Interoperability Test Report

Report No.: EM4124 Tags
GSRN.: 950110126000001862

Product Name	:	EM4124 Tags
Product Model Number(s)	:	N/A
Client	:	EM Microelectronic-Marin SA
Test Methodology	:	Interoperability Test System for EPC Compliant Class-1 Generation-2 UHF RFID devices – Interoperability Test Methodology Proposal v1.2.5

TABLE OF CONTENTS

1. IDENTIFICATION SUMMARY

1.1.Test Laboratory

Name:	MET Laboratories, Inc.
Address	914 W. Patapsco Avenue
City:	Baltimore, MD
Postal code:	21230
Country:	U.S.A.
Telephone:	410-354-3300
Fax:	410-354-3313
URL:	www.metlabs.com
Contact person:	
Name:	Dusmantha Tennakoon
e-mail:	dtennakoon@metlabs.com

Table 1. Test Laboratory Information

Competences and guarantees:

Met Laboratories Inc. is a testing laboratory competent to carry out the tests described in this report.

MET Laboratories guarantees the reliability of the data presented in this report, which is the result of tests performed to the item under test on the date and under the conditions stated on the report and is based on the knowledge and technical facilities available at MET Laboratories at the time of execution of the test.

1.2. Client

Name:	EM Microelectronic-Marin SA
Address:	2074 Marin
City:	Rue des Sors,
Postal code:	N/A
Country:	Switzerland
Telephone:	+41 32 755 5111
Contact person:	
Name:	Nicolas Pillin
e-mail:	npillin@emmicroelectronic.com

Table 2. Client Information

1.3. Manufacturer

NAME:	EM Microelectronic-Marin SA
Address:	2074 Marin
City:	Rue des Sors,
Postal code:	N/A
Country:	Switzerland
Telephone:	+41 32 755 5111
Contact person:	
Name:	Nicolas Pillin
e-mail:	npillin@emmicroelectronic.com

Table 3. Manufacturer Information

1.4. Implementation Under Test

PRODUCT NAME:	EM4124 TAG
Certified IC used:	EM4124 IC
Hw version:	V200
IS and IXIT:	SEE ANNEX C
Description of IUT :	UHF RFID tag

Table 4. Implementation Under Test Information

1.5. Test Parameters for QE Interrogators Used for Verification:

Reader name/GSRN number		**Impinj Speedway RFID Reader 950110126000000070 (hardware #)**		
Hardware Rev		**010-000.027255**		
Serial Number		**00-06-02-00020**		
Frequency (MHz)	**Tari(us)**	**Modulation**	**Backscatter encoding**	**Data rate (Kbps)**
902 - 928	25	PR-ASK	Miller 4	64

Table 5. Test Parameter for a QE Interrogator 1 Used for Verification

Reader name/GSRN number		Sirit INfinity 510 RFID Reader 950110126000000247 (hardware #)		
Version/Rev		1.2.rc4-5368/A		
Serial Number		03416500BC4357BD		
Frequency (MHz)	**Tari(us)**	**Modulation**	**Backscatter encoding**	**Data rate (Kbps)**
902 - 928	6.25	DSB-ASK	Miller 4	42.75
902 – 928	12.5	PR-ASK	FMO	320
865.7-867.5	12.5	PR-ASK	Miller 8	40

Table 6. Test Parameters for a QE Interrogator 2 Used for Verification

1.6. Test conditions

NOMINAL

TEMPERATURE IN THE RANGE 18ºC TO 27 ºC	20 °C

Table 7. Nominal Temperature

1.7. Record of agreement

The following samples were used for testing.

METRAK NO.:	SERIAL NO.:	DATE OF RECEPTION:
36857	NA	11/05/2012

Table 8. Record of Agreement Information

1.8. Limits and reservations

The test results presented in this test report apply only to the particular Implementation Under Test (IUT) declared in section 1.4 of this report, for the functionality described in the relevant Implementation Statement (IS), as presented for test on the date(s) declared in section 1.7 and configured as declared in the relevant Implementation eXtra Information for Testing (IXIT).

This test report does not constitute or imply, by its own, to be an approval of the product by Qualification Bodies, Certification Bodies or competent Authorities.

This document is only valid if complete; no partial reproduction can be made without written approval of the Test Laboratory.

This test report cannot be used partially or in full for publicity and/or promotional purposes without previous written approval of the Test Laboratory.

2.0 Summary

2.1 QE Interoperability

Was Tag(s) Interoperable with QE Readers?	Yes
QE Readers used to verify the device:	Impinj Speedway RFID Reader and Sirit INfinity 510 RFID Reader

Table 9. QE Interoperability Information

2.2. Interoperability Test Case Checklists

2.2.1. Access Memory

Access_memory.txt	Pass?	N/A?	Test case	Lock	Action	AP State	AP
	Yes		R_AP_1		-	zero	Correct
	Yes		R_AP_2		-	zero	Incorrect
	Yes		R_AP_3		-	non-zero	Correct
	Yes		R_AP_4		-	non-zero	Incorrect
	Yes		R_AP_5		-	non-zero	None
	Yes		R_AP_L_1	L	-	zero	Correct
	Yes		R_AP_L_2	L	-	zero	Incorrect
	Yes		R_AP_L_3	L	-	non-zero	Correct
	Yes		R_AP_L_4	L	-	non-zero	Incorrect
	Yes		R_AP_L_5	L	-	non-zero	None
	Yes		W_AP_Z		zero	non-zero	Correct
	Yes		W_AP_NZ_1		non-zero	zero	Correct
	Yes		W_AP_NZ_2		non-zero	non-zero	Correct
	Yes		W_AP_NZ_3		non-zero	non-zero	Incorrect
	Yes		W_AP_NZ_4		non-zero	non-zero	None
	Yes		W_AP_L_NZ_1	L	non-zero	zero	Correct
	Yes		W_AP_L_NZ_2	L	non-zero	zero	Incorrect
	Yes		W_AP_L_NZ_3	L	non-zero	non-zero	Correct
	Yes		W_AP_L_NZ_4	L	non-zero	non-zero	Incorrect
	Yes		W_AP_L_NZ_5	L	non-zero	non-zero	None
	Yes		L_AP_1		-	zero	Correct

	Yes		L_AP_2		-	zero	Incorrect
	Yes		L_AP_3		-	non-zero	Correct
	Yes		L_AP_4		-	non-zero	Incorrect
	Yes		L_AP_5		-	non-zero	None
	Yes		U_AP_L_1	L	-	zero	Correct
	Yes		U_AP_L_2	L	-	zero	Incorrect
	Yes		U_AP_L_3	L	-	non-zero	Correct
	Yes		U_AP_L_4	L	-	non-zero	Incorrect

Table 10. Access Memory Test Cases

2.2.2. Inventory Single

Inventory_single.txt	Pass?	N/A?	Test case	Lock	Action
	Yes		I		Non-select inventory
	Yes		I_L	L	Non-select inventory
	Yes		SI_E		Select EPC complete
	Yes		SI_E_L	L	Select EPC partial
	Yes		SI_T		Select TID complete
	Yes		SI_T_L	L	Select TID partial
		Yes	SI_U		Select User complete
		Yes	SI_U_L	L	Select User partial

Table 11. Inverntory Single Test Cases

2.2.3. Write Read

Write_read.txt	Pass?	N/A?	Test case	Lock	Action
	Yes		R_KP		Kill password
	Yes		R_KP_L	L	Kill password
	Yes		W_KP_Z		Kill password zero
	Yes		W_KP_NZ		Kill password non-zero
	Yes		W_KP_L_NZ	L	Kill password non-zero
	Yes		L_KP		Kill password
	Yes		U_KP_L	L	Kill password
	Yes		K_Z		Kill zero password
	Yes		K_INZ		Kill incorrect non-zero password
	Yes		K_INZ_L	L	Kill incorrect non-zero password
	Yes		R_E_C		EPC complete
	Yes		R_E_P		EPC partial
	Yes		R_E_L_P	L	EPC partial
	Yes		W_E_C		EPC complete
	Yes		W_E_P		EPC partial
	Yes		W_E_L_P	L	EPC partial

	Yes		L_E		EPC memory
	Yes		U_E_L	L	EPC memory
	Yes		R_T_C		TID complete
	Yes		R_T_P		TID partial
	Yes		R_T_L_P	L	TID partial
	Yes		W_T_C		TID complete
	Yes		W_T_P		TID partial
	Yes		W_T_L_P	L	TID partial
	Yes		L_T		TID memory
	Yes		U_T_L	L	TID memory
		Yes	R_U_C		User complete
		Yes	R_U_P		User partial
		Yes	R_U_P_L	L	User partial
		Yes	W_U_C		User complete
		Yes	W_U_P		User partial
		Yes	W_U_L_P	L	User partial
		Yes	L_U		User memory
		Yes	U_U_L	L	User memory

Table 12. Write Read Test Cases

2.2.4. PermaUnlocked

PermaUnlocked.txt	Pass?	N/A?	Test case	Lock	Action	AP State	AP
	Yes		R_KP_PU	PU	Kill password		
	Yes		W_KP_PU_NZ	PU	Kill password non-zero		
	Yes		PU_KP		Kill password		
	Yes		U_KP_PU	PU	Kill password		
	Yes		L_KP_PU	PU	Kill password		
	Yes		PU_KP_PU	PU	Kill password		
	Yes		K_INZ_PU	PU	Kill incorrect non-zero password		
	Yes		R_E_PU_P	PU	EPC partial		
	Yes		W_E_PU_P	PU	EPC partial		
	Yes		PU_E		EPC memory		
	Yes		U_E_PU	PU	EPC memory		
	Yes		L_E_PU	PU	EPC memory		
	Yes		PL_E_PU	PU	EPC memory		
	Yes		PU_E_PU	PU	EPC memory, should fail		
	Yes		R_T_PU_P	PU	TID partial		
	Yes		W_T_PU_P	PU	TID partial		
	Yes		U_T_PU	PU	TID memory		
	Yes		L_T_PU	PU	TID memory		
	Yes		PL_T_PU	PU	TID memory		
	Yes		PU_T_PU	PU	TID memory		

		Yes	R_U_P_PU	PU	User partial		
		Yes	W_U_PU_P	PU	User partial		
		Yes	PU_U		User memory		
		Yes	U_U_PU	PU	User memory		
		Yes	L_U_PU	PU	User memory		
		Yes	PL_U_PU	PU	User memory		
	Yes		R_AP_PU_1	PU	-	zero	Correct
	Yes		R_AP_PU_2	PU	-	zero	Incorrect
	Yes		R_AP_PU_3	PU	-	non-zero	Correct
	Yes		R_AP_PU_4	PU	-	non-zero	Incorrect
	Yes		R_AP_PU_5	PU	-	non-zero	None
	Yes		W_AP_PU_Z	PU	zero	non-zero	Correct
	Yes		W_AP_PU_NZ_1	PU	non-zero	zero	Correct
	Yes		W_AP_PU_NZ_2	PU	non-zero	non-zero	Correct
	Yes		W_AP_PU_NZ_3	PU	non-zero	non-zero	Incorrect
	Yes		W_AP_PU_NZ_4	PU	non-zero	non-zero	None
	Yes		PU_AP		-	zero	Correct
	Yes		L_AP_PU_1	PU	-	zero	None
	Yes		L_AP_PU_3	PU	-	non-zero	None
	Yes		PU_AP_PU_1	PU	-	zero	None
	Yes		PU_AP_PU_2	PU	-	non-zero	Correct
	Yes		PU_AP_PU_3	PU	-	non-zero	None
	Yes		SI_E_PU	PU	Select EPC partial		
	Yes		SI_T_PU	PU	Select TID partial		
		Yes	SI_U_PU	PU	Select User		

					partial		
		Yes	PU_U_PU	PU	User memory		
	Yes		PU_T		TID memory		

Table 13. PermaUnlocked Test Cases

2.2.5. PermaLocked_L

PermaLocked_L.txt	Pass?	N/A?	Test case	Lock	Action	AP State	AP
	Yes		PL_AP_L_1	L	-	non-zero	Correct
	Yes		PL_AP_L_2	L	-	non-zero	None

Table 14. PermaLocked_L Test Cases

2.2.6. PermaLocked_APZ

PermaLocked_APZ.txt	Pass?	N/A?	Test case	Lock	Action	AP State	AP
	Yes		R_KP_PL_1	PL	Kill password; Access password zero		
	Yes		W_KP_PL_NZ_1	PL	Kill password non-zero; Access password zero		

	Yes		PL_KP		Kill password		
	Yes		PL_KP_L_1	L	Kill password; Access password zero		
	Yes		U_KP_PL_1	PL	Kill password; Access password zero		
	Yes		L_KP_PL_1	PL	Kill password; Access password zero		
	Yes		PU_KP_PL_1	PL	Kill password; Access password zero		
	Yes		K_INZ_PL	PL	Kill incorrect non-zero password		
	Yes		K_NZ		Kill non-zero password		
	Yes		R_E_PL_P_1	PL	EPC partial; Access password zero		

	Yes		W_E_PL_P_1	PL	EPC partial; Access password zero		
	Yes		PL_E_L_1	L	EPC memory; Access password zero		
	Yes		U_E_PL_1	PL	EPC memory; Access password zero		
	Yes		L_E_PL_1	PL	EPC memory; Access password zero		
	Yes		PU_E_PL_1	PL	EPC memory; Access password zero		
	Yes		R_T_PL_P_1	PL	TID partial; Access password zero		
	Yes		W_T_PL_P_1	PL	TID partial; Access password		

					zero		
	Yes		PL_T_L_1	L	TID memory; Access password zero		
	Yes		U_T_PL_1	PL	TID memory; Access password zero		
	Yes		L_T_PL_1	PL	TID memory; Access password zero		
	Yes		PU_T_PL_1	PL	TID memory; Access password zero		
		Yes	R_U_P_PL_1	PL	User partial; Access password zero		
		Yes	W_U_PL_P_1	PL	User partial; Access password zero		
		Yes	PL_U_L_1	L	User memory; Access password		

					zero		
		Yes	U_U_PL_1	PL	User memory; Access password zero		
		Yes	L_U_PL_1	PL	User memory; Access password zero		
		Yes	PU_U_PL_1	PL	User memory; Access password zero		
	Yes		PL_AP_L	L	-	zero	Correct
	Yes		U_AP_PL_1	PL	-	zero	Correct
	Yes		L_AP_PL_1	PL	-	zero	Correct
	Yes		PU_AP_PL_1	PL	-	zero	Correct
	Yes		I_PU	PU	Non-select inventory		
	Yes		I_PL	PL	Non-select inventory		
	Yes		SI_E_PL	PL	Select EPC partial		
	Yes		SI_T_PL	PL	Select TID partial		
		Yes	SI_U_PL	PL	Select User		

					partial		

Table 15. PermaLocked_APZ Test Cases

2.2.7. PermaLocked_APNZ

PermaLocked_APNZ.txt	Pass?	N/A?	Test case	Lock	Action	AP State	AP
	Yes		R_KP_PL_2	PL	Kill password; Access password non-zero		
	Yes		W_KP_PL_NZ_2	PL	Kill password non-zero; Access password non-zero		
	Yes		PL_KP_L_2	L	Kill password; Access password non-zero		
	Yes		U_KP_PL_2	PL	Kill password; Access password non-zero		
	Yes		L_KP_PL_2	PL	Kill password; Access password non-zero		
	Yes		PU_KP_PL_2	PL	Kill password; Access password non-zero		

	Yes		R_E_PL_P_2	PL	EPC partial; Access password non-zero		
	Yes		W_E_PL_P_2	PL	EPC partial; Access password non-zero		
	Yes		PL_E_L_2	L	EPC memory; Access password non-zero		
	Yes		U_E_PL_2	PL	EPC memory; Access password non-zero		
	Yes		L_E_PL_2	PL	EPC memory; Access password non-zero		
	Yes		PU_E_PL_2	PL	EPC memory; Access password non-zero		
	Yes		R_T_PL_P_2	PL	TID partial; Access password non-zero		
	Yes		W_T_PL_P_2	PL	TID partial; Access password non-		

					zero		
	Yes		PL_T_L_2	L	TID memory; Access password non-zero		
	Yes		U_T_PL_2	PL	TID memory; Access password non-zero		
	Yes		L_T_PL_2	PL	TID memory; Access password non-zero		
	Yes		PU_T_PL_2	PL	TID memory; Access password non-zero		
		Yes	R_U_P_PL_2	PL	User partial; Access password non-zero		
		Yes	W_U_PL_P_2	PL	User partial; Access password non-zero		
		Yes	PL_U_L_2	L	User memory; Access password non-zero		

		Yes	U_U_PL_2	PL	User memory; Access password non-zero		
		Yes	L_U_PL_2	PL	User memory; Access password non-zero		
		Yes	PU_U_PL_2	PL	User memory; Access password non-zero		
	Yes		R_AP_PL_1	PL	-	non-zero	Correct
	Yes		R_AP_PL_2	PL	-	non-zero	None
	Yes		W_AP_PL_NZ_1	PL	non-zero	non-zero	Correct
	Yes		W_AP_PL_NZ_2	PL	non-zero	non-zero	None
	Yes		L_AP_PU_2	PU	-	non-zero	Correct
	Yes		U_AP_PL_2	PL	-	non-zero	Correct
	Yes		U_AP_PL_3	PL	-	non-zero	None
	Yes		L_AP_PL_2	PL	-	non-zero	Correct
	Yes		L_AP_PL_3	PL	-	non-zero	None
	Yes		PU_AP_PL_2	PL	-	non-zero	Correct
	Yes		PU_AP_PL_3	PL	-	non-zero	None

Table 16. PermaLocked_APNZ Test Cases

2.2.8. Inventory Multiple

Inventory_multiple.txt	Pass?	N/A?	Test case	Action
	Yes		I_MH	Non-select inventory homogeneous
	Yes		SI_MH_E	Select EPC complete homogeneous
	Yes		SI_MH_T	Select TID complete homogeneous
		Yes	SI_MH_U	Select User complete homogeneous
	Yes		I_MM	Non-select inventory mixed
	Yes		SI_MM_E	Select EPC complete mixed
	Yes		SI_MM_T	Select TID complete mixed
		Yes	SI_MM_U	Select User complete mixed

Table 17. Inventory MultipleTest Cases

2.2.9. Select/Query EPC Memory

sq_epc_test.txt	Pass?	N/A?	Test case	Target	Action	Pointer	Length	Truncate	Lock	Memory
	Fail*		SQ_E_S0_1	S0	0	32	96	0		EPC
	Fail*		SQ_E_S0_2	S0	1	34	64	0		EPC
	Fail*		SQ_E_S0_3	S0	10	37	48	0		EPC
	Fail*		SQ_E_S0_4	S0	11	43	24	0		EPC
	Fail*		SQ_E_S0_5	S0	100	55	12	0		EPC
	Fail*		SQ_E_S0_6	S0	101	79	6	0		EPC
	Fail*		SQ_E_S0_7	S0	110	95	3	0		EPC
	Fail*		SQ_E_S0_8	S0	111	127	1	0		EPC
	Yes		SQ_E_S1_1	S1	0	117	1	0	L	EPC
	Yes		SQ_E_S1_2	S1	1	97	3	0	L	EPC
	Yes		SQ_E_S1_3	S1	10	83	6	0	L	EPC
	Yes		SQ_E_S1_4	S1	11	64	12	0	L	EPC
	Yes		SQ_E_S1_5	S1	100	46	24	0	L	EPC
	Yes		SQ_E_S1_6	S1	101	42	48	0	L	EPC
	Yes		SQ_E_S1_7	S1	110	38	64	0	L	EPC
	Yes		SQ_E_S1_8	S1	111	32	96	0	L	EPC
	Yes		SQ_E_S2_1	S2	0	32	96	0		EPC
	Yes		SQ_E_S2_2	S2	1	32	95	0		EPC
	Yes		SQ_E_S2_3	S2	10	44	48	0		EPC
	Yes		SQ_E_S2_4	S2	11	56	30	0		EPC
	Yes		SQ_E_S2_5	S2	100	67	20	0		EPC
	Yes		SQ_E_S2_6	S2	101	75	6	0		EPC
	Yes		SQ_E_S2_7	S2	110	127	0	0		EPC
	Yes		SQ_E_S2_8	S2	111	126	1	0		EPC
	Yes		SQ_E_S3_1	S3	0	124	1	0	L	EPC

