example – One single namespace with multiple locking ranges – case 1

Locking Object	Namespace
Global	NS1 Global
LO1	NS1 LBA Range 1
LO2	NS1 LBA Range 2
LO3	NS1 LBA Range 3

3.6 One single namespace with multiple locking ranges – case 2

The Locking SP is in the Multiple LO / Multiple NS mode when only one single namespace exists in the NVM subsystem with LBA ranges under the control of other locking objects by using the Assign method (not by using the Set method). Note that the global range for the namespace is under the control of a non-Global Range locking object in this case (it is under the control of a Namespace Global Range locking object). Note that the NSID column value will be 1 for LO1, LO2, and LO3 in this case.

Table 28 Example – One single namespace with multiple locking ranges – case 2

Locking Object	Namespace
Global	
LO1	NS1 LBA Global Range
LO2	NS1 LBA Range 1
LO3	NS1 LBA Range 2

3.7 Multiple namespaces, some assigned

The Locking SP is in the Multiple LO / Multiple NS mode when multiple namespaces exist in the NVM subsystem with some assigned to other locking objects.

Table 29 Example – Multiple namespaces, some assigned

Locking Object	Namespace
Global	NS1 NS3 NS5
LO1	NS2 LBA Global Range
LO2	NS4 LBA Global Range
LO3	NS6 LBA Global Range

3.8 Multiple namespaces, some assigned with multiple locking ranges

The Locking SP is in the Multiple LO / Multiple NS mode when multiple namespaces exist in the NVM subsystem with some assigned to other locking objects and some with multiple locking ranges.

Table 30 Example – Multiple namespaces, some assigned with multiple locking ranges

Locking Object	Namespace
Global	NS1 NS3 NS5
LO1	NS2 LBA Global Range
LO2	NS4 LBA Global Range
LO3	NS2 LBA Range 1
LO4	NS2 LBA Range 2
LO5	NS4 LBA Range 1

3.9 Example Locking SP Mode Selection Process at Power On

The following is an example Locking SP mode selection process at power on based on the state of the system.

- If NamespaceID == 0 in all Locking objects
  and RangeLength == 0 in all Locking objects
  - Locking SP is in Global LO / Multiple NS mode
  - Unused Key Count = Maximum Key Count – number of existing namespaces
- Else if NamespaceID == 0 in all Locking objects
  and RangeLength != 0 in at least one Locking object
  - Note that only a single namespace exists
    - All other possibilities at this point are invalid
  - Locking SP is in Multiple LO / Single NS mode
  - Number of configured Non-Global Range Locking objects = number of Non-Global Range Locking objects with RangeLength != 0
  - Unused Key Count = Maximum Key Count – (1 + number of configured Non-Global Range Locking objects)
- Else
  - Note that NamespaceID != 0 in at least one Locking object
  - Locking SP is in Multiple LO / Multiple NS mode
  - Number of configured Non-Global Range Locking objects = number of Non-Global Range Locking objects with NamespaceID != 0
  - Number of configured Namespace Non-Global Range Locking objects = number of configured Non-Global Range Locking objects with NamespaceGlobalRange = FALSE
    - Presumably RangeLength != 0 for these Locking objects but it is not a requirement
  - Unused Key Count = Maximum Key Count – (number of existing namespaces + number of configured Namespace Non-Global Range Locking objects)

4 Revert / Revert SP usage examples

4.1 Overview

The definitions of the table columns are the same as in section 2 for these examples.

4.2 Use Revert method

Assume the following starting state

Table 31 Example – State before Revert is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4
LO1	NS5	TRUE	All	K5
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS6	TRUE	All except LBA[10..19]	K6
LO4	NS6	FALSE	LBA[10..19]	K7
LO5	NS3	FALSE	LBA[20..29]	K8
LO6	NS3	FALSE	LBA[0..9]	K9

Unused Key Count = 6

After a Revert method is executed with success, the following is the final state.

Table 32 Example – State after Revert is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS3 NS4 NS5 NS6	Ignored (T)	All	K10 K11 K12 K13 K14 K15

Unused Key Count = 9

Note that the Revert method and the RevertSP method with KeepGlobalRangeKey = FALSE functionally have the same effect with regards to the Configurable Namespace Locking feature. See Table 33 and Table 34.

