CVT Test Case Encyclopedia

16 MAY 2016

Radical Blue Gaming, Inc.
1770 S. Randall Road, Suite A319
Geneva, Illinois 60134
call us: +1.312.897.3737
visit us: www.radblue.com
drop us an email: sales@radblue.com

Need help?

Find out more about the GSA protocols

If you want to find out more about the Gaming Standards Association and the work being done in the area of protocol standardization for the gaming industry, we encourage you to visit their website at www.gamingstandards.com.

How to Read a Test Case Entry

Detailed information is provided for each CVT test case. Test case entry information is organized into several sections:

Section	Description
Criteria	Requirements for running the test: protocol version, GSA class, endpoint type (EGM or host) and the required number of hosts.
Objectives	Goal of the test case.
Required Devices	Devices required to perform test, and whether the other endpoint must be an owner or a guest.
Requirements Under Test	GSA requirements, found in the *Certification Requirement Checklists* document, that are tested by the test case.
Test Procedure	The steps taken to execute the test.
Test Type	Type of test performed for the test case: • **QUICK** - Test is run using an "as is" configuration. • **SUFFICIENT** - Full test, including validation of all configuration settings. • **CONTINUOUS** - Test that continuously validates G2S messages. • **COVERAGE** - Test that is run if the host does not send a command.

About the Test Case Naming Convention

Each test case is assigned a unique identifier. The naming convention for test cases is:

[class]-[functional group]-*nnnnn* (where *nnnnn* is a unique 5-digit number)

Class-level abbreviations correspond with the class abbreviations used by the Gaming Standards Association (GSA). Functional group abbreviations for Transport, Host and EGM are unique *within each class*. Functional group abbreviations are the same for both CVT Host or CVT EGM.

Once a test case is approved by GSA, it cannot be updated. When a test case requires a change, the old test case is deleted and a new, updated test case is created. The new test case has its own unique identifier.

Class-Level Abbreviations

Class-level abbreviations appear in the first two characters in the test case identifier. The following chart shows the 2-letter abbreviation for each class. Note that G2S Transport is denoted by **MS**.

Class	Abbreviation	Class	Abbreviation
bonus	BN	hopper	HP
cabinet	CB	idReader	ID
cashout	CO	informedPlayer	IP
central	CL	mediaDisplay	MD
coinAcceptor	CA	meters	MT
commConfig	CC	noteAcceptor	NA
communications	CM	noteDispenser	ND
deviceConfig	DC	optionConfig	OC
dft	DF	player	PR
download	DL	printer	PT
employee	EM	progressive	PG
eventHandler	EH	smartCard	SC
gamePlay	GP	storage	ST
gameTheme	GT	tournament	TR
gat	GA	voucher	VC
handpay	JP	wat	WT
hardware	HW		

Transport (MS) Functional Group Abbreviations

Functional Group Abbreviation	Functional Group Name
COR	Core Transport Functionality
WSDL Support	
SW2	S2S v1.3 or v1.2.2
SW1	S2S v1.2.1
GW2	G2S v1.3 or v1.1.2
GW1	G2S v1.1.1
Message Size	
4MB	4 Megabyte Message Support
Transport Options SOAP Client	
SCN	Client - No GZIP
SCH	Client - GZIP in HTTP Stack
SCP	Client - GZIP in Payload
CNT	Client - No Transport Options
CTR	Support Client Transport Options
Transport Options SOAP Server	
SSN	Server - No GZIP
SSH	Server - GZIP in HTTP Stack
SSP	Server - GZIP in Payload
SNT	Server - No Transport Option
STR	Support Server Transport Options
Security and Authentication - Certificate Options	
SYM	Change Symmetric Key on Demand
TLS	TLS Re-Handshake
GNC	GetNextCACert Support
SAN	Verify Domain by Subject Alternate Name
Security and Authentication - Symmetric Algorithms Supported	
AES	AES
OTH	Other
Security and Authentication - OCSP Client Support	
NCM	No Communications When OCSP Server Is Offline
CRT	Use Previously Good Certificates for a Limited Time
GCT	Use Prev Good Cert for Limited Time Then Use CRLs

Functional Group Abbreviation	Functional Group Name
CRL	Use CRLs When OCSP Is Offline
Security and Authentication - OCSP Client Options	
NON	Nonce Support
NUS	Next Update Support
DHCP Support	
DHC	DHCP Client Support
Multicast Support	
MLS	Multicast Listener
MHT	Multicast Host

Functional Group Abbreviations by Class

Note that functional group abbreviations are for both CVT EGM and CVT Host.

Class	Functional Group Abbreviation	Functional Group Name
bonus (BN)		
	COR	Core Bonus Functionality
	LIM	Bonus Award Limits (Extension igtBonus)
	MJT	Multiple Jackpot Time Bonus Support (Extension igtMJT)
	WGR	Wager Match Bonus Support (Extension igtWM)
cabinet (CB)		
	COR	Core Cabinet Functionality
	OHS	Operating Hours Support (Extension gtkOH)
	MRS	Master Reset Support (Extension GtkMR)
	OCC	Occupancy Meter Support (Extension g2sOC)
	PLY	Player-Initiated Configuration Changes
	RRS	Remote Reset Support
	TZO	Time Zone Offset Support
cashout (CO)		
	COR	Core Remote Cash-Out Functionality (Extension GtkCO)
central (CL)		
	COR	Core Central Determination Functionality
coinAcceptor (CA)		
	COR	Core Coin Acceptor Functionality
	PRO	Promotional and Non-Cashable Coin/Token Support

Class	Functional Group Abbreviation	Functional Group Name
commConfig (CC)		
	COR	Core Communications Configuration Functionality
communications (CM)		
	COR	Core Communications Functionality
	MUL	Multicast Message Support
deviceConfig (DC)		
	COR	Core Device Configuration Functionality
dft (DF)		
	COR	Core Direct Funds Transfer Functionality (Extension g2sDF)
download (DL)		
	COR	Core Software Download Functionality
	PRA	Pause/Resume/Abort Support (Extension gtkDL)
	UPL	Software Upload Support
Employee (EM)		
	COR	Core Employee Tracking Functionality (Extension g2sEM)
	ACR	Employee Activity Code Reporting
eventHandler (EH)		
	COR	Core Event Reporting Functionality
gamePlay (GP)		
	COR	Core Game Play Functionality
	AGD	Configure Accessible Games and Denominations (Extension gtkGC)
	GOS	Game Outcome Support (Extension 1k)
gameTheme (GT)		
	COR	Core Game Theme Functionality
gat (GA)		
	COR	Core Game Authentication Functionality
	SPE	Special Function Support
handpay (JP)		
	COR	Core HandPay Functionality
hardware (HW)		
	COR	Core Hardware Inventory Functionality (Extension g2sHW)
hopper (HP)		
	COR	Core Coin Hopper Functionality

Class	Functional Group Abbreviation	Functional Group Name
	PRO	Promotional and Non-Cashable Note/Token Support
idReader (ID)		
	COR	Core ID Reader Functionality
	EGM	EGM-Controlled ID Reader
	HST	Host-Controlled ID Reader
	MLS	Multi-Lingual Support
informedPlayer (IP)		
	COR	Core Informed Player Functionality (Extension tleIP)
	PIN	Player Authentication Using PINs
mediaDisplay (MD)		
	COR	Core Media Display (PUI) Functionality (Extension igtMediaDisplay)
meters (MT)		
	COR	Core Meter Reporting Functionality
	AUD	Audit Meter Support (Extension g2sAM)
noteAcceptor (NA)		
	COR	Core Note Acceptor Functionality
	PRO	Promotional and Non-Cashable Coin/Script Support
noteDispenser (ND)		
	COR	Core Note Dispenser Functionality
	PRO	Promotional and Non-Cashable Note/Script Support
optionConfig (OC)		
	COR	Core Option Configuration Functionality
player (PR)		
	COR	Core Player Tracking Functionality
	LIM	Display Limit Support (Extension igtPlayer-limits)
	MLS	Multi-Lingual Support
	WGR	Wager Match Player Support (Extension igtWMP)
printer (PT)		
	COR	Core Printer Functionality
	HST	Host-Initiated Printing Support
	RSC	Restrict Printing to Specific Players (Extension igtPrn)
progressive (PG)		
	COR	Core Progressive Functionality

Class	Functional Group Abbreviation	Functional Group Name
	DIS	EGM Discovery of Host Progressive Configuration
smartCard (SC)		
	COR	Core Smart Card Functionality (Extension igtSC)
storage (ST)		
	COR	Core Storage Requirements Functionality (Extension gtkST)
tournament (TR)		
	COR	Core Tournament Functionality (Extension g2sTR)
	EGM	EGM-Controlled Tournament Registration
	TSS	Tournament Standings Support
voucher (VC)		
	COR	Core Voucher Functionality
	ISS	Issue Voucher Support
	RDM	Redeem Voucher Support
wat (WT)		
	COR	Core Wagering Account Functionality
	EGM	EGM-Controlled User Interface

This chapter provides a cross-reference between requirements and test cases. For each G2S requirement, all cases that test that requirement are listed. Click the test case link to go directly to the associated test case.

1.1 XML and XML Schema

1.1.1

Test Case	EGM	Host
CM-COR-00001	X	
CM-COR-00060		X

1.1.2

Test Case	EGM	Host
CM-COR-00015	X	
CM-COR-00060		X

1.2 Transport Requirements

1.2.1

Test Case	EGM	Host
CM-COR-00001	X	
CM-COR-00060		X

1.2.4

Test Case	EGM	Host
CM-COR-00015	X	
CM-COR-00060		X

1.2.8

Test Case	EGM	Host
CM-COR-00015	X	
CM-COR-00060		X

1.2.9

Test Case	EGM	Host
CM-COR-00015	X	
CM-COR-00060		X

1.3 Point-to-Point Message Handling

1.3.1

Test Case	EGM	Host
CM-COR-00017	X	
CM-COR-00018	X	
CM-COR-00029	X	
CM-COR-00061		X

1.4 Point-to-Point Message Handling

1.4.1

Test Case	EGM	Host
CM-COR-00002	X	
CM-COR-00062		X

1.4.2

Test Case	EGM	Host
CM-COR-00002	X	
CM-COR-00062		X

1.4.4

Test Case	EGM	Host
CB-COR-00022		X
CB-MRS-00006		X
CM-COR-00063		X
EH-COR-00032		X
GP-COR-00022		X
JP-COR-00024		X
MT-COR-00019		X

1.4.5

Test Case	EGM	Host
CB-COR-00023		X
CB-MRS-00007		X
CM-COR-00063		X
EH-COR-00033		X

Test Case	EGM	Host
GP-COR-00023		X
JP-COR-00025		X
MT-COR-00020		X

1.4.6

Test Case	EGM	Host
CB-COR-00023		X
CB-MRS-00007		X
CM-COR-00063		X
EH-COR-00033		X
GP-COR-00023		X
JP-COR-00024		X
MT-COR-00019		X

1.4.7

Test Case	EGM	Host
CB-COR-00023		X
CB-MRS-00007		X
CM-COR-00063		X
EH-COR-00033		X
GP-COR-00030		X
JP-COR-00024		X
MT-COR-00020		X

1.5 Classes & Devices

1.5.1

Test Case	EGM	Host
CM-COR-00079	X	

1.5.4

Test Case	EGM	Host
CM-COR-00079	X	

1.5.5

Test Case	EGM	Host
CB-COR-00049	X	
EH-COR-00021	X	

1.6 Multiple Hosts

1.6.7

Test Case	EGM	Host
CM-COR-00026	X	

1.7 Device Identifiers

1.7.1

Test Case	EGM	Host
CM-COR-00092	X	

1.7.3

Test Case	EGM	Host
CM-COR-00092	X	

1.7.4

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00005	X	
GP-COR-00008	X	
JP-COR-00005	X	
MT-COR-00002	X	

1.8 Device Subscriptions

1.8.2

Test Case	EGM	Host
CM-COR-00027	X	

1.8.5

Test Case	EGM	Host
CB-COR-00001	X	
EH-COR-00003	X	
GP-COR-00001	X	
JP-COR-00001	X	
MT-COR-00003	X	

1.8.6

Test Case	EGM	Host
CM-COR-00064		X

1.8.7

Test Case	EGM	Host
CM-COR-00065		X

1.8.11

Test Case	EGM	Host
EH-COR-00049	X	

1.8.13

Test Case	EGM	Host
CM-COR-00018	X	

1.8.14

Test Case	EGM	Host
CB-COR-00002	X	
CB-RRS-00001	X	
EH-COR-00003	X	
EH-COR-00004	X	
GP-COR-00002	X	
JP-COR-00001	X	
JP-COR-00002	X	
MT-COR-00003	X	
MT-COR-00004	X	

1.9 Disabled Devices

1.9.6

Test Case	EGM	Host
CB-COR-00005	X	
EH-COR-00007	X	

1.9.9

Test Case	EGM	Host
CM-COR-00065		X

1.13 Host Registration

1.13.6

Test Case	EGM	Host
CM-COR-00033	X	

1.14 Host Identifiers

1.14.2

Test Case	EGM	Host
CM-COR-00066		X

1.14.11

Test Case	EGM	Host
CM-COR-00067		X

1.14.13

Test Case	EGM	Host
CM-COR-00018	X	

1.15 EGM Identifiers

1.15.1

Test Case	EGM	Host
CM-COR-00028	X	

1.15.3

Test Case	EGM	Host
CM-COR-00017	X	

1.17 Date/Time Format

1.17.5

Test Case	EGM	Host
CM-COR-00068		X
CM-COR-00080	X	

1.19 Transaction Logs

1.19.2

Test Case	EGM	Host
GP-COR-00008	X	
JP-COR-00008	X	

1.19.6

Test Case	EGM	Host
CM-COR-00081	X	

1.19.9

Test Case	EGM	Host
GP-COR-00027	X	

1.19.10

Test Case	EGM	Host
EH-COR-00050	X	
GP-COR-00027	X	

1.19.11

Test Case	EGM	Host
EH-COR-00051	X	

1.19.12

Test Case	EGM	Host
EH-COR-00005	X	

1.19.13

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21 Querying Transaction Logs

1.21.2

Test Case	EGM	Host
EH-COR-00003	X	
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	
MT-COR-00002	X	

1.21.3

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21.4

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21.5

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21.6

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21.7

Test Case	EGM	Host
EH-COR-00005	X	
GP-COR-00008	X	
JP-COR-00008	X	

1.21.8

Test Case	EGM	Host
GP-COR-00008	X	
JP-COR-00008	X	

1.22 Event Identifiers

1.22.1

Test Case	EGM	Host
CM-COR-00004	X	

1.22.2

Test Case	EGM	Host
GP-COR-00027	X	

1.22.4

Test Case	EGM	Host
CM-COR-00004	X	

1.22.5

Test Case	EGM	Host
CM-COR-00004	X	

1.22.6

Test Case	EGM	Host
CM-COR-00004	X	

1.23 Event Subscriptions

1.23.1

Test Case	EGM	Host
CB-COR-00004	X	
EH-COR-00052	X	
GP-COR-00027	X	

1.23.3

Test Case	EGM	Host
CB-COR-00004	X	
EH-COR-00007	X	
GP-COR-00027	X	

1.23.23

Test Case	EGM	Host
CB-COR-00050	X	
EH-COR-00052	X	

1.23.25

Test Case	EGM	Host
CB-COR-00038	X	
EH-COR-00038	X	
GP-COR-00006	X	
JP-COR-00006	X	
MT-COR-00006	X	

1.23.26

Test Case	EGM	Host
CB-COR-00038	X	
EH-COR-00038	X	
GP-COR-00006	X	
JP-COR-00006	X	
MT-COR-00006	X	

1.23.29

Test Case	EGM	Host
EH-COR-00042	X	

1.25 g2sBody Element

1.25.4

Test Case	EGM	Host
CM-COR-00017	X	
CM-COR-00018	X	

1.25.5

Test Case	EGM	Host
CM-COR-00067		X

1.25.7

Test Case	EGM	Host
CM-COR-00018	X	
CM-COR-00067		X

1.25.8

Test Case	EGM	Host
CM-COR-00069		X

1.25.10

Test Case	EGM	Host
CM-COR-00047	X	
CM-COR-00070		X

1.25.11

Test Case	EGM	Host
CM-COR-00047	X	
CM-COR-00070		X

1.26 g2sAck Element

1.26.1

Test Case	EGM	Host
CM-COR-00048	X	
CM-COR-00071		X

1.26.6

Test Case	EGM	Host
CM-COR-00018	X	
CM-COR-00067		X

1.26.7

Test Case	EGM	Host
CM-COR-00018	X	
CM-COR-00067		X

1.26.8

Test Case	EGM	Host
CM-COR-00017	X	
CM-COR-00067		X

1.26.9

Test Case	EGM	Host
CM-COR-00069		X

1.26.11

Test Case	EGM	Host
CM-COR-00069		X

1.26.13

Test Case	EGM	Host
CM-COR-00005	X	
CM-COR-00018	X	
CM-COR-00067		X

1.26.14

Test Case	EGM	Host
CM-COR-00005	X	
CM-COR-00017	X	
CM-COR-00067		X

1.26.15

Test Case	EGM	Host
CM-COR-00034	X	
CM-COR-00064		X

1.26.16

Test Case	EGM	Host
CM-COR-00005	X	
CM-COR-00064		X

1.27 Class-Level Element

1.27.1

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00064		X

1.27.3

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00064		X

1.27.7

Test Case	EGM	Host
CM-COR-00072		X

1.27.8

Test Case	EGM	Host
CM-COR-00072		X

1.27.9

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00072		X

1.27.26

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00072		X

1.27.27

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00072		X

1.27.28

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00072		X

1.27.29

Test Case	EGM	Host
CM-COR-00035	X	
CM-COR-00072		X

1.27.33

Test Case	EGM	Host
CB-COR-00003	X	
CB-COR-00024		X
CB-MRS-00008		X
CM-COR-00060		X
EH-COR-00016	X	
EH-COR-00034		X
GP-COR-00024		X
JP-COR-00026		X
JP-COR-00032	X	
MT-COR-00021		X
MT-COR-00032	X	

1.28 Command Retry

1.28.3

Test Case	EGM	Host
CM-COR-00030	X	
CM-COR-00031	X	
CM-COR-00096		X

1.28.8

Test Case	EGM	Host
EH-COR-00020	X	
MT-COR-00009	X	

1.28.9

Test Case	EGM	Host
CB-MRS-00009		X
CM-COR-00073		X
EH-COR-00035		X
JP-COR-00027		X
MT-COR-00022		X
MT-COR-00033	X	

1.28.10

Test Case	EGM	Host
EH-COR-00020	X	
JP-COR-00011	X	

1.28.13

Test Case	EGM	Host
EH-COR-00020	X	

1.29 Message Too Large

1.29.1

Test Case	EGM	Host
CM-COR-00007	X	

1.29.2

Test Case	EGM	Host
CM-COR-00007	X	

1.29.6

Test Case	EGM	Host
CM-COR-00074		X

1.30 Processing Order

1.30.2

Test Case	EGM	Host
EH-COR-00001	X	

1.30.3

Test Case	EGM	Host
CM-COR-00008	X	
CM-COR-00096		X

1.30.4

Test Case	EGM	Host
CM-COR-00008	X	
CM-COR-00096		X

1.30.5

Test Case	EGM	Host
CM-COR-00049	X	
CM-COR-00075		X

1.30.6

Test Case	EGM	Host
CM-COR-00049	X	

1.30.7

Test Case	EGM	Host
CB-COR-00025		X
CM-COR-00078		X
EH-COR-00036		X
GP-COR-00025		X
MT-COR-00023		X

1.32 Communications States

1.32.7

Test Case	EGM	Host
CM-COR-00023	X	

1.36 EGM Activation

1.36.9

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00005	X	
JP-COR-00014	X	
JP-COR-00015	X	

1.36.15

Test Case	EGM	Host
CB-COR-00039	X	
CB-COR-00041	X	

1.36.16

Test Case	EGM	Host
CB-COR-00034	X	

1.36.17

Test Case	EGM	Host
CB-COR-00034	X	

1.37 EGM Deactivation

1.37.1

Test Case	EGM	Host
EH-COR-00043	X	

1.38 Standardized Options

1.38.1

Test Case	EGM	Host
EH-COR-00019	X	
GP-AGD-00001	X	
GP-COR-00004	X	
JP-COR-00004	X	

1.39 Persistent Memory

1.39.2

Test Case	EGM	Host
CM-COR-00032	X	

1.39.3

Test Case	EGM	Host
CM-COR-00082	X	

1.41 Schema Validation

1.41.2

Test Case	EGM	Host
CB-MRS-00008		X
CM-COR-00017	X	
CM-COR-00018	X	
CM-COR-00076		X
EH-COR-00034		X
GP-COR-00007	X	
GP-COR-00031		X
JP-COR-00007	X	
JP-COR-00026		X
MT-COR-00008	X	
MT-COR-00021		X

1.41.5

Test Case	EGM	Host
GP-COR-00007	X	
JP-COR-00007	X	
MT-COR-00008	X	

1.41.6

Test Case	EGM	Host
CB-MRS-00008		X
CM-COR-00076		X
EH-COR-00034		X
JP-COR-00026		X
MT-COR-00021		X

1.41.8

Test Case	EGM	Host
CM-COR-00010	X	

1.41.9

Test Case	EGM	Host
CM-COR-00050	X	

1.41.10

Test Case	EGM	Host
CB-COR-00008	X	
CM-COR-00011	X	
EH-COR-00017	X	
GP-COR-00003	X	
JP-COR-00003	X	
MT-COR-00007	X	

1.41.13

Test Case	EGM	Host
CM-COR-00076		X

1.41.14

Test Case	EGM	Host
CM-COR-00076		X

1.41.15

Test Case	EGM	Host
CB-COR-00026		X
CM-COR-00076		X
EH-COR-00032		X
GP-COR-00026		X
JP-COR-00028		X
MT-COR-00025		X

1.42 Error Code Extensions

1.42.2

Test Case	EGM	Host
CM-COR-00048	X	
CM-COR-00069		X

1.42.3

Test Case	EGM	Host
CB-COR-00023		X
CB-MRS-00007		X
EH-COR-00037		X
GP-COR-00023		X

Test Case	EGM	Host
JP-COR-00031		X
MT-COR-00023		X
MT-COR-00034	X	

1.44 Error Reporting

1.44.2

Test Case	EGM	Host
CB-MRS-00009		X
CM-COR-00065		X
GP-COR-00009	X	
MT-COR-00010	X	

1.999 Host Coverage Commands

1.999.999

Test Case	EGM	Host
CB-COR-00017		X
CB-COR-00018		X
CB-COR-00019		X
CB-COR-00020		X
CB-COR-00021		X
CB-MRS-00002		X
CB-MRS-00003		X
CB-MRS-00004		X
CB-MRS-00005		X
CB-MRS-00011		X
CB-OHS-00001		X
CB-OHS-00002		X
CB-RRS-00002		X
CB-TZO-00002		X
CB-TZO-00003		X
CM-COR-00055		X
CM-COR-00056		X
CM-COR-00057		X

Test Case	EGM	Host
CM-COR-00058		X
CM-COR-00059		X
EH-COR-00023		X
EH-COR-00024		X
EH-COR-00025		X
EH-COR-00026		X
EH-COR-00027		X
EH-COR-00028		X
EH-COR-00029		X
EH-COR-00030		X
EH-COR-00031		X
GP-COR-00015		X
GP-COR-00016		X
GP-COR-00017		X
GP-COR-00018		X
GP-COR-00019		X
GP-COR-00020		X
GP-COR-00021		X
GP-GOS-00002		X
GP-GOS-00003		X
JP-COR-00018		X
JP-COR-00019		X
JP-COR-00020		X
JP-COR-00021		X
JP-COR-00022		X
JP-COR-00023		X
MT-AUD-00001		X
MT-COR-00015		X
MT-COR-00016		X
MT-COR-00017		X
MT-COR-00018		X