	Yes		SQ_E_S3_2	S3	1	102	3	0	L	EPC
	Yes		SQ_E_S3_3	S3	10	87	6	0	L	EPC
	Yes		SQ_E_S3_4	S3	11	59	12	0	L	EPC
	Yes		SQ_E_S3_5	S3	100	45	24	0	L	EPC
	Yes		SQ_E_S3_6	S3	101	41	48	0	L	EPC
	Yes		SQ_E_S3_7	S3	110	36	64	0	L	EPC
	Yes		SQ_E_S3_8	S3	111	32	96	0	L	EPC
	Yes		SQ_E_SL_1	SL	0	127	2	1		EPC
	Yes		SQ_E_SL_2	SL	1	63	64	1		EPC
	Yes		SQ_E_SL_3	SL	10	127	1	1		EPC
	Yes		SQ_E_SL_4	SL	11	63	0	1		EPC
	Yes		SQ_E_SL_5	SL	100	58	12	1		EPC
	Yes		SQ_E_SL_6	SL	101	81	16	1		EPC
	Yes		SQ_E_SL_7	SL	110	32	0	1		EPC
	Yes		SQ_E_SL_8	SL	111	124	3	1		EPC
	Yes		SQ_E_SL_9	SL	0	32	96	1	L	EPC
	Yes		SQ_E_SL_10	SL	1	32	95	1	L	EPC
	Yes		SQ_E_SL_11	SL	10	44	48	1	L	EPC
	Yes		SQ_E_SL_12	SL	11	56	30	1	L	EPC
	Yes		SQ_E_SL_13	SL	100	67	20	1	L	EPC
	Yes		SQ_E_SL_14	SL	101	75	6	1	L	EPC
	Yes		SQ_E_SL_15	SL	110	127	0	1	L	EPC
	Yes		SQ_E_SL_16	SL	111	126	1	1	L	EPC

Table 18. Select/Query EPC Test Cases

2.2.10 Select/Query TID Memory

sq_tid_test.txt	Pass?	N/A?	Test case	Target	Action	Pointer	Length	Truncate	Lock	Memory
	Yes		SQ_T_S0_1	S0	0	32	Full	0		TID
	Yes		SQ_T_S0_2	S0	1	32	Half	0		TID
	Yes		SQ_T_S1_1	S1	10	32	Full	0	L	TID
	Yes		SQ_T_S1_2	S1	11	32	Half	0	L	TID
	Yes		SQ_T_S2_1	S2	100	32	Full	0		TID
	Yes		SQ_T_S2_2	S2	101	32	Half	0		TID
	Yes		SQ_T_S3_1	S3	110	32	Full	0	L	TID
	Yes		SQ_T_S3_2	S3	111	32	Half	0	L	TID
	Yes		SQ_T_SL_1	SL	0	32	Full	0		TID
	Yes		SQ_T_SL_2	SL	1	32	Half	0		TID

Table. 19 Select/Query TID Test Cases

2.2.11 Select/Query User Memory

sq_user_test.txt	Pass?	N/A?	Test case	Target	Action	Pointer	Length	Truncate	Lock	Memory
		Yes	SQ_U_S0_1	S0	0	32	Full	0		User
		Yes	SQ_U_S0_2	S0	1	32	Half	0		User
		Yes	SQ_U_S1_1	S1	10	32	Full	0	L	User
		Yes	SQ_U_S1_2	S1	11	32	Half	0	L	User
		Yes	SQ_U_S2_1	S2	100	32	Full	0		User
		Yes	SQ_U_S2_2	S2	101	32	Half	0		User
		Yes	SQ_U_S3_1	S3	110	32	Full	0		User
		Yes	SQ_U_S3_2	S3	111	32	Half	0		User
		Yes	SQ_U_SL_1	SL	0	32	Full	0	L	User
		Yes	SQ_U_SL_2	SL	1	32	Half	0	L	User

Table. 20 Select/Query User Test Cases

Annex A: Test Log Files

1. Access_Memory (A_M.out.txt)

```
# This section initializes the tag
#
Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.
Write pass 00000000 kpass
   Write result: 1
   Write SUCCESSFUL.
# End of initialize tag

# === Access Password Section ===
# Access password only matters if it non-zero and the memory is locked.
#
# Tag is initially set with all zero AP and KP and unlocked memory
# AP = access password; U/L is unlocked/locked memory; TC = Test Case

# Running the unlocked Access Commands here

Read pass 00000000 apass
   Read: 00000000
   Read SUCCESSFUL.
# AP = 00000000, U; Verifies AP is all zeros; TC = R_AP_1
# This TC reads the correct AP when the AP is zero.

Read fail 00000000 apass EEEEEEEE
   Read: FAILED
   Read SUCCESSFUL.
# AP = 00000000, U; AP read w/ incorrect AP; TC = R_AP_2
# This TC reads the correct AP when the AP is zero, using an incorrect AP.

Lock fail apass DDDDDDDD
   Lock result: 0
   Lock SUCCESSFUL.
# AP = 00000000, U; Attempted lock w/ incorrect AP; TC = L_AP_2
# This TC attempts to lock the AP in the zero state using an incorrect AP.

WrRd pass 11111111 apass
   WrRd write result: 1
   WrRd read result: 11111111
```

WrRd SUCCESSFUL.
AP = 11111111, U; Set initial AP; TC = W_AP_NZ_1, R_AP_3
This TC writes and then reads a non-zero AP, using a correct all-zero AP.

Read pass 11111111 apass
Read: 11111111
Read SUCCESSFUL.
AP = 11111111, U; Read AP from open state; TC = R_AP_5
This TC attempts to read a non-zero AP using no AP.

Lock fail apass
Lock result: 0
Lock SUCCESSFUL.
AP = 11111111, U; Attempted Lock w/ nonzero AP; TC = L_AP_5
This TC attempts to lock the AP from the open state, when the AP is non-zero and no AP is given.

Write pass 22222222 apass 11111111
Write result: 1
Write SUCCESSFUL.
Read pass 22222222 apass 22222222
Read: 22222222
Read SUCCESSFUL.
AP = 22222222, U; Verifies AP change after incorrect AP operation; TC = W_AP_NZ_2
This TC writes and then reads a non-zero AP, using another correct non-zero AP.

Write pass 33333333 apass
Write result: 1
Write SUCCESSFUL.
AP = 33333333, U; Verifies AP change in open state; TC = W_AP_NZ_4
This TC writes a non-zero AP, using no AP, when the AP is non-zero.

Read fail 44444444 apass 44444444
Read: FAILED
Read SUCCESSFUL.
AP = 33333333, U; Attempted Read of AP w/ incorrect AP; TC = R_AP_4
This TC reads the AP using an incorrect non-zero AP.

Write fail 99999999 apass FFFFFFF
Write result: 0
Write SUCCESSFUL.
Read pass 33333333 apass
Read: 33333333
Read SUCCESSFUL.
AP = 33333333, U; Attempted AP change w/ incorrect AP; TC = W_AP_NZ_3

This TC attempts to write a non-zero AP, using an incorrect non-zero AP. It reads the correct AP from the open state.

Write pass 00000000 apass 33333333
Write result: 1
Write SUCCESSFUL.
Read pass 00000000 apass 00000000
Read: 00000000
Read SUCCESSFUL.
AP = 00000000, U; AP changed to all zeros; TC = W_AP_Z
This TC writes a zero AP, using a correct non-zero AP. It then reads that it is correct.

Running the Locked Access Commands here

Lock pass apass
Lock result: 1
Lock SUCCESSFUL.
AP = 00000000, L; Verifies Lock operation; TC = L_AP_1
This TC locks the AP from the secured state, when the AP is zero.
Read pass 00000000 apass
Read: 00000000
Read SUCCESSFUL.
AP = 00000000, L; Verifiying all zero AP; TC = R_AP_L_1
This TC reads the locked AP when the AP is zero and correct.

Read fail 00000000 apass CCCCCCCC
Read: FAILED
Read SUCCESSFUL.
AP = 00000000, L; Attempted Read w/ incorrect AP; TC = R_AP_L_2
This TC attempts to read the locked AP, when the AP is zero, using an incorrect AP.

Write fail 44444444 apass BBBBBBBB
Write result: 0
Write SUCCESSFUL.
Read pass 00000000 apass
Read: 00000000
Read SUCCESSFUL.
AP = 00000000, L; Attempted Write w/ incorrect AP; TC = W_AP_L_NZ_2
This TC attempts to write a non-zero AP, when the AP is locked and zero, using an incorrect AP.
The AP is read as unchanged.

Write pass 11111111 apass 00000000
Write result: 1
Write SUCCESSFUL.
Read pass 11111111 apass 11111111

Read: 11111111
Read SUCCESSFUL.
AP = 11111111, L; Change AP in Locked State; TC = W_AP_L_NZ_1, R_AP_L_3
This TC writes a non-zero AP, when the AP is locked and zero, using the correct AP.
The AP is read as changed.

Read fail 11111111 apass
Read: FAILED
Read SUCCESSFUL.
AP = 11111111, L; Attempting read w/o AP (open state read); TC = R_AP_L_5
This TC attempts to read the correct AP, when the AP is locked and non-zero, using no AP.
This TC fails because the read can't be done from the open state.

Read fail 33333333 apass AAAAAAAA
Read: FAILED
Read SUCCESSFUL.
AP = 11111111, L; Attempted Read w/ incorrect AP; TC = R_AP_L_4
This TC reads the incorrect AP, when the AP is locked and non-zero, using an incorrect AP.

Write pass 22222222 apass 11111111
Write result: 1
Write SUCCESSFUL.
Read pass 22222222 apass 22222222
Read: 22222222
Read SUCCESSFUL.
AP = 22222222, L; Verifies AP change in locked state; TC = W_AP_L_NZ_3
This TC writes to the AP, when the AP is locked and non-zero, using the correct AP.

Write fail 66666666 apass 55555555
Write result: 0
Write SUCCESSFUL.
Read pass 22222222 apass 22222222
Read: 22222222
Read SUCCESSFUL.
AP = 22222222, L; Attempted Write w/ incorrect AP; TC = W_AP_L_NZ_4
This TC attempts to write to the AP, when the AP is locked and non-zero, using an incorrect AP.

Write fail 66666666 apass
Write result: 0
Write SUCCESSFUL.
Read pass 22222222 apass 22222222
Read: 22222222
Read SUCCESSFUL.

AP = 22222222, L; Attempted Write to locked memory in open state; TC = W_AP_L_NZ_5
This TC attempts to write to the AP, when the AP is locked and non-zero, using no AP.

#
Run the Unlock Commands here

Unlock fail apass AAAAAAAA
   UnLock result: 0
   UnLock SUCCESSFUL.
AP = 22222222, L; Attempted Unlock w/ incorrect AP; TC = U_AP_L_4
Write fail 33333333 apass
   Write result: 0
   Write SUCCESSFUL.
AP = 22222222, L; Verifies cannot write from open state
Read fail 22222222 apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 22222222, L; Verifies cannot read from open state
Read pass 22222222 apass 22222222
   Read: 22222222
   Read SUCCESSFUL.
AP = 22222222, L; Verifes read from secured state
This TC attempts to unlock the AP, when the AP is locked and non-zero, using an incorrect AP.

Unlock pass apass 22222222
   UnLock result: 1
   UnLock SUCCESSFUL.
AP = 22222222, U; Successful Unlock w/ correct AP; TC = U_AP_L_3
WrRd pass 33333333 apass
   WrRd write result: 1
   WrRd read result: 33333333
   WrRd SUCCESSFUL.
AP = 33333333, U; Verifies can write and read from open state
Write pass 44444444 apass 33333333
   Write result: 1
   Write SUCCESSFUL.
Read pass 44444444 apass 44444444
   Read: 44444444
   Read SUCCESSFUL.
AP = 44444444, U; Verifies can write and read from secured state
This TC unlocks the AP, when the AP is formerly locked and non-zero, using the correct AP.
It also verifies the AP is unlocked.

Lock pass apass 44444444
   Lock result: 1
   Lock SUCCESSFUL.
AP = 44444444, L; Verifies lock from secured state; TC = L_AP_3
Write fail 33333333 apass
   Write result: 0
   Write SUCCESSFUL.
AP = 44444444, L; Verifies cannot write from open state
Read fail 44444444 apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 44444444, L; Verifies cannot read from open state
Read pass 44444444 apass 44444444
   Read: 44444444
   Read SUCCESSFUL.
AP = 44444444, L; Verifes read from secured state
This TC locks the AP, when the AP is unlocked and non-zero, using the correct AP.
Both open and secured states are verified.

Unlock pass apass 44444444
   UnLock result: 1
   UnLock SUCCESSFUL.
AP = 44444444, U; Unlocks AP for the next test.

Lock fail apass 42424242
   Lock result: 0
   Lock SUCCESSFUL.
AP = 44444444, U; Verifies lock with incorrect password; TC = L_AP_4
WrRd pass 33333333 apass
   WrRd write result: 1
   WrRd read result: 33333333
   WrRd SUCCESSFUL.
AP = 33333333, U; Verifies can write and read from open state
Write pass 44444444 apass 33333333
   Write result: 1
   Write SUCCESSFUL.
Read pass 44444444 apass 44444444
   Read: 44444444
   Read SUCCESSFUL.
AP = 44444444, U; Verifies can write and read from secured state
This TC attempts to lock the AP, when the AP is unlocked and non-zero, using an incorrect AP.

Write pass 00000000 apass 44444444
   Write result: 1

```
   Write SUCCESSFUL.
# AP = 00000000, U; Reset AP to all zeros
Read pass 00000000 apass
   Read: 00000000
   Read SUCCESSFUL.
# AP = 00000000, U; Verifies AP is all zeros
Lock pass apass
   Lock result: 1
   Lock SUCCESSFUL.
# AP = 00000000, L; Locks AP

Unlock pass apass 00000000
   UnLock result: 1
   UnLock SUCCESSFUL.
# AP = 00000000, U; Locks AP to all zeros; TC = U_AP_L_1
WrRd pass 33333333 apass
   WrRd write result: 1
   WrRd read result: 33333333
   WrRd SUCCESSFUL.
# AP = 33333333, U; Verifies can write and read from open state
Write pass 44444444 apass 33333333
   Write result: 1
   Write SUCCESSFUL.
Read pass 44444444 apass 44444444
   Read: 44444444
   Read SUCCESSFUL.
# AP = 44444444, U; Verifies can write and read from secured state
# This TC unlocks the AP, when the AP is locked and zero, using the correct zero AP.

Write pass 00000000 apass 44444444
   Write result: 1
   Write SUCCESSFUL.
# AP = 00000000, U; Reset AP to all zeros
Read pass 00000000 apass
   Read: 00000000
   Read SUCCESSFUL.
# AP = 00000000, U; Verifies AP all zeros
Lock pass apass
   Lock result: 1
   Lock SUCCESSFUL.
# AP = 00000000, L; Locks the AP

Unlock fail apass BBBBBBBB
   UnLock result: 0
   UnLock SUCCESSFUL.
# AP = 00000000, L; Unsuccessful unlock w/ incorrect AP; TC = U_AP_L_2
```

Write pass 11111111 apass 00000000
Write result: 1
Write SUCCESSFUL.
AP = 11111111, L; Verifies Write from secured state (dropped into secured state because AP is all zeros)
Read pass 11111111 apass 11111111
Read: 11111111
Read SUCCESSFUL.
AP = 11111111, L; Verifes read from secured state
This TC attempts to unlock the AP, when the AP is locked and zero, using an incorrect AP.

At the end of the script, the AP should be locked and the AP should be 0x11111111.

disconnect
Disconnected.

OVERALL RESULT: SUCCESS

2. Inventory_Single (I_S.out.txt)

```
# This section initializes the tag
WrRd pass 3000300833B2DDD9014035050000 epc
   WrRd write result: 1
   WrRd read result: 3000300833B2DDD9014035050000
   WrRd SUCCESSFUL.
# WrRd pass E200B080 tid
#
# Initialize TID if it is not Permalocked
#
Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.
#
# WrRd pass 0000111122223333 user
# Initialize user memory if available on chip
# Requires tag vendor customization as this feature is optional in Gen2
# This is an example syntax - total memory varies by vendor
#
# End of initialize tag

# === Inventory Section ===
#
# Access password only matters if it non-zero and the memory is locked.
#
# Tag is initially set with all zero AP and KP and unlocked memory
# EPC = 3000300833B2DDD9014035050000
# AP = access password; U/L is unlocked/locked memory; TC = Test Case
#
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
# TID = E2001050; Verify TID for tags w/ Permalock TID

# Now do inventory cases with unlocked memory locations

Inventory pass 1
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
# EPC = 3000300833B2DDD9014035050000, U; AP = 00000000; TC = I
Read pass 3000300833B2DDD9014035050000 epc
   Read: 3000300833B2DDD9014035050000
   Read SUCCESSFUL.
# EPC = 3000300833B2DDD9014035050000, U; AP = 00000000; Verifies EPC
Inventory pass 1 300833B2DDD9014035050000 epc
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
# This TC does a non-select inventory of the tag and reads its EPC number.

# EPC = 3000300833B2DDD9014035050000, U; AP = 00000000; TC = SI_E
Read pass 3000300833B2DDD9014035050000 epc
   Read: 3000300833B2DDD9014035050000
   Read SUCCESSFUL.
```

```
# EPC = 3000300833B2DDD9014035050000, U; AP = 00000000; Verifies EPC
# This TC does a complete selection of the single tag's EPC number.

Inventory pass 1 E200B080 tid
   Using TID E200B080
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
# TID = E2001050, U; AP = 00000000; TC = SI_T
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
# TID = E2001050, U; AP = 00000000; TC = SI_T; Verifies TID
# This TC does a complete selection of the single tag's TID number.

# This section is for tags with user memory
# Currently this functionality has not been verified
# Remove the comment lines on Inventory and Read directives to test the functionality
# This is an example syntax - total memory varies by vendor
#
#XXX Inventory pass 1 0000111122223333 user
# USER = 0000111122223333, U; AP = 00000000; TC = SI_U
#XXX Read pass 0000111122223333 user
# USER = 0000111122223333, U; AP = 00000000; Verifies EPC
# This TC does a complete selection of the single tag's user data.

# Now perform Inventory cases with memory locked

Write pass 11111111 apass
   Write result: 1
   Write SUCCESSFUL.
Lock pass epc 11111111
   Lock result: 1
   Lock SUCCESSFUL.
# EPC = 300833B2DDD9014035050000, L; AP = 11111111
Write fail 3000000011112222333344445555 epc
   Write result: 0
   Write SUCCESSFUL.
# Verified the EPC memory is locked and not writable
Lock pass tid 11111111
   Lock result: 1
   Lock SUCCESSFUL.
# TID = E2001050, L; AP = 11111111;
Write fail F3112161 tid
   Write result: 0
   Write SUCCESSFUL.
# Verifies the TID is locked or Permalocked
#
# This section is for tags with user memory
# Currently this functionality has not been verified
# Remove the comment lines on Inventory and Read directives to test the functionality
# This is an example syntax - total memory varies by vendor
#
#XXX Lock pass user 11111111
# USER = 0000111122223333, L; AP = 11111111;
```

#XXX Write fail FFFFFFFFFFFFFFFF user

Inventory pass 1
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; TC = I_L
Read pass 3000300833B2DDD9014035050000 epc 11111111
   Read: 3000300833B2DDD9014035050000
   Read SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Verifies EPC
This TC inventories the single tag while the AP, epc, and TID are locked/permalocked.