4.3 Use RevertSP method with KeepGlobalRangeKey = FALSE

Assume the following starting state

Table 33 Example – State before RevertSP (KeepGlobalRangeKey = FALSE) is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4
LO1	NS5	TRUE	All	K5
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS6	TRUE	All except LBA[10..19]	K6
LO4	NS6	FALSE	LBA[10..19]	K7
LO5	NS3	FALSE	LBA[20..29]	K8
LO6	NS3	FALSE	LBA[0..9]	K9

Unused Key Count = 6

After a RevertSP (KeepGlobalRangeKey = FALSE) method is executed with success, the following is the final state.

Table 34 Example – State after RevertSP (KeepGlobalRangeKey = FALSE) is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS3 NS4 NS5 NS6	Ignored (T)	All	K10 K11 K12 K13 K14 K15

Unused Key Count = 9

Note that the Revert method and the RevertSP method with KeepGlobalRangeKey = FALSE functionally have the same effect with regards to the Configurable Namespace Locking feature. See Table 33 and Table 34.

4.4 Use RevertSP method with KeepGlobalRangeKey = TRUE

Assume the following starting state

Table 35 Example – Initial state before RevertSP (KeepGlobalRangeKey = TRUE) is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4
LO1	NS5	TRUE	All	K5
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3
LO3	NS6	TRUE	All except LBA[10..19]	K6
LO4	NS6	FALSE	LBA[10..19]	K7
LO5	NS3	FALSE	LBA[20..29]	K8
LO6	NS3	FALSE	LBA[0..9]	K9

Unused Key Count = 6

After a RevertSP (KeepGlobalRangeKey = TRUE) method is executed with success, the following is the final state.

Table 36 Example – State after RevertSP (KeepGlobalRangeKey = TRUE) is executed

Locking Object	Namespace	NS Global Range	Range in NS	MEK
NA….	NS1 NS2 NS3 NS4 NS5 NS6	Ignored (T)	All	K1 K2 K10 K4 K11 K12

Unused Key Count = 9

Note: Global Range Locking object is inactive.

5 Single User Mode Feature Set Interaction Examples

5.1 Overview

Each example builds upon the previous example as in section 2. Also the definitions of the table columns are the same as in section 2. Note that this feature set causes changes to base Single User Mode behavior only when RangeStartLengthPolicy is 0.

5.2 Initial state

Assume the following starting state

Table 37 Example – Initial state for Single User Mode examples

Locking Object	Namespace	NS Global Range	Range in NS	MEK	Single User Mode
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4	N/A
LO1	NS5	TRUE	All	K5	N/A
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3	N/A
LO3	NS6	TRUE	All except LBA[10..19]	K6	N/A
LO4	NS6	FALSE	LBA[10..19]	K7	N/A
LO5	NS3	FALSE	LBA[20..29]	K8	N/A
LO6	NS3	FALSE	LBA[0..9]	K9	N/A

5.3 Reactivate Global Range Locking object into Single User Mode

Reactivate ([Global], RangeStartLengthPolicy = 0)

Table 38 Example – Initial state for Single User Mode examples

Locking Object	Namespace	NS Global Range	Range in NS	MEK	Single User Mode
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4	Yes
LO1	NS5	TRUE	All	K5	No
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3	No
LO3	NS6	TRUE	All except LBA[10..19]	K6	No
LO4	NS6	FALSE	LBA[10..19]	K7	No
LO5	NS3	FALSE	LBA[20..29]	K8	No
LO6	NS3	FALSE	LBA[0..9]	K9	No

5.4 Assign new Namespace Global Range Locking object does not work if Global Range Locking object is in Single User Mode

Assign (NSID = NS1) returns NOT_AUTHORIZED

Reason for failure: NS1 control cannot be moved the Global Range Locking object is in Single User Mode and currently controls NS1.

Results:

a) Locking SP is unchanged:

Note: RangeStartLengthPolicy being 0 causes this interaction (behavior change).