2.1 General requirements

2.1.2

Test Case	EGM	Host
CM-COR-00082	X	

2.1.3

Test Case	EGM	Host
CM-COR-00082	X	

2.2 Sending, Resending, and commandIds

2.2.2

Test Case	EGM	Host
CM-COR-00072		X

2.2.4

Test Case	EGM	Host
CM-COR-00012	X	

2.2.5

Test Case	EGM	Host
CM-COR-00072		X

2.2.8

Test Case	EGM	Host
CM-COR-00012	X	
CM-COR-00072		X

2.2.10

Test Case	EGM	Host
CM-COR-00012	X	
CM-COR-00096		X

2.3 Association Attributes

2.3.3

Test Case	EGM	Host
CM-COR-00038	X	

2.3.4

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00040	X	
CM-COR-00041	X	
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	

2.3.5

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00040	X	
CM-COR-00041	X	
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	

2.4 closed State

2.4.2

Test Case	EGM	Host
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00040	X	
CM-COR-00041	X	

2.5 opening State

2.5.1

Test Case	EGM	Host
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	

2.5.2

Test Case	EGM	Host
CM-COR-00045	X	
CM-COR-00046	X	

2.5.3

Test Case	EGM	Host
CM-COR-00038	X	

2.5.4

Test Case	EGM	Host
CM-COR-00038	X	
CM-COR-00045	X	

2.5.5

Test Case	EGM	Host
CM-COR-00045	X	

2.5.6

Test Case	EGM	Host
CM-COR-00083	X	

2.5.7

Test Case	EGM	Host
CM-COR-00046	X	

2.5.11

Test Case	EGM	Host
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00040	X	
CM-COR-00041	X	
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	

2.6 sync State

2.6.1

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	
CM-COR-00063		X
CM-COR-00065		X
CM-COR-00078		X

2.6.2

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00094		X

2.6.3

Test Case	EGM	Host
CM-COR-00023	X	

2.6.4

Test Case	EGM	Host
CM-COR-00038	X	

2.6.5

Test Case	EGM	Host
CM-COR-00023	X	

2.6.6

Test Case	EGM	Host
CM-COR-00036	X	

2.6.7

Test Case	EGM	Host
CM-COR-00084	X	

2.6.8

Test Case	EGM	Host
CM-COR-00039	X	

2.6.12

Test Case	EGM	Host
CM-COR-00037	X	
CM-COR-00041	X	

2.6.13

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00040	X	
CM-COR-00041	X	
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	
CM-COR-00046	X	

2.7 onLine State

2.7.2

Test Case	EGM	Host
CM-COR-00040	X	

2.7.3

Test Case	EGM	Host
CM-COR-00040	X	

2.7.4

Test Case	EGM	Host
CM-COR-00085	X	

2.7.5

Test Case	EGM	Host
CM-COR-00039	X	
CM-COR-00046	X	

2.7.9

Test Case	EGM	Host
CM-COR-00023	X	
CM-COR-00036	X	
CM-COR-00037	X	
CM-COR-00039	X	
CM-COR-00046	X	

2.7.10

Test Case	EGM	Host
CM-COR-00041	X	

2.9 closing State

2.9.1

Test Case	EGM	Host
CM-COR-00043	X	

2.9.2

Test Case	EGM	Host
CM-COR-00042	X	

2.9.3

Test Case	EGM	Host
CM-COR-00042	X	

2.9.4

Test Case	EGM	Host
CM-COR-00038	X	

2.9.5

Test Case	EGM	Host
CM-COR-00042	X	

2.9.6

Test Case	EGM	Host
CM-COR-00042	X	
CM-COR-00043	X	
CM-COR-00044	X	
CM-COR-00045	X	

2.10 Transport-Related Events

2.10.5

Test Case	EGM	Host
CM-COR-00052	X	

2.10.6

Test Case	EGM	Host
CM-COR-00093	X	

2.10.11

Test Case	EGM	Host
CM-COR-00093	X	

2.10.14

Test Case	EGM	Host
CM-COR-00037	X	
CM-COR-00041	X	

2.10.15

Test Case	EGM	Host
CM-COR-00037	X	
CM-COR-00041	X	

2.10.17

Test Case	EGM	Host
CM-COR-00044	X	

2.20 Owner-Controlled Parameters

2.20.2

Test Case	EGM	Host
CM-COR-00087	X	

2.21 setCommsState Command

2.21.3

Test Case	EGM	Host
CM-COR-00053	X	

2.22 commsStatus Command

2.22.1

Test Case	EGM	Host
CM-COR-00013	X	

2.22.3

Test Case	EGM	Host
CM-COR-00089	X	

2.23 commsProfile Command

2.23.2

Test Case	EGM	Host
CM-COR-00053	X	

2.24 commsOnline Command

2.24.1

Test Case	EGM	Host
CM-COR-00014	X	

2.24.2

Test Case	EGM	Host
CM-COR-00061		X

2.24.3

Test Case	EGM	Host
CM-COR-00065		X

2.24.5

Test Case	EGM	Host
CM-COR-00016	X	
CM-COR-00039	X	
CM-COR-00046	X	

2.24.6

Test Case	EGM	Host
CM-COR-00088	X	

2.24.7

Test Case	EGM	Host
CM-COR-00088	X	

2.26 commsOnlineAck Command

2.26.8

Test Case	EGM	Host
CM-COR-00094		X

2.26.9

Test Case	EGM	Host
CM-COR-00069		X

2.28 commsDisabledAck Command

2.28.1

Test Case	EGM	Host
CM-COR-00063		X
CM-COR-00065		X
CM-COR-00078		X

2.28.3

Test Case	EGM	Host
CM-COR-00094		X

2.30 commsClosingAck Command

2.30.1

Test Case	EGM	Host
CM-COR-00095		X

2.31 getDescriptor Command

2.31.1

Test Case	EGM	Host
CM-COR-00019	X	

2.36 setKeepAlive Command

2.36.1

Test Case	EGM	Host
CM-COR-00020	X	

2.37 keepAlive Command

2.37.1

Test Case	EGM	Host
CM-COR-00020	X	

2.42 G2S_CME002 Device Not Disabled by EGM

2.42.2

Test Case	EGM	Host
CM-COR-00045	X	

2.46 G2S_CME006 Device Configuration Changed by Operator

2.46.1

Test Case	EGM	Host
CM-COR-00090	X	

2.57 G2S_CME120 - Comms Host Unreachable

2.57.2

Test Case	EGM	Host
CM-COR-00037	X	
CM-COR-00041	X	

2.58 G2S_CME121 - Comms Transport Up

2.58.1

Test Case	EGM	Host
CM-COR-00037	X	
CM-COR-00041	X	
CM-COR-00044	X	
CM-COR-00093	X	

2.58.4

Test Case	EGM	Host
CB-COR-00040	X	

2.65 G2S_CME150 - Certificate Reenrollment Failure

2.65.1

Test Case	EGM	Host
CM-COR-00089	X	

2.66 G2S_CME151 - Certificate Reenrollment Failure Cleared

2.66.1

Test Case	EGM	Host
CM-COR-00089	X	

3.1 Introduction

3.1.1

Test Case	EGM	Host
CB-COR-00015	X	

3.2 Locale Identifier

3.2.1

Test Case	EGM	Host
CB-COR-00015	X	
CB-COR-00016	X	

3.3 Disable, Lockout, and Cabinet State

3.3.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00005	X	
CB-COR-00009	X	
CB-COR-00010	X	
CB-COR-00011	X	
CB-COR-00027	X	

3.3.2

Test Case	EGM	Host
CB-COR-00005	X	

3.3.3

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	
CM-COR-00053	X	

3.3.7

Test Case	EGM	Host
CB-COR-00005	X	
CM-COR-00054	X	

3.3.8

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	
CM-COR-00054	X	
JP-COR-00014	X	
JP-COR-00015	X	

3.3.10

Test Case	EGM	Host
CB-COR-00042	X	
CM-COR-00054	X	

3.3.11

Test Case	EGM	Host
CB-COR-00042	X	
CM-COR-00054	X	

3.3.12

Test Case	EGM	Host
CB-COR-00043	X	

3.3.16

Test Case	EGM	Host
CB-COR-00027	X	
CM-COR-00053	X	
JP-COR-00014	X	
JP-COR-00015	X	

3.3.17

Test Case	EGM	Host
CB-COR-00005	X	
JP-COR-00014	X	

3.3.18

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00027	X	
CM-COR-00053	X	
JP-COR-00009	X	
JP-COR-00014	X	

3.4 Cabinet Doors

3.4.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	
CB-COR-00010	X	

3.5 Text Messages

3.5.2

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	
JP-COR-00009	X	

3.5.4

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	

3.5.6

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	

3.5.12

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	
GP-COR-00012	X	

3.5.14

Test Case	EGM	Host
CB-COR-00044	X	

3.5.16

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	
CM-COR-00053	X	
JP-COR-00014	X	

3.5.19

Test Case	EGM	Host
CB-COR-00010	X	

3.6 Request-Response Pairs

3.6.1

Test Case	EGM	Host
CB-COR-00002	X	
CB-MRS-00010		X

3.6.3

Test Case	EGM	Host
CB-COR-00001	X	
CB-COR-00002	X	
CB-RRS-00001	X	

3.6.4

Test Case	EGM	Host
CB-COR-00005	X	

3.8 setCabinetState Command

3.8.4

Test Case	EGM	Host
CB-COR-00029	X	

3.8.5

Test Case	EGM	Host
CB-COR-00010	X	
CB-COR-00011	X	

3.8.6

Test Case	EGM	Host
CB-COR-00028	X	

3.9 cabinetStatus Command

3.9.1

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	
CB-COR-00045	X	
CB-COR-00046	X	
CB-COR-00047	X	

3.9.2

Test Case	EGM	Host
CB-COR-00035	X	

3.9.3

Test Case	EGM	Host
CB-COR-00036	X	

3.9.6

Test Case	EGM	Host
CB-COR-00045	X	

3.9.7

Test Case	EGM	Host
CB-COR-00046	X	

3.9.8

Test Case	EGM	Host
CB-COR-00047	X	

3.9.9

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	
CB-COR-00045	X	
CB-COR-00046	X	
CB-COR-00047	X	

3.9.15

Test Case	EGM	Host
CB-COR-00030	X	

3.10 cabinetProfile Command

3.10.13

Test Case	EGM	Host
GP-AGD-00001	X	

3.10.14

Test Case	EGM	Host
CB-COR-00031	X	
CB-COR-00032	X	
CB-COR-00033	X	

3.10.20

Test Case	EGM	Host
CB-COR-00031	X	
CB-COR-00032	X	
CB-COR-00033	X	
CB-COR-00034	X	
CB-COR-00035	X	
CB-COR-00036	X	

3.10.22

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	

3.10.23

Test Case	EGM	Host
CB-COR-00048	X	

3.10.30

Test Case	EGM	Host
CB-COR-00013	X	
CB-COR-00014	X	
CB-MRS-00001	X	
CB-OCC-00001	X	
CB-TZO-00001	X	

3.11 setCabinetLockOut Command

3.11.1

Test Case	EGM	Host
CB-COR-00011	X	

3.12 setDateTime Command

3.12.1

Test Case	EGM	Host
CB-COR-00007	X	

3.12.2

Test Case	EGM	Host
CB-COR-00007	X	

3.17 masterReset Command

3.17.13

Test Case	EGM	Host
CB-MRS-00010		X

3.25 Data Types

3.25.1

Test Case	EGM	Host
CB-COR-00015	X	

3.26 G2S_CBE001 Device Disabled by EGM

3.26.2

Test Case	EGM	Host
CB-COR-00009	X	
CB-COR-00027	X	

3.27 G2S_CBE002 Device Not Disabled by EGM

3.27.1

Test Case	EGM	Host
CB-COR-00009	X	
CB-COR-00027	X	

3.28 G2S_CBE003 Device Disabled by Host

3.28.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	

3.29 G2S_CBE004 Device Not Disabled by Host

3.29.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	

3.32 G2S_CBE009 Device Locked by Host

3.32.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00011	X	

3.33 G2S_CBE010 Device Not Locked by Host

3.33.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00011	X	

3.34 G2S_CBE101 Host Disabled Game Play

3.34.1

Test Case	EGM	Host
CB-COR-00005	X	

3.35 G2S_CBE102 Host Enabled Game Play

3.35.1

Test Case	EGM	Host
CB-COR-00005	X	

3.36 G2S_CBE103 Host Disabled Money In

3.36.1

Test Case	EGM	Host
CB-COR-00005	X	

3.37 G2S_CBE104 Host Enabled Money In

3.37.1

Test Case	EGM	Host
CB-COR-00005	X	

3.40 G2S_CBE203 Device Action Disabled EGM

3.40.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	
CB-COR-00027	X	

3.41 G2S_CBE204 Host Command Disabled EGM

3.41.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00010	X	

3.42 G2S_CBE205 EGM Enabled and Playable

3.42.1

Test Case	EGM	Host
CB-COR-00005	X	
CB-COR-00009	X	
CB-COR-00010	X	
CB-COR-00011	X	
CB-COR-00027	X	

3.43 G2S_CBE206 Operator Menu Activated

3.43.1

Test Case	EGM	Host
CB-COR-00051	X	

3.44 G2S_CBE207 Demo Mode Activated

3.44.1

Test Case	EGM	Host
CB-COR-00052	X	

3.45 G2S_CBE208 Meters/Audit Mode Initiated

3.45.1

Test Case	EGM	Host
CB-COR-00053	X	

3.46 G2S_CBE209 EGM Locked – Operator Menu

3.46.1

Test Case	EGM	Host
CB-COR-00054	X	

3.48 G2S_CBE211 Host Action Locked EGM

3.48.1

Test Case	EGM	Host
CB-COR-00011	X	

3.49 G2S_CBE301 Service Lamp On

3.49.1

Test Case	EGM	Host
CB-COR-00006	X	

3.50 G2S_CBE302 Service Lamp Off

3.50.1

Test Case	EGM	Host
CB-COR-00006	X	

3.51 G2S_CBE303 Logic Door Open

3.51.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.51.2

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.52 G2S_CBE304 Logic Door Closed

3.52.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.52.2

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.53 G2S_CBE305 Auxiliary Door Open

3.53.1

Test Case	EGM	Host
CB-COR-00009	X	

3.53.2

Test Case	EGM	Host
CB-COR-00009	X	

3.54 G2S_CBE306 Auxiliary Door Closed

3.54.1

Test Case	EGM	Host
CB-COR-00009	X	

3.54.2

Test Case	EGM	Host
CB-COR-00009	X	

3.55 G2S_CBE307 Cabinet Door Open

3.55.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.55.2

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.56 G2S_CBE308 Cabinet Door Closed

3.56.1

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.56.2

Test Case	EGM	Host
CB-COR-00004	X	
CB-COR-00009	X	

3.57 G2S_CBE309 General Cabinet Fault

3.57.1

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	

3.57.2

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	

3.58 G2S_CBE310 Video Display Error

3.58.1

Test Case	EGM	Host
CB-COR-00045	X	

3.59 G2S_CBE311 Non-Volatile Storage Fault

3.59.1

Test Case	EGM	Host
CB-COR-00046	X	

3.59.2

Test Case	EGM	Host
CB-COR-00046	X	

3.60 G2S_CBE312 General Memory Fault

3.60.1

Test Case	EGM	Host
CB-COR-00047	X	

3.60.2

Test Case	EGM	Host
CB-COR-00047	X	

3.61 G2S_CBE313 All Cabinet Faults Cleared

3.61.1

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	

3.61.2

Test Case	EGM	Host
CB-COR-00035	X	
CB-COR-00036	X	

3.62 G2S_CBE314 Game Combo Change Committed

3.62.1

Test Case	EGM	Host
GP-COR-00011	X	

3.65 G2S_CBE328 EGM Idle

3.65.1

Test Case	EGM	Host
CB-COR-00030	X	

3.66 G2S_CBE329 EGM Not Idle

3.66.1

Test Case	EGM	Host
CB-COR-00030	X	

3.68 G2S_CBE315 Date/Time Changed

3.68.1

Test Case	EGM	Host
CB-COR-00007	X	

3.71 G2S_CBE316 Cash-Out Button Pressed

3.71.1

Test Case	EGM	Host
CB-COR-00012	X	
GP-COR-00011	X	

3.71.2

Test Case	EGM	Host
CB-COR-00012	X	

3.72 G2S_CBE317 Power Off – Logic Door Open

3.72.1

Test Case	EGM	Host
CB-COR-00055	X	

3.72.2

Test Case	EGM	Host
CB-COR-00055	X	

3.73 G2S_CBE318 Power Off – Auxiliary Door Open

3.73.1

Test Case	EGM	Host
CB-COR-00055	X	

3.73.2

Test Case	EGM	Host
CB-COR-00055	X	

3.74 G2S_CBE319 Power Off – Cabinet Door Open

3.74.1

Test Case	EGM	Host
CB-COR-00055	X	

3.74.2

Test Case	EGM	Host
CB-COR-00055	X	

3.75 G2S_CBE320 Operator Reset Cabinet

3.75.1

Test Case	EGM	Host
CB-COR-00056	X	

3.76 G2S_CBE321 Life-To-Date Meters Reset

3.76.1

Test Case	EGM	Host
CB-COR-00057	X	

3.76.3

Test Case	EGM	Host
CB-COR-00057	X	

3.77 G2S_CBE322 Non-Volatile Storage Cleared

3.77.1

Test Case	EGM	Host
CB-COR-00037	X	

3.77.2

Test Case	EGM	Host
CB-COR-00037	X	

3.77.3

Test Case	EGM	Host
CB-COR-00037	X	

3.79 G2S_CBE325 EGM Power Up/Restart

3.79.1

Test Case	EGM	Host
CB-COR-00049	X	

3.79.2

Test Case	EGM	Host
CB-COR-00049	X	

3.96 Device Option Configuration

3.96.1

Test Case	EGM	Host
CB-COR-00013	X	

4.1 General Requirements

4.1.1

Test Case	EGM	Host
EH-COR-00018	X	

4.1.2

Test Case	EGM	Host
EH-COR-00018	X	

4.1.3

Test Case	EGM	Host
EH-COR-00018	X	

4.1.4

Test Case	EGM	Host
EH-COR-00018	X	

4.3 Event Persistence

4.3.1

Test Case	EGM	Host
EH-COR-00007	X	

4.3.3

Test Case	EGM	Host
EH-COR-00007	X	
EH-COR-00020	X	

4.3.6

Test Case	EGM	Host
EH-COR-00005	X	

4.3.7

Test Case	EGM	Host
EH-COR-00007	X	

4.4 Event Processing

4.4.4

Test Case	EGM	Host
EH-COR-00007	X	

4.4.5

Test Case	EGM	Host
EH-COR-00044	X	
EH-COR-00045	X	

4.5 Request-Response Pairs

4.5.1

Test Case	EGM	Host
EH-COR-00033		X
EH-COR-00035		X
EH-COR-00036		X
EH-COR-00037		X
EH-COR-00039	X	

4.5.3

Test Case	EGM	Host
EH-COR-00004	X	

4.5.5

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.5.6

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.7 setEventHandlerState Command

4.7.1

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.7.2

Test Case	EGM	Host
EH-COR-00053	X	

4.7.3

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.7.4

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.8 Disabled by EGM

4.8.4

Test Case	EGM	Host
EH-COR-00054	X	

4.8.5

Test Case	EGM	Host
EH-COR-00055	X	

4.12 eventHandlerProfile Command

4.12.4

Test Case	EGM	Host
EH-COR-00043	X	

4.12.5

Test Case	EGM	Host
EH-COR-00046	X	

4.12.7

Test Case	EGM	Host
EH-COR-00056	X	

4.12.8

Test Case	EGM	Host
EH-COR-00056	X	

4.12.9

Test Case	EGM	Host
EH-COR-00056	X	

4.12.10

Test Case	EGM	Host
EH-COR-00019	X	

4.13 getSupportedEvents Command

4.13.1

Test Case	EGM	Host
EH-COR-00013	X	

4.13.2

Test Case	EGM	Host
EH-COR-00013	X	

4.14 supportedEvents Command

4.14.1

Test Case	EGM	Host
EH-COR-00013	X	

4.14.2

Test Case	EGM	Host
EH-COR-00013	X	

4.15 setEventSub Command

4.15.1

Test Case	EGM	Host
EH-COR-00015	X	

4.15.2

Test Case	EGM	Host
EH-COR-00015	X	

4.15.3

Test Case	EGM	Host
EH-COR-00015	X	

4.15.4

Test Case	EGM	Host
EH-COR-00021	X	

4.15.5

Test Case	EGM	Host
EH-COR-00057	X	

4.15.6

Test Case	EGM	Host
EH-COR-00015	X	

4.15.7

Test Case	EGM	Host
EH-COR-00015	X	

4.15.8

Test Case	EGM	Host
EH-COR-00040	X	

4.15.9

Test Case	EGM	Host
EH-COR-00015	X	

4.17 getEventSub Command

4.17.1

Test Case	EGM	Host
EH-COR-00015	X	

4.17.2

Test Case	EGM	Host
EH-COR-00015	X	

4.17.3

Test Case	EGM	Host
EH-COR-00015	X	

4.17.4

Test Case	EGM	Host
EH-COR-00015	X	

4.18 eventSubList Command

4.18.1

Test Case	EGM	Host
EH-COR-00014	X	
EH-COR-00015	X	

4.19 clearEventSub Command

4.19.1

Test Case	EGM	Host
EH-COR-00015	X	
EH-COR-00022	X	

4.19.2

Test Case	EGM	Host
EH-COR-00047	X	

4.19.3

Test Case	EGM	Host
EH-COR-00022	X	

4.19.4

Test Case	EGM	Host
EH-COR-00015	X	
EH-COR-00022	X	

4.19.5

Test Case	EGM	Host
EH-COR-00015	X	
EH-COR-00022	X	

4.19.6

Test Case	EGM	Host
EH-COR-00015	X	
EH-COR-00022	X	

4.21 eventReport Command

4.21.2

Test Case	EGM	Host
CM-COR-00091	X	

4.21.3

Test Case	EGM	Host
EH-COR-00035		X

4.21.4

Test Case	EGM	Host
EH-COR-00035		X

4.21.5

Test Case	EGM	Host
EH-COR-00014	X	

4.23 G2S_EHE001 Device Disabled by EGM

4.23.1

Test Case	EGM	Host
EH-COR-00048	X	

4.24 G2S_EHE002 Device Not Disabled by EGM

4.24.1

Test Case	EGM	Host
EH-COR-00048	X	

4.25 G2S_EHE003 Device Disabled by Host

4.25.1

Test Case	EGM	Host
EH-COR-00058	X	

4.26 G2S_EHE004 Device Not Disabled by Host

4.26.1

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	

4.27 G2S_EHE005 Device Configuration Changed by Host

4.27.1

Test Case	EGM	Host
EH-COR-00038	X	

4.28 G2S_EHE006 Device Configuration Changed by Operator

4.28.1

Test Case	EGM	Host
EH-COR-00041	X	

4.29 G2S_EHE101 Event Subscription Changed

4.29.1

Test Case	EGM	Host
EH-COR-00005	X	
EH-COR-00007	X	
EH-COR-00015	X	

4.30 G2S_EHE102 Event Handler Queue Overflow

4.30.1

Test Case	EGM	Host
EH-COR-00059	X	

4.31 G2S_EHE103 Event Handler Queue Overflow Cleared

4.31.1

Test Case	EGM	Host
EH-COR-00059	X	

4.32 Command Examples

4.32.1

Test Case	EGM	Host
EH-COR-00014	X	

This chapter contains error codes that can be generated during CVT testing. Generic error codes, which can be generated during a step in most test cases, are grouped by type: Send Request Errors, Event Checking Errors, Event Not Expected Errors and DUT Error. Following the generic errors is a list of all errors that can occur during testing.

Send Request Errors

Error codes associated with the CVT sending a request to the EGM.

Error Code	Description
E-XX-00002	CVT timed out waiting for the response.
E-XX-00003	The wrong response was received. It MUST be ${expected}, but was ${actual}.
E-XX-00004	The wrong error was received. One of ${errorList} was expected, but ${actual} was received.
E-XX-00005	An unexpected error of ${errorCode} was received.
E-XX-00009	An unexpected response of ${response} was received. One of ${errorList} was expected.