Inventory pass 1 3008 epc
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; TC = SI_E_L
Read pass 3000300833B2DDD9014035050000 epc 11111111
   Read: 3000300833B2DDD9014035050000
   Read SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Verifies EPC
This TC selects a portion of the EPC for a single tag while the AP, epc, and TID are locked/permalocked.

Inventory pass 1 E200 tid
   Using TID E200
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
TID = E2001050, L; AP = 00000000; TC = SI_T_L
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
TID = E2001050, L; AP = 00000000; Verifies TID
This TC selects a portion of the TID for a single tag while the AP, epc, and TID are locked/permalocked.

This section is for tags with user memory
Currently this functionality has not been verified
Remove the comment lines on Inventory and Read directives to test the functionality
#
Inventory pass 1 00001111 user
USER = 0000111122223333, L; AP = 00000000; TC = SI_U_L
Read pass 0000111122223333 user,0,4
USER = 0000111122223333, L; AP = 00000000; Verifies user memory

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS

3. Write_Read (W_R.out.txt)

```
# This section initializes the tag
WrRd pass 3000300833B2DDD9014035050000 epc
   WrRd write result: 1
   WrRd read result: 3000300833B2DDD9014035050000
   WrRd SUCCESSFUL.
#
#TTT WrRd pass E200B080 tid
# Initialize TID if it is not Permalocked
#
Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.
#
# WrRd pass 0000111122223333 user
# Initialize user memory if available on chip
# Requires tag vendor customization as this feature is optional in Gen2
# This is an example syntax - total memory varies by vendor
#
# End of initialize tag

# === Write and Read Test Section ===
# ===  Section tests EPC memory   ===
#
# Access password only matters if it non-zero and the memory is locked.
#
# Tag is initially set with all zero AP and KP and unlocked memory
# EPC = 3000300833B2DDD9014035050000
# AP = access password; U/L is unlocked/locked memory; TC = Test Case
#
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
# TID = E2001050; Verify TID for tags w/ Permalock TID

WrRd pass 3000300833B2DDD9014035050000 epc
   WrRd write result: 1
   WrRd read result: 3000300833B2DDD9014035050000
   WrRd SUCCESSFUL.
# EPC = 3000300833B2DDD9014035050000, U; TC = W_E_C, R_E_C
# These TCs write and then read the complete EPC number.

WrRd pass 3009 epc,2
   WrRd write result: 1
   WrRd read result: 3009
   WrRd SUCCESSFUL.
# EPC = 3000300933B2DDD9014035050000, U; TC = W_E_P, R_E_P
# These TCs write and then read a portion of the EPC number.

# Now run test cases for locked EPC

Write pass 11111111 apass
   Write result: 1
```

Write SUCCESSFUL.
EPC = 3000300933B2DDD9014035050000, U; AP = 11111111; Sets AP
Lock pass epc 11111111
Lock result: 1
Lock SUCCESSFUL.
EPC = 3000300933B2DDD9014035050000, L; AP = 11111111; TC = L_E
This TC locks the epc number.

Write fail 3000300833B2DDD9014035050000 epc
Write result: 0
Write SUCCESSFUL.
EPC = 3000300933B2DDD9014035050000, L; AP = 11111111; Attempted Write from open state;
Write fail 3000300833B2DDD9014035050000 epc FFFFFFFF
Write result: 0
Write SUCCESSFUL.
EPC = 3000300933B2DDD9014035050000, L; AP = 11111111; Attempted Write incorrect AP;
WrRd pass 3008 epc,2 11111111
WrRd write result: 1
WrRd read result: 3008
WrRd SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; TC = W_E_L_P and R_E_L_P;
This TC writes and then reads a portion of the EPC from the secured state, while the EPC memory is locked.

Now run test cases to unlock EPC

Unlock fail epc
UnLock result: 0
UnLock SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Unlock w/o AP
Write fail 3000300933B2DDD9014035050000 epc
Write result: 0
Write SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Write from open state;
Write fail 3000300933B2DDD9014035050000 epc FFFFFFFF
Write result: 0
Write SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Write incorrect AP;

Unlock fail epc FFFFFFFF
UnLock result: 0
UnLock SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Unlock w/ incorrect AP
Write fail 3000300933B2DDD9014035050000 epc
Write result: 0
Write SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Write from open state;
Write fail 3000300933B2DDD9014035050000 epc FFFFFFFF
Write result: 0
Write SUCCESSFUL.
EPC = 3000300833B2DDD9014035050000, L; AP = 11111111; Attempted Write incorrect AP;

Unlock pass epc 11111111
UnLock result: 1
UnLock SUCCESSFUL.

```
# EPC = 3000300833B2DDD9014035050000, U; AP = 11111111; TC = U_E_L
WrRd pass 3009 epc,2 11111111
   WrRd write result: 1
   WrRd read result: 3009
   WrRd SUCCESSFUL.
# EPC = 3000300933B2DDD9014035050000, U; AP = 11111111; Write from secured state;
WrRd pass 3008 epc,2
   WrRd write result: 1
   WrRd read result: 3008
   WrRd SUCCESSFUL.
# EPC = 3000300833B2DDD9014035050000, U; AP = 11111111; Write from open state;
# This TC unlocks the EPC memory from the locked state.

WrRd pass 00000000 apass
   WrRd write result: 1
   WrRd read result: 00000000
   WrRd SUCCESSFUL.
# EPC = 3000300833B2DDD9014035050000, U; AP = 00000000; Resets AP


# === Write and Read Test Section ===
# ===  Section tests TID memory   ===
#
# Access password only matters if it non-zero and the memory is locked.
#
# Tag is initially set with all zero AP and KP and unlocked memory
# Assumes a 32-bit programmable TID
# TID = E2001050
# AP = access password; U/L is unlocked/locked memory; TC = Test Case
#
# NOTE: This section needs to be modified for each tag with a different TID
#
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
# TID = E2001050; Verify TID for tags w/ Permalock TID

#TTT WrRd pass E200B080 tid
# TID = E2001050, U; TC = W_T_C, R_T_C
# Not valid for permalocked TID
# This TC writes and then reads the complete TID.

#TTT WrRd pass 2222 tid,2
# TID = E2002222, U; TC = W_T_P, R_T_P
# Not valid for permalocked TID
# This TC writes and then reads a portion of the TID.

# Now run test cases for locked TID

Write pass 11111111 apass
   Write result: 1
   Write SUCCESSFUL.
# TID = E2001050, U; AP = 11111111; Sets AP
Lock pass tid 11111111
```

```
   Lock result: 1
   Lock SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; TC = L_T
Write fail E200B080 tid
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; AP = 11111111; Attempted Write from open state;
Write fail E200B080 tid FFFFFFFF
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Write incorrect AP;
# This TC locks the TID memory.

#TTT WrRd pass 3333 tid,2 11111111
# TID = E2003333, L; AP = 11111111; TC = W_T_L_P and R_T_L_P;
# Not valid for permalocked TID
# This TC writes and then reads a portion of the TID while the TID is locked.

# Now run test cases to unlock TID

Unlock fail tid
   UnLock result: 0
   UnLock SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Unlock w/o AP
Write fail E2004444 tid
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Write from open state;
Write fail E2004444 tid FFFFFFFF
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Write incorrect AP;

Unlock fail tid FFFFFFFF
   UnLock result: 0
   UnLock SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Unlock w/ incorrect AP
Write fail E2004444 tid
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Write from open state;
Write fail E2004444 tid FFFFFFFF
   Write result: 0
   Write SUCCESSFUL.
# TID = E2001050, L; AP = 11111111; Attempted Write incorrect AP;

## TID is Permalocked
Unlock fail tid 11111111
   UnLock result: 0
   UnLock SUCCESSFUL.
# TID = E2001050, U; AP = 11111111; TC = U_T_L
#WrRd pass 1111 tid,2 11111111
# TID = E2001111, U; AP = 11111111; Write from secured state;
#WrRd pass 0000 tid,2
# TID = E2000000, U; AP = 11111111; Write from open state;
```

This TC attempts to unlock the TID from the locked/permalocked state.
It fails because the TID is permalocked, and therefore the tag doesn't allow the reader to deselect its Lock/Action bits.

WrRd pass 00000000 apass
   WrRd write result: 1
   WrRd read result: 00000000
   WrRd SUCCESSFUL.
TID = E2001050, U; AP = 00000000; Resets AP

=== Write and Read Test Section ===
=== Section tests USER memory   ===
#
Access password only matters if it non-zero and the memory is locked.
#
Tag is initially set with all zero AP and KP and unlocked memory
Assumes a 64-bit programmable USER
USER = 0000111122223333
AP = access password; U/L is unlocked/locked memory; TC = Test Case
#
Note: This entire section requires tag vendor customization as this feature is optional in Gen2
This is an example syntax - total memory varies by vendor

#UUU WrRd pass 0000111122223333 user
USER = 0000111122223333, U; TC = W_U_C, R_U_C

#UUU WrRd pass 4444 user,4
USER = 0000111122224444, U; TC = W_U_P, R_U_P

Now run test cases for locked USER

Write pass 11111111 apass
   Write result: 1
   Write SUCCESSFUL.
USER = 0000111122224444, U; AP = 11111111; Sets AP
#UUU Lock pass user 11111111
USER = 0000111122224444, L; AP = 11111111; TC = L_U
#Write fail 0000111122223333 user
USER = 0000111122224444, L; AP = 11111111; AP = 11111111; Attempted Write from open state;
#Write fail 0000111122223333 user FFFFFFFF
USER = 0000111122224444, L; AP = 11111111; Attempted Write incorrect AP;
#UUU WrRd pass 3333 user,4 11111111
USER = 0000111122223333, L; AP = 11111111; TC = W_U_L_P and R_U_L_P;

Now run test cases to unlock USER

#UUU Unlock fail user
USER = 0000111122223333, L; AP = 11111111; Attempted Unlock w/o AP
#Write fail 0000111122224444 user
USER = 0000111122223333, L; AP = 11111111; Attempted Write from open state;
#Write fail 0000111122224444 user FFFFFFFF
USER = 0000111122223333, L; AP = 11111111; Attempted Write incorrect AP;

#UUU Unlock fail user FFFFFFFF
USER = 0000111122223333, L; AP = 11111111; Attempted Unlock w/ incorrect AP

#Write fail 0000111122224444 user
USER = 0000111122223333, L; AP = 11111111; Attempted Write from open state;
#Write fail 0000111122224444 user FFFFFFFF
USER = 0000111122223333, L; AP = 11111111; Attempted Write incorrect AP;

#UUU Unlock pass user 11111111
USER = 0000111122223333, U; AP = 11111111; TC = U_U_L
#UUU WrRd pass 4444 user,4 11111111
USER = 0000111122224444, U; AP = 11111111; Write from secured state;
#UUU WrRd pass 3333 user,4
USER = 0000111122223333, U; AP = 11111111; Write from open state;
Now run test cases for Kill

Write pass 00000000 kpass
   Write result: 1
   Write SUCCESSFUL.
KP = 00000000, U; AP = 11111111; TC = W_KP_Z
This TC writes to the KP when the KP is all zeros.

Read pass 00000000 kpass
   Read: 00000000
   Read SUCCESSFUL.
KP = 00000000, U; AP = 11111111; TC = R_KP
This TC reads the KP when the KP is unlocked.

Write pass 01234567 kpass
   Write result: 1
   Write SUCCESSFUL.
KP = 01234567, U; AP = 11111111; TC = W_KP_NZ
Read pass 01234567 kpass
   Read: 01234567
   Read SUCCESSFUL.
KP = 01234567, U; AP = 11111111;
This TC writes a non-zero KP to the KP memory.

Lock pass kpass 11111111
   Lock result: 1
   Lock SUCCESSFUL.
KP = 01234567, L; AP = 11111111; Locks the kill password; TC = L_KP
Read fail 01234567 kpass
   Read: FAILED
   Read SUCCESSFUL.
KP = 01234567, L; AP = 11111111; Attempt to read KP from open state
Write fail 33333333 kpass
   Write result: 0
   Write SUCCESSFUL.
KP = 01234567, L; AP = 11111111; Verifies cannot write from open state
Read fail 01234567 kpass 44444444
   Read: FAILED
   Read SUCCESSFUL.
AP = 44444444, L; AP = 11111111; Verifies cannot read w/ incorrect AP
This TC locks the KP.

Read pass 01234567 kpass 11111111
   Read: 01234567

```
   Read SUCCESSFUL.
# KP = 01234567, L; AP = 11111111; Verifes read from secured state; TC = R_KP_L
# This TC reads the KP when the KP is locked.

Write pass 76543210 kpass 11111111
   Write result: 1
   Write SUCCESSFUL.
# KP = 76543210, L; AP = 11111111; Verifes write from secured state; TC = W_KP_L_NZ
Read pass 76543210 kpass 11111111
   Read: 76543210
   Read SUCCESSFUL.
# KP = 76543210, L; AP = 11111111; Verifes read from secured state;
# This TC writes to the KP when the KP is locked and non-zero.

Kill fail 01234567
   Kill result: 0
   Kill SUCCESSFUL.
# KP = 76543210, L; AP = 11111111; TC = K_INZ_L
# This TC attempts to kill the tag with an incorrect, non-zero KP, while the KP is locked.

Unlock pass kpass 11111111
   UnLock result: 1
   UnLock SUCCESSFUL.
# KP = 76543210 , U; AP = 11111111; TC = U_KP_L
# This TC unlocks the KP when it was locked.

Kill fail FFFFFFFF
   Kill result: 0
   Kill SUCCESSFUL.
# KP = 76543210 , U; AP = 11111111; TC = K_INZ
Read pass 76543210 kpass
   Read: 76543210
   Read SUCCESSFUL.
# KP = 76543210 , U; AP = 11111111; Verifies tag is still alive
# This TC attempts to kill the tag using an incorrect, non-zero KP, while the KP is unlocked.

Write pass 00000000 kpass 11111111
   Write result: 1
   Write SUCCESSFUL.
# KP = 00000000, U; AP = 11111111; Sets KP to all zeros
Kill fail 00000000
   Kill result: 0
   Kill SUCCESSFUL.
# KP = 00000000 , U; AP = 11111111; TC = K_Z
# This TC attempts to kill the tag using an all-zero KP.

# At the end of this script, the tag is unlocked with an AP of 0x11111111 and a KP of 0x00000000

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS
```

4. PermaUnlocked (PU.out.txt)

```
# This script verifies PermaUnlock behavior with non-zero AP
# "unlocked" means that the bit pattern to the associated memory bank is "00"
# "locked" that the bit pattern to the associated memory bank is "10"
# "permaunlocked" means that the bit pattern to the associated memory bank is "01"
# "permalocked" means that the bit pattern to the associated memory bank is "11"

Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.
Read pass 00000000 apass
   Read: 00000000
   Read SUCCESSFUL.
Write pass AAAAAAAA kpass
   Write result: 1
   Write SUCCESSFUL.
Read pass AAAAAAAA kpass
   Read: AAAAAAAA
   Read SUCCESSFUL.
Unlock pass kpass
   UnLock result: 1
   UnLock SUCCESSFUL.
Unlock pass epc
   UnLock result: 1
   UnLock SUCCESSFUL.
# The following line is for rewritable TID
#TTT Unlock pass tid
# The following line is for Permalocked TID
Unlock fail tid
   UnLock result: 0
   UnLock SUCCESSFUL.
#UUU Unlock pass user
WrRd pass 300011112222333344445555666 epc
   WrRd write result: 1
   WrRd read result: 300011112222333344445555666
   WrRd SUCCESSFUL.
# EPC = 300011112222333344445555666
#TTT WrRd pass E200B080 tid
# Assumes a 48-bit programmable TID memory
# TID = E20010501111
#UUU WrRd pass 0000111122223333 user
# Assumes a 64-bit programmable USER memory
# USER = 0000111122223333

### ==== PermaUnlock Access Password Section ==== ###
Unlock pass apass
   UnLock result: 1
   UnLock SUCCESSFUL.
PermLock pass apass
   PermLock result: 1
   PermLock SUCCESSFUL.
# AP = 00000000, PU; Permanently unlocks the access password; TC = PU_AP
# Sets the PermaLock bit in the AP field to a one
# This TC permaunlocks the tag's AP with the correct, zero AP.
```

Write pass 22222222 apass
   Write result: 1
   Write SUCCESSFUL.
AP = 22222222, PU; Verifies can write from open state; TC = W_AP_PU_NZ_1
This TC writes a non-zero AP when the AP is permaunlocked and zero, using the correct AP.

Read pass 22222222 apass 22222222
   Read: 22222222
   Read SUCCESSFUL.
AP = 22222222, PU; Verifies can read from secured state; TC = R_AP_PU_3
This TC reads a non-zero AP, when the AP is permaunlocked and non-zero, using the correct AP.

Write pass 11111111 apass 22222222
   Write result: 1
   Write SUCCESSFUL.
AP = 11111111, PU; Verifies can write from secured state; TC = W_AP_PU_NZ_2
This TC writes a non-zero AP, when the AP is permaunlocked and non-zero, using the correct AP.

Read pass 11111111 apass
   Read: 11111111
   Read SUCCESSFUL.
AP = 11111111, PU; Verifies can read from open state; TC = R_AP_PU_5
This TC reads a non-zero AP, when the AP is permaunlocked and non-zero, using no AP.

Read fail 11111111 apass FFFFFFFF
   Read: FAILED
   Read SUCCESSFUL.
AP = 11111111, PU; Attempted read w incorrect AP; TC = R_AP_PU_4
This TC attempts to read a non-zero AP, when the AP is permaunlocked and non-zero, using an incorrect AP.

Read pass 11111111 apass 11111111
   Read: 11111111
   Read SUCCESSFUL.
AP = 11111111, PU; Verifies AP is all ones
Write pass 22222222 apass
   Write result: 1
   Write SUCCESSFUL.
AP = 22222222, PU; Verifies can write from open state; TC = W_AP_PU_NZ_4
This TC writes a non-zero AP, when the AP is permaunlocked and non-zero, using no AP.

Read pass 22222222 apass
   Read: 22222222
   Read SUCCESSFUL.
AP = 22222222, PU; Verifies can read from open state
Write fail 11111111 apass FFFFFFFF
   Write result: 0
   Write SUCCESSFUL.
AP = 22222222, PU; Attempted write with incorrect AP; TC = W_AP_PU_NZ_3
This TC attempts to write a non-zero AP, when the AP is permaunlocked and non-zero, using an incorrect AP.