5.5 Deassign Namespace Global Range Locking object does not work if Global Range Locking object is in Single User Mode

Deassign (LO1) returns NOT_AUTHORIZED

Reason for failure: NS5 control cannot be moved to the Global Range Locking object since the Global Range Locking object is in Single User Mode.

Results:

a) Locking SP is unchanged:

Note: RangeStartLengthPolicy being 0 causes this interaction (behavior change).

5.6 Reactivate into Single User Mode for some Locking Objects

Reactivate ([LO3, LO5], RangeStartLengthPolicy = 0)

Before the Reactivate call: **ACE_Assign BooleanExpr = {Admins, User1, User2, User4, User6, User7}**

Table 39 Example – State after Reactivate completes

Locking Object	Namespace	NS Global Range	Range in NS	MEK	Single User Mode
Global	NS1 NS2 NS4	Ignored	All	K1 K2 K4	No
LO1	NS5	TRUE	All	K5	No
LO2	NS3	TRUE	All **except** LBA[0..9] and LBA[20..29]	K3	No
LO3	NS6	TRUE	All except LBA[10..19]	K6	Yes
LO4	NS6	FALSE	LBA[10..19]	K7	No
LO5	NS3	FALSE	LBA[20..29]	K8	Yes
LO6	NS3	FALSE	LBA[0..9]	K9	No

After the Reactivate call: ACE_Assign BooleanExpr = {Admins, User1, User2, User7}

5.7 Assign Namespace Non-Global Range Locking object does not work if Namespace Global Range Locking object is in Single User Mode

Assign (NSID = NS6, RangeStart = 30, RangeLength = 10) returns NOT_AUTHORIZED

Reason for failure: LO3 which controls NS6 is in Single User Mode and hence a new Namespace Non-Global Range Locking object cannot be created for NS6.

Results:

a) Locking SP is unchanged:

Note: RangeStartLengthPolicy being 0 causes this interaction (behavior change).

5.8 Deassign does not work if Locking object is in Single User Mode

Deassign (LO5) returns NOT_AUTHORIZED

Reason for failure: LO5 is in Single User Mode and hence cannot be deassigned.

Results:

a) Locking SP is unchanged:

Note: RangeStartLengthPolicy being 0 causes this interaction (behavior change).

5.9 Deassign Namespace Non-Global Range Locking object does not work if corresponding Namespace Global Range Locking object is in Single User Mode

Deassign (LO4) returns NOT_AUTHORIZED

Reason for failure: LO3 which controls the global range for NS6 is in Single User Mode and hence LO4 which is a Namespace Non-Global Range Locking object for NS6 cannot be deassigned.

Results:

a) Locking SP is unchanged:

Note: RangeStartLengthPolicy being 0 causes this interaction (behavior change).

5.10 Single User Mode Setup

The following is an example of setting up multiple ranges in Single User Mode. Namespace 1 is to be controlled by two Single User Mode authorities while Namespace 2 is to be left alone.

- Create Namespaces 1 and 2
- Assign Namespace 1 to LO1
- Assign Namespace 1 to LO2 without specifying a range
- Assign Namespace 2 to LO3
- Modify ACE to also allow User6 to call Assign
- Reactivate with LO1 and LO2 in Single User Mode and with RangeStartLengthPolicy = 0
- Authenticate as Single User Mode authority User3
- Set a non-zero LBA range for Namespace 1 in LO2

- Authenticate as User6
- Assign Namespace 2 non-zero LBA range to LO4
    - Namespace Global Range Locking Object for Namespace 2 (LO3) is not in Single User Mode and hence Assign is allowed specifying Namespace 2

5.11 Update to add additional Locking Object in Single User Mode Setup

The following example continues on from the above example. The goal is to have a third Single User Mode authority added to control Namespace 1. However the Namespace Global Range Locking Object for Namespace 1 (LO1) is in Single User Mode and hence Assign is not allowed specifying Namespace 1.

- Reactivate with no Single User Mode locking objects defined
- Assign Namespace 1 to LO5
- Reactivate with LO1, LO2, and LO5 in Single User Mode