Event Checking Errors

Error codes associated with checking events that should be generated during a step in a test case.

Error Code	Description
E-EH-00013	Event ${eventCode} status attribute ${attributeName} of ${actual} is wrong it should be ${expected}.
E-EH-00014	Event ${eventCode} log attribute ${attributeName} of ${actual} is wrong it should be ${expected}.
E-EH-00015	Event ${eventCode} meter value ${meterName} of ${actual} is wrong it should be ${expected}.
E-EH-00016	Event ${eventCode} has the wrong device class of ${actual} it should be ${expected}.
E-CB-00017	Event ${eventCode} has too many device transaction elements.
E-EH-00018	Event ${eventCode} log element is in the wrong namespace of ${actual} it should be ${expected}.
E-EH-00019	Event ${eventCode} log element has the wrong name of ${actual} it should be ${expected}.
E-CB-00020	Event ${eventCode} should have a device log.
E-CB-00021	Event ${eventCode} meter info type of ${actual} is wrong it should be ${expected}.
E-EH-00022	Event ${eventCode} should not have ${meterLevel} for ${device}.
E-EH-00023	Meters are not balanced for ${eventCode}.
E-EH-00024	Event ${eventCode} should have a device status of ${expected}.
E-EH-00025	Event ${eventCode} should not have a device status of ${actual}.
E-EH-00026	Event ${eventCode} status element is in the wrong namespace of ${actual} it should be ${expected}.
E-EH-00027	Event ${eventCode} status element has the wrong name of ${actual} it should be ${expected}.

Error Code	Description
E-EH-00030	Event ${eventCode} was expected, but it was not received.
E-EH-00032	One of ${eventList} was expected, but it was not received.
E-EH-00038	Event ${eventCode} was supposed to be retried, but it was not received.
E-EH-00041	Event ${eventCode} transaction ID MUST be zero (0).
E-EH-00042	Event ${eventCode} transaction ID must not be zero (0) for events associated with transactions.
E-EH-00056	One of ${eventList} was supposed to be retried, but it was not received.
E-EH-00058	Event ${eventCode} should have a meter set.

Event Not Expected Error

Error code associated with an event that should not have been generated during a step in a test case.

Error Code	Description
E-EH-00028	Event ${eventCode} should not be generated at this time.

DUT Error

Error code associated with the user canceling a test case.

Error Code	Description
E-XX-00001	The user cancelled the operation.

All Errors

Error Code	Description
E-CB-00001	The EGM is still EGM-disabled.
E-CB-00002	The cabinet device MUST be required for play, but it is not.
E-CB-00003	The cabinet device is host-disabled.
E-CB-00004	The cabinet device is host-enabled.
E-CB-00005	Money-in is disabled.
E-CB-00006	Money-in is enabled.
E-CB-00007	The EGM is disabled.
E-CB-00008	The EGM is not disabled.
E-CB-00009	The locked device MUST be gamePlay. It is ${deviceClass}.
E-CB-00010	The locked device MUST be cabinet. It is {$deviceClass}.
E-CB-00011	The locked device ID MUST be ${deviceId}.
E-CB-00012	The EGM was not configured.
E-CB-00013	Disable text MUST NOT be displayed.
E-CB-00014	Required text is not displayed.
E-CB-00015	The inserted bill MUST be accepted.

Error Code	Description
E-CB-00016	The inserted bill MUST be accepted.
E-CB-00017	Player cashable meter is wrong. It MUST be ${expected}, but it was ${actual}.
E-CB-00018	The EGM state of ${actual} is wrong it MUST be ${expected}.
E-CB-00019	While disabled, EGM MUST NOT process a lock out command.
E-CB-00020	Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.
E-CB-00021	The ${class} profile attribute ${attribute} is wrong since cabinetProfile.enhancedConfigMode is false. It MUST be ${expected}, but it was ${actual}.
E-CB-00022	EGM MUST only expose one cabinet device.
E-CB-00023	Locale identifiers MUST be constructed as language_country.
E-CB-00024	Cabinet style MUST be a unique 64 identifier data type.
E-CB-00025	Cabinet profile attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.
E-CB-00026	Cash out device of ${device} is not a valid cash out device.
E-CB-00027	The EGM MUST cash out the player when cabinetProfile.cashOutOnDisable is G2S_forceCashOut.
E-CB-00028	The EGM MUST NOT cash out the player when the EGM is disabled and cabinetProfile.cashOutOnDisable is ${cashOutOnDisableType}.
E-CB-00029	The EGM MUST allow the player to cash out when the EGM is disabled and cabinetProfile.cashOutOnDisable is G2S_allowCashOut.
E-CB-00030	The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.
E-CB-00031	The reelsTilted attribute of ${actual} in the cabinet status is not properly formatted. Only vertical bars and decimal values are allowed.
E-CB-00032	The user failed to clear the non-volatile storage.
E-CB-00033	While host locked, the EGM must not allow ${activity}.
E-CB-00034	The user failed to credit meter to the max credit meter value of ${maxCredit}.
E-CB-00035	The EGM MUST NOT have a credit meter value greater than ${maxCredit}.
E-CC-00001	Device identifier for host-oriented devices MUST be equal to the host identifier of the owner host.
E-CC-00002	EGM MUST have device class ${deviceClass} for host ID ${hostId}.
E-CC-00003	Host-oriented devices MUST not be owned by unregistered hosts.
E-CC-00004	EGM MUST not own host-oriented devices.
E-CM-00001	Endpoint must send valid XML.
E-CM-00002	Endpoint used the wrong end point ID ${endpointId}.
E-CM-00003	Device ${deviceClass} ${deviceId} structure should not have changed.
E-CM-00004	Invalid host ID of ${hostId}.
E-CM-00005	Device {$deviceId} does not have an owner.
E-CM-00006	An EGM ID of '${egmID}' is not valid.

Error Code	Description
E-CM-00007	Command ID of ${commandId} must be greater than ${lastCommandId}.
E-CM-00008	Command dateTime of ${dateTime} must be greater than ${lastDateTime}.
E-CM-00009	Command ${command} MUST NOT be sent as a notification.
E-CM-00010	Endpoint did not send a valid response to ${command}.
E-CM-00011	Endpoint MUST NOT send application-level response for a notification.
E-CM-00012	Invalid device ID for ${deviceClass} ${deviceId}.
E-CM-00013	Invalid host ID of ${hostId} in g2sAck.
E-CM-00014	Invalid EGM ID of '${egmId}' in g2sAck.
E-CM-00015	Error code MUST be G2S_none, except for application-level errors.
E-CM-00016	Error text MUST be empty if error code is G2S_none.
E-CM-00017	EGM MUST not send ${errorCode}.
E-CM-00018	Message MUST be syntactically correct per the G2S schema.
E-CM-00019	Device ID of ${deviceId} in a response MUST match the corresponding request device ID.
E-CM-00020	Application responses MUST use an existing session ID.
E-CM-00021	Notifications and responses MUST have a time-to-live value of zero.
E-CM-00022	The sessionMore attribute MUST be set to false.
E-CM-00023	Host MUST NOT send ${errorCode}.
E-CM-00024	Endpoint MUST support standard P2P transport.
E-CM-00025	Endpoint MUST support ${xmlEncoding}.
E-CM-00026	Endpoint MUST validate egmId and hostId in ${location}.
E-CM-00027	Host-owned devices MUST have the same device ID as the owner's host ID.
E-CM-00028	The endpoint MUST validate user-entered value for ${feature}.
E-CM-00029	Device ${device} has an invalid configurator of ${configuratorId}
E-CM-00030	Transaction ID MUST be continuous when the CVT owns all transaction log devices.
E-CM-00031	Transaction ID ${transactionId} MUST be greater than ${lastTransactionId}.
E-CM-00032	The endpoint MUST return an error for commands from unknown classes.
E-CM-00033	The endpoint MUST return an error for unknown commands.
E-CM-00034	G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.
E-CM-00035	The error code of ${errorCode} is not valid for a g2sAck. It MUST be G2S_none or start with G2S_ MSX.
E-CM-00036	The EGM must not allow commands from the non-guest host.
E-CM-00037	The EGM must not allow control commands from a guest host.
E-CM-00038	The EGM must generate a response to a request from the host.
E-CM-00039	The EGM must validate the G2S request.

Error Code	Description
E-CM-00040	The EGM must only allow control commands from the owner host.
E-CM-00041	Command ${commandName} has the wrong session type of ${actual} it should be ${expected}.
E-CM-00042	Retried commands MUST have new dateTimeSent values.
E-CM-00043	Command IDs MUST strictly increase by one. Command ID ${commandId} MUST be one greater than ${lastCommandId}.
E-CM-00044	The g2sProtocol attribute in commsStatus commands MUST always be set to true.
E-CM-00045	EGM initiated commands MUST always be sent to the device's owner host.
E-CM-00046	Device ${deviceClass}[${deviceId}] ownership, configurator or guest permission should not have changed.
E-CM-00047	Device ${deviceClass}[${deviceId}] MUST not be included in descriptor list for ${permissionType} devices.
E-CM-00048	Device ${deviceClass}[${deviceId}] MUST be included in descriptor list for ${permissionType} devices.
E-CM-00049	EGM MUST NOT send a keepAlive at this time.
E-CM-00050	CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.
E-CM-00051	EGM-originated request ${request} is not allowed when the comms state is ${commsState}.
E-CM-00052	CommsOnLine commands cannot be sent with a frequency less than one second.
E-CM-00053	CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.
E-CM-00054	EGM MUST report a command ID out of order error.
E-CM-00055	EGM MUST report an unknown namespace at the class level error.
E-CM-00056	EGM MUST transition to closing state when the host is unavailable.
E-CM-00057	The user failed to power cycle the EGM.
E-CM-00058	The user failed to change the device permission from the administrative interface.
E-CM-00059	Resent commands MUST have the same command dateTime values.
E-CM-00060	The endpoint MUST respond to a request.
E-CM-00061	CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.
E-CM-00062	Registered hosts MUST have a communications device.
E-CM-00063	Unregistered hosts MUST NOT have a communications device.
E-CM-00064	The user failed to deactivate a device.
E-CM-00065	The user failed to activate a device.
E-CM-00066	CommsOnLine MUST have deviceChanged attribute set to false if no device changes have occurred for the EGM.
E-CM-00067	The user failed to reset the commsDevice.
E-CM-00068	Subscriptions were lost but the commsOnLine.subscriptionLost was set to false.

Error Code	Description
E-CM-00069	Meters were reset but commsOnLine.metersReset was set to false.
E-CM-00070	The user failed to revoke the EGM's certificate.
E-CM-00071	CommsOnLine MUST have ${attribute} set to true when the EGM has its non-volatile storage cleared.
E-CM-00072	The user failed to change the device permission for ${device}.
E-CM-00073	The syncTimer MUST NOT be less than 15 seconds.
E-EH-00001	Event ${eventCode} contains additional data from a known namespace.
E-EH-00002	Event ${eventCode} was not expected based on host subscription and forced subscription.
E-EH-00003	Event ${eventCode} contains additional statusInfo elements for known namespaces.
E-EH-00004	Event ${eventCode} contains additional known meter set that is not affected by the event.
E-EH-00005	Event ${eventCode} contains additional transactionInfo elements for a known namespace.
E-EH-00006	Event ${eventCode} is missing affected data.
E-EH-00007	Event ${eventCode} did not update affected data per the protocol.
E-EH-00008	Event ${eventCode} did not update affected meters per the protocol.
E-EH-00009	Event ${eventCode} was generated when the affected data was not changed on the EGM.
E-EH-00010	Event ${eventCode} MUST be in the supported event list.
E-EH-00011	Event ${eventCode} has the wrong device class of ${actual} it should be ${expected}.
E-EH-00012	Event ${eventCode} has the wrong session type of ${actual} it should be ${expected}.
E-EH-00013	Event ${eventCode} status attribute ${attributeName} of ${actual} is wrong it should be ${expected}.
E-EH-00014	Event ${eventCode} log attribute ${attributeName} of ${actual} is wrong it should be ${expected}.
E-EH-00015	Event ${eventCode} ${classOrDevice} level meter ${meterName}${meterCategory} ${deltaOrValue} of ${actual} is wrong it should be ${expected}.
E-EH-00016	Event ${eventCode} has the wrong device class of ${actual} it should be ${expected}.
E-EH-00017	Event ${eventCode} has too many device transaction elements.
E-EH-00018	Event ${eventCode} log element is in the wrong namespace of ${actual} it should be ${expected}.
E-EH-00019	Event ${eventCode} log element has the wrong name of ${actual} it should be ${expected}.
E-EH-00020	Event ${eventCode} should have a device log.
E-EH-00021	Event ${eventCode} meter info type of ${actual} is wrong it should be ${expected}.
E-EH-00022	Event ${eventCode} should not have ${meterLevel} for ${device}.
E-EH-00023	Meters are not balanced for ${eventCode}.
E-EH-00024	Event ${eventCode} should have a device status of ${expected}.
E-EH-00025	Event ${eventCode} should not have a device status of ${actual}.
E-EH-00026	Event ${eventCode} status element is in the wrong namespace of ${actual} it should be ${expected}.

Error Code	Description
E-EH-00027	Event ${eventCode} status element has the wrong name of ${actual} it should be ${expected}.
E-EH-00028	Event ${eventCode} should not be generated at this time.
E-EH-00029	The ${eventCode} had invalid data per the contract.
E-EH-00030	Event ${eventCode} was expected, but it was not received.
E-EH-00031	Event ${eventCode} had invalid data per the contract.
E-EH-00032	One of ${eventList} was expected, but it was not received.
E-EH-00033	The event ID ${eventId} has already been used.
E-EH-00034	Event IDs MUST be contiguous when all events are subscribed to.
E-EH-00035	Event ${eventId} MUST have an event date/time after ${lastEventDateTime}.
E-EH-00036	Event ${eventCode} has different eventDateTime values for the same event ID.
E-EH-00037	Event ${eventCode} MUST have the same event code for the same event ID.
E-EH-00038	Event ${eventCode} was supposed to be retried, but it was not received.
E-EH-00039	Event ${eventCode} MUST not have meter definitions attributes.
E-EH-00040	Wild cards are not allowed in eventSubList responses.
E-EH-00041	Event ${eventCode} transaction ID MUST be zero.
E-EH-00042	Event ${eventCode} transaction ID must not be zero for events associated with transactions.
E-EH-00043	Event log list MUST be contiguous. A skip was detected at ${sequenceNumber}.
E-EH-00044	Event log MUST start with logSequence number ${sequenceNumber}.
E-EH-00045	Event log MUST have ${logEntries} entries.
E-EH-00046	Event log MUST be empty when requesting a lastSequence number not in the log.
E-EH-00047	EGM MUST return an empty supported event list for ${deviceClass}[${deviceId}].
E-EH-00048	Wild cards are not allowed in supportEvents responses.
E-EH-00049	Supported events list has an unknown device ${deviceClass}[${deviceId}].
E-EH-00050	Supported events list MUST include events for ${deviceClass}[${deviceId}].
E-EH-00051	EGM MUST NOT accept an event subscription for ${deviceClass}[${deviceId}].
E-EH-00052	EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.
E-EH-00053	Event subscription for ${deviceClass}[${deviceId}] ${eventCode} MUST have ${attribute} set to ${expected}.
E-EH-00054	EGM must not have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.
E-EH-00055	EGM MUST return an empty event subscription list for ${deviceClass}[${deviceId}], event code ${eventCode}.
E-EH-00056	One of ${eventList} was supposed to be retried, but it was not received.
E-EH-00057	Event ${eventCode} is not allowed from ${eventState} state.
E-EH-00058	Event ${eventCode} should have a meter set.
E-EH-00059	Event ${eventCode} should have an entry in the event handler log list.

Error Code	Description
E-EH-00060	Event ${eventCode} MUST have an affected device entry for ${device}.
E-EH-00061	Event ${eventCode} MUST NOT be sent to the CVT since the CVT does not have permission for ${device}.
E-EH-00062	Event ${eventCode} MUST re-gather affected ${type} ${name} when it is resent.
E-EH-00063	Event subscriptions must be persisted.
E-EH-00064	Event handler status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.
E-EH-00065	Non-persisted events MUST NOT be stored in the event handler log.
E-EH-00066	Event queue is not full.
E-EH-00067	Event log MUST be persisted.
E-EH-00068	Event log MUST persist affected data sets for events.
E-EH-00069	Event log attribute eventAck must be false for unacknowledged event reports.
E-EH-00070	Event log attribute eventAck must be true for acknowledged event reports.
E-EH-00071	EGM MUST have a forced subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.
E-EH-00072	EGM MUST clear the event subscriptions for a device when the host no longer owns nor is it a guest of that device.
E-GP-00001	Recall log list MUST be contiguous. A skip was detected at ${sequenceNumber}.
E-GP-00002	Recall log MUST start with logSequence number ${sequenceNumber}.
E-GP-00003	Recall log MUST have ${logEntries} entries.
E-GP-00004	Recall log MUST be empty when requesting a lastSequence number not in the log.
E-GP-00005	Game denom ${denomId} must be ${activeState}.
E-GP-00006	Inactive game denom ${denomId} MUST NOT be playable for game play device ${deviceId}.
E-GP-00007	Enabled game play devices MUST have at least one active game denom.
E-GP-00008	EGM MUST only report denom ${denomId} once in game denom list.
E-GP-00009	Game recall log entry does not match game play event value. Attribute ${attribute} MUST be ${expected}, but it was ${actual}.
E-GP-00010	Game play status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.
E-GP-00011	Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.
E-GP-00012	Recall Log MUST have a winLevelItem when initialWin or secondaryWin is non-zero.
E-GP-00013	Win level index of ${winLevelIndex} MUST exist in the gamePlayProfile.
E-GP-00014	User observed game play results do not match game play recall log.
E-GP-00015	Game play device that allows progressives MUST have a win-level that allows progressives.
E-GP-00016	Game play device that allows secondary games MUST have a win-level that allows secondary games.
E-GP-00017	Game play device that MUST have at least one win-level.
E-GP-00018	A win level index, denomation and number of credits wagered (${winLevelIndex}/${denom} /${credits}) for a game play device MUST NOT be linked to more than one progressive level.

Error Code	Description
E-JP-00001	Handpay log list MUST be contiguous. A skip was detected at ${sequenceNumber}.
E-JP-00002	Handpay log MUST start with logSequence number ${sequenceNumber}.
E-JP-00003	Handpay log MUST have ${logEntries} entries.
E-JP-00004	Handpay log MUST be empty when requesting a lastSequence number not in the log.
E-JP-00005	EGM MUST only expose one handpay device.
E-JP-00006	Handpay type must be ${handpayType}.
E-JP-00007	Handpay ${cashType} requested amount must be ${requestAmt}.
E-JP-00008	Handpay source reference list must be empty for a cancel-credit handpay.
E-JP-00009	Retried handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.
E-JP-00010	Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.
E-JP-00011	Cancel-credit handpays must overwrite the last handpay log entry if that entry is a cancelled cancel-credit handpay.
E-JP-00012	Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.
E-JP-00013	Handpay requests MUST have unique transaction IDs, a handpay request for ${transactionId} already exists.
E-JP-00014	Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.
E-JP-00015	Handpay remote key-off ack MUST set transactionId to ${expected}, but was ${actual}.
E-JP-00016	Handpay source references MUST be included in non-cancelled-credit handpays.
E-JP-00017	The total value of source references of ${sourceAmount} MUST equal EGM paid amount ${egmPaidAmount} plus request amount ${requestAmount}.
E-JP-00018	EGM MUST return a G2S_JPX003 Transaction Is Not Currently Pending error for an transaction that has already been keyed-off.
E-JP-00019	The user failed to create a handpay.
E-JP-00020	In G2S 2.1, handpay request commands MUST set requestPromoAmt and requestNonCashAmt to zero.
E-JP-00021	EGM MUST NOT send a handpayRequest when the handpay is no longer in the G2S_ handpayRequest state.
E-MS-00001	The endpoint did not reconnect in ${timeOut} seconds.
E-MS-00002	The endpoint did not query the OCSP server for ${expectedCertificate}.
E-MS-00003	The endpoint unexpectedly disconnected from CVT.
E-MS-00004	Endpoint MUST disconnect from CVT.
E-MS-00005	Endpoint MUST NOT call the getTransportOptions method.
E-MS-00006	The endpoint MUST NOT compress the message using ${compressionType}.
E-MS-00007	The wrong transport protocol was received. It MUST be ${expected}, but it was ${actual}.
E-MS-00008	The wrong protocol was in the egmLocation. It MUST be ${expected}, but it was ${actual}.

Error Code	Description
E-MS-00009	The endpoint MUST NOT close the connection using "Connection: close".
E-MS-00010	The endpoint MUST NOT use ${actual} compression. It MUST use ${expected}.
E-MS-00011	The endpoint dateTimeSent is not within 5.1 seconds of the CVT dateTimeSent value.
E-MS-00012	The endpoint URL of ${actual} MUST be an absolute URI.
E-MS-00013	Endpoint MUST support TLS 1.0.
E-MS-00014	Endpoint MUST support SHA-1.
E-MS-00015	Certificates MUST be X.509 v3.
E-MS-00016	The RSA certificate key size of ${actual} is less than 1024 bits.
E-MS-00017	A cipher of ${actual} MUST have a key strength of 112-bit key strength.
E-MS-00018	A cipher of ${actual} MUST have an authentication algorithm.
E-MS-00019	The MD5 algorithm authentication code size of ${actual} is less than 128 bits.
E-MS-00020	The SHA-1 algorithm authentication code size of ${actual} is less than 160 bits.
E-MS-00021	${algorithm} authentication code size of ${actual} is less than 80 bits.
E-MS-00022	The endpoint MUST have a new certificate after the previous certificate expired.
E-MS-00023	The endpoint MUST use a certificate to authenticate communications.
E-MS-00024	The CA certificate MUST have all basic fields.
E-MS-00025	The CA certificate MUST have identical issue and subject fields.
E-MS-00026	The CA certificate does not have a basic constraint extension marked as critical.
E-MS-00027	The CA certificate pathLenConstraint file is less than zero.
E-MS-00028	The endpoint certificate MUST have all basic fields.
E-MS-00029	The endpoint certificate MUST have different issue and subject fields.
E-MS-00030	The endpoint certificate MUST have a subject alternative name set.
E-MS-00031	The endpoint certificate MUST have a distinguished name.
E-MS-00032	Wrong common name. It MUST be ${expected}, but it was ${actual}.
E-MS-00033	Wrong organization name. It MUST be ${expected}, but it was ${actual}.
E-MS-00034	The endpoint MUST support RSA certificates with key pairs of size ${size}.
E-MS-00035	The endpoint MUST have an administrative interface for ${feature}.
E-MS-00036	The endpoint MUST have a new certificate after regenerating public-private key pairs.
E-MS-00037	The endpoint MUST contact the SCEP server.
E-MS-00038	The endpoint MUST NOT have connected to CVT.
E-MS-00039	The endpoint MUST create a new TLS session.
E-MS-00040	The endpoint MUST have a new certificate.
E-MS-00041	Wrong egmLocation. The URL MUST only have a dotted-IP address.
E-MS-00042	The endpoint MUST have a distinguished name.