Write pass 00000000 apass 22222222
   Write result: 1

Write SUCCESSFUL.
AP = 00000000, PU; Resets AP to all zeros; TC = W_AP_PU_Z
This TC writes a zero AP, when the AP is permaunlocked and non-zero, using the correct AP.

Read pass 00000000 apass 00000000
Read: 00000000
Read SUCCESSFUL.
AP = 00000000, PU; Verifies AP all zeros; TC = R_AP_PU_1
This TC reads a zero AP, when the AP is permaunlocked and zero, using the correct AP.

Read fail FFFFFFFF apass
Read: 00000000
Read SUCCESSFUL.
AP = 00000000, PU; Verifies failure to read despite with incorrect AP; TC = R_AP_PU_2
This TC attempts to read a zero AP, when the AP is permaunlocked and zero, using no AP.

PermUnLock fail apass
PermUnLock result: 0
PermUnLock SUCCESSFUL.
AP = 00000000, PU; Tries to deselect the permalock bit to the access password; TC = PU_AP_PU_1
This TC attempts to change the permalock bit, when the AP is permaunlocked and zero, using no AP.

Write pass 22222222 apass
Write result: 1
Write SUCCESSFUL.
AP = 22222222, PU; Verifies can write from open state
Read pass 22222222 apass
Read: 22222222
Read SUCCESSFUL.
AP = 22222222, PU; Verifies can read from open state

PermUnLock fail apass
PermUnLock result: 0
PermUnLock SUCCESSFUL.
AP = 22222222, PU; Tries to deselect the permalock bit for the access password from open state; TC = PU_AP_PU_3
This TC attempts to change the permalock bit, when the AP is permaunlocked and non-zero, using no AP.

PermUnLock fail apass 22222222
PermUnLock result: 0
PermUnLock SUCCESSFUL.
AP = 22222222, PU; Tries to deselect the permalock bit for the access password from secured state; TC = PU_AP_PU_2
This TC attempts to change the permalock bit, when the AP is permaunlocked and non-zero, using the correct AP.

Lock fail apass
Lock result: 0
Lock SUCCESSFUL.
AP = 22222222, PU; Tries to lock the access password from open state; TC = L_AP_PU_3
#
This TC attempts to set the lock bit, when the AP is permaunlocked and non-zero, using no AP.

Lock fail apass 22222222

Lock result: 0
Lock SUCCESSFUL.
AP = 22222222, PU; Tries to lock the access password using the correct access password, when the access password is non-zero; TC = L_AP_PU_2
This TC attempts to set the lock bit, when the AP is permaunlocked and non-zero, using the correct AP.

Write pass 00000000 apass 22222222
Write result: 1
Write SUCCESSFUL.
AP = 00000000, PU; Sets tag AP to all zeros
Read pass 00000000 apass
Read: 00000000
Read SUCCESSFUL.
Tag AP is in PermaUnlocked state with all zeros.

Continue with the PermaUnlocked EPC, TID, and USER memory test cases
and Single tag Inventory cases after EPC

Now working on EPC memory
PermLock pass epc
PermLock result: 1
PermLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PU; AP = 00000000, PU; Permanently unlocks the EPC bank; TC = PU_E
Sets the PermaLock bit in the AP field to a one
This TC puts the EPC in permaunlock, when the AP is permaunlocked and zero, using no AP.

Lock fail epc
Lock result: 0
Lock SUCCESSFUL.
EPC = 3000111122223333444455556666, PU; AP = 00000000, PU; TC = L_E_PU
Verifies the epc bank is PermaUnlocked and that it can't be locked from the open state
This TC attempts to set the EPC in permalock, when the AP is permaunlocked and zero and the EPC is permaunlocked, using no AP.

Unlock pass epc
UnLock result: 1
UnLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PU; AP = 00000000, PU; TC = U_E_PU
This TC unsets the EPC lock bit (or confirms that it is unset in this case), when the AP is permaunlocked and zero and the EPC is permaunlocked.

PermUnLock fail epc
PermUnLock result: 0
PermUnLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PU; AP = 00000000, PU; TC = PU_E_PU
This TC attempts to unset the EPC permalock bit, when the AP is permaunlocked and zero and the EPC is permaunlocked, using no AP.

Lock fail epc
Lock result: 0
Lock SUCCESSFUL.
EPC = 3000111122223333444455556666, PU; AP = 00000000, PU
Verifies the EPC bank is still PermaUnlocked
Write pass 7777 epc,5

Write result: 1
Write SUCCESSFUL.
EPC = 3000111122223333777755556666, PU; AP = 00000000, PU; TC = W_E_PU_P
This TC writes to a portion of the EPC, when the AP is permaunlocked and zero and the EPC is permaunlocked.

Read pass 7777 epc,5,1
Read: 7777
Read SUCCESSFUL.
EPC = 3000111122223333777755556666, PU; AP = 00000000, PU; TC = R_E_PU_P
This TC reads a portion of the EPC, when the AP is permaunlocked and zero and the EPC is permaunlocked, using no AP.

#Lock pass epc 00000000
EPC = 3000111122223333777755556666, PL; AP = 00000000, PU; TC = L_E_PU, PL_E_PU
Verifies the epc bank is PermaLocked and that it can be permalocked from the secure state
This test case is taken care of by PL_E_L_2

Now working on TID memory
This section is for generic TID
Assumes a 48-bit programmable TID memory
TID = E20010501111
#TTT PermLock pass tid
TID = E20010501111, PU; AP = 00000000, PU; Permanently unlocks the EPC bank; TC = PU_T
Sets the PermaLock bit in the AP field to a one
#TTT Lock fail tid
TID = E20010501111, PU; AP = 00000000, PU; TC = L_T_PU
Verifies the epc bank is PermaUnlocked
#TTT Unlock pass tid
TID = E20010501111, PU; AP = 00000000, PU; TC = U_T_PU
#TTT PermUnLock fail tid
TID = E20010501111, PU; AP = 00000000, PU; TC = PU_T_PU
#TTT Lock fail tid
TID = E20010501111, PU; AP = 00000000, PU
Verifies the TID bank is still PermaUnlocked
#TTT Write pass 2222 tid,2
TID = E20010502222, PU; AP = 00000000, PU; TC = W_T_PU_P
#TTT Read pass 2222 tid,2,1
TID = E20010502222, PU; AP = 00000000, PU; TC = R_T_PU_P

This section is for tags with PermaUnlocked TID memory
Note: This section is not applicable for tags with Permalocked TID memory
#
#PermLock pass tid
TID = E2001050, PL; AP = 00000000, PU; Permanently unlocks the TID bank; TC = PU_T
Sets the PermaLock bit in the AP field to a one
#Lock pass tid
TID = E2001050, PL; AP = 00000000, PU; TC = L_T_PU
Verifies the tid bank is PermaUnlocked
#Unlock fail tid
TID = E2001050, PL; AP = 00000000, PU; TC = U_T_PU
#PermUnLock fail tid
TID = E2001050, PL; AP = 00000000, PU; TC = PU_T_PU
#Lock pass tid
TID = E2001050, PL; AP = 00000000, PU

Verifies the TID bank is still PermaUnlocked
#Write fail 1111 tid,1
TID = E2001050, PL; AP = 00000000, PU; TC = W_T_PU_P
#Read pass B080 tid,1,1
TID = E2001050, PL; AP = 00000000, PU; TC = R_T_PU_P

Now working on USER memory
Assumes a 64-bit programmable USER
USER = 0000111122223333
Note: This section is not applicable for tags with no user memory
#UUU Unlock pass user
#UUU PermLock pass user
USER = 0000111122223333, PU; AP = 00000000, PU; Permanently unlocks the USER bank; TC = PU_U
Sets the PermaLock bit in the AP field to a one
#UUU Lock fail user 00000000
USER = 0000111122223333, PU; AP = 00000000, PU; Attempt to lock the USER bank; TC= L_U_PU
#UUU Unlock pass user
USER = 0000111122223333, PU; AP = 00000000, PU; TC = U_U_PU
#PermUnLock fail user
USER = 0000111122223333, PU; AP = 00000000, PU; TC = PU_U_PU
Verifies the User memory is still Permaunlocked
#UUU Write pass 4444 user,3
USER = 0000111122224444, PU; AP = 00000000, PU; TC = W_U_PU_P
#UUU Read pass 4444 user,3,1
USER = 0000111122224444, PU; AP = 00000000, PU; TC = R_U_PU_P

Now working on Kill password
PermLock pass kpass
   PermLock result: 1
   PermLock SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU; Permanently unlocks the EPC bank; TC = PU_KP
This TC puts the KP in permaunlock, when the AP is zero, using no AP.

Sets the PermaLock bit in the AP field to a one
Lock fail kpass
   Lock result: 0
   Lock SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU; TC = L_KP_PU
Verifies the kill password is PermaUnlocked and that it can't lock it from the open state.
This TC attempts to set the lock bit for the KP, using no AP, when the KP is permaunlocked.

Unlock pass kpass
   UnLock result: 1
   UnLock SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU; TC = U_KP_PU
This TC unsets the lock bit to the KP (or confirms that it is already unset) when the KP is permaunlocked.

PermUnLock fail kpass
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU; TC = PU_KP_PU
This TC attempts to unset the permalock bit, while the KP is permaunlocked.

Lock fail kpass
   Lock result: 0
   Lock SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU
Verifies the Kill password is still PermaUnlocked
Write pass BBBBBBBB kpass
   Write result: 1
   Write SUCCESSFUL.
KP = BBBBBBBB, PU; AP = 00000000, PU; TC = W_KP_PU_NZ
This TC writes a non-zero KP, when the AP is zero and the KP is permaunlocked and non-zero.

Read pass BBBBBBBB kpass
   Read: BBBBBBBB
   Read SUCCESSFUL.
KP = BBBBBBBB, PU; AP = 00000000, PU; TC = R_KP_PU
This TC reads a non-zero KP, when the AP is zero and the KP is permaunlocked and non-zero.

Kill fail ABBAABBA
   Kill result: 0
   Kill SUCCESSFUL.
KP = AAAAAAAA, PU; AP = 00000000, PU; TC = K_INZ_PU
This TC attempts to kill the tag using an incorrect, non-zero KP, while the KP is permaunlocked.

#=== Now run the Inventory cases ===#

Inventory pass 1
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
TC = I_PU
This TC does an inventory of a single tag.

Inventory pass 1 11112222 epc
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
EPC = 300011112222333377775555666, PU; AP = 00000000; TC = SI_E_PU
Read pass 300011112222333377775555666 epc
   Read: 300011112222333377775555666
   Read SUCCESSFUL.
EPC = 300011112222333377775555666, PU; AP = 00000000; Verifies EPC
This TC inventories a single tag, based on a portion of the EPC being present.

Inventory pass 1 E200 tid
   Using TID E200
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
TID = E2001050, PU/PL; AP = 00000000; TC = SI_T_PU
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
TID = E2001050, PU/PL; AP = 00000000; Verifies TID
This TC inventories a single tag, based on a portion of the TID being present.

#UUU Inventory pass 1 00001111 user
USER = 0000111122224444, PU; AP = 00000000; TC = SI_U_PU

#UUU Read pass 0000111122224444 user,0,4
USER = 0000111122224444, PU; AP = 00000000; Verifies user memory
This TC inventories a single tag, based on a portion of the User data being present.

disconnect
Disconnected.

OVERALL RESULT: SUCCESS

5. PermaLocked_L (PL_L.out.txt)

"unlocked" means that the bit pattern to the associated memory bank is "00"
"locked" that the bit pattern to the associated memory bank is "10"
"permaunlocked" means that the bit pattern to the associated memory bank is "01"
"permalocked" means that the bit pattern to the associated memory bank is "11"
#
Write pass 00000000 apass
Write result: 1
Write SUCCESSFUL.
Read pass 00000000 apass
Read: 00000000
Read SUCCESSFUL.
Write pass AAAAAAAA kpass
Write result: 1
Write SUCCESSFUL.
Read pass AAAAAAAA kpass
Read: AAAAAAAA
Read SUCCESSFUL.
Unlock pass kpass
UnLock result: 1
UnLock SUCCESSFUL.
Unlock pass epc
UnLock result: 1
UnLock SUCCESSFUL.
The following line is for rewritable TID
#TTT Unlock pass tid
The following line is for Permalocked TID
Unlock fail tid
UnLock result: 0
UnLock SUCCESSFUL.
#UUU Unlock pass user
WrRd pass 300011112222333344445555666 epc
WrRd write result: 1
WrRd read result: 300011112222333344445555666
WrRd SUCCESSFUL.
EPC = 300011112222333344445555666
#TTT WrRd pass E200B080 tid
Assumes a 48-bit programmable TID memory
TID = E20010501111
#UUU WrRd pass 0000111122223333 user
Assumes a 64-bit programmable USER memory
USER = 0000111122223333

Now working on the rest of the AP memory
Write pass 12345678 apass
   Write result: 1
   Write SUCCESSFUL.
Read pass 12345678 apass 12345678
   Read: 12345678
   Read SUCCESSFUL.
AP=12345678, U, KP=AAAAAAAA
lock pass apass 12345678
   Lock result: 1
   Lock SUCCESSFUL.
Putting tag in locked state for PL test
Permlock fail apass
   PermLock result: 0
   PermLock SUCCESSFUL.
Attempting to put in Permalock while in open state. TC=PL_AP_L_2
Write fail 33333333 apass
   Write result: 0
   Write SUCCESSFUL.
AP=12345678, L; Verifies can't write from open state
Read fail 33333333 apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 12345678, L; Verifies can't read 33333333 from open state
Read fail 12345678 apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 12345678, L; Verifies can't read 12345678 from open state
Read pass 12345678 apass 12345678
   Read: 12345678
   Read SUCCESSFUL.
AP = 12345678, L; Verifies can read from secured state
Write pass 11111111 apass 12345678
   Write result: 1
   Write SUCCESSFUL.
AP = 12345678, L; Verifies can write from secured state;
Read pass 11111111 apass 11111111
   Read: 11111111
   Read SUCCESSFUL.
AP = 11111111, L; Verifies AP is 12345678 from the secured state;
This TC attempts to set the permalock bit for the AP, while the tag's AP is locked and non-zero, using no AP

PermLock pass apass 11111111
   PermLock result: 1
   PermLock SUCCESSFUL.
Tag now in PL state with non-zero password. TC=PL_AP_L_1
Write fail 33333333 apass 11111111
   Write result: 0
   Write SUCCESSFUL.
AP=11111111, PL; Verifies can't write from secured state
Read fail 33333333 apass 33333333
   Read: FAILED
   Read SUCCESSFUL.
AP = 11111111, PL; Verifies AP is not 33333333

Read fail 11111111 apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 11111111, PL; Verifies can't read from open state.
Read fail 11111111 apass 11111111
   Read: FAILED
   Read SUCCESSFUL.
AP = 11111111, PL; Verifies can read from secured state
Lock pass apass 11111111
   Lock result: 1
   Lock SUCCESSFUL.
AP = 11111111, PL; Verifies still can lock the tag
Write fail 00000000 apass 11111111
   Write result: 0
   Write SUCCESSFUL.
Read fail 00000000 apass 00000000
   Read: FAILED
   Read SUCCESSFUL.
AP = 11111111, PL; Verifies still can't write to the AP
This TC sets the permalock bit for the AP and puts the tag in permalock, while the tag's AP is locked and non-zero, using the correct AP.

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS

6. PermaLocked_APZ (PL_APZ.out.txt)

This script verifies PermaLock behavior with zero AP
"unlocked" means that the bit pattern to the associated memory bank is "00"
"locked" that the bit pattern to the associated memory bank is "10"
"permaunlocked" means that the bit pattern to the associated memory bank is "01"
"permalocked" means that the bit pattern to the associated memory bank is "11"

Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.
Read pass 00000000 apass
   Read: 00000000
   Read SUCCESSFUL.
Write pass AAAAAAAA kpass
   Write result: 1
   Write SUCCESSFUL.
Read pass AAAAAAAA kpass
   Read: AAAAAAAA
   Read SUCCESSFUL.
Unlock pass kpass
   UnLock result: 1
   UnLock SUCCESSFUL.
Unlock pass epc
   UnLock result: 1
   UnLock SUCCESSFUL.
The following line is for rewritable TID

#TTT Unlock pass tid
The following line is for Permalocked TID
Unlock fail tid
   UnLock result: 0
   UnLock SUCCESSFUL.
#UUU Unlock pass user
WrRd pass 300011112222333344445555666 epc
   WrRd write result: 1
   WrRd read result: 300011112222333344445555666
   WrRd SUCCESSFUL.
EPC = 300011112222333344445555666
#TTT WrRd pass E200B080 tid
Assumes a 48-bit programmable TID memory
TID = E20010501111
#UUU WrRd pass 0000111122223333 user
Assumes a 64-bit programmable USER memory
USER = 0000111122223333

==== PermaLock Access Password Section ====
Lock pass apass
   Lock result: 1
   Lock SUCCESSFUL.
AP = 00000000, L; Locks the access password
Sets the Lock bit in the AP field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass apass 00000000
   PermLock result: 1
   PermLock SUCCESSFUL.
AP = 00000000, PL; Permanently locks the access password; TC = PL_AP_L_3
Sets the PermaLock bit in the AP field to a one
Write fail FFFFFFFF apass
   Write result: 0
   Write SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot write from open state
Read fail FFFFFFFF apass
   Read: FAILED
   Read SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot read from open state
Write fail 00000000 apass 00000000
   Write result: 0
   Write SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot write from secured state
Read fail 00000000 apass 00000000
   Read: FAILED
   Read SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot read from secured state
This TC permalocks the AP when the AP is zero and correct. It's also the TC where it permalocks the AP with it already being locked.

Unlock fail apass 00000000
   UnLock result: 0
   UnLock SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot be unlocked from secured state; TC = U_AP_PL_1
This TC attempts to unlock the AP after it has been permalocked using the correct, zero AP.
PermUnLock fail apass 00000000

PermUnLock result: 0
PermUnLock SUCCESSFUL.
AP = 00000000, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_AP_PL_1
This TC attempts to deset the permalock bit to the AP after it has been permalocked using the correct, zero AP.

Lock pass apass 00000000
Lock result: 1
Lock SUCCESSFUL.
AP = 00000000, PL; Verifies memory can be Permaunlocked and locked from secured state; TC = L_AP_PL_1
This TC confirms that the lock bit is set to the AP after it has been permalocked using the correct, zero AP.

=== Now working on EPC memory ===
Lock pass epc
Lock result: 1
Lock SUCCESSFUL.
AP = 00000000, L; Locks the EPC memory
Sets the Lock bit in the EPC field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass epc 00000000
PermLock result: 1
PermLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Permanently locks the EPC bank from secured state; TC = PL_E_L_1
This TC permalocks the EPC, after it's locked, when the AP is correct and zero.

Sets the PermaLock bit in the AP field to a one
Write fail 7777 epc,5
Write result: 0
Write SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot write from open state; TC = W_E_PL_P_1
This TC attempts to write a portion of the EPC, when the AP is zero and permalocked.

Read pass 4444 epc,5,1
Read: 4444
Read SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot read from open state; TC = R_E_PL_P_1
This TC attempts to read a portion of the EPC, when the AP is zero and permalocked.