Error Code	Description
E-MS-00043	The endpoint MUST have an organizational name of ${expected}.
E-MS-00044	The endpoint did not query SCEP server during the last half of the validity period.
E-MS-00045	DHCP values MUST be used when DHCP is enabled.
E-MS-00046	The endpoint MUST NOT be using transport option ${transportOption}.
E-MS-00047	The endpoint MUST NOT implement the getTransportOptions SOAP method.
E-MS-00048	The endpoint MUST include ${transportOption} in the getTransportOptions response.
E-MT-00001	Ever-increasing meters MUST always be increasing.
E-MT-00002	Meter values are not in balance.
E-MT-00003	EGM MUST return a G2S_MTX001 Invalid meterSubType Specified error for an unknown meterSubType.
E-MT-00004	EGM MUST return a G2S_MTX002 Invalid Periodic Meter Values error for ${attribute} = ${value}.
E-MT-00005	Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.
E-MT-00006	Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.
E-MT-00007	Wild cards are not allowed in ${command} commands.
E-MT-00008	Meter ${meterName} value of ${actual} is wrong it should be ${expected}.
E-MT-00009	Device ${device} was not expected in ${meterType} meter list.
E-MT-00010	Device ${device} was expected in ${meterType} meter list.
E-MT-00011	Meter ${meterName} in ${device} was expected to have meter definitions.
E-MT-00012	Meter ${meterName} in ${device} was not expected to have meter definitions.
E-MT-00013	Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.
E-MT-00014	Meter ${meterName} in ${device} sub values of ${subValue} for ${type} does not equal the device value of ${deviceValue}.
E-MT-00015	Periodic meter subscription should have a frequency of ${expected}, but it was ${actual}.
E-MT-00016	Meter info type is wrong. It MUST be ${expected}, but it was ${actual}.
E-MT-00017	Wilcard meter subscriptions must be expanded in the meterSubList.
E-MT-00018	Wrong meter subscription type of ${actual} was sent in the meterSubList, it should be ${expected}.
E-MT-00019	Meter subscription ${meterSubType} MUST be persisted, the ${attribute} value of ${actual} is wrong, is should be ${expected}.
E-MT-00020	Periodic meter subscription date/time of ${meterDateTime} MUST be sent at intervals of ${periodicInterval} from an offset of ${periodicBase} seconds from midnight.
E-MT-00021	Meter info list for unknown device class or invalid device MUST be empty.
E-MT-00022	EGM MUST send end-of-day meters after the eodBase trigger point of ${eodBase}.
E-MT-00023	The sum of wager category wagered amounts of MUST equal the actual amount wagered.
E-MT-00024	Wager category ${wagerCategory} played count meter must be increment by one when a game is

Error Code	Description
	played for that wager category.
E-MT-00025	The game play device MUST have at least one game denomination at the ${levelName} level.
E-MT-00026	The sum of game denomination meter deltas of ${actual} for ${meterName} does not match device performance meter delta ${expected}.
E-MT-00027	EGM must report ${field} for each distinct reported game denomination.
E-MT-00028	EGM must respond to queries for game denom meters.
E-MT-00029	Game denom wager amount of ${wageredAmt} is not a multiple of reported denomination ${denomValue}.
E-OC-00001	EGM MUST set optionChangeStatus attribute ${attribute} to ${expected}, but was ${actual}.
E-OC-00002	Option config ${paramId} MUST have canModRemote set to ${canModRemote}.
E-OC-00003	Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.
E-OC-00004	Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.
E-OC-00005	Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.
E-OC-00006	Option config parameter ${paramId} is not configured properly. Parameter MUST exist.
E-OC-00007	Option item ${optionId} MUST exist.
E-XX-00001	The user cancelled the operation.
E-XX-00002	CVT timed out waiting for the response.
E-XX-00003	The wrong response was received. It MUST be ${expected}, but was ${actual}.
E-XX-00004	The wrong error was received. One of ${errorList} was expected, but ${actual} was received.
E-XX-00005	An unexpected error of ${errorCode} was received.
E-XX-00006	The user stopped the test session.
E-XX-00007	CVT timed out waiting for the communications state to reach ${commsState}.
E-XX-00008	CVT timed out during an instruction.
E-XX-00009	An unexpected response of ${response} was received. One of ${errorList} was expected.
E-XX-00010	The device ${device} is not ready to execute a test case (${reason}).
E-XX-00011	An unknown exception ${exceptionMessage} has caused the step to fail (${exceptionId}).
E-XX-00012	There were no transcript records for the assertion phase to evaluate.
E-XX-00013	CVT timed out waiting for ${expectedRequest} from the EGM.
E-XX-00014	EGM request of ${summary} MUST have a timeToLive of ${expected} which is the same value as the ${profileCommand}.timeToLive value.
E-XX-00015	EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).
E-XX-00016	Each log MUST have a separate counter.
E-XX-00017	The ${command} request should not be generated at this time.

Error Code	Description
E-XX-00018	CVT timed out waiting for ${expectedRequest} from the host.
E-XX-00019	An unexpected response of ${response} was received.
E-XX-00020	The user did not cause device fault for event ${eventCode}.
E-XX-00021	The ${deviceClass} log sequence number MUST increment by one for a new log entry.
E-XX-00022	The user failed to fund the EGM.
E-XX-00023	The EGM MUST persist ${dataStructure} for the ${class} device. (${attribute} : ${originalValue} /${currentValue}])
E-XX-00024	Log sequence number for ${deviceClass} MUST be persisted.
E-XX-00025	Each log MUST have its own sequence number.
E-XX-00026	The user failed to press the cash-out button.

cabinet

Cabinet
Functional Groups

Test Case: CB-COR-00001

This case verifies that the EGM does not accept commands from non-guest hosts.

Objective

Verify that the EGM does not accept commands from non-guest hosts.

Requirements Under Test

- 1.8.5
- 3.6.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	non-guest

Test Procedure

1. **Send a `setCabinetState` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
   {c2cce615-17b0-4678-809b-8ff75fe58118}

   Command
   - `cabinet.setCabinetState`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

2. **Send a `setCabinetLockOut` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
   {9a4457f8-e9af-43fe-8090-f1de3bfa0a9a}

   Command
   - `cabinet.setCabinetLockOut`

Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

3. **Send a `setDateTime` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
{7aeab468-d987-468d-00ea-81f364bbb451}

   Command

   - cabinet.setDateTime

     Expect **G2S_APX010** or **G2S_APX012**

   Errors

   - Send Request Errors
   - E-CM-00036 [The EGM must not allow commands from the non-guest host.]

4. **Send a `getCabinetStatus` command, and verify that EGM responds with `G2S_APX012`.**
{5fd6f807-df6c-48c9-806c-228f82033960}

   Command

   - cabinet.getCabinetStatus

     Expect **G2S_APX012**

   Errors

   - Send Request Errors
   - E-CM-00036 [The EGM must not allow commands from the non-guest host.]

5. **Send a `getCabinetProfile` command, and verify that the EGM responds with `G2S_APX012`.**
{44952334-1fc2-467b-00e6-ff3939cdcccc}

   Command

   - cabinet.getCabinetProfile

     Expect **G2S_APX012**

   Errors

   - Send Request Errors
   - E-CM-00036 [The EGM must not allow commands from the non-guest host.]

6. **Send a `getDateTime` command, and verify that EGM responds with `G2S_APX012`.**
{188c02a7-dab8-4dfd-80dc-0ffd097de920}

   Command

- `cabinet.getDateTime`
  Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

Test Case: CB-COR-00002

This case verifies that the EGM does not accept control commands from a guest host.

Objectives

- Verify that the EGM does not accept control commands from a guest host.
- Verify that the EGM accepts non-control commands from a guest host.

Requirements Under Test

- 1.8.14
- 3.6.1
- 3.6.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest

Test Procedure

1. **Send a `setCabinetState` command, and verify that the EGM responds with `G2S_APX010`.**
   {f12ea029-4a2b-4ae5-80d7-149f7e8d4447}

   Command
   - `cabinet.setCabinetState`

     Expect **G2S_APX010**

   Errors
   - Send Request Errors
   - `E-CM-00037` [The EGM must not allow control commands from a guest host.]

2. **Send a `setCabinetLockOut` command, and verify that the EGM responds with `G2S_APX010`.**
   {a587e9d0-59a4-4a15-804f-dbf50fe68faa}

   Command
   - `cabinet.setCabinetLockOut`

Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00037` [The EGM must not allow control commands from a guest host.]

3. **Send a `setDateTime` command, and verify that the EGM responds with `G2S_APX010`.**
{26d082b8-e1c0-4abe-0020-bf8e8208469e}

Command

- `cabinet.setDateTime`

Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00037` [The EGM must not allow control commands from a guest host.]

4. **Send a `getCabinetStatus` command, and verify that the EGM responds properly.**
{ad5ad9b8-fe93-48b4-8051-e9e1aef354c1}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

5. **Send a `getCabinetProfile` command, and verify that the EGM responds properly.**
{c42ba1f2-6294-4d52-8000-1b79b7224dc9}

Command

- `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

6. **Send a `getDateTime` command, and verify that the EGM responds properly.**
{c70ad206-dfcb-441c-00a3-e44d41e8c25d}

Command

- `cabinet.getDateTime`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

Test Case: CB-COR-00003

This case tests that the EGM supports the required `cabinetDateTime` attribute in the `cabinet.setDateTime` command.

Objective

Verify that the EGM validates the `cabinet.setDateTime` command.

Requirements Under Test

- 1.27.33

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send a `cabinet.setDateTime` command to the EGM without a `cabinetDateTime` attribute.**
   {e71ead74-9bcf-448b-8e96-325f3d675e52}

   #### Command

   - `cabinet.setDateTime`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   #### Errors

   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

Test Case: CB-COR-00004

This case verifies that `cabinet` event subscriptions associated with disabling the cabinet are correct.

Objectives

- Verify that event subscriptions are evaluated at the time an event is generated.
- Verify that affected data is collected when the `eventReport` command is sent.
- Verify that `cabinet` events have the correct affected data.
- Verify that the EGM generated the appropriate events when the `cabinetStatus.egmState` attribute is changed.
- Verify that the `cabinetStatus` is properly updated when the `cabinet` device is disabled.

Requirements Under Test

- 1.23.1
- 1.23.3
- 1.36.9
- 3.3.1
- 3.4.1
- 3.40.1
- 3.51.1
- 3.51.2
- 3.52.1
- 3.52.2
- 3.55.1
- 3.55.2
- 3.56.1
- 3.56.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager to suppress responses to `eventReport` commands.**
{20dbe466-542a-4340-00c4-41b1902afd3c}

2. **Open the cabinet door.**
{ec655015-18d2-487a-00f5-cede6a34cc5c}

   Action
   - Request CABINET door OPEN.

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE001` [EGM Disabled Cabinet] or `G2S_CBE203` [Device Fault Disabled EGM] or `G2S_CBE307` [Cabinet Door Open]

   Errors
   - Event Checking Errors
   - DUT Error

3. **Open the logic door.**
{96da3fe3-4f38-4b96-006b-bb0a1da42748}

   Action
   - Request LOGIC door OPEN.

   Event
   (Time out: ${retry.timeOut})
   - `G2S_CBE001` [EGM Disabled Cabinet] or `G2S_CBE203` [Device Fault Disabled EGM] or `G2S_CBE303` [Logic Door Open] or `G2S_CBE307` [Cabinet Door Open]

   Errors
   - Event Checking Errors
   - DUT Error

4. **Wait for the EGM to retry an `eventReport` command.**
{15fd1db2-a163-42bf-8ce1-f8d2f0b86b38}

   Event
   (Time out: ${retry.timeOut})
   - `G2S_CBE001` [EGM Disabled Cabinet] or `G2S_CBE203` [Device Fault Disabled EGM] or `G2S_CBE303` [Logic Door Open] or `G2S_CBE307` [Cabinet Door Open]

   Error
   - Event Checking Errors

5. **Set the response manager to respond to `eventReport` commands.**
{1f8fd8f7-0856-4efc-003b-993d94d7b01b}

Events
(Time out: ${retry.timeOut})

- `G2S_CBE307` [Cabinet Door Open]
- `G2S_CBE001` [EGM Disabled Cabinet]
- `G2S_CBE203` [Device Fault Disabled EGM]
- `G2S_CBE303` [Logic Door Open]

Error

- Event Checking Errors

6. **Set the event subscription for meters only.**
{87f451d2-4af7-4e43-bc7b-55c187e02f77}

Commands

- `eventHandler.clearEventSub`
- `eventHandler.setEventSub`
- `eventHandler.getEventSub`

Error

- Send Request Errors

7. **Close the logic door.**
{f0248912-d420-460f-8079-5f72aed6c516}

Action

- Request LOGIC door CLOSE.

Event
(Time out: ${event.timeOut})

- `G2S_CBE304` [Logic Door Closed]

Errors

- Event Checking Errors
- DUT Error

8. **Set the event subscription for no affected data.**
{969425fe-05e6-4a89-004e-1c5050faac63}

Commands

- `eventHandler.getEventSub`
- `eventHandler.clearEventSub`
- `eventHandler.setEventSub`

Error

- Send Request Errors

9. **Close the cabinet door.**
{af3fc493-e9e8-45ed-00d2-e85eb10def9f}

Action

- Request CABINET door CLOSE.

Events
(Time out: ${event.timeOut})

- `G2S_CBE308` [Cabinet Door Closed]
- `G2S_CBE002` [EGM Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

10. **Verify the EGM state is "enabled."**
{08f2f36a-2a2a-485f-b225-47a62f141f30}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

Test Case: CB-COR-00005

This case verifies that the EGM supports cabinet disables and locks.

Objectives

- Verify that the EGM handles the `setCabinetState` command.
- Verify that the EGM handles the `setCabinetLockOut` command.
- Verify that the EGM properly sets the `cabinetStatus` command when the EGM transitions between EGM states.

Requirements Under Test

- 1.9.6
- 1.36.9
- 3.3.1
- 3.3.2
- 3.3.3
- 3.3.7
- 3.3.8
- 3.3.17
- 3.3.18
- 3.6.4
- 3.10.22
- 3.28.1
- 3.29.1
- 3.32.1
- 3.33.1
- 3.34.1
- 3.35.1
- 3.36.1
- 3.37.1
- 3.41.1
- 3.42.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Get the cabinet status.**
   {db378d50-697c-4457-969e-0dffa514194b}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

2. **Disable the cabinet.**
   {c4026cf2-348c-457e-8018-14acc2279965}

   Command
   - `cabinet.setCabinetState`

   Events
   (Time out: ${event.timeOut})
   - `G2S_CBE003` [Host Disabled Cabinet]
   - `G2S_CBE204` [Host Command Disabled EGM]

   Errors
   - Send Request Errors
   - Event Checking Errors

3. **Verify that the EGM state is "host disabled."**
   {00a45e1b-fb56-408d-aec7-576286261e48}

   Command
   - `cabinet.getCabinetStatus`

   Errors
   - Send Request Errors
   - `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

4. **Lock the cabinet.**
   {ef5f045e-8c2b-4830-008a-afb9b97c3e1d}

   Command
   - `cabinet.setCabinetLockOut`

     Expect **G2S_APX016**

   Errors

- Send Request Errors
- E-CB-00019 [While disabled, EGM MUST NOT process a lock out command.]

5. **Enable the cabinet, but disable game play and money in.**
{eadd6b3e-74e3-4d93-00a0-268b9ec72be5}

Command
- cabinet.setCabinetState

Events
(Time out: ${event.timeOut})
- G2S_CBE004 [Host Enabled Cabinet]
- G2S_CBE101 [Host Disabled Game Play]
- G2S_CBE103 [Host Disabled Money In]
- G2S_CBE205 [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

6. **Enable game play and money in.**
{20b4a8ab-d23b-4b4f-0022-0500dbe04158}

Command
- cabinet.setCabinetState

Events
(Time out: ${event.timeOut})
- G2S_CBE102 [Host Enabled Game Play]
- G2S_CBE104 [Host Enabled Money In]

Errors
- Send Request Errors
- Event Checking Errors
- Event Not Expected Error
- E-CB-00018 [The EGM state of ${actual} is wrong it MUST be ${expected}.]

7. **Lock the cabinet.**
{8d1df4a8-60bb-457b-0092-002edf262e77}

Command
- cabinet.setCabinetLockOut

Events
(Time out: ${event.timeOut})

- G2S_CBE009 [Host Locked Cabinet]
- G2S_CBE211 [Host Action Locked EGM]

Errors

- Send Request Errors
- Event Checking Errors

8. **Verify that the EGM state is "host locked."**
{01483e68-472c-4af0-bc76-ae78714b3940}

Command

- cabinet.getCabinetStatus

Errors

- Send Request Errors
- E-CB-00018 [The EGM state of ${actual} is wrong it MUST be ${expected}.]

9. **Disable the cabinet.**
{f756924d-d8b2-4b1a-003e-a80293fba86d}

Command

- cabinet.setCabinetState

Event

(Time out: ${event.timeOut})

- G2S_CBE003 [Host Disabled Cabinet]

Errors

- Send Request Errors
- Event Checking Errors
- Event Not Expected Error
- E-CB-00018 [The EGM state of ${actual} is wrong it MUST be ${expected}.]

10. **Unlock the cabinet.**
{7097418a-bd07-4751-8002-61b7001ad432}

Command

- cabinet.setCabinetLockOut

Events

(Time out: ${event.timeOut})

- G2S_CBE010 [Host Unlocked Cabinet]
- G2S_CBE204 [Host Command Disabled EGM]

Errors

- Send Request Errors
- Event Checking Errors
- Event Not Expected Error

11. **Verify that the EGM state is "host disabled."**
{03268fbd-c081-4971-829a-9a6237a05903}

Command
- `cabinet.getCabinetStatus`

Errors
- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

12. **Enable the cabinet.**
{048d9585-c550-432e-003a-db1ab6ebc629}

Command
- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})
- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

13. **Verify that the EGM state is "enabled."**
{0449d995-e378-46fe-a700-41bc0f3886e8}

Command
- `cabinet.getCabinetStatus`

Errors
- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

Test Case: CB-COR-00006

This case tests that the EGM generates the proper events when the service lamp is activated and deactivated.

Objectives

- Verify that the EGM sends a `G2S_CBE301` event when the service lamp is activated.
- Verify that the EGM sends a `G2S_CBE302` event when the service lamp is deactivated.

Requirements Under Test

- 3.49.1
- 3.50.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Activate the service lamp.**
   {c6e32522-4871-4132-009f-ddeb6c4f4e2b}

   #### Action

   - Instruct the user to 'Press the service lamp button'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE301` [Service Lamp On]

   #### Errors

   - Event Checking Errors
   - DUT Error

2. **Deactivate the service lamp.**
   {262d10cd-5b48-41c8-00e8-b8eb4197ce88}

   #### Action

   - Instruct the user to 'Press the service lamp button'.

Event

(Time out: ${event.timeOut})

- `G2S_CBE302` [Service Lamp Off]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00007

This case tests that the EGM sends a `G2S_CBE315 Date/Time Changed` event when the cabinet time is changed by more than 5 seconds.

Objective

Verify that the EGM sends a `G2S_CBE315 Date/Time Changed` event when the cabinet time is changed by more than 5 seconds.

Requirements Under Test

- 3.12.1
- 3.12.2
- 3.68.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Determine if the EGM supports setting the cabinet date/time.**
   {2dfa419f-0b7b-4233-bb2c-88ef48755081}

   Command
   - `cabinet.setDateTime`

     Expect **G2S_CBX001** or **Normal Response**

   Error
   - Send Request Errors

2. **If the EGM supports the `setDateTime` command for the `cabinet` class:**

   a. **Instruct user to set the cabinet time to 1 minute in the past.**
   {8a8f7707-c905-4fd2-00a7-566abde063ab}

   Action
   - Instruct the user to 'Set the cabinet date/time to ${g_past}'.

   Event

(Time out: ${event.timeOut})

- G2S_CBE315 [Date/Time Changed]

Errors

- Event Checking Errors
- DUT Error
- E-MS-00035 [The endpoint MUST have an administrative interface for ${feature}.]

b. **Set the time to 10 seconds in the past.**
{d70de258-7f49-40f9-007e-cfa6af8763e3}

Command

- cabinet.setDateTime

Event
(Time out: ${event.timeOut})

- G2S_CBE315 [Date/Time Changed]

Errors

- Send Request Errors
- Event Checking Errors

c. **Set the time to the current time.**
{5f2900c1-6185-4307-8022-5af8f896fad2}

Command

- cabinet.setDateTime

Event
(Time out: ${event.timeOut})

- G2S_CBE315 [Date/Time Changed]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: CB-COR-00008

This case tests that the EGM rejects unknown commands.

Objective

Verify that the EGM sends the proper error for unknown commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Send a `cvtTesting` command to the EGM.**
   {fc5f6072-d262-4c3a-80c2-b293b5252e7a}

   Command
   - `cabinet.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   Errors
   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: CB-COR-00009

This case verifies that `cabinet` events associated with opening and closing the EGM doors are correct.

Objectives

- Verify that `cabinet` events have the correct affected data.
- Verify that the EGM generates the appropriate events when the EGM doors are opened and closed.

Requirements Under Test

- 3.3.1
- 3.4.1
- 3.26.2
- 3.27.1
- 3.40.1
- 3.42.1
- 3.51.1
- 3.51.2
- 3.52.1
- 3.52.2
- 3.53.1
- 3.53.2
- 3.54.1
- 3.54.2
- 3.55.1
- 3.55.2
- 3.56.1
- 3.56.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Open the cabinet door.**
   {013a2040-5244-44f1-8c79-0942c4de30be}

Action

- Request CABINET door OPEN.

Events
(Time out: ${event.timeOut})

- G2S_CBE307 [Cabinet Door Open]
- G2S_CBE001 [EGM Disabled Cabinet]
- G2S_CBE203 [Device Fault Disabled EGM]

Errors

- Event Checking Errors
- DUT Error

2. **Open the logic door.**
{02e0e7fa-f36d-4974-8483-70807acf5464}

Action

- Request LOGIC door OPEN.

Event
(Time out: ${retry.timeOut})

- G2S_CBE303 [Logic Door Open]

Errors

- Event Checking Errors
- DUT Error

3. **Close the logic door.**
{03dc35a8-2868-41d4-a1da-a96aa263396b}

Action

- Request LOGIC door CLOSE.

Event
(Time out: ${event.timeOut})

- G2S_CBE304 [Logic Door Closed]

Errors

- Event Checking Errors
- DUT Error

4. **Close the cabinet door.**
{04600820-0bd1-4f7b-a87e-9d9e1495ae08}

Action

- Request CABINET door CLOSE.

Events
(Time out: ${event.timeOut})

- G2S_CBE308 [Cabinet Door Closed]
- G2S_CBE002 [EGM Enabled Cabinet]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

5. **Open the auxiliary door.**
{05a067af-a6a9-416e-94fe-a4772ae9be44}

Action

- Request AUXILIARY door OPEN.

Events
(Time out: ${retry.timeOut})

- G2S_CBE305 [Auxiliary Door Open]
- G2S_CBE001 [EGM Disabled Cabinet]
- G2S_CBE203 [Device Fault Disabled EGM]

Errors

- Event Checking Errors
- DUT Error

6. **Close the auxiliary door.**
{06198f39-821e-4ebf-92cb-04f8f49a4eb5}

Action

- Request AUXILIARY door CLOSE.

Events
(Time out: ${event.timeOut})

- G2S_CBE306 [Auxiliary Door Closed]
- G2S_CBE002 [EGM Enabled Cabinet]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

7. **Verify the EGM state is "enabled."**
{07847ab7-e426-4aae-9af8-71dcf7d0b295}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

Test Case: CB-COR-00010

This case tests that the disable text is displayed when the cabinet is disabled.

Objective

Verify that the EGM displays the disable text from the `setCabinetState` command.

Requirements Under Test

- 3.3.1
- 3.3.3
- 3.3.8
- 3.4.1
- 3.5.2
- 3.5.4
- 3.5.6
- 3.5.12
- 3.5.16
- 3.5.19
- 3.8.5
- 3.10.22
- 3.28.1
- 3.29.1
- 3.41.1
- 3.42.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Get the cabinet status.**
   {01a38462-ba2d-426d-8fea-8a0ad1caca8c}

   Command
   - `cabinet.getCabinetStatus`

   Error

- Send Request Errors

2. **Disable the cabinet.**
{02bbd85f-39f5-4f48-8fd6-18628f601622}

Command
- `cabinet.setCabinetState`

Actions
- Verify that text 'CB-COR-00010 Step #2' appears onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_CBE003` [Host Disabled Cabinet]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00004` [The cabinet device is host-enabled.]
- `E-CB-00014` [Required text is not displayed.]

3. **Open the cabinet door.**
{0382e076-3c0d-42ac-b985-b8eb30460435}

Actions
- Request CABINET door OPEN.
- Verify that text 'CB-COR-00010 Step #2' appears onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_CBE307` [Cabinet Door Open]
- `G2S_CBE001` [EGM Disabled Cabinet]

Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00014` [Required text is not displayed.]