Write fail 7777 epc,5 00000000
Write result: 0
Write SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot write from secured state
Read pass 4444 epc,5,1 00000000
Read: 4444
Read SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot read from secured state
Unlock fail epc 00000000

UnLock result: 0
UnLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot be unlocked from secured state; TC = U_E_PL_1
This TC attempts to unlock the EPC, when the AP is zero and permalocked.

PermUnLock fail epc 00000000
PermUnLock result: 0
PermUnLock SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_E_PL_1
This TC attempts to unset the permalock bit to the EPC, when the AP is zero and permalocked.

Lock pass epc 00000000
Lock result: 1
Lock SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000, PL; Verifies memory can be locked from secured state; TC = L_E_PL_1
This TC attempts to set the lock bit to the EPC, (or confirm that it's set) when the AP is zero and permalocked.

Now working on TID memory
This section is for generic TID
Assumes a 48-bit programmable TID memory
TID = E20010501111
#TTT Lock pass tid
TID = E20010501111, L; AP = 00000000, PL; Locks the TID memory
Sets the Lock bit in the TID field to a one
This action is required first in order to Permalock using PermLock directive
#TTT PermLock pass tid 00000000
TID = E20010501111, PL; AP = 00000000, PL; Permanently locks the TID bank from secured state; TC = PL_T_L_1
Sets the PermaLock bit in the AP field to a one
#TTT Write fail 2222 tid,2
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot write from open state; TC = W_T_PL_P_1
#TTT Read pass 1111 tid,2,1
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot read from open state; TC = R_T_PL_P_1
#TTT Write fail 2222 tid,2 00000000
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot write from secured state
#TTT Read pass 1111 tid,2,1 00000000
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot read from secured state
#TTT Unlock fail tid 00000000
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot be unlocked from secured state; TC = U_T_PL_1
#TTT PermUnLock fail tid 00000000
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_T_PL_1
#TTT Lock pass tid 00000000
TID = E20010501111, PL; AP = 00000000, PL; Verifies memory can be locked from secured state; TC = L_T_PL_1

This section is for PermaLocked TID
TID = E2001050

Lock pass tid
   Lock result: 1
   Lock SUCCESSFUL.
TID = E2001050, L; AP = 00000000, PL; Locks the TID memory
Sets the Lock bit in the TID field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass tid 00000000
   PermLock result: 1
   PermLock SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Permanently locks the TID bank from secured state; TC = PL_T_L_1
This TC permalocks the TID (or confirms it's permalocked) when the tag's TID is locked and the AP is zero.

Sets the PermaLock bit in the TID field to a one
Write fail 2222 tid,1
   Write result: 0
   Write SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory cannot write from open state; TC = W_T_PL_P_1
This TC attempts to write to the TID when the TID is permalocked, using an all-zero AP.

Read pass B080 tid,1,1
   Using TID B080
   Read: B080
   Read SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory can read from open state; TC = R_T_PL_P_1
This TC reads a known value from the TID when the TID is permalocked, using an all-zero AP.

Write fail 2222 tid,1 00000000
   Write result: 0
   Write SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory cannot write from secured state
Read pass B080 tid,1,1 00000000
   Using TID B080
   Read: B080
   Read SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory cannot read from secured state
Unlock fail tid 00000000
   UnLock result: 0
   UnLock SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory cannot be unlocked from secured state; TC = U_T_PL_1
This TC attempts to unlock the TID from the permalocked state when the AP is zero.

PermUnLock fail tid 00000000
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_T_PL_1
This TC attempts to unset the TID's permalock bit when the AP is zero.

Lock pass tid 00000000
   Lock result: 1

Lock SUCCESSFUL.
TID = E2001050, PL; AP = 00000000, PL; Verifies memory can be locked; TC = L_T_PL_1
This TC attempts to set the lock bit (it confirms a locked TID) when the TID is permalocked and the AP is zero.

Now working on USER memory
Assumes a 64-bit programmable USER
USER = 0000111122223333
#UUU Lock pass user
USER = 0000111122223333, L; Locks the USER memory
Sets the Lock bit in the USER field to a one
This action is required first in order to Permalock using PermLock directive
#UUU PermLock pass user 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Permanently locks the USER bank from secured state; TC = PL_U_L_1
Sets the PermaLock bit in the USER field to a one
#UUU Write fail 4444 user,3
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory cannot write from open state; TC = W_U_PL_P_1
#UUU Read pass 3333 user,3,1
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory can read from open state; TC = R_U_PL_P_1
#UUU Write fail 4444 user,3 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory cannot write from secured state
#UUU Read pass 3333 user,3,1 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory cannot read from secured state
#UUU Unlock fail user 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory cannot be unlocked from secured state; TC = U_U_PL_1
#UUU PermUnLock fail user 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_U_PL_1
#UUU Lock pass user 00000000
USER = 0000111122223333, PL; AP = 00000000, PL; Verifies memory can be locked from secured state; TC = L_U_PL_1

Now working on Kill password
Lock pass kpass
Lock result: 1
Lock SUCCESSFUL.
KP = AAAAAAAA, L; Locks the Kill password
Sets the Lock bit in the KP field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass kpass 00000000
PermLock result: 1
PermLock SUCCESSFUL.
KP = AAAAAAAA, PL; Permanently locks the kill password from secured state; TC = PL_KP_L_1, PL_KP
Sets the PermaLock bit in the KP field to a one
This TC permalocks the KP, after it was locked, using a zero AP.

Write fail FFFFFFFF kpass
Write result: 0
Write SUCCESSFUL.

KP = AAAAAAAA, PL; Verifies memory cannot write from open state; TC = W_KP_PL_NZ_1
This TC writes a non-zero KP, when the KP is permalocked, using a zero AP.

Read fail AAAAAAAA kpass
   Read: FAILED
   Read SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot read from open state; TC = R_KP_PL_1
This TC attempts to read the KP, when the KP is permalocked, using a zero AP.

Write fail FFFFFFFF kpass 00000000
   Write result: 0
   Write SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot write from secured state
Read fail AAAAAAAA kpass 00000000
   Read: FAILED
   Read SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot read from secured state
Unlock fail kpass 00000000
   UnLock result: 0
   UnLock SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot be unlocked from secured state; TC = U_KP_PL_1

PermUnLock fail kpass 00000000
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_KP_PL_1
This TC attempts to deset the permalock bit, when the KP is permalocked, using a zero AP.

Lock pass kpass 00000000
   Lock result: 1
   Lock SUCCESSFUL.
KP = AAAAAAAA, PL; AP = 00000000, PL; Verifies memory can be locked from secured state; TC = L_KP_PL_1
This TC sets the lock bit to the KP, (or confirms that it's set) when the KP is permalocked and the AP is zero.

Kill fail ABBAABBA
   Kill result: 0
   Kill SUCCESSFUL.
KP = AAAAAAAA, PL; AP = 00000000, PL; TC = K_INZ_PL
This TC attempts to kill the tag, while the KP is permalocked, using an incorrect, non-zero KP.

#=== Now run the Inventory cases ===#

Inventory pass 1
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
TC = I_PL
This TC inventories the single tag while all memory is permalocked.

Inventory pass 1 11112222 epc
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
EPC = 300011112222333344445555666, PL; AP = 00000000; TC = SI_E_PL

Read pass 3000111122223333444455556666 epc
   Read: 3000111122223333444455556666
   Read SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000; Verifies EPC
This TC inventories the tag based on a portion of the EPC while the tag's memory is permalocked.

Inventory pass 1 E200 tid
   Using TID E200
   Inventoried 1 unique tags.
   Inventory SUCCESSFUL.
TID = E2001050, PU/PL; AP = 00000000; TC = SI_T_PL
Read pass E200B080 tid
   Using TID E200B080
   Read: E200B080
   Read SUCCESSFUL.
TID = E2001050, PU/PL; AP = 00000000; Verifies TID
This TC inventories the tag based on a portion of the TID while the tag's memory is permalocked.

#UUU Inventory pass 1 00001111 user
USER = 0000111122223333, PL; AP = 00000000; TC = SI_U_PL
#UUU Read pass 0000111122223333 user
USER = 0000111122223333, PL; AP = 00000000; Verifies user memory

Kill pass AAAAAAAA
   Kill result: 1
   Kill SUCCESSFUL.
KP = AAAAAAAA, PL; AP = 00000000, PL; TC = K_NZ
Dead tag
Inventory fail 1 11112222 epc
   Inventoried 0 unique tags.
   Inventory SUCCESSFUL.
EPC = 3000111122223333444455556666, PL; AP = 00000000
Inventory fail 1 E200 tid
   Using TID E200
   Inventoried 0 unique tags.
   Inventory SUCCESSFUL.
TID = E2001050, PU/PL; AP = 00000000
Inventory fail 1 00001111 user
   Inventoried 0 unique tags.
   Inventory SUCCESSFUL.
USER = 0000111122223333, PL; AP = 00000000
This TC kills the tag using a non-zero KP.

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS

7. PermaLocked_APNZ (PL_APNZ.out.txt)

```
# This script verifies PermaLock behavior with nonzero AP
# "unlocked" means that the bit pattern to the associated memory bank is "00"
# "locked" that the bit pattern to the associated memory bank is "10"
# "permaunlocked" means that the bit pattern to the associated memory bank is "01"
# "permalocked" means that the bit pattern to the associated memory bank is "11"

Write pass ABCDEF01 apass
   Write result: 1
   Write SUCCESSFUL.
Read pass ABCDEF01 apass
   Read: ABCDEF01
   Read SUCCESSFUL.
Write pass AAAAAAAA kpass ABCDEF01
   Write result: 1
   Write SUCCESSFUL.
Read pass AAAAAAAA kpass ABCDEF01
   Read: AAAAAAAA
   Read SUCCESSFUL.
Unlock pass kpass ABCDEF01
   UnLock result: 1
   UnLock SUCCESSFUL.
Unlock pass epc ABCDEF01
   UnLock result: 1
   UnLock SUCCESSFUL.
# The following line is for rewritable TID
#TTT Unlock pass tid ABCDEF01
# The following line is for Permalocked TID
Unlock fail tid ABCDEF01
   UnLock result: 0
   UnLock SUCCESSFUL.
# The following line is for User memory
#UUU Unlock pass user
WrRd pass 3000AAAABBBBCCCCDDDDEEEEFFFF epc ABCDEF01
   WrRd write result: 1
   WrRd read result: 3000AAAABBBBCCCCDDDDEEEEFFFF
   WrRd SUCCESSFUL.
# EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF
#TTT WrRd pass E200B080 tid ABCDEF01
# Assumes a 48-bit programmable TID memory
# TID = E20010501111
#UUU WrRd pass FFFFEEEEDDDDCCCC user ABCDEF01
# Assumes a 64-bit programmable USER memory
# USER = FFFFEEEEDDDDCCCC

### ==== PermaLock Access Password Section ==== ###
Lock pass apass ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
# AP = ABCDEF01, L; Locks the access password
# Sets the Lock bit in the AP field to a one
# This action is required first in order to Permalock using PermLock directive
PermLock pass apass ABCDEF01
   PermLock result: 1
```

PermLock SUCCESSFUL.
AP = ABCDEF01, PL; Permanently locks the access password; same as TC = L_AP_PU_2
Sets the PermaLock bit in the AP field to a one
This TC puts the AP in permalocked state, which is what happens when the AP is locked from the PU state, using a non-zero and correct AP.

Write fail FFFFFFFF apass
Write result: 0
Write SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot write from open state
Read fail ABCDEF01 apass
Read: FAILED
Read SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot read from open state
Write fail FFFFFFFF apass ABCDEF01
Write result: 0
Write SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot write from secured state; TC = W_AP_PL_NZ_1
This TC writes a non-zero AP, when the AP is permalocked and non-zero, using the correct AP.

Read fail ABCDEF01 apass ABCDEF01
Read: FAILED
Read SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot read from secured state; TC = R_AP_PL_1
This TC attempts to read the AP, when the AP is permalocked and non-zero, using the correct AP.

Write fail FFFFFFFF apass
Write result: 0
Write SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot write from open state; TC = W_AP_PL_NZ_2
This TC attempts to write a non-zero AP, when the AP is permalocked and non-zero, using no AP.

Read fail ABCDEF01 apass
Read: FAILED
Read SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot read from open state; TC = R_AP_PL_2
This TC attempts to read the AP, when the AP is permalocked and non-zero, using no AP.

Unlock fail apass ABCDEF01
UnLock result: 0
UnLock SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot be unlocked from secured state; TC = U_AP_PL_2
This TC attempts to deset the lock bit to the AP, when the AP is permalocked and non-zero, using the correct AP.

Unlock fail apass
UnLock result: 0
UnLock SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot be unlocked from open state; TC = U_AP_PL_3
This TC attempts to unlock the AP, when the AP is permalocked and non-zero, using no AP.

PermUnLock fail apass ABCDEF01
PermUnLock result: 0
PermUnLock SUCCESSFUL.

AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_AP_PL_2
This TC attempts to deset the peramlock bit to the AP, when the AP is permalocked and non-zero, using the correct AP.

PermUnLock fail apass
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from open state; TC = PU_AP_PL_3
This TC attempts to deset the peramlock bit to the AP, when the AP is permalocked and non-zero, using no AP.

Lock pass apass ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = L_AP_PL_2
This TC locks the AP (or confirmes that the lock bit is set), when the AP is permalocked and non-zero, using the correct AP.

Lock fail apass
   Lock result: 0
   Lock SUCCESSFUL.
AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from open state; TC = L_AP_PL_3
This TC attempts to lock the AP (or confirm that the lock bit is set), when the AP is permalocked and non-zero, using no AP.

=== Now working on EPC memory ===
Lock pass epc ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
AP = ABCDEF01, L; Locks the EPC memory
Sets the Lock bit in the EPC field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass epc ABCDEF01
   PermLock result: 1
   PermLock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Permanently locks the EPC bank from secured state; TC = PL_E_L_2
Sets the PermaLock bit in the AP field to a one
This TC permalocks the EPC, when the EPC is locked and the AP is non-zero.

Write fail 7777 epc,5
   Write result: 0
   Write SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot write from open state; TC = W_E_PL_P_2
This TC attempts to write to a portion of the EPC, when the AP is non-zero and the EPC is permalocked. See following command for write from secured state.

Read pass DDDD epc,5,1
   Read: DDDD
   Read SUCCESSFUL.

EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory can be read from open state; TC = R_E_PL_P_2
This TC reads a portion of the EPC, when the AP is non-zero and the EPC is permalocked. See following command for read from secured state.

Write fail 7777 epc,5 ABCDEF01
Write result: 0
Write SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot write from secured state

Read pass DDDD epc,5,1 ABCDEF01
Read: DDDD
Read SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory can be read from secured state

Unlock fail epc ABCDEF01
UnLock result: 0
UnLock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot be unlocked from secured state; TC = U_E_PL_2
This TC attempts to deset the lock bit, when the AP is non-zero and the EPC is permalocked.

PermUnLock fail epc ABCDEF01
PermUnLock result: 0
PermUnLock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_E_PL_2
This TC attempts to deset the permalock bit, when the AP is non-zero and the EPC is permalocked.

Lock fail epc 12345678
Lock result: 0
Lock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from secured state with incorrect AP

Lock fail epc
Lock result: 0
Lock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from open state
Lock pass epc ABCDEF01
Lock result: 1
Lock SUCCESSFUL.
EPC = 3000AAAABBBBCCCCDDDDEEEEFFFF, PL; AP = ABCDEF01, PL; Verifies memory can be locked from secured state; TC = L_E_PL_2
This TC attempts to set the lock bit, when the AP is non-zero and the EPC is permalocked, using an incorrect AP, no AP, and the correct AP.

Now working on TID memory
This section is for generic TID
Assumes a 48-bit programmable TID memory
TID = E20010501111
#TTT Lock pass tid ABCDEF01

TID = E20010501111, L; AP = ABCDEF01, PL; Locks the TID memory
Sets the Lock bit in the TID field to a one
This action is required first in order to Permalock using PermLock directive
#TTT PermLock pass tid ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Permanently locks the TID bank from secured state; TC = PL_T_L_2
Sets the PermaLock bit in the AP field to a one
#TTT Write fail 2222 tid,2
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot write from open state; TC = W_T_PL_P_2
#TTT Read pass 1111 tid,2,1
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory can be read from open state; TC = R_T_PL_P_2
#TTT Write fail 2222 tid,2 ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot write from secured state
#TTT Read pass 1111 tid,2,1 ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory can be read from secured state
#TTT Unlock fail tid ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot be unlocked from secured state; TC = U_T_PL_2
#TTT PermUnLock fail tid ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_T_PL_2
#TTT Lock fail tid 12345678
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from secured state with incorrect AP
#TTT Lock fail tid
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from open state
#TTT Lock pass tid ABCDEF01
TID = E20010501111, PL; AP = ABCDEF01, PL; Verifies memory can be locked from secured state; TC = L_T_PL_2

This section is for PermaLocked TID
TID = E2001050
Lock pass tid ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
TID = E2001050, L; AP = ABCDEF01, PL; Locks the TID memory
Sets the Lock bit in the TID field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass tid ABCDEF01
   PermLock result: 1
   PermLock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Permanently locks the TID bank from secured state; TC = PL_T_L_2
This TC makes the TID permalocked (or confirms that it's permalocked) when the AP is non-zero and the TID is locked.

Sets the PermaLock bit in the TID field to a one
Write fail 2222 tid,1
   Write result: 0
   Write SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot write from open state; TC = W_T_PL_P_2

This TC (and the subsequent command for the secured state) attempt to write to a portion of the TID, when the TID is permalocked and the AP is non-zero.

Read pass B080 tid,1,1
   Using TID B080
   Read: B080
   Read SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory can be read from open state; TC = R_T_PL_P_2
This TC (and the subsequent command for the secured state) attempt to read from a portion of the TID, when the TID is permalocked and the AP is non-zero.

Write fail 2222 tid,1 ABCDEF01
   Write result: 0
   Write SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot write from secured state
Read pass B080 tid,1,1 ABCDEF01
   Using TID B080
   Read: B080
   Read SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory can be read from secured state

Unlock fail tid ABCDEF01
   UnLock result: 0
   UnLock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot be unlocked from secured state; TC = U_T_PL_2
This TC attempts to deset the lock bit to the TID, when the TID is permalocked and the AP is non-zero.

PermUnLock fail tid ABCDEF01
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_T_PL_2
This TC attempts to deset the permalock bit to the TID, when the TID is permalocked and the AP is non-zero.

Lock fail tid 12345678
   Lock result: 0
   Lock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from secured state with incorrect AP
Lock fail tid
   Lock result: 0
   Lock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from open state
Lock pass tid ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
TID = E2001050, PL; AP = ABCDEF01, PL; Verifies memory can be locked; TC = L_T_PL_2
This TC attempts to set the lock bit, when the AP is non-zero and the TID is permalocked, using an incorrect AP, no AP, and the correct AP.