4. **Close the cabinet door.**
{044c1e05-ba92-4191-830c-9c58b6f518ab}

Actions
- Request CABINET door CLOSE.

- Verify that text 'CB-COR-00010 Step #2' appears onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_CBE308` [Cabinet Door Closed]
- `G2S_CBE002` [EGM Enabled Cabinet]

Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00014` [Required text is not displayed.]

5. **Disable the cabinet with an empty disable text.**
{058ffefb-4189-46fc-b341-63027c20db2a}

Command
- `cabinet.setCabinetState`

Actions
- Verify that text 'CB-COR-00010 Step #5' does NOT appear onscreen.

Errors
- Send Request Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00004` [The cabinet device is host-enabled.]
- `E-CB-00013` [Disable text MUST NOT be displayed.]

6. **Disable the cabinet with a known disable text.**
{06ee0aa3-fc17-493b-a365-3e0ec022339f}

Command
- `cabinet.setCabinetState`

Actions
- Verify that text 'CB-COR-00010 Step #6' appears onscreen.

Errors
- Send Request Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00004` [The cabinet device is host-enabled.]
- `E-CB-00014` [Required text is not displayed.]

7. **Enable the cabinet.**
{07c41579-bdae-495b-ad9d-84f99c693e46}

Command
- `cabinet.setCabinetState`

Actions
- Verify that text 'CB-COR-00010 Step #7' does NOT appear onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00003` [The cabinet device is host-disabled.]
- `E-CB-00013` [Disable text MUST NOT be displayed.]

8. **Verify that the EGM state is enabled.**
{08adee22-3060-422d-8fe8-63fe3120e038}

Command
- `cabinet.getCabinetStatus`

Errors
- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

Test Case: CB-COR-00011

This case tests that the disable text is displayed when the cabinet is locked.

Objective

Verify that the EGM displays the disable text from the `setCabinetLockOut` command.

Requirements Under Test

- 3.3.1
- 3.5.2
- 3.5.4
- 3.5.6
- 3.5.12
- 3.5.16
- 3.8.5
- 3.11.1
- 3.32.1
- 3.33.1
- 3.42.1
- 3.48.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Get the cabinet status.**
   {01472d77-49fd-4cec-88f0-62b4f2c1c68f}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

2. **Lock the cabinet.**
   {02c26938-4e36-40da-9f79-1c7312ad7289}

Command
- `cabinet.setCabinetLockOut`

Actions
- Verify that text 'CB-COR-00011 Step #2' appears onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_CBE009` [Host Locked Cabinet]
- `G2S_CBE211` [Host Action Locked EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00014` [Required text is not displayed.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Lock the cabinet with an empty disable text.**
{03378676-bb2c-412b-8259-0aeba1cbb5cd}

Command
- `cabinet.setCabinetLockOut`

Actions
- Verify that text 'CB-COR-00011 Step #2' does NOT appear onscreen.

Errors
- Send Request Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

4. **Lock the cabinet with a known lock text.**
{04d86a38-35b6-4357-966d-a554077819d0}

Command
- `cabinet.setCabinetLockOut`

Actions
- Verify that text 'CB-COR-00011 Step #4' appears onscreen.

Errors

- Send Request Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00014` [Required text is not displayed.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

5. **Verify that the lock text is not shown when the lock expires.**
{054e8848-c834-464d-a324-5b3c72e0732c}

Command

- `cabinet.setCabinetLockOut`

Actions

- Verify that text 'CB-COR-00011 Step #5' does NOT appear onscreen.

Events
(Time out: ${event.timeOut})

- `G2S_CBE010` [Host Unlocked Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

6. **Verify that the EGM state is "enabled."**
{06a81ded-4075-465d-9845-4baef44b9fc0}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]

Test Case: CB-COR-00012

This case tests that the EGM generates the proper event when the cash-out button is pressed.

Objective

Verify that the EGM sends a `G2S_CBE316 Cash-Out Button Pressed` event each time the cash-out button is pressed.

Requirements Under Test

- 3.71.1
- 3.71.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to press the cash-out button.**
   {01e64085-b1fe-40be-80be-1117dbabc36f}

   Action
   - Cash Out.

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE316` [Cash-Out Button Pressed]

   Errors
   - Event Checking Errors
   - DUT Error

2. **Instruct user to press the cash-out button.**
   {02bb09ef-5f3f-4654-80f4-3d79d03f8960}

   Action
   - Cash Out.

   Event

(Time out: ${event.timeOut})

- `G2S_CBE316` [Cash-Out Button Pressed]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00013

This case tests that standard configuration options are available in `optionConfig`, and that they are consistent with the `cabinetProfile`.

Objectives

- Verify that the `optionConfig` parameters for option ID `G2S_protocolOptions` are configured properly.
- Verify that the `optionConfig` parameters for option ID `G2S_cabinetOptions` are configured properly.
- Verify that the `optionConfig` parameters for option ID `G2S_cabinetLimits` are configured properly.
- Verify that the `cabinetProfile` attribute values match the `optionList` parameter values.

Requirements Under Test

- 3.10.30
- 3.96.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_cabinetOptions`.**
   {01fcc01d-b7cc-4a8d-0098-cbf0c314427c}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `G2S_protocolOptions` has the correct parameter configuration.**
   {026b7bf8-1869-4d9c-800c-e7413c4885ec}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]
- `E-OC-00006` [Option config parameter ${paramId} is not configured properly. Parameter MUST exist.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

3. **Assert that `G2S_cabinetOptions` has the correct parameter configuration.**
{03e454f3-10f8-479f-80ca-302edcf298aa}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

4. **Assert that `G2S_cabinetLimits` has the correct parameter configuration.**
{04387cd3-87bd-41c6-80ff-e2909b22eb81}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

5. **Assert that the `cabinetProfile` attribute values match the `optionConfig` parameter values.**
{059f75e6-0a27-4596-001d-8f2296cde4e1}

Command

- `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: CB-COR-00014

This case tests that standard configuration options in G2S 2.1 are available in `optionConfig`, and that they are consistent with the `cabinetProfile`.

Objectives

- Verify that the `optionConfig` parameters for option ID `G2S_configDelayOptions` are configured properly.
- Verify that the `optionConfig` parameter for option ID `G2S_cashOutOnDisableOptions` is configured properly.
- Verify that the `optionConfig` parameter for option ID `G2S_faultsSupportedOption` is configured properly.
- Verify that the `optionConfig` parameter for option ID `G2S_propertyIdOptions` is configured properly.
- Verify that the `cabinetProfile` attribute values match the `optionList` parameter values.

Requirements Under Test

- 3.10.30

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_cabinetOptions`.**
   {012a0ee8-23ab-4355-00e0-0cbfbaa95a7a}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `G2S_configDelayOptions` has the correct parameter configuration.**
{0204b03a-b25a-486d-0079-0ae679e1373d}

   Errors
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
   - `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]
   - `E-OC-00007` [Option item ${optionId} MUST exist.]

3. **Assert that `G2S_cashOutOnDisableOptions` has the correct parameter configuration.**
{03ed3328-98d9-45fd-80ed-6dd498cac479}

   Errors
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
   - `E-OC-00007` [Option item ${optionId} MUST exist.]

4. **Assert that `G2S_faultsSupportedOption` has the correct parameter configuration.**
{048d6900-aef4-44d3-0057-93fec55ea202}

   Errors
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
   - `E-OC-00007` [Option item ${optionId} MUST exist.]

5. **Assert that `G2S_propertyIdOptions` has the correct parameter configuration.**
{059422b2-c436-4458-80fe-aaa8e6dae358}

   Errors
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
   - `E-OC-00007` [Option item ${optionId} MUST exist.]

6. **Assert that the `cabinetProfile` attribute values match the `optionList` parameter values.**
{060871eb-ab47-4c98-803f-82a6828f8318}

   Command
   - `cabinet.getCabinetProfile`

   Errors
   - Send Request Errors
   - `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: CB-COR-00015

This case verifies `cabinet` device configuration.

Objectives

- Verify that the EGM only exposes one active cabinet device.
- Verify that the locale identifier in cabinetProfile is properly formed.
- Verify that the `cabinetStyle` enumeration complies with the unique 64 identifier data type.

Requirements Under Test

- 3.1.1
- 3.2.1
- 3.25.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_communications	owner
G2S_optionConfig	owner

Test Procedure

1. **Get a descriptor list and verify that there is only one active cabinet device.**
   {01517aa6-171b-4170-007e-568e74a38fd6}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CB-00022` [EGM MUST only expose one cabinet device.]

2. **Verify that the locale identifier in the `cabinetProfile` is properly formed.**
   {02dbe52d-ae28-4926-0097-b59d6e8a7477}

   Command
   - `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-CB-00023` [Locale identifiers MUST be constructed as language_country.]

3. **Get the `G2S_cabinetOptions` options and verify the cabinetStyle and localeId enumerations are valid.**
{03761115-cce7-46a0-8063-c7a3d4751de1}

Command

- `optionConfig.getOptionList`

Errors

- Send Request Errors
- `E-CB-00023` [Locale identifiers MUST be constructed as language_country.]
- `E-CB-00024` [Cabinet style MUST be a unique 64 identifier data type.]

Test Case: CB-COR-00016

This case verifies that `cabinetStatus.localeId` in G2S 2.1 is properly formed.

Objective

Verify that the locale identifier in cabinetStatus is properly formed.

Requirements Under Test

- 3.2.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Verify that the `localeId` in the cabinet status is properly formed.**
   {01c3bc9e-4051-4d6d-8001-f6ef204b46b6}

   #### Command

   - `cabinet.getCabinetStatus`

   #### Errors

   - Send Request Errors
   - `E-CB-00023` [Locale identifiers MUST be constructed as language_country.]

Test Case: CB-COR-00017

This case verifies that the host sends a valid getCabinetStatus command.

Objective

Verify the getCabinetStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getCabinetStatus command from the host.**
   {0127f629-b5aa-469d-8ad7-f58e929c4fd1}

   #### Command

   - `cabinet.getCabinetStatus`

   #### Actions

   - Instruct the user to 'Send getCabinetStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-COR-00018

This case verifies that the host sends a valid getCabinetProfile command.

Objective

Verify the getCabinetProfile command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getCabinetProfile command from the host.**
   {01f791d6-0056-41b7-84a5-f88f4707d7fa}

   #### Command

   - `cabinet.getCabinetProfile`

   #### Actions

   - Instruct the user to 'Send getCabinetProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-COR-00019

This case verifies that the host sends a valid setCabinetState command.

Objective

Verify the setCabinetState command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send setCabinetState command from the host.**
   {0187d29d-6b99-41d4-9cbd-7c93fe901d94}

   #### Command

   - `cabinet.setCabinetState`

   #### Actions

   - Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-COR-00020

This case verifies that the host sends a valid setDateTime command.

Objective

Verify the setDateTime command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send setDateTime command from the host.**
   {01ed3d1d-da67-4bdf-b477-955dbdb5edef}

   #### Command

   - `cabinet.setDateTime`

   #### Actions

   - Instruct the user to 'Send setDateTime to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-COR-00021

This case verifies that the host sends a valid getDateTime command.

Objective

Verify the getDateTime command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getDateTime command from the host.**
   {01f71dfc-820f-4038-8d3b-ef69a7e86947}

   #### Command

   - `cabinet.getDateTime`

   #### Actions

   - Instruct the user to 'Send getDateTime to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-COR-00022

This case verifies that the host does not process a cabinet response when sent as a request.

Objective

Verify that the host sends a G2S_APX008 Command Not Supported error when a cabinet response is sent as a request.

Requirements Under Test

- 1.4.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send a cabinetStatus as a request and verify host responds with a G2S_APX008 error.**
   {01dfab61-9840-4380-8c5c-a3ab26c79577}

   Command
   - `cabinet.cabinetStatus`

     Expect **G2S_APX008**

   Error
   - Send Request Errors

Test Case: CB-COR-00023

This case verifies that the host handles error responses to host cabinet requests.

Objectives

- Verify that the host is still working after responding with a cabinetProfile to a setCabinetState command.
- Verify that the host is still working after responding with a setCabinetState response to a setCabinetState command.
- Verify that the host is still working after responding with a cabinetStatus notification to a setCabinetState command.
- Verify that the host is still working after responding with an unknown error code to a setCabinetState command.

Requirements Under Test

- 1.4.5
- 1.4.6
- 1.4.7
- 1.42.3

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to respond with a cabinetProfile response to a setCabinetState request.**
   {01fa0551-c488-41c6-a042-9f7457fd8dcd}

2. **Instruct user to send setCabinetState command from the host.**
   {023a2c9f-6903-4c8f-a35a-35f6f574f377}

   Command
   - `cabinet.setCabinetState`

Actions

- Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId} (1 of 4).'.

Errors

- Expected Request Errors
- DUT Error

3. **Set the response manager to respond with a setCabinetState response to a setCabinetState request.**
{03bb3f8c-d37b-46d7-85cb-2e793112d7f0}

4. **Instruct user to send setCabinetState command from the host.**
{04c0c47c-15e2-4b69-97c2-d6aa3ebbefa3}

Command

- `cabinet.setCabinetState`

Actions

- Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId} (2 of 4).'.

Errors

- Expected Request Errors
- DUT Error

5. **Set the response manager to respond with a cabinetStatus notification to a setCabinetState request.**
{0587596d-582a-4093-9d6e-7b7e410a1ca6}

6. **Instruct user to send setCabinetState command from the host.**
{06ebd84b-c88d-44bf-9545-1d5ad39f2da2}

Command

- `cabinet.setCabinetState`

Actions

- Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId} (3 of 4).'.

Errors

- Expected Request Errors
- DUT Error

7. **Set the response manager to respond with 'CVT_ error' error code to a setCabinetState request.**
{077dfe46-e5b6-4d03-be08-6fb869d060bc}

8. **Instruct user to send setCabinetState command from the host.**
{088af4ff-58ca-4094-9ed5-6ccec4653034}

   Command
   - `cabinet.setCabinetState`

   Actions
   - Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId} (4 of 4).'.

   Errors
   - Expected Request Errors
   - DUT Error

9. **Reset the response manager.**
{09b94fd8-d120-4561-bea6-7f1d65d06d18}

10. **Send a keepAlive to the host to verify that it is still working.**
{104c86c0-3dfe-482b-b356-f1b4d874e34b}

    Command
    - `communications.keepAlive`

    Error
    - Send Request Errors

Test Case: CB-COR-00024

This case verifies that host the handles invalid XML in a cabinet response.

Objective

Verify that the host is still working after responding with invalid XML to a setCabinetState command.

Requirements Under Test

- 1.27.33

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to respond with an invalid cabinetStatus response to a setCabinetState request.**
   {01cd817f-c349-421d-b27c-f6374fab23c4}

2. **Instruct user to send setCabinetState command from the host.**
   {02499e34-ae0e-475d-b0e3-c3b4023ef53a}

   #### Command

   - `cabinet.setCabinetState`

   #### Actions

   - Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {033e141e-9649-4bc2-9cb0-c474f8d2e2d6}

4. **Send a keepAlive to the host to verify that it is still working.**
   {04add63f-c7c1-4f1c-9b21-f8cb6d4dc3f4}

Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: CB-COR-00025

This case verifies that the host processes delayed g2sAcks in cabinet commands.

Objective

Verify that the host is still working after delaying the g2sAck for a setCabinetState request.

Requirements Under Test

- 1.30.7

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to delay the g2sAck for setCommsState command.**
   {010ff20b-5005-4fa8-af06-712e7cdaedac}

2. **Instruct user to send setCabinetState command from the host.**
   {0233ce6d-9a41-4c51-a270-e9216572fc5b}

   Command
   - `cabinet.setCabinetState`

   Actions
   - Instruct the user to 'Send setCabinetState to device ${deviceUnderTest.deviceId}.'.

   Errors
   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {034e64c6-3f51-4cf0-9ce0-f84d5133aa00}

4. **Send a keepAlive to the host to verify that it is still working.**
   {045f9632-5f26-4de5-8c77-23ea4f4ee776}

   Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: CB-COR-00026

This case verifies that the host handles unknown cabinet commands.

Objectives

- Verify that the host sends G2S_APX008 Command Not Supported or G2S_APX015 Unknown Command Encountered error for an unknown cabinet command.
- Verify that if the host sends G2S_APX015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send an unknown cabinet command and verify G2S_APX008 or G2S_APX015 error.**
   {0188ac01-326d-4cbb-b725-bd976015ae8d}

   #### Command

   - `cabinet.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: CB-COR-00027

This test case verifies that the EGM sets the `cabinetStatus.egmState` to `G2S_egmDisabled` when the cabinet device is disabled.

Objective

Verify that the EGM is EGM disabled when the cabinet device is EGM disabled.

Requirements Under Test

- 3.3.1
- 3.3.16
- 3.3.18
- 3.26.2
- 3.27.1
- 3.40.1
- 3.42.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Determine the current cabinetStatus.deviceClass and cabinetStatus.deviceId values.**
   {010a5a6d-2380-4b75-a081-2ba359fc9996}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

2. **Instruct the user to open the cabinet door.**
   {02bea81f-ea08-4772-a4bb-bf831f91ee99}

   Action
   - Request CABINET door OPEN.

   Events

(Time out: ${event.timeOut})

- `G2S_CBE001` [EGM Disabled Cabinet]
- `G2S_CBE203` [Device Fault Disabled EGM]

Errors

- Event Checking Errors
- DUT Error

3. **Get the cabinet status and verify that the egmState is G2S_egmDisabled.**
{031d32ab-8e01-4f9e-b97b-b7d2b588c895}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

4. **Instruct the user to close the cabinet door.**
{04420b22-1b15-4226-a379-9ac624cb869c}

Action

- Request CABINET door CLOSE.

Events

(Time out: ${event.timeOut})

- `G2S_CBE002` [EGM Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00028

This case continuously tests that `cabinetStatus.enableMoneyOut` is set to true.

Objectives

- Continuously verify that cabinetStatus.enableMoneyOut is set to true.
- Continuously verify that all cabinet events that have a statusInfo element have cabinetStatus.enableMoneyOut is set to true.

Requirements Under Test

- 3.8.6

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Assert that cabinetStatus.enableMoneyOut is true.**
   {01b03fc4-fe8e-433b-9079-3625582e295a}

   Error
   - `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

2. **Assert that all cabinet events that have a statusInfo element has cabinetStatus.enableMoneyOut set to true.**
   {02fa9466-1216-47aa-bad5-ed49aca9cd0b}

   Error
   - `E-EH-00013` [Event ${eventCode} status attribute ${attributeName} of ${actual} is wrong it should be ${expected}.]

Test Case: CB-COR-00029

This test case verifies that if the EGM is required to cash-out the player when disabled, that the EGM reports the cash-out transactions.

Objective

Verify that the EGM reports cash-out transactions if the EGM is required to cash-out the player.

Requirements Under Test

- 3.8.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Test Procedure

1. **Get the note acceptor profile.**
   {01315ed8-eade-4227-897b-3f45aff7a661}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {02221f10-51f7-4071-8748-367d5c3202ac}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

3. **Instruct the user to insert a note.**
   {03e3fdc3-9bac-42c0-bb16-a209c4915515}

Action

- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

4. **Send a setCabinetState to disable the cabinet device, wait for potential cash out and then verify if the EGM was cashed out that the EGM meters have been updated.**
{0460229d-025a-481a-b38d-b118b6aca641}

Commands

- `cabinet.setCabinetState`
- `meters.getMeterInfo`

Actions

- Instruct the user to 'Did the EGM cash out when it was disabled? Press 'True' only after cash out is complete.'.

Events
(Time out: ${event.timeOut})

- `G2S_CBE003` [Host Disabled Cabinet]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors

- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00026` [Cash out device of ${device} is not a valid cash out device.]
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]
- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

5. **Host enable the cabinet device.**
{0520aeac-c776-4a8c-b91d-480dcdc19ace}

Command

- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})

- G2S_CBE004 [Host Enabled Cabinet]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

6. **If the EGM still has credits on the credit meter.**

    a. **Instruct user to cash out EGM**
    {06c47c2c-262e-4e91-a9a4-b7c5271a3d2f}

    Action

    - Cash Out.

    Error

    - DUT Error

Test Case: CB-COR-00030

This test case verifies that the EGM properly supports `g2s1` extension in the `cabinetStatus`.

Objective

Verify that the EGM reports `egmIdle` properly in the `cabinet.cabinetStatus`.

Requirements Under Test

- 3.9.15
- 3.65.1
- 3.66.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Test Procedure

1. **Get the note acceptor profile.**
   {01fce36e-bc95-4386-a152-c0cc1968e42e}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {02b30eed-c3a6-40a9-ae5a-1a77256c9621}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

3. **Get the cabinet profile and status.**
{03222be5-3fef-4d5c-b00a-9b0a5f936506}

Commands
- `cabinet.getCabinetProfile`
- `cabinet.getCabinetStatus`

Error
- Send Request Errors

4. **If EGM is not idle**

a. **Wait cabinetProfile.idlePeriodTime for the EGM to become idle.**
{0450e94a-38aa-40e6-a9cf-e35a65a46ba4}

Event
(Time out: ${event.timeOut})
- `G2S_CBE328` [EGM Idle]

Error
- Event Checking Errors

5. **Instruct the user to insert a note and verify G2S_CBE329 is generated.**
{0520d19f-20f9-456f-b55c-78cdd87076f7}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]
- `G2S_CBE329` [EGM Not Idle]

Errors
- Event Checking Errors
- DUT Error

6. **Disable the response to event reports and instruct the user to cash out the EGM.**
{06e19b5b-bf25-4a62-9fd2-9c267f717b4c}

Actions
- Cash Out.

Error
- DUT Error

7. **Enable event reports and wait for cabinetProfile.idleTimePeriod to expire and verify G2S_CBE328 is generated.**
{076c7bac-2463-4299-a64f-6c5fdfbb870f}

Command

- `cabinet.getCabinetStatus`

Event

(Time out: ${event.timeOut})

- `G2S_CBE328` [EGM Idle]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: CB-COR-00031

This test case verifies that the EGM performs a cash-out when it is forced to cash-out the player when disabled.

Objective

Verify that the EGM performs a cash-out when `cabinetProfile.cashOutOnDisable` is `G2S_forceCashOut`.

Requirements Under Test

- 3.10.14
- 3.10.20

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Required Configurations

Option	Value
gamePlayProfile.cashOutOnDisable	G2S_forceCashOut

Test Procedure

1. **Get the note acceptor profile.**
   {012aca47-5e4a-446e-9bfc-c19af4d43599}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {020c96ff-0164-4538-81e8-cf3e7ff7aba5}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

3. **Instruct the user to insert a note.**
{03a2f169-39fb-47f8-a1a4-05cb68d4c9ba}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

4. **Send a setCabinetState to disable the cabinet device, wait for potential cash out and then get the device meters to verify the player was cashed out.**
{04516894-1a37-4af3-8fa4-071b0a51dce6}

Commands
- `cabinet.setCabinetState`
- `meters.getMeterInfo`

Actions
- Instruct the user to 'Did the EGM cash out when it was disabled? Press 'True' only after cash out is complete.'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE003` [Host Disabled Cabinet]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00027` [The EGM MUST cash out the player when cabinetProfile.cashOutOnDisable is G2S_forceCashOut.]

- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

5. **Host enable the cabinet device.**
{05b59af8-7bbe-40e4-927c-17917e817ac9}

Command
- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})
- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

6. **If the EGM still has credits on the credit meter.**

a. **Instruct user to cash out EGM**
{06ea0b35-6356-453c-9a43-4a4a9bb98322}

Action
- Cash Out.

Error
- DUT Error

Test Case: CB-COR-00032

This test case verifies that the EGM allows a player to cash-out when it is allowed to cash-out the player when disabled.

Objective

Verify that the EGM performs a cash-out when `cabinetProfile.cashOutOnDisable` is `G2S_allowCashOut`.