Now working on USER memory
Assumes a 64-bit programmable USER

USER = FFFFEEEEDDDDCCCC
#UUU Lock pass user ABCDEF01
USER = FFFFEEEEDDDDCCCC, L; Locks the USER memory
Sets the Lock bit in the USER field to a one
This action is required first in order to Permalock using PermLock directive
#UUU PermLock pass user ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Permanently locks the USER bank from secured state; TC = PL_U_L_2
Sets the PermaLock bit in the USER field to a one
#UUU Write fail 4444 user,3
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot write from open state; TC = W_U_PL_P_2
#UUU Read pass 3333 user,3,1
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory can be read from open state; TC = R_U_PL_P_2
#UUU Write fail 4444 user,3 ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot write from secured state
#UUU Read pass 3333 user,3,1 ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory can be read from secured state
#UUU Unlock fail user ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot be unlocked from secured state; TC = U_U_PL_2
#UUU PermUnLock fail user ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_U_PL_2
#UUU Lock fail user 12345678
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from secured state with incorrect AP
#UUU Lock fail user
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from open state
#UUU Lock pass user ABCDEF01
USER = FFFFEEEEDDDDCCCC, PL; AP = ABCDEF01, PL; Verifies memory can be locked while Permaunlocked from secured state; TC = L_U_PL_2

Now working on Kill password
Lock pass kpass ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
KP = AAAAAAAA, L; Locks the Kill password
Sets the Lock bit in the KP field to a one
This action is required first in order to Permalock using PermLock directive
PermLock pass kpass ABCDEF01
   PermLock result: 1
   PermLock SUCCESSFUL.
KP = AAAAAAAA, PL; Permanently locks the kill password from secured state; TC = PL_KP_L_2
Sets the PermaLock bit in the KP field to a one
This TC makes the KP permalocked, when the KP was locked and the AP is non-zero.

Write fail FFFFFFFF kpass
   Write result: 0
   Write SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot write from open state; TC = W_KP_PL_NZ_2

This TC (and the subsequent directive from the secured state) attempt to write to the KP, when the KP is permalocked and non-zero and the AP is non-zero.

Read fail AAAAAAAA kpass
   Read: FAILED
   Read SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot read from open state; TC = R_KP_PL_2
This TC (and the subsequent directive from the secured state) attempt to read the KP, when the KP is permalocked and non-zero and the AP is non-zero.

Write fail FFFFFFFF kpass ABCDEF01
   Write result: 0
   Write SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot write from secured state
Read fail AAAAAAAA kpass ABCDEF01
   Read: FAILED
   Read SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot read from secured state

Unlock fail kpass ABCDEF01
   UnLock result: 0
   UnLock SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot be unlocked from secured state; TC = U_KP_PL_2
This TC attempts to deset the lock bit to the KP, when the KP is permalocked and the AP is non-zero.

PermUnLock fail kpass ABCDEF01
   PermUnLock result: 0
   PermUnLock SUCCESSFUL.
KP = AAAAAAAA, PL; Verifies memory cannot be Permaunlocked from secured state; TC = PU_KP_PL_2
This TC attempts to deset the permalock bit to the KP, when the KP is permalocked and the AP is non-zero.

Lock fail kpass 12345678
   Lock result: 0
   Lock SUCCESSFUL.
KP = AAAAAAAA, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from secured state with incorrect AP
Lock fail kpass
   Lock result: 0
   Lock SUCCESSFUL.
KP = AAAAAAAA, PL; AP = ABCDEF01, PL; Verifies memory cannot be locked from open state
Lock pass kpass ABCDEF01
   Lock result: 1
   Lock SUCCESSFUL.
KP = AAAAAAAA, PL; AP = ABCDEF01, PL; Verifies lock bit still set from secured state; TC = L_KP_PL_2
This TC attempts to set the lock bit, when the AP is non-zero and the KP is permalocked, using an incorrect AP, no AP, and the correct AP.

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS

8. Inventory_Multiple (I_M.out.txt)

```
# This script is modified to accomodate tags with unlockable TID memory.

# Assume 10 tags with the following EPCs
# EPC = 3000300933B2DDD9014035050000
# EPC = 3000300833B2DDD9014035050000
# EPC = 3000300733B2DDD9014035050000
# EPC = 3000300633B2DDD9014035050000
# EPC = 3000300533B2DDD9014035050001
# EPC = 3000300533B2DDD9014035050002
# EPC = 3000300533B2DDD9014035050003
# EPC = 3000300433B2DDD9014035050000
# EPC = 3000300333B2DDD9014035050000
# EPC = 3000300233B2DDD9014035050000
#
# Assume that TID length equals 32 bits for programmable TID chips
# TID = E2004420
# TID = E200C0E2
# TID = E2001071
# TID = E2001050
# TID = E2001050
# TID = E2006001
# TID = E2006001
# TID = E2001050
# TID = E2006004
# TID = E2003412
#
# Assume user memory is at least 64 bits
# USER = 300933B2DDD90140
# USER = 300833B2DDD90140
# USER = 300733B2DDD90140
# USER = 300633B2DDD90140
# USER = 300533B2DDD90140
# USER = 300433B2DDD90140
# USER = 300333B2DDD90140
# USER = 300233B2DDD90140
# USER = 300133B2DDD90140
# USER = 300033B2DDD90140
#
# === Inventory Multiple Tags Section ===
#
# Tag is initially set with all zero AP and KP and unlocked memory
# AP = access password; U/L is unlocked/locked memory; TC = Test Case
#

Inventory pass 10
   Inventoried 10 unique tags.
   300733B2DDD9014035050000 :5
   300333B2DDD9014035050000 :5
   300533B2DDD9014035050001 :2
   300533B2DDD9014035050002 :5
   300633B2DDD9014035050000 :5
   300833B2DDD9014035050000 :5
   300533B2DDD9014035050003 :5
```

```
   300933B2DDD9014035050000 :5
   300433B2DDD9014035050000 :5
   300233B2DDD9014035050000 :5
   Inventory SUCCESSFUL.
# TC = I_MH or I_MM
# This TC inventories 10 tags, which either have the same IC or an IC from different vendors.

Inventory pass 1 300633B2DDD9014035050000 epc
   Inventoried 1 unique tags.
   300633B2DDD9014035050000 :28
   Inventory SUCCESSFUL.
# TC = SI_MH_E or SI_MM_E
# This TC inventories 1 tag of the 10, based on its complete EPC.

# This test case for tags with writeable TID
Inventory pass 10 E20080B0 tid
   Inventoried 10 unique tags.
   300533B2DDD9014035050001 :2
   300333B2DDD9014035050000 :4
   300433B2DDD9014035050000 :4
   300733B2DDD9014035050000 :3
   300833B2DDD9014035050000 :3
   300633B2DDD9014035050000 :2
   300533B2DDD9014035050003 :3
   300533B2DDD9014035050002 :3
   300933B2DDD9014035050000 :3
   300233B2DDD9014035050000 :3
   Inventory SUCCESSFUL.
# TC = SI_MH_T or SI_MM_T
# This test case for tags with permalocked TID
#Inventory pass 10 E2004420 tid
# TC = SI_MH_T or SI_MM_T
# This TC inventories 10 tags with the same TID, based on their complete TID.

#
# This section is for tags with user memory
# Assumption is at least 64 bits filled with alternating 10 pattern
# Individual vendors will need to adjust for their particular product
# Remove comment before Inventory directive for tags with user memory
#
# Inventory pass 1 300633B2DDD90140 user
# TC = SI_MH_U or SI_MM_U

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS
```

9. Inventory_Multiple_Mixed (I_M_Mout.txt)

```
# This script is modified to accomodate tags with unlockable TID memory.

# Assume 10 tags with the following EPCs
```

```
# EPC = 3000300933B2DDD9014035050000
# EPC = 3000300833B2DDD9014035050000
# EPC = 3000300733B2DDD9014035050000
# EPC = 3000300633B2DDD9014035050000
# EPC = 3000300533B2DDD9014035050001
# EPC = 3000300533B2DDD9014035050002
# EPC = 3000300533B2DDD9014035050003
# EPC = 3000300433B2DDD9014035050000
# EPC = 3000300333B2DDD9014035050000
# EPC = 3000300233B2DDD9014035050000
#
# Assume that TID length equals 32 bits for programmable TID chips
# TID = E2004420
# TID = E200C0E2
# TID = E2001071
# TID = E2001050
# TID = E2001050
# TID = E2006001
# TID = E2006001
# TID = E2001050
# TID = E2006004
# TID = E2003412
#
# Assume user memory is at least 64 bits
# USER = 300933B2DDD90140
# USER = 300833B2DDD90140
# USER = 300733B2DDD90140
# USER = 300633B2DDD90140
# USER = 300533B2DDD90140
# USER = 300433B2DDD90140
# USER = 300333B2DDD90140
# USER = 300233B2DDD90140
# USER = 300133B2DDD90140
# USER = 300033B2DDD90140
#
# === Inventory Multiple Tags Section ===
#
# Tag is initially set with all zero AP and KP and unlocked memory
# AP = access password; U/L is unlocked/locked memory; TC = Test Case

Inventory pass 10
   Inventoried 10 unique tags.
   300533B2DDD9014035050002 :14
   300733B2DDD9014035050000 :17
   300533B2DDD9014035050003 :14
   300233B2DDD9014035050000 :13
   300933B2DDD9014035050000 :12
   300433B2DDD9014035050000 :13
   300633B2DDD9014035050000 :13
   300333B2DDD9014035050000 :13
   300533B2DDD9014035050001 :13
   300833B2DDD9014035050000 :12
   Inventory SUCCESSFUL.
# TC = I_MH or I_MM
# This TC inventories 10 tags, which either have the same IC or an IC from different vendors.
```

```
Inventory pass 1 300633B2DDD9014035050000 epc
   Inventoried 1 unique tags.
   300633B2DDD9014035050000 :29
   Inventory SUCCESSFUL.
# TC = SI_MH_E or SI_MM_E
# This TC inventories 1 tag of the 10, based on its complete EPC.

# This test case for tags with permalocked TID
Inventory pass 1 E20080B0 tid
   Inventoried 1 unique tags.
   300933B2DDD9014035050000 :24
   Inventory SUCCESSFUL.
# TC = SI_MH_T or SI_MM_T
# This test case for tags with wrieable TID
#Inventory pass 1 E2004420 tid
# TC = SI_MH_T or SI_MM_T
# This TC inventories 10 tags with the same TID, based on their complete TID.

# This section is for tags with user memory
# Assumption is at least 64 bits filled with alternating 10 pattern
# Individual vendors will need to adjust for their particular product
# Remove comment before Inventory directive for tags with user memory
#
# Inventory pass 1 300633B2DDD90140 user
# TC = SI_MH_U or SI_MM_U

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS
```

10. SelectQuery_EPC (SQ_EPC.out.txt)

```
# Program tag with 96-bit EPC "111122223333444455556666"
WrRd pass 300011112222333344445555666 6 epc
   WrRd write result: 1
   WrRd read result: 300011112222333344445555666 6
   WrRd SUCCESSFUL.

# Write non-zero access password
Write pass 11111111 apass
   Write result: 1
   Write SUCCESSFUL.


# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
```

```
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 sl 100 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################
# SQ_E_S0_1
#
# Matching EPC, matching Query target (A->A)
SQ pass epc 96 s0 000 32 96 111122223333444455556666 0 s0 00 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s0 000 32 96 111122223333444455556666 0 s0 00 B
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 000 32 96 111122223333444455556667 0 s0 00 B
   Data length = 24 Data = 111122223333444455556667
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->B)
SQ fail epc 96 s0 000 32 96 111122223333444455556667 0 s0 11 B
   Data length = 24 Data = 111122223333444455556667
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.


# SQ_E_S0_2
#
# Matching EPC, matching Query target (A->A)
SQ pass epc 96 s0 001 34 64 44448888cccd1111 0 s0 00 A
   Data length = 16 Data = 44448888cccd1111
```

```
   SQ result: 1
   EPC result: 11112222333344445555666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s0 001 34 64 44448888cccd1111 0 s0 11 A
   Data length = 16 Data = 44448888cccd1111
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s0 001 34 64 44448888cccd11FF 0 s0 00 A
   Data length = 16 Data = 44448888cccd11FF
   SQ result: 1
   EPC result: 11112222333344445555666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A)
SQ fail epc 96 s0 001 34 64 44448888cccd11FF 0 s0 00 B
   Data length = 16 Data = 44448888cccd11FF
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_3
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s0 010 37 48 222444466668 0 s0 00 A
   Data length = 12 Data = 222444466668
   SQ result: 1
   EPC result: 11112222333344445555666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s0 010 37 48 222444466668 0 s0 00 B
   Data length = 12 Data = 222444466668
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 010 37 48 2224444666FF 0 s0 00 B
   Data length = 12 Data = 2224444666FF
   SQ result: 1
   EPC result: 11112222333344445555666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Session (A->B)
SQ fail epc 96 s0 010 37 48 2224444666FF 0 s0 00 A
   Data length = 12 Data = 2224444666FF
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_4
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 011 43 24 891111 0 s0 00 B
   Data length = 6 Data = 891111
   SQ result: 1
```

```
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Session (A->B)
SQ fail epc 96 s0 011 43 24 891111 0 s1 00 B
   Data length = 6 Data = 891111
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s0 011 43 24 89111F 0 s0 00 A
   Data length = 6 Data = 89111F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A)
SQ fail epc 96 s0 011 43 24 89111F 0 s0 00 B
   Data length = 6 Data = 89111F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_5
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 100 55 12 111 0 s0 00 B
   Data length = 3 Data = 1110
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->B)
SQ fail epc 96 s0 100 55 12 111 0 s0 00 A
   Data length = 3 Data = 1110
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->A)
SQ pass epc 96 s0 100 55 12 11F 0 s0 00 A
   Data length = 3 Data = 11F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s0 100 55 12 11F 0 s0 11 A
   Data length = 3 Data = 11F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_6
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 101 79 6 A0 0 s0 00 B
   Data length = 2 Data = A0
   SQ result: 1
   EPC result: 111122223333444455556666
```

```
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Session (A->B)
SQ fail epc 96 s0 101 79 6 A0 0 s3 00 B
   Data length = 2 Data = A0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s0 101 79 6 AF 0 s0 00 A
   Data length = 2 Data = AF
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A)
SQ fail epc 96 s0 101 79 6 AF 0 s0 00 B
   Data length = 2 Data = AF
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_7
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s0 110 95 3 2 0 s0 00 A
   Data length = 1 Data = 20
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s0 110 95 3 2 0 s0 00 B
   Data length = 1 Data = 20
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->A)
SQ pass epc 96 s0 110 95 3 F 0 s0 00 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s0 110 95 3 F 0 s0 11 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S0_8
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s0 111 127 1 0 0 s0 00 A
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
```

```
# Matching EPC, non-matching Query Sel (A)
SQ fail epc 96 s0 111 127 1 0 0 s0 11 A
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s0 111 127 1 F 0 s0 00 B
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A->B)
SQ fail epc 96 s0 111 127 1 F 0 s0 00 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################

# Lock epc
Lock pass epc 11111111
   Lock result: 1
   Lock SUCCESSFUL.

# SQ_E_S1_1
#
# Matching EPC, matching Query target (A->A)
SQ pass epc 96 s1 000 117 1 8 0 s1 00 A
   Data length = 1 Data = 80
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s1 000 117 1 8 0 s1 00 B
   Data length = 1 Data = 80
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s1 000 117 1 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Session (A->B)
SQ fail epc 96 s1 000 117 1 0 0 s1 00 A
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_2
#
```

```
# Matching EPC, matching Query target (B->A)
SQ pass epc 96 s1 001 97 3 A 0 s1 00 A
   Data length = 1 Data = A0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (B->A)
SQ fail epc 96 s1 001 97 3 A 0 s1 11 A
   Data length = 1 Data = A0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s1 001 97 3 F 0 s1 00 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A)
SQ fail epc 96 s1 001 97 3 F 0 s1 00 B
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_3
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s1 010 83 6 20 0 s1 00 A
   Data length = 2 Data = 20
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s1 010 83 6 20 0 s1 00 B
   Data length = 2 Data = 20
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s1 010 83 6 2F 0 s1 00 B
   Data length = 2 Data = 2F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Session (A->B)
SQ fail epc 96 s1 010 83 6 2F 0 s0 00 B
   Data length = 2 Data = 2F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
```

```
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_4
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s1 011 64 12 333 0 s1 00 B
   Data length = 3 Data = 3330
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Give an extra command so that the flag will toggle back to A
#Read pass - epc
# Matching EPC, non-matching Query Sel (B->A)
SQ fail epc 96 s1 011 64 12 333 0 s1 11 B
   Data length = 3 Data = 3330
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s1 011 64 12 33F 0 s1 00 A
   Data length = 3 Data = 33F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Give flag time to toggle, due to matching Query
#Read pass - epc 11111111
# Non-matching EPC, non-matching Query target (A)
SQ fail epc 96 s1 011 64 12 33F 0 s1 00 B
   Data length = 3 Data = 33F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
   Data length = 1 Data = 00
```

```
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_5
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s1 100 46 24 48888C 0 s1 00 B
   Data length = 6 Data = 48888C
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->B)
SQ fail epc 96 s1 100 46 24 48888C 0 s1 00 A
   Data length = 6 Data = 48888C
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s1 100 46 24 48888F 0 s1 00 A
   Data length = 6 Data = 48888F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s1 100 46 24 48888F 0 s1 11 A
   Data length = 6 Data = 48888F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_6
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s1 101 42 48 448888cccd11 0 s1 00 B
   Data length = 12 Data = 448888cccd11
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (B->B)
```

SQ fail epc 96 s1 101 42 48 448888cccd11 0 s1 00 A
Data length = 12 Data = 448888cccd11
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching EPC, matching Query target (B)
SQ pass epc 96 s1 101 42 48 448888cccd1F 0 s1 00 B
Data length = 12 Data = 448888cccd1F
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Non-matching EPC, non-matching Query Session (B)
SQ fail epc 96 s1 101 42 48 448888cccd1F 0 s3 00 B
Data length = 12 Data = 448888cccd1F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