Requirements Under Test

- 3.10.14
- 3.10.20

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Required Configurations

Option	Value
gamePlayProfile.cashOutOnDisable	G2S_allowCashOut

Test Procedure

1. **Get the note acceptor profile.**
   {017d423d-3f80-4b23-a6b5-7fdf33d90006}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {021f4783-0040-4e81-b9c3-0bffce51a92c}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

3. **Instruct the user to insert a note.**
{03255972-a39b-466d-a7f9-7a2a546d192b}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

4. **Send a setCabinetState to disable the cabinet device, wait for potential cash out and then get the device meters to verify the player was not cashed out.**
{04afdcd2-592b-4333-aaf9-50b5ca2d9da1}

Commands
- `cabinet.setCabinetState`
- `meters.getMeterInfo`

Actions
- Instruct the user to 'Did the EGM cash out when it was disabled? Press 'True' only after cash out is complete.'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE003` [Host Disabled Cabinet]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00028` [The EGM MUST NOT cash out the player when the EGM is disabled and cabinetProfile.cashOutOnDisable is ${cashOutOnDisableType}.]

- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

5. **If the EGM still has credits on the credit meter.**

    a. **Instruct user to cash out EGM**
    {05c3e5ad-112f-415a-b58e-30c6d59061cf}

    Command
    - `meters.getMeterInfo`

    Actions
    - Cash Out.
    - Instruct the user to 'Did the EGM allow the cash out? Press 'True' only after cash out is complete.'.

    Errors
    - Send Request Errors
    - DUT Error
    - `E-CB-00029` [The EGM MUST allow the player to cash out when the EGM is disabled and cabinetProfile.cashOutOnDisable is G2S_allowCashOut.]
    - `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

6. **Host enable the cabinet device.**
{0626c176-6593-465a-a84c-a73e4472d0eb}

    Command
    - `cabinet.setCabinetState`

    Events
    (Time out: ${event.timeOut})
    - `G2S_CBE004` [Host Enabled Cabinet]
    - `G2S_CBE205` [EGM Enabled and Playable]

    Errors
    - Send Request Errors
    - Event Checking Errors

Test Case: CB-COR-00033

This test case verifies that the EGM does not performs a cash-out when it is disabled.

Objective

Verify that the EGM does not perform a cash-out when `cabinetProfile.cashOutOnDisable` is `G2S_preventCashOut`.

Requirements Under Test

- 3.10.14
- 3.10.20

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Required Configurations

Option	Value
gamePlayProfile.cashOutOnDisable	G2S_preventCashOut

Test Procedure

1. **Get the note acceptor profile.**
   {013edee3-9179-43b7-bb24-c847357af8d5}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {02a2a0c2-896f-402c-9824-e1fc64438e9b}

Command
- meters.getMeterInfo

Error
- Send Request Errors

3. **Instruct the user to insert a note.**
{0361718e-c236-4664-aacd-0ef1fa6e980b}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- G2S_NAE116 [Note In Escrow]
- G2S_NAE114 [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

4. **Send a setCabinetState to disable the cabinet device, wait for potential cash out and then get the device meters to verify the player was not cashed out.**
{044ed8b7-3bd6-4ffd-8d81-56fef4b9d75a}

Commands
- cabinet.setCabinetState
- meters.getMeterInfo

Actions
- Instruct the user to 'Did the EGM cash out when it was disabled? Press 'True' only after cash out is complete.'.

Events
(Time out: ${event.timeOut})
- G2S_CBE003 [Host Disabled Cabinet]
- G2S_CBE204 [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- E-CB-00028 [The EGM MUST NOT cash out the player when the EGM is disabled and cabinetProfile.cashOutOnDisable is ${cashOutOnDisableType}.]

- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

5. **Host enable the cabinet device.**
{05dd06c2-bd78-498f-ad42-213ff0b66728}

Command
- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})
- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

6. **If the EGM still has credits on the credit meter.**

a. **Instruct user to cash out EGM**
{06ee4709-d5c6-4643-8154-98b459cf5949}

Action
- Cash Out.

Error
- DUT Error

Test Case: CB-COR-00034

This test case verifies that the EGM properly sets the `cabinetStatus.hostEnabled` attribute after an EGM restart.

Objectives

- Verify that the EGM has `cabinetProfile.restartStatusMode` to `TRUE` for extension `g2sA`.
- Verify that the EGM sets the `cabinetStatus.hostEnabled` based on `cabinetProfile.restartStatus` and `cabinetProfile.restartStatusMode` attributes.

Requirements Under Test

- 1.36.16
- 1.36.17
- 3.10.20

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_communications	owner

Test Procedure

1. **Get the cabinet profile to determine the expected hostEnable value after a restart.** {0184ec48-31f3-4045-811a-33dca7c112e4}

   #### Command

   - `cabinet.getCabinetProfile`

   #### Errors

   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **If the cabinet device should be disabled before the restart.**

    a. **Host disable the cabinet device.**
    {0264db99-7e3c-45e4-ba04-59f924193b0e}

    Command
    - `cabinet.setCabinetState`

    Event
    (Time out: ${event.timeOut})
    - `G2S_CBE003` [Host Disabled Cabinet]

    Errors
    - Send Request Errors
    - Event Checking Errors
    - `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Stop responding to eventReports and instruct the user to power cycle the EGM.**
{03a6763e-9a8a-4992-850e-d1bc25fd6397}

    Actions
    - Instruct the user to 'Please power cycle the EGM.'.

    Errors
    - DUT Error
    - `E-CM-00057` [The user failed to power cycle the EGM.]

4. **Start responding to eventReports and bring the communication channel to the SYNC state.**
{0465e8d2-4373-4632-92a3-b36b53384224}

    Commands
    - `communications.commsOnLine`
    - `communications.commsDisabled`

    Error
    - Expected Request Errors

5. **Get the cabinet status and verify the cabinetStatus.hostEnabled attribute is set correctly.**
{05988d40-818f-4b29-b0cc-3620bf7a7826}

    Command
    - `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

6. **Enable the communications channel and verify expected events.**
{060466a1-2b66-4907-8559-736c6cfe538b}

Command

- `communications.setCommsState`

Event

(Time out: ${event.timeOut})

- `G2S_CBE325` [EGM Power Up]

Errors

- Send Request Errors
- Event Checking Errors
- Event Not Expected Error
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]

Test Case: CB-COR-00035

This test case verifies that the EGM reports general faults properly.

Objectives

- Verify that the EGM has `cabinetProfile.faultsSupported` set to `G2S_true` for extension `g2sA`.
- Verify that the EGM sets the `cabinetStatus.generalFault` based on `cabinetProfile.faultsSupported` attribute.

Requirements Under Test

- 3.9.1
- 3.9.2
- 3.9.9
- 3.10.20
- 3.57.1
- 3.57.2
- 3.61.1
- 3.61.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Get the cabinet profile to determine the expected cabinetStatus.generalFault attribute value.**
   {0117c6da-fa0a-4931-8f93-a5d159716d37}

   #### Command
   - `cabinet.getCabinetProfile`

   #### Errors
   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **Instruct user to cause a general fault and verify G2S_CBE309 event is generated. If the EGM was disabled then verify G2S_CBE001 and G2S_CBE203 events are generated.**
{02b1eb54-2295-4fdb-a099-c3a1014328e3}

   Command
   - `cabinet.getCabinetStatus`

   Actions
   - Instruct the user to cause a device fault to generate 'G2S_CBE309'.

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE309` [General Cabinet Tilt]

   Errors
   - Send Request Errors
   - Event Checking Errors
   - Event Not Expected Error
   - DUT Error
   - `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]
   - `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]
   - `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

3. **Instruct the user to clear the general fault and verify G2S_CBE313 event is generated. If the EGM was disabled then verify that G2S_CBE002 and G2S_CBE205 events are generated.**
{03aeb5de-8bb3-4872-b21f-5528b3b703e9}

   Action
   - Instruct the user to clear a device fault to generate 'G2S_CBE313'.

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE313` [Cabinet Tilt Cleared]

   Errors
   - Event Checking Errors
   - Event Not Expected Error
   - DUT Error
   - `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]

Test Case: CB-COR-00036

This test case verifies that the EGM reports reel tilts properly.

Objectives

- Verify that the EGM has `cabinetProfile.faultsSupported` set to `G2S_true` for extension `g2sA`.
- Verify that the EGM sets the `cabinetStatus` attributes `reelTilt` and `reelTilted` based on `cabinetProfile.faultsSupported` attribute.

Requirements Under Test

- 3.9.1
- 3.9.3
- 3.9.9
- 3.10.20
- 3.57.1
- 3.57.2
- 3.61.1
- 3.61.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Get the cabinet profile to determine the expected cabinetStatus.generalFault attribute value.**
   {015be9e8-c6a8-4c42-ae3b-96da7b550814}

   Command
   - `cabinet.getCabinetProfile`

   Errors
   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **Instruct user to cause a reel tilt fault and verify the expected events are generated.**
{02a16c94-68d5-4f0e-b0f7-9d5e9c71585e}

Command
- `cabinet.getCabinetStatus`

Actions
- Instruct the user to cause a device fault to generate 'G2S_CBE309'.

Event
(Time out: ${event.timeOut})
- `G2S_CBE309` [General Cabinet Tilt]

Errors
- Send Request Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]
- `E-CB-00031` [The reelsTilted attribute of ${actual} in the cabinet status is not properly formatted. Only vertical bars and decimal values are allowed.]
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]
- `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

3. **Instruct the user to clear the general fault and verify expected events are generated.**
{0300735b-29a9-4247-be08-a6574a16abcc}

Action
- Instruct the user to clear a device fault to generate 'G2S_CBE313'.

Event
(Time out: ${event.timeOut})
- `G2S_CBE313` [Cabinet Tilt Cleared]

Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]

Test Case: CB-COR-00037

This test case verifies that the EGM sets the `commsOnLine` attribute properly after non-volatile storage is cleared.

Objectives

- Verify that the EGM sets `deviceReset`, `deviceChanged`, `subscriptionLost`, and `metersReset` to `TRUE` after non-volatile storage is cleared.
- Verify sends `G2S_CBE322 Non-Volatile Storage Cleared` event when the non-volatile storage is cleared.
- Verify that `G2S_CBE322 Non-Volatile Storage Cleared` has no affected data.

Requirements Under Test

- 3.77.1
- 3.77.2
- 3.77.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_communications	owner

Test Procedure

1. **Stop responding to eventReports.**
   {0154c101-6120-48fd-ac07-938d36d093b3}
2. **Instruct user to clear the non-volatile storage and wait for the EGM to reconnect.**
   {02d2cd33-5cd1-41d3-a861-0862d1ec09ea}

   #### Command

   - `communications.commsOnLine`

   #### Actions

   - Instruct the user to 'Please clear the non-volatile storage.'.

Errors

- Expected Request Errors
- DUT Error
- `E-CB-00032` [The user failed to clear the non-volatile storage.]
- `E-CM-00071` [CommsOnLine MUST have ${attribute} set to true when the EGM has its non-volatile storage cleared.]

3. **Start responding to eventReports.**
{036a3205-19fb-4727-92a3-95a0ece26a1e}

4. **Start the communications channel and verify that G2S_CBE322 is generated.**
{048f425a-9599-47e9-b05d-c6f43c3ad798}

Command

- `communications.setCommsState`

Event

(Time out: ${event.timeOut})

- `G2S_CBE322` [NVM Cleared]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: CB-COR-00038

This case verifies that the EGM handles forced subscriptions properly in the cabinet class.

Objectives

- Verify that the EGM treats forced subscriptions in the cabinet class as if set by the host.
- Verify that the EGM "ORs" host subscriptions and forced subscriptions in the cabinet class.

Requirements Under Test

- 1.23.25
- 1.23.26

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner
G2S_eventHandler	owner

Test Procedure

1. **Clear the event subscription for G2S_CBE307 Cabinet Door Open and G2S_CBE316 Cash-Out Button Pressed.**
   {01003f38-2402-44eb-bd1c-7cb6c41ed001}

   Command
   - `eventHandler.clearEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Error
   - Event Checking Errors

2. **Set a host subscription for G2S_CBE307 to send device meters.**
   {024fc42c-5e21-46da-93ed-e53b96b6b964}

Command
- `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

3. **Set a forced subscription for G2S_CBE307 to send device status and G2S_CBE316.**
{03294317-6de5-42ac-a88e-306700bc2de1}

Command
- `optionConfig.setOptionChange`

Event
(Time out: ${event.timeOut})
- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

4. **Instruct the user to press the cash out button and verify G2S_CBE316 event is generated.**
{047cb7ab-70c6-425d-a556-ab4e483b3dc0}

Action
- Instruct the user to 'Please press the cash out button.'.

Event
(Time out: ${event.timeOut})
- `G2S_CBE316` [Cash-Out Button Pressed]

Errors
- Event Checking Errors
- DUT Error

5. **Instruct user to open the cabinet door and verify that G2S_CBE307 has device status and device meters.**
{05606f91-ad36-441f-a890-30f9e2ffb7ec}

Action

- Request CABINET door OPEN.

Event
(Time out: ${event.timeOut})

- `G2S_CBE307` [Cabinet Door Open]

Errors

- Event Checking Errors
- DUT Error

6. **Instruct the user to close the cabinet door.**
{06939800-9900-49f6-bdcd-0fbb6ae337d1}

Action

- Request CABINET door CLOSE.

Event
(Time out: ${event.timeOut})

- `G2S_CBE308` [Cabinet Door Closed]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00039

This case verifies that the EGM unlocks a cabinet device when restarted.

Objective

Verify that the EGM unlocks the cabinet device after a restart.

Requirements Under Test

- 1.36.15

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_communications	owner

Test Procedure

1. **Host lock the cabinet device.**
   {01aa335e-4347-495b-9523-483f1b3e58bd}

   #### Command
   - `cabinet.setCabinetLockOut`

   #### Events
   (Time out: ${event.timeOut})
   - `G2S_CBE009` [Host Locked Cabinet]
   - `G2S_CBE211` [Host Action Locked EGM]

   #### Errors
   - Send Request Errors
   - Event Checking Errors
   - `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

2. **Stop responding to eventReports and instruct the user to power cycle the EGM.**
   {0228294f-20f3-4cad-b28d-1ea3ffa5ac02}

   #### Actions
   - Instruct the user to 'Please power cycle the EGM.'.

Errors

- DUT Error
- `E-CM-00057` [The user failed to power cycle the EGM.]

3. **Start responding to eventReports and bring the communication channel to the SYNC state.**
{03f854ac-1192-4eac-ba89-058f8a689dcd}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Error

- Expected Request Errors

4. **Get the cabinet status and verify the cabinetStatus.hostLocked attribute is false.**
{046fc8f9-db9c-4a91-a105-d91c0ecfe3f1}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

5. **Enable the communications channel and verify expected events.**
{051482d3-3af2-4135-af2d-a5939a444b0f}

Command

- `communications.setCommsState`

Events

(Time out: ${event.timeOut})

- `G2S_CBE010` [Host Unlocked Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: CB-COR-00040

This case verifies that the EGM releases the EGM from transport-disabled state when the transport is available.

Objective

Verify that the EGM releases the `egmState` when the transport is available.

Requirements Under Test

- 2.58.4

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_communications	owner

Test Procedure

1. **Determine the timeToLive and noResponseTimer for the communications device.**
   {017c20a5-2929-4731-80c7-d13c1cfd8938}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **If the communications device is requiredForPlay.**

   a. **Stop the CVT web server and wait for the EGM to transition to the closed state.**
   {02629314-4af4-41f6-a9d8-da1aa7e70636}

   b. **Restart the CVT web server.**
   {038a766d-ea91-49ee-b240-941920eefad8}

   c. **Wait for the EGM to reconnect.**
   {04dd6c5d-31e1-4167-a9c7-442a226d81d7}

   Commands
   - `communications.commsOnLine`

- `communications.commsDisabled`

Error

- Expected Request Errors

d. **Get the cabinet status and verify the cabinetStatus.egmState attribute is not G2S_transportDisabled.**
{054ba742-9cf7-497c-baed-4a543154baff}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: CB-COR-00041

This case verifies that the EGM does not initiate any games, accept any money-in or dispense any money-out when host locked.

Objectives

- Verify that the EGM does not initiate any games when the cabinet device is host locked.
- Verify that the EGM does not accept any money-in when the cabinet device is host locked.
- Verify that the EGM does not dispense any money-out when the cabinet device is host locked.

Requirements Under Test

- 1.36.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_meters	owner
G2S_noteAcceptor	guest or owner

Test Procedure

1. **Get the note acceptor profile.**
   {01a26dfe-b74f-4bfd-a015-a0dd7d0412e6}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Get device and currency meters.**
   {0234f00c-9b99-48b0-93d0-63bfb1c17fa6}

   Command
   - `meters.getMeterInfo`

   Error

- Send Request Errors

3. **Instruct the user to insert a note.**
{03236b1a-2b3b-48bb-8bac-0872646581ea}

Action

- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

4. **Host lock the cabinet device.**
{0449337d-6786-4e15-9265-9dc17f9a33d0}

Command

- `cabinet.setCabinetLockOut`

Events
(Time out: ${event.timeOut})

- `G2S_CBE009` [Host Locked Cabinet]
- `G2S_CBE211` [Host Action Locked EGM]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

5. **Instruct user to verify that game play has been disabled.**
{058f7b1c-0c6e-4145-8705-4b212ad4d9b4}

Action

- Instruct the user to 'Please verify that game play has been disabled. Select 'False' if game is playable.'.

Errors

- DUT Error
- `E-CB-00033` [While host locked, the EGM must not allow ${activity}.]

6. **Instruct user to verify that the EGM will not accept money-in.**
{060832bd-e378-4005-8cb9-c7b168a794c1}

Action

- Instruct the user to 'Please verify that money-in has been disabled. Select 'False' if money-in is enabled.'.

Errors

- DUT Error
- `E-CB-00033` [While host locked, the EGM must not allow ${activity}.]

7. **Instruct user to verify that the EGM will not dispense money-out.**
{071a0a06-5365-41ce-b123-e3e2026e1f6f}

Action

- Instruct the user to 'Please verify that money-out has been disabled. Select 'False' if money-out is enabled.'.

Errors

- DUT Error
- `E-CB-00033` [While host locked, the EGM must not allow ${activity}.]

8. **Unlock the cabinet device.**
{08785969-0576-4e53-bde6-44bf2e82e610}

Command

- `communications.setCabinetLockOut`

Events

(Time out: ${event.timeOut})

- `G2S_CBE010` [Host Unlocked Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: CB-COR-00042

This case verifies that the EGM updates the `egmState`, `deviceClass` and `deviceId` attributes to reflect the current state when the cabinet device is locked or disabled.

Objectives

- Verify that the EGM sets `egmState`, `deviceClass` and `deviceId` to the cabinet device when the cabinet is host disabled.
- Verify that the EGM sets `egmState`, `deviceClass` and `deviceId` to the cabinet device when the cabinet is host locked.
- Verify that the EGM maintains `deviceClass` and `deviceId` attributes when the operator menu is entered.
- Verify that the EGM sets `egmState` to `G2S_operatorMode` when the operator menu is entered.
- Verify that the EGM maintains `deviceClass` and `deviceId` attributes when the operator menu is exited.
- Verify that the EGM sets `egmState` to `G2S_hostDisabled` or `G2S_hostLocked` when the operator menu is exited.
- Verify that the EGM resets `deviceClass` and `deviceId` to the original device when the cabinet is not locked or disabled.

Requirements Under Test

- 3.3.10
- 3.3.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Get the cabinet status to determine the current value for deviceClass and deviceId attributes.**

{0155de86-f7a1-4cd7-b192-b8f215caf40c}

Command
- `cabinet.getCabinetStatus`

Error
- Send Request Errors

2. **Host disable the cabinet device and verify egmState, deviceClass and deviceId attributes are correct in the cabinetStatus.**
{0259cf3f-f780-4dc6-8746-46b315053638}

Command
- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})
- `G2S_CBE003` [Host Disabled Cabinet]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Instruct user to enter the operator menu.**
{03eaf4db-a3bf-43d8-a2db-cfbc857a4914}

Action
- Instruct the user to 'Please enter the operator menu (1 of 2).'.

Event
(Time out: ${event.timeOut})
- `G2S_CBE206` [Operator Menu Activated]

Errors
- Event Checking Errors
- DUT Error

4. **Verify cabinetStatus.egmState attribute is G2S_operatorMode, and that deviceClass and deviceId attributes are correct in the cabinetStatus.**
{04761615-14f2-430e-96cb-46b3d2550dc7}

Command
- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

5. **Instruct user to exit the operator menu.**
{05cf9b53-c992-404d-8e45-6ff6f576f3e8}

Action

- Instruct the user to 'Please exit the operator menu (1 of 2).'.

Event
(Time out: ${event.timeOut})

- `G2S_CBE204` [Host Command Disabled EGM]

Errors

- Event Checking Errors
- DUT Error

6. **Verify cabinetStatus.egmState attribute is G2S_hostDisabled, and that deviceClass and deviceId attributes are correct in the cabinetStatus.**
{06e67c98-d563-4764-af3a-764652cf700d}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

7. **Host enable the cabinet device and verify egmState, deviceClass and deviceId attributes are correct in the cabinetStatus.**
{073a1f31-a6be-411e-8d4f-8e91148e6e79}

Command

- `cabinet.setCabinetState`

Events
(Time out: ${event.timeOut})

- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

8. **Host lock the cabinet device and verify egmState, deviceClass and deviceId attributes are correct in the cabinetStatus.**
{08157c75-c5a1-40a3-902c-931c72500a08}

Command
- cabinet.setCabinetLockOut

Events
(Time out: ${event.timeOut})
- G2S_CBE009 [Host Locked Cabinet]
- G2S_CBE211 [Host Action Locked EGM]

Errors
- Send Request Errors
- Event Checking Errors
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

9. **Instruct user to enter the operator menu.**
{09a3b4a6-5468-4004-baca-1bb25ca63cd3}

Action
- Instruct the user to 'Please enter the operator menu (2 of 2).'.

Event
(Time out: ${event.timeOut})
- G2S_CBE206 [Operator Menu Activated]

Errors
- Event Checking Errors
- DUT Error

10. **Verify cabinetStatus.egmState attribute is G2S_operatorMode, and that deviceClass and deviceId attributes are correct in the cabinetStatus.**
{108f4693-f46b-4324-bdf5-b060d79ed146}

Command
- cabinet.getCabinetStatus

Errors

- Send Request Errors
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

11. **Instruct user to exit the operator menu.**
{1169c4ee-2674-424e-8ae0-3ee12e3b22d5}

Action

- Instruct the user to 'Please exit the operator menu (2 of 2).'.

Event
(Time out: ${event.timeOut})

- G2S_CBE211 [Host Action Locked EGM]

Errors

- Event Checking Errors
- DUT Error

12. **Verify cabinetStatus.egmState attribute is G2S_hostLocked, and that deviceClass and deviceId attributes are correct in the cabinetStatus.**
{12dc2895-d007-49bc-a980-8299632841ec}

Command

- cabinet.getCabinetStatus

Errors

- Send Request Errors
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

13. **Host unlock the cabinet device and verify egmState, deviceClass and deviceId attributes are correct in the cabinetStatus.**
{13849fe5-cada-4071-b016-af8980dc38c4}

Commands

- cabinet.setCabinetState
- cabinet.setCabinetLockOut

Events
(Time out: ${event.timeOut})

- G2S_CBE010 [Host Unlocked Cabinet]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: CB-COR-00043

This case verifies that the EGM does not accept money in when money-in is disabled.

Objective

Verify that the EGM does not accept money-in when the `cabinetStatus.enableMoneyIn` attribute is set to false.

Requirements Under Test

- 3.3.12

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Disable money-in and verify G2S_CBE103 Host Disabled Money In event is generated.**
   {01ad752d-c782-4733-b068-275a60c6fadf}

   #### Command
   - `cabinet.setCabinetState`

   #### Event
   (Time out: ${event.timeOut})
   - `G2S_CBE103` [Host Disabled Money In]

   #### Errors
   - Send Request Errors
   - Event Checking Errors
   - `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

2. **Instruct user to verify that the EGM will not accept money-in.**
   {028cfec5-a016-4cb7-b25a-8c2ea34061ad}

   #### Action
   - Instruct the user to 'Please verify that money-in has been disabled. Select 'False' if money-in is enabled.'.