Send SELECT with 0 length mask and Action 000 to send Sx INV to A
Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
Data length = 1 Data = 00
SQ result: 0
EPC result:
SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
Data length = 1 Data = 00
SQ result: 0
EPC result:
SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
Data length = 1 Data = 00
SQ result: 0
EPC result:
SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
Data length = 1 Data = 00
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_E_S1_7
#
Matching EPC, matching Query target (A)
SQ pass epc 96 s1 110 38 64 4448888cccd11115 0 s1 00 A
Data length = 16 Data = 4448888cccd11115
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Matching EPC, non-matching Query Session (B)
SQ fail epc 96 s1 110 38 64 4448888cccd11115 0 s0 00 B
Data length = 16 Data = 4448888cccd11115
SQ result: 0
EPC result:
SQ SUCCESSFUL.

```
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s1 110 38 64 4448888cccd1111F 0 s1 00 A
   Data length = 16 Data = 4448888cccd1111F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->A)
SQ fail epc 96 s1 110 38 64 4448888cccd1111F 0 s1 00 B
   Data length = 16 Data = 4448888cccd1111F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S1_8
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s1 111 32 96 111122223333444455556666 0 s1 00 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s1 111 32 96 111122223333444455556666 0 s1 00 B
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A->B)
SQ pass epc 96 s1 111 32 96 11112222333344445555666F 0 s1 00 B
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (B->A)
SQ fail epc 96 s1 111 32 96 11112222333344445555666F 0 s1 11 A
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Unlock epc
UnLock pass epc 11111111
   UnLock result: 1
   UnLock SUCCESSFUL.

# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################

# SQ_E_S2_1
#
# Matching EPC, matching Query target (A->A)
SQ pass epc 96 s2 000 32 96 111122223333444455556666 0 s2 00 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s2 000 32 96 111122223333444455556666 0 s2 11 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s2 000 32 96 11112222333344445555666F 0 s2 00 B
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->B)
SQ fail epc 96 s2 000 32 96 11112222333344445555666F 0 s2 00 A
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_2
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->A)
SQ pass epc 96 s2 001 32 95 111122223333444455556666 0 s2 00 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s2 001 32 95 111122223333444455556666 0 s2 00 B
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s2 001 32 95 111122223333444455556666F 0 s2 00 A
   Data length = 24 Data = 111122223333444455556666F
   SQ result: 1
   EPC result: 11112222333344445555666 6
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A)
SQ fail epc 96 s2 001 32 95 111122223333444455556666F 0 s2 11 A
   Data length = 24 Data = 111122223333444455556666F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_3
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s2 010 44 48 122223333444 0 s2 00 A
   Data length = 12 Data = 122223333444
   SQ result: 1
   EPC result: 11112222333344445555666 6
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s2 010 44 48 122223333444 0 s2 00 B
   Data length = 12 Data = 122223333444
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s2 010 44 48 12222333344F 0 s2 00 B
   Data length = 12 Data = 12222333344F
   SQ result: 1
   EPC result: 11112222333344445555666 6
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query session (B->B)
SQ fail epc 96 s2 010 44 48 12222333344F 0 s1 00 B
   Data length = 12 Data = 12222333344F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_4
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->A)
SQ pass epc 96 s2 011 56 30 22333344 0 s2 00 A
   Data length = 8 Data = 22333344
   SQ result: 1
   EPC result: 11112222333344445555666 6
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query session (A->B)
SQ fail epc 96 s2 011 56 30 22333344 0 s0 00 B
   Data length = 8 Data = 22333344
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B)
SQ pass epc 96 s2 011 56 30 2233334F 0 s2 00 B
   Data length = 8 Data = 2233334F
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (B)
SQ fail epc 96 s2 011 56 30 2233334F 0 s2 00 A
   Data length = 8 Data = 2233334F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_5
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->B)
SQ pass epc 96 s2 100 67 20 999A2 0 s2 00 B
   Data length = 5 Data = 999A20
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (B->B)
SQ fail epc 96 s2 100 67 20 999A2 0 s2 00 A
   Data length = 5 Data = 999A20
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s2 100 67 20 999AF 0 s2 00 A
   Data length = 5 Data = 999AF0
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s2 100 67 20 999AF 0 s2 11 A
   Data length = 5 Data = 999AF0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_6
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s2 101 75 6 98 0 s2 00 B
   Data length = 2 Data = 98
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (B->B)
SQ fail epc 96 s2 101 75 6 98 0 s2 00 A
   Data length = 2 Data = 98
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B)
SQ pass epc 96 s2 101 75 6 9F 0 s2 00 B
   Data length = 2 Data = 9F
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query session (B)
SQ fail epc 96 s2 101 75 6 9F 0 s3 00 B
   Data length = 2 Data = 9F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_7
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B)
SQ pass epc 96 s2 110 83 3 2 0 s2 00 B
   Data length = 1 Data = 20
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Session (B)
SQ fail epc 96 s2 110 83 3 2 0 s1 00 B
   Data length = 1 Data = 20
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s2 110 83 3 F 0 s2 00 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s2 110 83 3 F 0 s2 00 B
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S2_8
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s2 111 126 1 8 0 s2 00 A
   Data length = 1 Data = 80
   SQ result: 1
   EPC result: 11112222333334444555566666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s2 111 126 1 8 0 s2 00 B
   Data length = 1 Data = 80
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s2 111 126 1 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (B->A)
SQ fail epc 96 s2 111 126 1 0 0 s2 11 A
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Lock epc
Lock pass epc 11111111
   Lock result: 1
   Lock SUCCESSFUL.

# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################

# SQ_E_S3_1
#
# Matching EPC, matching Query target (A->A)
SQ pass epc 96 s3 000 124 1 0 0 s3 00 A
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s3 000 124 1 0 0 s3 11 B
```

```
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s3 000 124 1 F 0 s3 00 B
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (B->B)
SQ fail epc 96 s3 000 124 1 F 0 s3 00 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_2
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->A)
SQ pass epc 96 s3 001 102 3 4 0 s3 00 A
   Data length = 1 Data = 40
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s3 001 102 3 4 0 s3 00 B
   Data length = 1 Data = 40
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A)
SQ pass epc 96 s3 001 102 3 4 0 s3 00 A
   Data length = 1 Data = 40
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A)
SQ fail epc 96 s3 001 102 3 4 0 s3 11 B
   Data length = 1 Data = 40
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_3
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s3 010 87 6 20 0 s3 00 A
   Data length = 2 Data = 20
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s3 010 87 6 20 0 s3 00 B
```

```
   Data length = 2 Data = 20
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s3 010 87 6 2F 0 s3 00 B
   Data length = 2 Data = 2F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query session (B->B)
SQ fail epc 96 s3 010 87 6 2F 0 s1 00 B
   Data length = 2 Data = 2F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_4
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->A)
SQ pass epc 96 s3 011 59 12 119 0 s3 00 A
   Data length = 3 Data = 1190
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query session (A->B)
SQ fail epc 96 s3 011 59 12 119 0 s2 00 B
   Data length = 3 Data = 1190
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B)
SQ pass epc 96 s3 011 59 12 11F 0 s3 00 B
   Data length = 3 Data = 11F0
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (B)
SQ fail epc 96 s3 011 59 12 11F 0 s3 00 A
   Data length = 3 Data = 11F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_5
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B->B)
SQ pass epc 96 s3 100 45 24 244446 0 s3 00 B
   Data length = 6 Data = 244446
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (B->B)
```

```
SQ fail epc 96 s3 100 45 24 244446 0 s3 00 A
   Data length = 6 Data = 244446
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s3 100 45 24 24444F 0 s3 00 A
   Data length = 6 Data = 24444F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (A->A)
SQ fail epc 96 s3 100 45 24 24444F 0 s3 11 A
   Data length = 6 Data = 24444F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_6
#
# Matching EPC, matching Query target (A->B)
SQ pass epc 96 s3 101 41 48 224444666688 0 s3 00 B
   Data length = 12 Data = 224444666688
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query session (B->B)
SQ fail epc 96 s3 101 41 48 224444666688 0 s0 00 B
   Data length = 12 Data = 224444666688
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B)
SQ pass epc 96 s3 101 41 48 22444466668F 0 s3 00 B
   Data length = 12 Data = 22444466668F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (B)
SQ fail epc 96 s3 101 41 48 22444466668F 0 s3 00 A
   Data length = 12 Data = 22444466668F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_7
#
# Tag is in B from previous test ( persistent session )
# Matching EPC, matching Query target (B)
SQ pass epc 96 s3 110 36 64 1112222333344445 0 s3 00 B
   Data length = 16 Data = 1112222333344445
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query session (B)
```

```
SQ fail epc 96 s3 110 36 64 1112222333344445 0 s1 00 B
   Data length = 16 Data = 1112222333344445
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (B->A)
SQ pass epc 96 s3 110 36 64 111222233334444F 0 s3 00 A
   Data length = 16 Data = 111222233334444F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (A->A)
SQ fail epc 96 s3 110 36 64 111222233334444F 0 s3 00 B
   Data length = 16 Data = 111222233334444F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_S3_8
#
# Matching EPC, matching Query target (A)
SQ pass epc 96 s3 111 32 96 111122223333444455556666 0 s3 00 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (A)
SQ fail epc 96 s3 111 32 96 111122223333444455556666 0 s3 00 B
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query target (A->B)
SQ pass epc 96 s3 111 32 96 11112222333344445555666F 0 s3 00 B
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 1
   EPC result: 111122223333444455556666
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (B->A)
SQ fail epc 96 s3 111 32 96 11112222333344445555666F 0 s3 11 B
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Unlock epc
UnLock pass epc 11111111
   UnLock result: 1
   UnLock SUCCESSFUL.

# Send SELECT with 0 length mask and Action 000 to send Sx INV to A
# Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail epc 96 s0 000 32 0 0 0 s0 00 B
   Data length = 1 Data = 00
   SQ result: 0
```

```
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s1 000 32 0 0 0 s1 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s2 000 32 0 0 0 s2 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
SQ fail epc 96 s3 000 32 0 0 0 s3 00 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################

# SQ_E_SL_1
#
# Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 000 126 2 8 1 s0 11 A
   Data length = 1 Data = 80
   SQ result: 1
   EPC result: 00000000000000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 000 126 2 8 1 s0 11 B
   Data length = 1 Data = 80
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 000 126 2 F 1 s0 10 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 000000000000000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL->~SL)
SQ fail epc 96 sl 000 126 2 F 1 s0 11 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_2
#
# Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 001 63 64 1999a2222aaab333 1 s0 11 A
   Data length = 16 Data = 1999a2222aaab333
   SQ result: 1
   EPC result: A2222AAAB3330C1A00000000
   SQ SUCCESSFUL.
```

```
# Matching EPC, non-matching Query Sel (SL)
SQ fail epc 96 sl 001 63 64 1999a2222aaab333 1 s0 10 A
   Data length = 16 Data = 1999a2222aaab333
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 001 63 64 1999a2222aaab33F 1 s0 11 A
   Data length = 16 Data = 1999a2222aaab33F
   SQ result: 1
   EPC result: A2222AAAB33F300200000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 001 63 64 1999a2222aaab33F 1 s0 11 B
   Data length = 16 Data = 1999a2222aaab33F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_3
#
# Tag has SL from previous test ( SL flag has persistence )
# Matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 010 127 1 0 1 s0 11 A
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 000000000000000000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Target (SL)
SQ fail epc 96 sl 010 127 1 0 1 s0 11 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 010 127 1 F 1 s0 10 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 000000000000000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL->~SL)
SQ fail epc 96 sl 010 127 1 F 1 s0 11 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_4
#
# Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 011 63 8 19 1 s0 11 A
   Data length = 2 Data = 19
   SQ result: 1
   EPC result: A2222AAAB3330C1A00000000
   SQ SUCCESSFUL.
```

```
# Matching EPC, non-matching Query target (SL->~SL)
SQ fail epc 96 sl 011 63 8 19 1 s0 10 B
   Data length = 2 Data = 19
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 011 63 8 1F 1 s0 10 A
   Data length = 2 Data = 1F
   SQ result: 1
   EPC result: 24444666688800000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL)
SQ fail epc 96 sl 011 63 8 1F 1 s0 11 A
   Data length = 2 Data = 1F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_5
#
# Matching EPC, matching Query Sel (~SL->~SL)
SQ pass epc 96 sl 100 58 12 88C 1 s0 10 A
   Data length = 3 Data = 88C0
   SQ result: 1
   EPC result: CD1111555599986000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (~SL->~SL)
SQ fail epc 96 sl 100 58 12 88C 1 s0 11 A
   Data length = 3 Data = 88C0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 100 58 12 88F 1 s0 11 A
   Data length = 3 Data = 88F0
   SQ result: 1
   EPC result: 0022224444666688800000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (SL->SL)
SQ fail epc 96 sl 100 58 12 88F 1 s0 11 B
   Data length = 3 Data = 88F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_6
#
# Tag has SL from previous test ( SL flag has persistence )
# Matching EPC, matching Query SL (SL->~SL)
SQ pass epc 96 sl 101 81 16 8888 1 s0 10 A
   Data length = 4 Data = 8888
   SQ result: 1
   EPC result: AAAACCCC3000000000000000
   SQ SUCCESSFUL.
```

```
# Matching EPC, non-matching Query target (~SL->~SL)
SQ fail epc 96 sl 101 81 16 8888 1 s0 10 B
   Data length = 4 Data = 8888
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 101 81 16 888F 1 s0 10 A
   Data length = 4 Data = 888F
   SQ result: 1
   EPC result: 300222244400000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL)
SQ fail epc 96 sl 101 81 16 8888 1 s0 11 A
   Data length = 4 Data = 8888
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_7
#
# Matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 110 32 5 10 1 s0 10 A
   Data length = 2 Data = 10
   SQ result: 1
   EPC result: 222233334444555566661800
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (~SL)
SQ fail epc 96 sl 110 32 5 10 1 s0 11 A
   Data length = 2 Data = 10
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 110 32 5 1F 1 s0 11 A
   Data length = 2 Data = 1F
   SQ result: 1
   EPC result: 111122223333444455556600
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (SL->SL)
SQ fail epc 96 sl 110 32 5 1F 1 s0 11 B
   Data length = 2 Data = 1F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_8
#
# Tag has SL from previous test ( SL flag has persistence )
# Matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 111 124 3 6 1 s0 11 A
   Data length = 1 Data = 60
   SQ result: 1
   EPC result: 000000000000000000000000
   SQ SUCCESSFUL.
```

```
# Matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 111 124 3 6 1 s0 11 B
   Data length = 1 Data = 60
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 111 124 3 F 1 s0 10 A
   Data length = 1 Data = F0
   SQ result: 1
   EPC result: 00000000000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL->SL)
SQ fail epc 96 sl 111 124 3 F 1 s0 10 A
   Data length = 1 Data = F0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Send SELECT with 0 length mask and Action 011 to send SL to ~SL
# Do not send correct query to avoid flipping S0 INV flag to B by ACKing it
SQ fail epc 96 sl 011 32 0 0 0 s0 10 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

################################################################################

# Lock epc
Lock pass epc 11111111
   Lock result: 1
   Lock SUCCESSFUL.

# SQ_E_SL_9
#
# Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 000 32 96 111122223333444455556666 1 s0 11 A
   Data length = 24 Data = 111122223333444455556666
   SQ result: 1
   EPC result: 222233334444555566661835
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 000 32 96 111122223333444455556666 1 s0 11 B
   Data length = 24 Data = 111122223333444455556666
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 000 32 96 11112222333344445555666F 1 s0 10 A
   Data length = 24 Data = 11112222333344445555666F
   SQ result: 1
   EPC result: 2222333344445555666F3002
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (~SL->~SL)
```

SQ fail epc 96 sl 000 32 96 111122223333444455556666F 1 s0 11 A
Data length = 24 Data = 111122223333444455556666F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_E_SL_10
#
Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 001 32 95 111122223333444455556666 1 s0 11 A
Data length = 24 Data = 111122223333444455556666
SQ result: 1
EPC result: 222233334444555566661800
SQ SUCCESSFUL.
Matching EPC, non-matching Query Sel (SL)
SQ fail epc 96 sl 001 32 95 111122223333444455556666 1 s0 10 A
Data length = 24 Data = 111122223333444455556666
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 001 32 95 111122223333444455556666F 1 s0 11 A
Data length = 24 Data = 111122223333444455556666F
SQ result: 1
EPC result: 2222333344445555666E6000
SQ SUCCESSFUL.
Non-matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 001 32 95 111122223333444455556666F 1 s0 11 B
Data length = 24 Data = 111122223333444455556666F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_E_SL_11
#
Tag has SL from previous test (SL flag has persistence)
Matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 010 44 48 122223333444 1 s0 11 A
Data length = 12 Data = 122223333444
SQ result: 1
EPC result: 34444555566661830000
SQ SUCCESSFUL.
Matching EPC, non-matching Query target (SL)
SQ fail epc 96 sl 010 44 48 122223333444 1 s0 11 B
Data length = 12 Data = 122223333444
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 010 44 48 12222333344F 1 s0 10 A
Data length = 12 Data = 12222333344F
SQ result: 1
EPC result: 344F3002222444460000
SQ SUCCESSFUL.
Non-matching EPC, non-matching Query Sel (~SL->~SL)

SQ fail epc 96 sl 010 44 48 12222333344F 1 s0 11 A
Data length = 12 Data = 12222333344F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_E_SL_12
#
Matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 011 56 30 22333344 1 s0 11 A
Data length = 8 Data = 22333344
SQ result: 1
EPC result: 44555566661800000000
SQ SUCCESSFUL.
Matching EPC, non-matching Query Sel (SL->~SL)
SQ fail epc 96 sl 011 56 30 22333344 1 s0 11 A
Data length = 8 Data = 22333344
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 011 56 30 2233334F 1 s0 10 A
Data length = 8 Data = 2233334F
SQ result: 1
EPC result: C0088891111900000000
SQ SUCCESSFUL.
Non-matching EPC, non-matching Query target (~SL)
SQ fail epc 96 sl 011 56 30 2233334F 1 s0 10 B
Data length = 8 Data = 2233334F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_E_SL_13
#
Matching EPC, matching Query Sel (~SL->~SL)
SQ pass epc 96 sl 100 67 20 999A2 1 s0 10 A
Data length = 5 Data = 999A20
SQ result: 1
EPC result: 2222AAAB3330C10000000000
SQ SUCCESSFUL.
Matching EPC, non-matching Query target (~SL->~SL)
SQ fail epc 96 sl 100 67 20 999A2 1 s0 10 B
Data length = 5 Data = 999A20
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 100 67 20 999AF 1 s0 11 A
Data length = 5 Data = 999AF0
SQ result: 1
EPC result: F30022224444660000000000
SQ SUCCESSFUL.
Non-matching EPC, non-matching Query Sel (SL->SL)
SQ fail epc 96 sl 100 67 20 999AF 1 s0 10 A

```
   Data length = 5 Data = 999AF0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_14
#
# Tag has SL from previous test ( SL flag has persistence )
# Matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 101 75 6 9A 1 s0 10 A
   Data length = 2 Data = 9A
   SQ result: 1
   EPC result: AB333000000000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (~SL->~SL)
SQ fail epc 96 sl 101 75 6 9A 1 s0 11 A
   Data length = 2 Data = 9A
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 101 75 6 9F 1 s0 10 A
   Data length = 2 Data = 9F
   SQ result: 1
   EPC result: 088891111900000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (~SL)
SQ fail epc 96 sl 101 75 6 9F 1 s0 10 B
   Data length = 2 Data = 9F
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_15
#
# Matching EPC, matching Query Sel (~SL)
SQ pass epc 96 sl 110 95 13 2AAA 1 s0 10 A
   Data length = 4 Data = 2AAA
   SQ result: 1
   EPC result: 0000000000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query target (~SL)
SQ fail epc 96 sl 110 95 13 2AAA 1 s0 10 B
   Data length = 4 Data = 2AAA
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (~SL->SL)
SQ pass epc 96 sl 110 95 13 2AA0 1 s0 11 A
   Data length = 4 Data = 2AA0
   SQ result: 1
   EPC result: 0000000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query Sel (SL->SL)
SQ fail epc 96 sl 110 95 13 2AA0 1 s0 10 A
```

```
   Data length = 4 Data = 2AA0
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# SQ_E_SL_16
#
# Tag has SL from previous test ( SL flag has persistence )
# Matching EPC, matching Query Sel (SL)
SQ pass epc 96 sl 111 126 1 8 1 s0 11 A
   Data length = 1 Data = 80
   SQ result: 1
   EPC result: 00000000000000000000
   SQ SUCCESSFUL.
# Matching EPC, non-matching Query Sel (SL)
SQ fail epc 96 sl 111 126 1 8 1 s0 10 A
   Data length = 1 Data = 80
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
# Non-matching EPC, matching Query Sel (SL->~SL)
SQ pass epc 96 sl 111 126 1 0 1 s0 10 A
   Data length = 1 Data = 00
   SQ result: 1
   EPC result: 000000000000000000000000
   SQ SUCCESSFUL.
# Non-matching EPC, non-matching Query target (~SL->SL)
SQ fail epc 96 sl 111 126 1 0 1 s0 11 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.

# Unlock epc
UnLock pass epc 11111111
   UnLock result: 1
   UnLock SUCCESSFUL.

# Send SELECT with 0 length mask and Action 011 to send SL to ~SL
# Do not send correct query to avoid flipping S0 INV flag to B by ACKing it
SQ fail epc 96 sl 011 32 0 0 0 s0 10 B
   Data length = 1 Data = 00
   SQ result: 0
   EPC result:
   SQ SUCCESSFUL.
################################################################################
# Write zero access password
Write pass 00000000 apass
   Write result: 1
   Write SUCCESSFUL.

disconnect
   Disconnected.

   OVERALL RESULT: SUCCESS
```

11. SelectQuery_TID (SQ_TID.out.txt)

Confirm that the TID is E2001050
Read pass E200B080 tid
  Using TID E200B080
  Read: E200B080
  Read SUCCESSFUL.
Set the AP to a non-zero value
Write pass 11111111 apass
  Write result: 1
  Write SUCCESSFUL.

Send SELECT with 0 length mask and Action 000 to send Sx INV to A
Do not send correct query to avoid flipping flag back to B by ACKing it
SQ fail tid 96 s0 000 0 0 0 0 s0 00 B
  Data length = 1 Data = 00
  SQ result: 0
  EPC result:
  SQ SUCCESSFUL.
SQ fail tid 96 s1 000 0 0 0 0 s1 00 B
  Data length = 1 Data = 00
  SQ result: 0
  EPC result:
  SQ SUCCESSFUL.
SQ fail tid 96 s2 000 0 0 0 0 s2 00 B
  Data length = 1 Data = 00
  SQ result: 0
  EPC result:
  SQ SUCCESSFUL.
SQ fail tid 96 s3 000 0 0 0 0 s3 00 B
  Data length = 1 Data = 00
  SQ result: 0
  EPC result:
  SQ SUCCESSFUL.
SQ fail tid 96 sl 100 0 0 0 0 s3 00 B
  Data length = 1 Data = 00

SQ result: 0

EPC result:

SQ SUCCESSFUL.

##

SQ_T_S0_1

Matching TID, matching Query target (A->A)

SQ pass tid 96 s0 000 0 16 E200 0 s0 00 A

Data length = 4 Data = E200

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Matching TID, non-matching Query Sel (A->A)

SQ fail tid 96 s0 000 0 16 E200 0 s0 11 B

Data length = 4 Data = E200

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query target (A->B)

SQ pass tid 96 s0 000 0 16 E20F 0 s0 00 B

Data length = 4 Data = E20F

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Non-matching TID, non-matching Query target (A->B)

SQ fail tid 96 s0 000 0 16 E20F 0 s0 00 A

Data length = 4 Data = E20F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

SQ_T_S0_2

Matching TID, matching Query target (A->A)

SQ pass tid 96 s0 001 0 32 E200B080 0 s0 00 A

Data length = 8 Data = E200B080

SQ result: 1

EPC result: 11112222333344445555666

SQ SUCCESSFUL.

Matching TID, non-matching Query target (A->A)

SQ fail tid 96 s0 001 0 32 E200B080 0 s0 00 B

Data length = 8 Data = E200B080

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query target (A)

SQ pass tid 96 s0 001 0 32 E200104F 0 s0 00 A

Data length = 8 Data = E200104F

SQ result: 1

EPC result: 11112222333344445555666

SQ SUCCESSFUL.

Non-matching TID, non-matching Query Sel (A)

SQ fail tid 96 s0 001 0 32 E200104F 0 s0 11 B

Data length = 8 Data = E200104F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

##

Lock the TID

Lock pass tid 11111111

Lock result: 1

Lock SUCCESSFUL.

SQ_T_S1_1

Matching TID, matching Query target (A)

SQ pass tid 96 s1 010 0 16 E200 0 s1 00 A

Data length = 4 Data = E200

SQ result: 1

EPC result: 11112222333344445555666

SQ SUCCESSFUL.

Matching TID, non-matching Query target (A)

SQ fail tid 96 s1 010 0 16 E200 0 s1 00 B

Data length = 4 Data = E200

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query target (A->B)

SQ pass tid 96 s1 010 0 16 E20F 0 s1 00 B

Data length = 4 Data = E20F

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Non-matching TID, non-matching Query session (B->B)

SQ fail tid 96 s1 010 0 16 E20F 0 s1 00 A

Data length = 4 Data = E20F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Send SELECT with 0 length mask and Action 000 to send S1 INV to A

Do not send correct query to avoid flipping S0 INV flag to B by ACKing it

SQ fail tid 96 s1 000 0 0 0 0 s1 00 B

Data length = 1 Data = 00

SQ result: 0

EPC result:

SQ SUCCESSFUL.

SQ_T_S1_2

#

Matching TID, matching Query target (A->B)

SQ pass tid 96 s1 011 0 32 E200B080 0 s1 00 B

Data length = 8 Data = E200B080

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Give the flag time to toggle with the next command

#Read pass - tid

Matching TID, non-matching Query session (B->A)

SQ fail tid 96 s1 011 0 32 E200B080 0 s1 00 B

Data length = 8 Data = E200B080

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query target (A)

SQ pass tid 96 s1 011 0 32 E200104F 0 s1 00 A

Data length = 8 Data = E200104F

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Give the flag time to toggle with the next command

#Read pass - tid

Non-matching TID, non-matching Query target (A)

SQ fail tid 96 s1 011 0 32 E200104F 0 s1 00 B

Data length = 8 Data = E200104F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

If TID capable of being unlocked:

#Unlock pass tid 11111111

Send SELECT with 0 length mask and Action 000 to send S1 INV to A

Do not send correct query to avoid flipping S0 INV flag to B by ACKing it

SQ fail tid 96 s1 000 0 0 0 0 s1 00 B

Data length = 1 Data = 00

SQ result: 0

EPC result:

SQ SUCCESSFUL.

##

SQ_T_S2_1

#

Matching TID, matching Query target (A->B)

SQ pass tid 96 s2 100 0 16 E200 0 s2 00 B

Data length = 4 Data = E200

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Matching TID, non-matching Query target (B->B)
SQ fail tid 96 s2 100 0 16 E200 0 s2 00 A
Data length = 4 Data = E200
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching TID, matching Query target (B->A)
SQ pass tid 96 s2 100 0 16 E20F 0 s2 00 A
Data length = 4 Data = E20F
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Non-matching TID, non-matching Query target (A->A)
SQ fail tid 96 s2 100 0 16 E20F 0 s2 00 B
Data length = 4 Data = E20F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_T_S2_2
Matching TID, matching Query target (A->B)
SQ pass tid 96 s2 101 0 32 E200B080 0 s2 00 B
Data length = 8 Data = E200B080
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Matching TID, non-matching Query Session (B->B)
SQ fail tid 96 s2 101 0 32 E200B080 0 s0 00 B
Data length = 8 Data = E200B080
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching TID, matching Query target (B)
SQ pass tid 96 s2 101 0 32 E200104F 0 s2 00 B
Data length = 8 Data = E200104F
SQ result: 1

EPC result: 11112222333344445555666 6

SQ SUCCESSFUL.

Non-matching TID, non-matching Query target (B)

SQ fail tid 96 s2 101 0 32 E200104F 0 s2 00 A

Data length = 8 Data = E200104F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Lock the TID

Lock pass tid 11111111

Lock result: 1

Lock SUCCESSFUL.

Send SELECT with 0 length mask and Action 000 to send S2 INV to A

Do not send correct query to avoid flipping S0 INV flag to B by ACKing it

SQ fail tid 96 s2 000 0 0 0 0 s2 00 B

Data length = 1 Data = 00

SQ result: 0

EPC result:

SQ SUCCESSFUL.

##

SQ_T_S3_1

#

Matching TID, matching Query target (A)

SQ pass tid 96 s3 110 0 16 E200 0 s3 00 A

Data length = 4 Data = E200

SQ result: 1

EPC result: 11112222333344445555666 6

SQ SUCCESSFUL.

Matching TID, non-matching Query target (A)

SQ fail tid 96 s3 110 0 16 E200 0 s3 00 B

Data length = 4 Data = E200

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query target (B->A)
SQ pass tid 96 s3 110 0 16 E20F 0 s3 00 A
Data length = 4 Data = E20F
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Non-matching TID, non-matching Query Sel (A->A)
SQ fail tid 96 s3 110 0 16 E20F 0 s3 11 A
Data length = 4 Data = E20F
SQ result: 0
EPC result:
SQ SUCCESSFUL.

SQ_T_S3_2
#
Matching TID, matching Query target (A)
SQ pass tid 96 s3 111 0 32 E200B080 0 s3 00 A
Data length = 8 Data = E200B080
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Matching TID, non-matching Query target (A)
SQ fail tid 96 s3 111 0 32 E200B080 0 s3 00 B
Data length = 8 Data = E200B080
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching TID, matching Query target (A->B)
SQ pass tid 96 s3 111 0 32 E200104F 0 s3 00 B
Data length = 8 Data = E200104F
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Non-matching TID, non-matching Query Sel (B->A)
SQ fail tid 96 s3 111 0 32 E200104F 0 s3 11 A
Data length = 8 Data = E200104F

SQ result: 0
EPC result:
SQ SUCCESSFUL.

If TID capable of being unlocked:
#Unlock pass tid 11111111

Send SELECT with 0 length mask and Action 000 to send S3 INV to A
Do not send correct query to avoid flipping S0 INV flag to B by ACKing it
SQ fail tid 96 s3 000 0 0 0 0 s3 00 B
Data length = 1 Data = 00
SQ result: 0
EPC result:
SQ SUCCESSFUL.
##
SQ_T_SL_1
#
Matching TID, matching Query Sel (~SL->SL)
SQ pass tid 96 sl 000 0 16 E200 0 s0 11 A
Data length = 4 Data = E200
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Matching TID, non-matching Query Sel (SL->SL)
SQ fail tid 96 sl 000 0 16 E200 0 s0 10 A
Data length = 4 Data = E200
SQ result: 0
EPC result:
SQ SUCCESSFUL.
Non-matching TID, matching Query Sel (SL->~SL)
SQ pass tid 96 sl 000 0 16 E20F 0 s0 10 A
Data length = 4 Data = E20F
SQ result: 1
EPC result: 111122223333444455556666
SQ SUCCESSFUL.
Non-matching TID, non-matching Query target (~SL->~SL)

SQ fail tid 96 sl 000 0 16 E20F 0 s0 10 B

Data length = 4 Data = E20F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

SQ_T_SL_2

#

Matching TID, matching Query Sel (~SL->SL)

SQ pass tid 96 sl 001 0 32 E200B080 0 s0 11 A

Data length = 8 Data = E200B080

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Matching TID, non-matching Query target (SL)

SQ fail tid 96 sl 001 0 32 E200B080 0 s0 11 B

Data length = 8 Data = E200B080

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Non-matching TID, matching Query Sel (SL)

SQ pass tid 96 sl 001 0 32 E200104F 0 s0 11 A

Data length = 8 Data = E200104F

SQ result: 1

EPC result: 111122223333444455556666

SQ SUCCESSFUL.

Non-matching TID, non-matching Query Sel (SL)

SQ fail tid 96 sl 001 0 32 E200104F 0 s0 10 A

Data length = 8 Data = E200104F

SQ result: 0

EPC result:

SQ SUCCESSFUL.

Send SELECT with 0 length mask and Action 011 to send SL to ~SL

Do not send correct query to avoid flipping S0 INV flag to B by ACKing it

SQ fail tid 96 sl 011 0 0 0 0 s0 10 B

Data length = 1 Data = 00

SQ result: 0

EPC result:

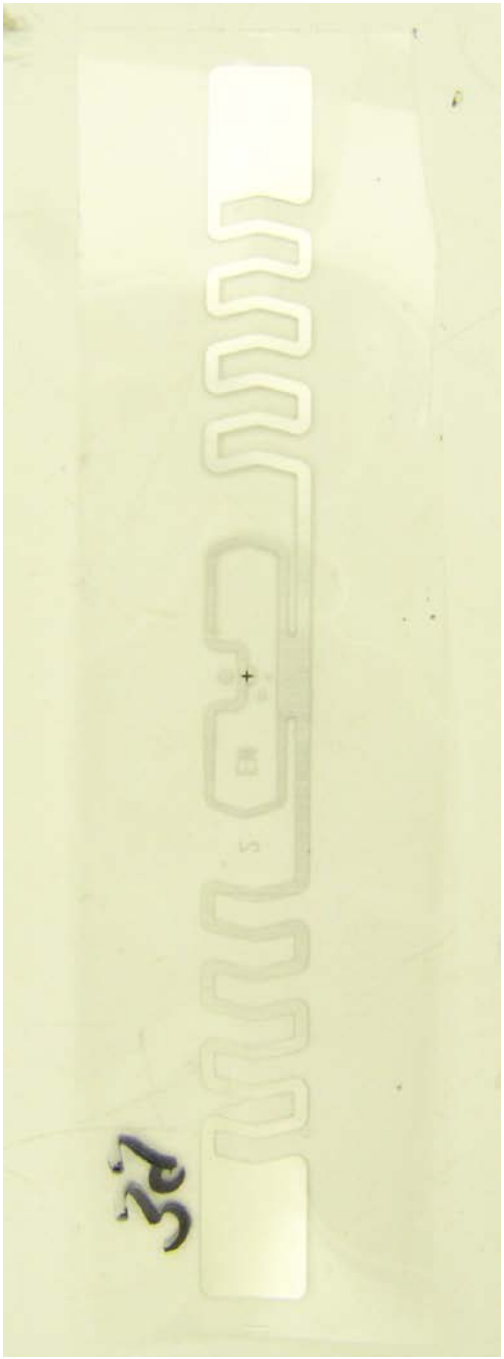
SQ SUCCESSFUL.

disconnect

Disconnected.

OVERALL RESULT: SUCCESS

Annex B: Pictures



Photograph 1. View of Tag

\

Annex C: IS Specification

Tag Role

Item	Memory bank	Document Reference	IS Reference	Status	Support
1	Reserved memory	6.3.2.1	A:1.4/1	m	y
2	EPC memory	6.3.2.1	A:1.4/2	m	y
3	TID memory	6.3.2.1	A:1.4/3	m	y
4	User memory	6.3.2.1	A:1.4/4	o	n

able A.4. Memory banks supported

Item	Password	Document Reference	IS Reference	Status	Support
1	Kill password	6.3.2.1.1	A:1.5/1	o	y
2	Access password	6.3.2.1.2	A:1.5/2	o	y

Table A.5. Stored passwords in Reserved memory bank

Item	EPC Data	Document Reference	IS Reference	Status	Support
1	CRC-16	6.3.2.1	A:1.6/1	m	y
2	PC (Protocol control)	6.3.2.1	A:1.6/2	m	y
3	Object identifier code	6.3.2.1	A:1.6/3	m	y

Table A.6. Stored data in EPC memory bank

Item	Object identifier	Document Reference	IS Reference	Status	Support
1	EPC (EPCglobal members)	6.3.2.1	A:1.7/1	o.1	y
2	Other object identifier code	6.3.2.1	A:1.7/2	o.1	y

Table A.7. Object identifier type

o.1: It is mandatory to support one of these items.
Comments:

Item	TID Data	Document Reference	IS Reference	Status	Support
1	8-bit ISO/IEC 15963 allocation class identifier	6.3.2.1	A:1.8/1	m	y
2	12-bit tag mask-designer identifier	6.3.2.1	A:1.8/2	c.1	y
3	12-bit tag model number	6.3.2.1	A:1.8/3	c.1	y
4	TID memory writeable	na	na	o	n

Table A.8. Stored data in TID memory bank

c.1: Mandatory for EPCglobal members (A:3.1.3.2/1 in Table 1.7)
Comments:

Item	States	Document Reference	IS Reference	Status	Support
1	Ready state	6.3.2.10.1.1	A:1.9/1	m	y
2	Arbitrate state	6.3.2.10.2.1	A:1.9/2	m	y
3	Reply state	6.3.2.10.2.2	A:1.9/3	m	y
4	Acknowledged state	6.3.2.10.2.3	A:1.9/4	m	y
5	Open state	6.3.2.10.2.4	A:1.9/5	m	y
6	Secured state	6.3.2.10.2.5	A:1.9/6	m	y
7	Killed state	6.3.2.10.3.1	A:1.9/7	m	y

Table A.9. States

Item	Command	Document Reference	IS Reference	Status	Support
1	Select	6.3.2.10.1.1	A:1.10/1	m	y
2	Query	6.3.2.10.2.1	A:1.10/2	m	y
3	QueryAdjust	6.3.2.10.2.2	A:1.10/3	m	y
4	QueryRep	6.3.2.10.2.3	A:1.10/4	m	y
5	ACK	6.3.2.10.2.4	A:1.10/5	m	y
6	NAK	6.3.2.10.2.5	A:1.10/6	m	y
7	Req_RN	6.3.2.10.3.1	A:1.10/7	m	y
8	Read	6.3.2.10.3.2	A:1.10/8	m	y
9	Write	6.3.2.10.3.3	A:1.10/9	m	y
10	Kill	6.3.2.10.3.4	A:1.10/10	m	y
11	Lock	6.3.2.10.3.5	A:1.10/11	m	y
12	Access	6.3.2.10.3.6	A:1.10/12	o	y
13	BlockWrite	6.3.2.10.3.7	A:1.10/13	o	n
14	BlockErase	6.3.2.10.3.8	A:1.10/14	o	n

Table A.10. Commands supported

Item	MemBank	Document Reference	IS Reference	Status	Support	Values	
						Allowed	Supported
1	Reserved memory bank	6.3.2.1	B.1.11/1	m	y	00	y
2	EPC memory bank	6.3.2.1	B.1.11/2	m	y	01	y
3	TID memory bank	6.3.2.1	B.1.11/3	m	y	10	y
4	User memory bank	6.3.2.1	B.1.11/4	m	y	11	y

Table A.11. Logical partitioning of the memory banks

Item	Data	Document Reference	IS Reference	Status	Support	Values	
						Allowed	Supported
1	Kill password in Reserved memory	6.3.2.1	B.1.12/1	c.1	y	00_h - $1F_h$	y
2	Access password in Reserved memory	6.3.2.1	B.1.12/2	c.2	y	20_h - $3F_h$	y
3	CRC-16 in EPC memory	6.3.2.1	B.1.12/3	m	y	00_h – $0F_h$	y
4	Protocol Control (PC) in EPC memory	6.3.2.1	B.1.12/4	m	y	10_h – $1F_h$	y
5	Object identifier code in EPC memory	6.3.2.1	B.1.12/5	m	y	Above $1F_h$	y
6	ISO/IEC 15963 allocation class identifier in TID memory	6.3.2.1	B.1.12/6	m	y	00_h - 07_h	y
7	Other identifier information in TID memory	6.3.2.1	B.1.12/7	m	y	Above 07_h	y
8	Tag mask-designer identifier in TID memory	6.3.2.1	B.1.12/8	c.3	y	08_h – 13_h	y
9	Tag model number in TID memory	6.3.2.1	B.1.12/9	c.3	y	14_h – $1F_h$	y

Table A.12. Memory locations of stored data in each memory bank

c.1: Mandatory for tags that implement the kill password (A:3.1.2/1 in Table 1.5)

c.2: Mandatory for tags that implement the access password (A:3.1.2/2 in Table 1.5)

c.3: Mandatory for EPCglobal members (A:3.1.3.2/1 in Table 1.7)