Errors

- DUT Error
- `E-CB-00033` [While host locked, the EGM must not allow ${activity}.]

3. **Enable money-in and verify G2S_CBE104 Host Enabled Money In event is generated.**
{039790aa-077f-4d7f-8dcc-d0e277cdd8e4}

Command

- `cabinet.setCabinetState`

Event

(Time out: ${event.timeOut})

- `G2S_CBE104` [Host Enabled Money In]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: CB-COR-00044

This case verifies that the EGM displays and removes text messages for the cabinet device.

Objectives

- Verify that the EGM displays text when it becomes eligible to be displayed.
- Verify that the EGM removes text when it is no longer eligible to be displayed.

Requirements Under Test

- 3.5.14

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Disable the cabinet device with specific disable text.**
   {0185f393-df0a-45a2-97b5-560b72d818b6}

   Command
   - `cabinet.setCabinetState`

   Events
   (Time out: ${event.timeOut})
   - `G2S_CBE003` [Host Disabled Cabinet]
   - `G2S_CBE204` [Host Command Disabled EGM]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Instruct user to verify disable text is displayed.**
   {0299d76a-3d0e-459e-ad16-220cb92edee4}

   Action
   - Verify that text 'CB-COR-00044 Step #1' appears onscreen.

Errors

- DUT Error
- `E-CB-00014` [Required text is not displayed.]

3. **Enable the cabinet device with specific disable text.**
{03365c2e-fe17-40f1-898d-811181cee532}

Command

- `cabinet.setCabinetState`

Events

(Time out: ${event.timeOut})

- `G2S_CBE004` [Host Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

4. **Instruct user to verify disable text is not displayed.**
{040b0c3f-716d-43c4-aea8-98b8d8d05339}

Action

- Verify that text 'CB-COR-00044 Step #3' does NOT appear onscreen.

Errors

- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]

Test Case: CB-COR-00045

This test case verifies that the EGM reports video faults properly.

Objectives

- Verify that the EGM has `cabinetProfile.faultsSupported` attribute set to `G2S_true` for extension `g2sA`.
- Verify that G2S_CBE310 Video Display Error is generated for a video display fault.
- Verify that the EGM sets `cabinetStatus.videoDisplayFault` attribute based on `cabinetProfile.faultsSupported` attribute.

Requirements Under Test

- 3.9.1
- 3.9.6
- 3.9.9
- 3.58.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Get the cabinet profile to determine the expected cabinetStatus.videoDisplayFault attribute value.**
   {01edf498-6294-4514-b178-ef2db5f3fb00}

   Command
   - `cabinet.getCabinetProfile`

   Errors
   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **Instruct user to cause a video display fault and verify G2S_CBE310 event is generated.**
   {0247bf65-a89e-4439-856b-e4b6b4913838}

Command

- `cabinet.getCabinetStatus`

Actions

- Instruct the user to cause a device fault to generate 'G2S_CBE310'.

Events

(Time out: ${event.timeOut})

- `G2S_CBE310` [Video Display Error]
- `G2S_CBE001` [EGM Disabled Cabinet]
- `G2S_CBE203` [Device Fault Disabled EGM]

Errors

- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]
- `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

3. **Instruct the user to clear the video display fault and verify G2S_CBE313 event is generated.**
{03c1b02b-12da-4c32-b046-d1307221e88a}

Action

- Instruct the user to clear a device fault to generate 'G2S_CBE313'.

Events

(Time out: ${event.timeOut})

- `G2S_CBE313` [Cabinet Tilt Cleared]
- `G2S_CBE002` [EGM Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00046

This test case verifies that the EGM reports non-volatile storage faults properly.

Objectives

- Verify that the EGM has `cabinetProfile.faultsSupported` set to `G2S_true` for extension `g2sA`.
- Verify that G2S_CBE311 Non-Volatile Storage Fault is generated for a non-volatile storage fault.
- Verify that the EGM sets `cabinetStatus.nvStorageFault` attribute based on `cabinetProfile.faultsSupported` attribute.

Requirements Under Test

- 3.9.1
- 3.9.7
- 3.9.9
- 3.59.1
- 3.59.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Get the cabinet profile to determine the expected cabinetStatus.nvStorageFault attribute value.**
   {0122dd0b-00f6-4814-8153-9cc3deb0ec1a}

   Command
   - `cabinet.getCabinetProfile`

   Errors
   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **Instruct user to cause a non-volatile storage fault and verify G2S_CBE311 event is generated.**
{02065fc5-6c32-49fd-b022-598a73691fa5}

Command
- `cabinet.getCabinetStatus`

Actions
- Instruct the user to cause a device fault to generate 'G2S_CBE311'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE311` [NVM Failure]
- `G2S_CBE001` [EGM Disabled Cabinet]
- `G2S_CBE203` [Device Fault Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]
- `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

3. **Instruct the user to clear the non-volatile fault and verify G2S_CBE313 event is generated.**
{03f22cad-e493-454b-b6cb-76af364ac353}

Action
- Instruct the user to clear a device fault to generate 'G2S_CBE313'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE313` [Cabinet Tilt Cleared]
- `G2S_CBE002` [EGM Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Event Checking Errors
- DUT Error

Test Case: CB-COR-00047

This test case verifies that the EGM reports general memory faults properly.

Objectives

- Verify that the EGM has `cabinetProfile.faultsSupported` set to `G2S_true` for extension `g2sA`.
- Verify that G2S_CBE312 General Memory Fault is generated for a general memory fault.
- Verify that the EGM sets `cabinetStatus.generalMemoryFault` attribute based on `cabinetProfile.faultsSupported` attribute.

Requirements Under Test

- 3.9.1
- 3.9.8
- 3.9.9
- 3.60.1
- 3.60.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Get the cabinet profile to determine the expected cabinetStatus.generalMemoryFault attribute value.**
   {0150b3c6-da4c-4854-b7aa-3b35c99fced8}

   Command
   - `cabinet.getCabinetProfile`

   Errors
   - Send Request Errors
   - `E-CB-00030` [The EGM MUST NOT have cabinet profile attribute ${attribute} set to ${value} if the EGM supports ${extension} extension.]

2. **Instruct user to cause a general memory fault and verify G2S_CBE312 event is generated.**
{02dda636-0189-408b-9e0d-d9450090e62f}

Command
- `cabinet.getCabinetStatus`

Actions
- Instruct the user to cause a device fault to generate 'G2S_CBE312'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE312` [General Memory Failure]
- `G2S_CBE001` [EGM Disabled Cabinet]
- `G2S_CBE203` [Device Fault Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]
- `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

3. **Instruct the user to clear the general memory fault and verify G2S_CBE313 event is generated.**
{03fee79d-a408-4294-9b32-baf7409ad409}

Action
- Instruct the user to clear a device fault to generate 'G2S_CBE313'.

Events
(Time out: ${event.timeOut})
- `G2S_CBE313` [Cabinet Tilt Cleared]
- `G2S_CBE002` [EGM Enabled Cabinet]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Event Checking Errors
- DUT Error

Test Case: CB-COR-00048

This test case verifies that the EGM does not exceed the `maxCreditMeter` on the credit meter.

Objective

Verify that the EGM does not allow the credit meter to exceeds `maxCreditMeter`.

Requirements Under Test

- 3.10.23

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner

Test Procedure

1. **Get the cabinet profile to determine the cabinetProfile.maxCreditMeter attribute value.**
   {011cff52-29a9-428e-8d00-17b52d4f0f89}

   Command
   - `cabinet.getCabinetProfile`

   Error
   - Send Request Errors

2. **Instruct user to fund the EGM to the max credit meter.**
   {023a81a4-6b0b-4b7c-b37e-a49bab9f963b}

   Action
   - Instruct the user to 'Please fund the credit meter to the max value of ${maxCreditValue.formatAsDollars()}.'.

   Errors
   - DUT Error
   - `E-XX-00022` [The user failed to fund the EGM.]

3. **Verify that the credit meter is equal to the max credit meter value.**
   {0373d5f8-5662-485e-9a59-8774a44bfc74}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-CB-00034` [The user failed to credit meter to the max credit meter value of ${maxCredit}.]

4. **Instruct user to verify that the EGM will not accept any additional funds.**
{047aa787-a476-4226-9dfa-3fc3bfdf689b}

Action

- Instruct the user to 'Please verify that the EGM does not accept any additional funds. Press 'False' if the EGM does.'.

Error

- DUT Error

5. **Verify that the credit meter is not greater than the max credit meter value.**
{055b2ce2-1ab4-4fb1-8509-63d8d264f9c4}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-CB-00035` [The EGM MUST NOT have a credit meter value greater than ${maxCredit}.]

Test Case: CB-COR-00049

This test case verifies that the EGM persists critical data structures for the cabinet device and sends `G2S_CBE325 EGM Power Up/Restart` event after a restart.

Objectives

- Verify that the EGM persists the `cabinetStatus` and `cabinetProfile`.
- Verify that the EGM generates `G2S_CBE325` after a restart.

Requirements Under Test

- 1.5.5
- 3.79.1
- 3.79.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_communications	owner

Test Procedure

1. **Get the cabinet profile.**
   {018d0f36-9426-40ba-b022-8a4d03bc479e}

   Command
   - `cabinet.getCabinetProfile`

   Error
   - Send Request Errors

2. **Get the cabinet status.**
   {02ef3f7a-d97f-4d7e-8fd8-9ca1e4c2ee0c}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

3. **Instruct the user to power off the EGM.**
{03c50abc-7706-499a-9a75-9a004c8bef0e}

   Action
   - Instruct the user to 'Please power off the EGM. Press 'TRUE' when the EGM has been powered off.'.

   Errors
   - DUT Error
   - `E-CM-00057` [The user failed to power cycle the EGM.]

4. **Instruct the user to power on the EGM and wait for the EGM to reconnect to the host.**
{04398727-bc1b-436c-8766-9da5b34d8068}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Actions
   - Instruct the user to 'Please power on the EGM.'.

   Errors
   - Expected Request Errors
   - DUT Error

5. **Enable communications and verify a G2S_CBE325 event is generated.**
{05df5d7a-03b6-4f4b-9273-ed1ffd9c34f6}

   Command
   - `communications.setCommsState`

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE325` [EGM Power Up]

   Errors
   - Send Request Errors
   - Event Checking Errors

6. **Get the cabinet profile to verify it did not change after the power cycle.**
{0606a49b-113b-4ebf-aaa1-6fa6ef4d332b}

   Command
   - `cabinet.getCabinetProfile`

   Errors

- Send Request Errors
- `E-XX-00023` [The EGM MUST persist ${dataStructure} for the ${class} device. (${attribute} : ${originalValue}/${currentValue}])]

7. **Get the cabinet status to verify it did not change after the power cycle.**
{072bbabe-1aed-47fc-9739-7787232ac67f}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-XX-00023` [The EGM MUST persist ${dataStructure} for the ${class} device. (${attribute} : ${originalValue}/${currentValue}])]

Test Case: CB-COR-00050

This test case verifies that the EGM persists affected data sets for cabinet events.

Objective

Verify that the EGM persists affected data sets for cabinet events.

Requirements Under Test

- 1.23.23

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_eventHandler	owner

Test Procedure

1. **Instruct the user to open the cabinet door.**
   {01635b5f-b0eb-4259-b496-142ecc45b0a7}

   Action

   - Request CABINET door OPEN.

   Event
   (Time out: ${event.timeOut})

   - `G2S_CBE307` [Cabinet Door Open]

   Errors

   - Event Checking Errors
   - DUT Error

2. **Instruct the user to close the cabinet door.**
   {027432e7-af20-4058-a2f1-e99e402253be}

   Action

   - Request CABINET door CLOSE.

   Event
   (Time out: ${event.timeOut})

- G2S_CBE308 [Cabinet Door Closed]

Errors

- Event Checking Errors
- DUT Error

3. **Verify that the cabinet device is in the affected device and meter list for G2S_CBE307 and G2S_CBE308.**
{038c01d5-9a66-4d33-b580-8e018c51a1c3}

Command

- eventHandler.getEventHandlerLog

Errors

- Send Request Errors
- E-EH-00068 [Event log MUST persist affected data sets for events.]

Test Case: CB-COR-00051

This test case verifies that events associated with activating the operator menu are correct.

Objectives

- Verify that `G2S_CBE206 Operator Menu Activated` event is generated when the operator menu is activated.
- Verify that `G2S_CBE205 EGM Enabled and Playable` event is generated when the operator menu is exited.

Requirements Under Test

- 3.43.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to activate the operator menu.**
   {01a0e6fd-dca1-4137-ac60-08b65e60d19f}

   #### Action

   - Instruct the user to 'Please activate the operator menu.'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE206` [Operator Menu Activated]

   #### Errors

   - Event Checking Errors
   - DUT Error

2. **Instruct the user to exit the operator menu.**
   {020dd251-5956-41d8-aa4f-68e5cb5f5e78}

   #### Action

- Instruct the user to 'Please exit the operator menu.'.

Event

(Time out: ${event.timeOut})

- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00052

This test case verifies that events associated with demo mode are correct.

Objectives

- Verify that `G2S_CBE207 Demo Mode Activated` event is generated when the demo mode is activated.
- Verify that `G2S_CBE205 EGM Enabled and Playable` event is generated when the demo mode is exited.

Requirements Under Test

- 3.44.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to activate the demo mode.**
   {01b8b956-f240-4f89-ae9f-9940c0a9f981}

   #### Action

   - Instruct the user to 'Please activate the demo mode.'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE207` [Demo Mode Activated]

   #### Errors

   - Event Checking Errors
   - DUT Error

2. **Instruct the user to exit the demo mode.**
   {02aa948c-85ce-4d56-80b9-c93caaba1273}

   #### Action

- Instruct the user to 'Please exit the demo mode.'.

Event

(Time out: ${event.timeOut})

- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00053

This test case verifies that events associated with meters/audit mode are correct.

Objectives

- Verify that `G2S_CBE208 Meters/Audit Mode Initiated` event is generated when the meters/audit mode is activated.
- Verify that `G2S_CBE205 EGM Enabled and Playable` event is generated when the meters/audit mode is exited.

Requirements Under Test

- 3.45.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to activate the meters/audit mode.**
   {01be186d-f13a-48c5-8d0f-f34075805cf8}

   Action

   - Instruct the user to 'Please activate the meters/audit mode.'.

   Event
   (Time out: ${event.timeOut})

   - `G2S_CBE208` [Meters/Audit Mode Initiated]

   Errors

   - Event Checking Errors
   - DUT Error

2. **Instruct the user to exit the demo mode.**
   {025f7368-cbe1-4c73-9c87-eb3494dd3d59}

   Action

- Instruct the user to 'Please exit the meters/audit mode.'.

Event

(Time out: ${event.timeOut})

- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00054

This test case verifies that events associated with operator lock are correct.

Objectives

- Verify that `G2S_CBE209 EGM Locked - Operator Menu` event is generated when the operator locks the EGM.
- Verify that `G2S_CBE205 EGM Enabled and Playable` event is generated when the operator lock is released.

Requirements Under Test

- 3.46.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to enter the operator menu.**
   {016e05cf-d9a7-4b21-b00c-500b3860984f}

   #### Action

   - Instruct the user to 'Please operator lock the EGM.'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE209` [EGM Locked - Operator Menu]

   #### Errors

   - Event Checking Errors
   - DUT Error

2. **Instruct the user to release the operator lock.**
   {0225afd1-6f1e-41b3-a500-26f11d296563}

   #### Action

- Instruct the user to 'Please release the operator lock.'.

Event

(Time out: ${event.timeOut})

- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Event Checking Errors
- DUT Error

Test Case: CB-COR-00055

This test case verifies that events associated with power off door open monitors are correct.

Objectives

- Verify that `G2S_CBE317 Power Off - Logic Door Open` event is generated when the logic door is opened with a power off door monitor installed.
- Verify that `G2S_CBE303 Logic Door Open` event is generated when the EGM is powered on.
- Verify that `G2S_CBE318 Power Off - Auxiliary Door Open` event is generated when the auxiliary door is opened with a power off door monitor installed.
- Verify that `G2S_CBE305 Auxiliary Door Open` event is generated when the EGM is powered on.
- Verify that `G2S_CBE319 Power Off - Cabinet Door Open` event is generated when the auxiliary door is opened with a power off door monitor installed.
- Verify that `G2S_CBE307 Cabinet Door Open` event is generated when the EGM is powered on.

Requirements Under Test

- 3.72.1
- 3.72.2
- 3.73.1
- 3.73.2
- 3.74.1
- 3.74.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Get the cabinet device meter.**
{0190b5b6-3944-49c8-823f-05e1bd895a32}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

2. **Instruct the user to power off the EGM.**
{02dd100f-4063-47cf-9ace-05a918d3922b}

   Action
   - Instruct the user to 'Please power off the EGM. Press 'TRUE' when the EGM has been powered off.'.

   Errors
   - DUT Error
   - `E-CM-00057` [The user failed to power cycle the EGM.]

3. **Instruct the user to open the cabinet door.**
{03e0f0c4-35f1-4b46-9a12-8c3a7a7147d8}

   Action
   - Instruct the user to 'Please open the cabinet door. Press 'TRUE' after opening the cabinet door.'.

   Errors
   - DUT Error
   - `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

4. **Instruct the user to open the logic door.**
{04023988-2ecc-4ea8-97fa-4fe8354392b3}

   Action
   - Instruct the user to 'Please open the logic door. Press 'TRUE' after opening the logic door.'.

   Errors
   - DUT Error
   - `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

5. **Instruct the user to open the auxiliary door.**
{0565ed87-856c-4195-bc48-ad320838057a}

   Action

- Instruct the user to 'Please open the auxiliary door. Press 'TRUE' after opening the auxiliary door.'.

Errors

- DUT Error
- `E-XX-00020` [The user did not cause device fault for event ${eventCode}.]

6. **Instruct the user to close all of the doors.**
{06db3bfa-2770-4598-bd02-cf5fe8cb2e1f}

Action

- Instruct the user to 'Please close the EGM doors. Press 'TRUE' after closing the doors.'.

Error

- DUT Error

7. **Instruct the user to power on the EGM and wait for the EGM to reconnect to the host.**
{07b02294-7d85-4ae3-857c-89db948d69d0}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Actions

- Instruct the user to 'Please power on the EGM.'.

Errors

- Expected Request Errors
- DUT Error

8. **Enable communications and verify the proper events are generated.**
{0802416f-58e7-42a1-b737-40bcb5173346}

Command

- `communications.setCommsState`

Events
(Time out: ${event.timeOut})

- `G2S_CBE303` [Logic Door Open]
- `G2S_CBE305` [Auxiliary Door Open]
- `G2S_CBE307` [Cabinet Door Open]
- `G2S_CBE317` [Power Off - Logic Door Open]
- `G2S_CBE318` [Power Off - Auxiliary Door Open]
- `G2S_CBE319` [Power Off - Cabinet Door Open]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: CB-COR-00056

This test case verifies that `G2S_CBE320 Operator Reset Cabinet` event is generated after the operator resets the cabinet.

Objective

Verify that `G2S_CBE320` event is generated when the operator resets the cabinet.

Requirements Under Test

- 3.75.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to reset the cabinet.**
   {01551997-d460-42f7-8558-e30ac6efdd83}

   #### Action

   - Instruct the user to 'Please reset the cabinet.'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE320` [Operator Reset Cabinet]

   #### Errors

   - Event Checking Errors
   - DUT Error

Test Case: CB-COR-00057

This test case verifies that `G2S_CBE321 Life-To-Date Meters Reset` event is generated after the life-to-date meters are reset.

Objective

Verify that `G2S_CBE321` event is generated when the life-to-date meters are reset.

Requirements Under Test

- 3.76.1
- 3.76.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct the user to reset the life-to-date meters.**
   {01b02fd6-f3e1-43fd-8312-996f617b277f}

   #### Action

   - Instruct the user to 'Please reset the life-to-date meters.'.

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_CBE321` [Life-To-Date Meters Reset]

   #### Errors

   - Event Checking Errors
   - DUT Error
   - `E-EH-00001` [Event ${eventCode} contains additional data from a known namespace.]

Test Case: CB-MRS-00001

This case tests that standard configuration option in the Master Reset Support (Extension GtkMR) functional group is available in `optionConfig`, and that it is consistent with the `cabinetProfile`.

Objectives

- Verify that the `optionConfig` parameter for option ID `GTK_protocolOptions` is configured properly.
- Verify that the `cabinetProfile` attribute value matches the `optionConfig` parameter value.

Requirements Under Test

- 3.10.30

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: cabinet
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_cabinetOptions`.**
   {014c4b9d-cb68-4f8d-80bf-19c8f4b0e11b}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `GTK_protocolOptions` has the correct parameter configuration.**
   {02c7c00c-9faf-42a9-806e-0bf27315608c}

   Errors

- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

3. **Assert that the `cabinetProfile` attribute value matches the `optionConfig` parameter value.**
{034aeee9-78b6-454c-0022-752a8594bbed}

Command

- `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: CB-MRS-00002

This case verifies that the host sends a valid masterReset command.

Objective

Verify the masterReset command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Set the response manager to respond with an application error to a masterReset request.**
   {01510177-dad1-42ef-941f-ea3b5b34d83c}

2. **Instruct user to send masterReset command from the host.**
   {0229810b-2eff-4e92-a7d6-d3ac3f11cec7}

   #### Command

   - `cabinet.masterReset`

   #### Actions

   - Instruct the user to 'Send masterReset to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03d146b8-6aa8-48c4-90f6-815c12e7a61e}

Test Case: CB-MRS-00003

This case verifies that the host sends a valid authorizeMasterReset command.

Objective

Verify the authorizeMasterReset command from the owner host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send authorizeMasterReset command from the host.**
   {012c67c5-59d9-4dfd-a9e9-e35f5d0f8b63}

   #### Command

   - `cabinet.authorizeMasterReset`

   #### Actions

   - Instruct the user to 'Send authorizeMasterReset to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-MRS-00004

This case verifies that the host sends a valid cancelMasterReset command.

Objective

Verify the cancelMasterReset command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send cancelMasterReset command from the host.**
   {0156c563-7962-45ef-8660-76e7dfd0ab06}

   #### Command

   - `cabinet.cancelMasterReset`

   #### Actions

   - Instruct the user to 'Send cancelMasterReset to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-MRS-00005

This case verifies that the host sends a valid getMasterResetStatus command.

Objective

Verify the getMasterResetStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getMasterResetStatus command from the host.**
   {010768bc-b5b4-4a76-b110-d0e7fba1847f}

   #### Command

   - `cabinet.getMasterResetStatus`

   #### Actions

   - Instruct the user to 'Send getMasterResetStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-MRS-00006

This case verifies that the host does not process a masterReset request.

Objective

Verify that the host sends G2S_APX008 Command Not Supported error when a masterReset is sent as a request.

Requirements Under Test

- 1.4.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send a masterReset as a request and verify host responds with a G2S_APX008 error.**
   {016156cb-1197-4978-b5ed-501c9a5b88af}

   #### Command

   - `cabinet.masterReset`

     Expect **G2S_APX008**

   #### Error

   - Send Request Errors

Test Case: CB-MRS-00007

This case verifies that the host handles errors responses to masterReset requests.

Objectives

- Verify that the host is still working after responding with a cabinetStatus to a masterReset command.
- Verify that the host is still working after responding with a masterReset response to a masterReset command.
- Verify that the host is still working after responding with a masterResetStatus notification to a masterReset command.
- Verify that the host is still working after responding with a unknown error code to a masterReset command.

Requirements Under Test

- 1.4.5
- 1.4.6
- 1.4.7
- 1.42.3

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Set the response manager to respond with a cabinetStatus response to a masterReset request.**
   {0127a535-f0ed-4f33-87de-e0bf066eac22}

2. **Instruct user to send masterReset command from the host.**
   {02d50c8a-c484-4a1f-b73f-179b3420dd12}

   Command
   - `cabinet.masterReset`

Actions

- Instruct the user to 'Send masterReset to device ${deviceUnderTest.deviceId} (1 of 4).'.

Errors

- Expected Request Errors
- DUT Error

3. **Set the response manager to respond with a masterReset response to a masterReset request.**
{038b25ac-f7b7-438a-9a7d-358bd1665044}

4. **Instruct user to send masterReset command from the host.**
{04c0ed2f-bf64-485a-bd19-b8458c2f9ce9}

Command

- `cabinet.masterReset`

Actions

- Instruct the user to 'Send masterReset to device ${deviceUnderTest.deviceId} (2 of 4).'.

Errors

- Expected Request Errors
- DUT Error

5. **Set the response manager to respond with a masterResetStatus notification to a masterReset request.**
{05b808b9-404a-4d7d-9d8c-a172049cd5e0}

6. **Instruct user to send masterReset command from the host.**
{060a20cc-d280-43c7-93e1-1ba0439d8e92}

Command

- `cabinet.masterReset`

Actions

- Instruct the user to 'Send masterReset to device ${deviceUnderTest.deviceId} (3 of 4)'.

Errors

- Expected Request Errors
- DUT Error

7. **Set the response manager to respond with 'CVT_ error' error code to a masterReset request.**
{07c52047-50bb-4534-9b59-bd56a1ec3c36}

8. **Instruct user to send masterReset command from the host.**
{08e8e0ab-745e-488b-8954-196cb8705bbe}

   Command
   - `cabinet.masterReset`

   Actions
   - Instruct the user to 'Send masterReset to device ${deviceUnderTest.deviceId} (4 of 4).'.

   Errors
   - Expected Request Errors
   - DUT Error

9. **Reset the response manager.**
{0936bbc2-3cda-42b1-8499-4e4323d034d2}

10. **Send a masterResetStatus with masterResetStatus='GTK_aborted' to the host to verify that it is still working.**
{1067b94a-4e1b-4d8b-8043-46f23bdc7191}

    Command
    - `cabinet.masterResetStatus`

    Error
    - Send Request Errors

Test Case: CB-MRS-00008

This case verifies that the host handles masterResetStatus commands with invalid XML.

Objective

Verify that the host sends G2S_MSX004 Incomplete/Malformed XML, G2S_MSX005 Invalid Data Type Encountered, or G2S_APX004 Incomplete/Malformed XML error to a masterResetStatus command with a missing requestId attribute.

Requirements Under Test

- 1.27.33
- 1.41.2
- 1.41.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send a masterResetStatus request with a missing requestId attribute and verify that the host responds with G2S_MSX004, G2S_MSX005, or G2S_APX004 error code.**
   {011bea53-85da-4b5a-88d6-9bc4cfcecd24}

   Command
   - `cabinet.masterResetStatus`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Error
   - Send Request Errors

Test Case: CB-MRS-00009

This case tests that the host processes duplicate masterResetStatus commands.

Objectives

- Verify that the host processes duplicate masterResetStatus commands.
- Verify that the host sends GTK_CBX005 MasterReset Not Pending, G2S_APX999 Undefined Error, or masterResetStatusAck response

Requirements Under Test

- 1.28.9
- 1.44.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Send five masterResetStatus commands in one G2S message. Verify the host sends five application-level errors.**
   {0177af0f-3964-490b-879d-94caa5675c3f}

   Command
   - `cabinet.masterResetStatus`

     Expect **GTK_CMX005** or **Normal Response**

   Error
   - Send Request Errors

Test Case: CB-MRS-00010

This case tests that the masterReset.timeOutDate attribute is within 24 hours.

Objective

Verify that if the timeOutDate attribute in the masterReset command is within 24 hours.

Requirements Under Test

- 3.6.1
- 3.17.13

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Set the response manager to respond to a masterReset command with a masterResetStatus response.**
   {01a0b2ed-f5e3-4578-8660-732ab7ddb30e}

2. **Instruct user to send a masterReset command.**
   {02ab9c2b-7719-4719-b44f-32225dd46fe6}

   Command
   - `cabinet.masterReset`

   Actions
   - Instruct the user to 'Send a masterReset to device ${deviceUnderTest.deviceId}.'.

   Errors
   - Expected Request Errors
   - DUT Error

3. **Send a masterResetStatus request.**
   {03a7c668-6aa8-4d80-a20c-bbb80d699774}

   Command
   - `cabinet.masterResetStatus`

Error

- Send Request Errors

Test Case: CB-MRS-00011

This case verifies that the host sends a valid authorizeMasterReset command from the guest host.

Objective

Verify the authorizeMasterReset command from the guest host is valid.

Requirements Under Test

- 1.999.999

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest

Test Procedure

1. **Start a master reset.**
   {0185d8e0-2733-4780-a901-721e24e47a4f}
2. **Wait for authorizeMasterReset from guest host.**
   {02e8ba57-7b7e-40f7-b0e4-5ff4ac5430f8}

   Command
   - `cabinet.authorizeMasterReset`

   Error
   - Expected Request Errors
3. **Wait for the host to enable communications.**
   {03e75cf2-e668-4033-be1d-de81a4f4d9fd}

Test Case: CB-OCC-00001

This case tests that standard configuration option in the Occupancy Meter Support (Extension g2sOC) functional group is available in `optionConfig`, and that it is consistent with the `cabinetProfile`.

Objectives

- Verify that the `optionConfig` parameter for option ID `G2S_occupancyMeterOptions` is configured properly.
- Verify that the `cabinetProfile` attribute values matches the `optionConfig` parameter value.

Requirements Under Test

- 3.10.30

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_cabinetOptions`.**
   {01a9bad1-f97e-44e4-0030-4ddb4883cb89}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `G2S_occupancyMeterOptions` has the correct parameter configuration.**
   {02dbc253-de34-4045-0042-a7e680f9a248}

   Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

3. **Assert that the `cabinetProfile` attribute value matches the `optionConfig` parameter value.**
{038d7bf0-10e2-4477-0080-9ea4e29de358}

Command

- `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: CB-OHS-00001

This case verifies that the host sends a valid setOperatingHours command.

Objective

Verify the setOperatingHours command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send setOperatingHours command from the host.**
   {01bfb1f4-d58c-4546-96bf-c3d49d7097f0}

   #### Command

   - `cabinet.setOperatingHours`

   #### Actions

   - Instruct the user to 'Send setOperatingHours to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-OHS-00002

This case verifies that the host sends a valid getOperatingHours command.

Objective

Verify the getOperatingHours command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getOperatingHours command from the host.**
   {01b16348-8d3c-4498-9cda-e40b9b7016aa}

   #### Command

   - `cabinet.getOperatingHours`

   #### Actions

   - Instruct the user to 'Send getOperatingHours to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-RRS-00001

This case verifies that the EGM does not accept the `remoteReset` command from non-owner hosts.

Objective

Verify that the EGM does not accept the `remoteReset` command from non-owner hosts.

Requirements Under Test

- 1.8.14
- 3.6.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	guest or non-guest

Test Procedure

1. **Send a `resetProcessor` command, and verify that the EGM responds with a `G2S_APX010` or `G2S_APX012` error code.**
   {c41b5019-fb5f-4de2-00da-6480749539b0}

   #### Command

   - `cabinet.resetProcessor`

     Expect **G2S_APX010** or **G2S_APX012**

   #### Errors

   - Send Request Errors
   - `E-CM-00040` [The EGM must only allow control commands from the owner host.]

Test Case: CB-RRS-00002

This case verifies that the host sends a valid resetProcessor command.

Objective

Verify the resetProcessor command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send resetProcessor command from the host.**
   {01826de0-06eb-4bf3-8d95-1e9ef33a2f39}

   #### Command

   - `cabinet.resetProcessor`

   #### Actions

   - Instruct the user to 'Send resetProcessor to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-TZO-00001

This case tests that standard configuration option in the Time Zone Offset Support functional group is available in `optionConfig`, and that they are consistent with the `cabinetProfile`.

Objectives

- Verify that the `optionConfig` parameter for option ID `G2S_timeZonesSupportedOption` is configured properly.
- Verify that the `cabinetProfile` command values match the `optionConfig` values.

Requirements Under Test

- 3.10.30

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: cabinet
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_cabinet	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_cabinetOptions`.**
   {01c859a0-f86c-4442-0025-8c0edd531cd5}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `G2S_timeZonesSupportedOption` has the correct parameter configuration.**
   {022d70d5-cd2d-4772-0095-67968c28c155}

   Errors

- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]
- `E-OC-00007` [Option item ${optionId} MUST exist.]

3. **Assert that the `cabinet` profile values match the `optionConfig` values.**
{03567735-624b-4340-8035-96df860427ef}

Command

- `cabinet.getCabinetProfile`

Errors

- Send Request Errors
- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: CB-TZO-00002

This case verifies that the host sends a valid setTimeZoneOffsets command.

Objective

Verify the setTimeZoneOffsets command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	owner

Test Procedure

1. **Instruct user to send setTimeZoneOffsets command from the host.**
   {016372be-8e3d-4346-8a80-c3912b03a3fe}

   #### Command

   - `cabinet.setTimeZoneOffsets`

   #### Actions

   - Instruct the user to 'Send setTimeZoneOffsets to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CB-TZO-00003

This case verifies that the host sends a valid getTimeZoneOffsets command.

Objective

Verify the getTimeZoneOffsets command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: cabinet

Required Devices

Device	Permissions
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to send getTimeZoneOffsets command from the host.**
   {013183eb-2917-45e4-967b-628038c92456}

   #### Command

   - `cabinet.getTimeZoneOffsets`

   #### Actions

   - Instruct the user to 'Send getTimeZoneOffsets to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Communications Functional Groups

- Core Functionality (COR)

Test Case: CM-COR-00001

This case continuously tests that messages are well-formed XML and valid according to the G2S schema.

Objective

Continuously verify that all messages are valid according to the G2S schema.

Requirements Under Test

- 1.1.1
- 1.2.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that all messages are well-formed XML.**
   {8887f646-9131-4fcd-a8be-d93ccdc9332d}

   Error
   - `E-CM-00001` [Endpoint must send valid XML.]

2. **Assert that all messages are valid according to the G2S schema.**
   {30f83f71-7570-4e71-87e9-48764126d61d}

   Error
   - `E-CM-00018` [Message MUST be syntactically correct per the G2S schema.]

Test Case: CM-COR-00002

This case continuously tests that point-to-point message processing is correct.

Objectives

- Continuously verify that only eventReport, meterInfo and smartCardMsg commands are sent as notifications.
- Continuously verify that all requests have a response or an error.

Requirements Under Test

- 1.4.1
- 1.4.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that only eventReport, meterInfo and smartCardMsg commands are sent as notifications.**
   {63cb6d18-c219-4d95-0021-4c4d157b8e5f}

   Error
   - `E-CM-00009` [Command ${command} MUST NOT be sent as a notification.]

2. **Assert that all requests have a response or an error.**
   {1babf0b7-7f56-4f8a-00fe-9c6b250a5305}

   Error
   - `E-CM-00010` [Endpoint did not send a valid response to ${command}.]

Test Case: CM-COR-00003

Deleted

This test case was deleted and replaced by test case CM-COR-00092.

Test Case: CM-COR-00004

This case continuously tests that event IDs are correct.

Objectives

- Continuously verify that all `eventReport.eventId` attributes are contiguous when subscribed to all events.
- Continuously verify that the `eventReport.eventId` attribute for an event is unique since the last clearing of the non-volatile storage.
- Continuously verify that the `eventReport.eventId` attribute is constantly increasing for each `eventReport,` in event ID order.

Requirements Under Test

- 1.22.1
- 1.22.4
- 1.22.5
- 1.22.6

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the `eventReport.eventId` attribute is unique since the last clearing of the non-volatile storage.**
   {cd4836e5-fa37-455d-00f2-b28f0d89b31b}

   Error
   - `E-EH-00033` [The event ID ${eventId} has already been used.]

2. **Assert that all `eventReport.eventId` attributes are contiguous when subscribed to all events.**
   {f0f5eddf-4c61-4593-0056-5c2d4f2d8810}

   Error
   - `E-EH-00034` [Event IDs MUST be contiguous when all events are subscribed to.]

3. **Assert that the `eventReport.eventId` attribute is constantly increasing for each `eventReport`, in `eventDateTime` order.**
{bcaa83e6-18d5-472f-002f-b099657fa0ee}

   Error
   - `E-EH-00035` [Event ${eventId} MUST have an event date/time after ${lastEventDateTime}.]

4. **Assert that the `eventReport.eventDateTime` attribute is the same as a previous event report for the same `eventReport.eventId`.**
{22d94a40-1cfc-497a-bf2d-2424e4115133}

   Error
   - `E-EH-00036` [Event ${eventCode} has different eventDateTime values for the same event ID.]

5. **Assert that the `eventReport.eventCode` attribute is the same as a previous event report for the same `eventReport.eventId`.**
{69121195-ff4c-40b0-9262-5db1a3413c4f}

   Error
   - `E-EH-00037` [Event ${eventCode} MUST have the same event code for the same event ID.]

Test Case: CM-COR-00005

This case continuously tests that `g2sAck` responses are correct.

Objectives

- Continuously verify that the `g2sAck` element's `hostId` is the host ID used during testing.
- Continuously verify that the `g2sAck` element's `egmId` is the EGM ID used during testing.
- Continuously verify that the `errorText` in the `g2sAck` response is empty if the `errorCode` value is `G2S_none`.

Requirements Under Test

- 1.26.13
- 1.26.14
- 1.26.16

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the `g2sAck` element's `hostId` is the host ID used during testing.**
   {0f607932-174f-49cd-000d-2eb8c4e517fc}

   Error
   - `E-CM-00013` [Invalid host ID of ${hostId} in g2sAck.]

2. **Assert that the `g2sAck` element's `egmId` is the EGM ID used during testing.**
   {d3d66618-6859-4209-0018-dfd81b27ea5d}

   Error
   - `E-CM-00014` [Invalid EGM ID of '${egmId}' in g2sAck.]

3. **Assert that the `errorText` in the `g2sAck` is empty if the `errorCode` is `G2S_none`.**
   {772a0050-5639-40c4-00d9-8b75c6a40eb9}

   Error
   - `E-CM-00016` [Error text MUST be empty if error code is G2S_none.]

Test Case: CM-COR-00006

Deleted

This test case was deleted and replaced by test case CM-COR-00035.

Test Case: CM-COR-00007

This case continuously tests that an error indicating a command or message is too large is not reported by the EGM.

Objectives

- Continuously verify that `G2S_MSX008 Inbound Message Too Large` is not reported.
- Continuously verify that `G2S_APX018 Inbound Command Too Large` is not reported.

Requirements Under Test

- 1.29.1
- 1.29.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that that `G2S_MSX008 Inbound Message Too Large` is not reported. The EGM should be able to handle all CVT tests.**
   {8c825c1f-97cf-402f-0019-cb4606b50fb2}

   Error
   - `E-CM-00017` [EGM MUST not send ${errorCode}.]

2. **Assert that `G2S_APX018 Inbound Command Too Large` is not reported. The EGM should be able to handle all CVT tests.**
   {91798aec-4e86-42e6-0035-c38e05934557}

   Error
   - `E-CM-00017` [EGM MUST not send ${errorCode}.]

Test Case: CM-COR-00008

This case continuously tests that commands are generated in proper processing order.

Objectives

- Continuously verify that command IDs are in command ID order.
- Continuously verify that command IDs are in command ID order in the `g2sBody` element.

Requirements Under Test

- 1.30.3
- 1.30.4

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that command IDs are in command ID order.**
   {e8bc22ff-64eb-478d-003e-c787756305ce}

   Error
   - `E-CM-00007` [Command ID of ${commandId} must be greater than ${lastCommandId}.]

2. **Assert that command IDs are in command ID order in the `g2sBody` element.**
   {e339a723-c01e-4d7e-80c8-83e6853216d3}

   Error
   - `E-CM-00007` [Command ID of ${commandId} must be greater than ${lastCommandId}.]

Test Case: CM-COR-00010

This case tests that the EGM supports unknown classes.

Objective

Verify that the EGM sends the proper error for unknown classes.

Requirements Under Test

- 1.41.8

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a command in the cvtTesting class to the EGM.**
   {743398e3-d5fc-40a5-b291-9e1799d49411}

   #### Command

   - `cvtTesting.keepAlive`

     Expect **G2S_APX007** or **G2S_APX014**

   #### Errors

   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: CM-COR-00011

This case tests that the EGM rejects unknown commands.

Objective

Verify that the EGM sends the proper error for unknown commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a communications.cvtTesting command to the EGM.**
   {fbfa3327-5aaa-43dd-a790-9b2bdc4b9dbb}

   #### Command

   - `communications.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: CM-COR-00012

This case continuously tests that commands IDs are correct per section 2.2 of G2S 2.1.

Objectives

- Continuously verify that the starting command ID that is communicated in the `commsOnLine` command is used by the EGM.
- Continuously verify that the command ID is strictly increasing by 1 (one).

Requirements Under Test

- 2.2.4
- 2.2.8
- 2.2.10

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that command IDs are strictly increasing by one (1) from the `commsOnLine` starting command ID.**
   {01f0a357-5279-4d42-80b9-814a484684e8}

   Errors
   - `E-CM-00042` [Retried commands MUST have new dateTimeSent values.]
   - `E-CM-00043` [Command IDs MUST strictly increase by one. Command ID ${commandId} MUST be one greater than ${lastCommandId}.]

Test Case: CM-COR-00013

This case continuously tests that `commsStatus.g2sProtocol` is set to true.

Objective

Continuously verify that `commsStatus.g2sProtocol` is set to true.

Requirements Under Test

- 2.22.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that `commsStatus.g2sProtocol` is set to true.**
   {01dcb4a7-dded-4b41-8037-02cfd2440f17}

   Error

   - `E-CM-00044` [The g2sProtocol attribute in commsStatus commands MUST always be set to true.]

Test Case: CM-COR-00014

This case continuously tests that that the EGM initiated commands are only sent to the owner host.

Objective

Continuously verify that EGM initiated commands are sent to the owner host.

Requirements Under Test

- 2.24.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that EGM initiated commands are sent to the owner host.**
   {01c1a583-9aa6-421f-80e5-691b475358f7}

   Error

   - `E-CM-00045` [EGM initiated commands MUST always be sent to the device's owner host.]

Test Case: CM-COR-00015

This case tests that the EGM supports the transport requirements.

Objectives

- Verify that EGM can process UTF-8 and UTF-16.
- Verify that the EGM uses standard P2P transport – SOAP over HTTPS.
- Verify that the EGM validates the egmId in the SOAP message and g2sBody element.
- Verify that the EGM validates the hostId in the SOAP message and g2sBody element.

Requirements Under Test

- 1.1.2
- 1.2.4
- 1.2.8
- 1.2.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a `keepAlive` command to the EGM using UTF-8.**
   {0b7bfad2-c672-4d68-002e-081743fe8e7c}

   Command
   - `communications.keepAlive`

   Errors
   - Send Request Errors
   - `E-CM-00025` [Endpoint MUST support ${xmlEncoding}.]

2. **Send a `keepAlive` command to the EGM using UTF-16.**
   {cf0ccae3-b871-44a1-0048-da39afc4060c}

   Command

- `communications.keepAlive`

Errors

- Send Request Errors
- `E-CM-00025` [Endpoint MUST support ${xmlEncoding}.]

3. **Assert that the EGM is using standard P2P transport.**
{41f0f0b1-f392-4544-80da-538700f29231}

Error

- `E-CM-00024` [Endpoint MUST support standard P2P transport.]

Test Case: CM-COR-00016

This case continuously tests that the device ownership, configurator and guest permissions are preserved unless the deviceChanged value is true.

Objective

Continuously verify that the device ownership, configurator and guest permissions are identical to the previous device values if `commsOnLine.deviceChanged` is **false**. If the device ownership, configuator or guest permissions has changed, that `deviceChanged` value is set to **true**.

Requirements Under Test

- 2.24.5

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the device ownership, configurator and guest permissions are identical to the previous device structure, if `commsOnLine.deviceChanged` is false. If the device ownership, configurator or device permissions has changed, that `deviceChanged` is set to true.**
   {019a0c10-8a39-42cb-80ab-b9afea708283}

   Error

   - `E-CM-00046` [Device ${deviceClass}[${deviceId}] ownership, configurator or guest permission should not have changed.]

Test Case: CM-COR-00017

This case tests that the EGM validates the EGM ID.

Objectives

- Verify that the EGM validates the `egmId` in `g2sBody` and WSDL calls.
- Verify that the EGM does not process commands from unknown hosts.

Requirements Under Test

- 1.3.1
- 1.15.3
- 1.25.4
- 1.26.8
- 1.26.14
- 1.41.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a `keepAlive` command to the EGM with an empty EGM ID in the WSDL call.**
   {0efdd689-fc21-4a2a-00f1-fd222f5f10b6}

   Command
   - `communications.keepAlive`

     Expect **G2S_MSX002**, **G2S_MSX003**, **G2S_MSX004** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

2. **Send a `keepAlive` command to the EGM with a wrong EGM ID in the WSDL call.**
   {1451da6c-3b53-4c7f-801a-87b02fb16624}

   Command
   - `communications.keepAlive`

Expect **G2S_MSX002**, **G2S_MSX003**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- E-CM-00026 [Endpoint MUST validate egmId and hostId in ${location}.]

3. **Send a `keepAlive` command to the EGM with an empty EGM ID in the `g2sBody`.**
{3ea65233-8468-4233-8007-20ba194b6b15}

Command

- communications.keepAlive

  Expect **G2S_APX002**, **G2S_MSX002**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- E-CM-00026 [Endpoint MUST validate egmId and hostId in ${location}.]

4. **Send a `keepAlive` command to the EGM with a wrong EGM ID in the `g2sBody`.**
{a1d7c939-f1ca-49a2-8057-be8a4d58358e}

Command

- communications.keepAlive

  Expect **G2S_APX002**, **G2S_MSX002**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- E-CM-00026 [Endpoint MUST validate egmId and hostId in ${location}.]

5. **Send a `keepAlive` command to the EGM with an empty EGM ID in the WSDL and `g2sBody`.**
{c3330151-0244-467d-008c-181e2f75fd59}

Command

- communications.keepAlive

  Expect **G2S_APX002**, **G2S_MSX002**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- E-CM-00026 [Endpoint MUST validate egmId and hostId in ${location}.]

6. **Send a `keepAlive` command to the EGM with a wrong EGM ID in the WSDL and `g2sBody`.**
{200b0a2f-1aaa-4d5f-807c-49e9a76f94de}

Command

- communications.keepAlive

Expect **G2S_APX002**, **G2S_MSX002**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- E-CM-00026 [Endpoint MUST validate egmId and hostId in ${location}.]

Test Case: CM-COR-00018

This case tests that the EGM validates the `hostId`.

Objective

Verify that the EGM validates the `hostId` in the `g2sBody` and WSDL calls.

Requirements Under Test

- 1.3.1
- 1.8.13
- 1.14.13
- 1.25.4
- 1.25.7
- 1.26.6
- 1.26.7
- 1.26.13
- 1.41.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a `keepAlive` command to the EGM with an empty host ID in the WSDL call.**
   {f3279c27-6d05-4e5a-00e4-fcb7127abe8b}

   Command
   - `communications.keepAlive`

     Expect **G2S_MSX001** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

2. **Send a `keepAlive` command to the EGM with a wrong host ID in the WSDL call.**
   {75a16f03-1640-48e2-00ae-1d2d1ef3773f}

   Command

- `communications.keepAlive`
  Expect **G2S_MSX001** or **G2S_MSX005**

Errors

- Send Request Errors
- `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

3. **Send a `keepAlive` command to the EGM with an empty host ID in the `g2sBody`.**
{3f7e8bb9-5fd2-4979-0074-6f858e1faabf}

Command

- `communications.keepAlive`
  Expect **G2S_MSX001**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

4. **Send a `keepAlive` command to the EGM with a wrong host ID in the `g2sBody`.**
{2690b83b-d33e-4f62-00ae-22a165f85fad}

Command

- `communications.keepAlive`
  Expect **G2S_MSX001** or **G2S_MSX003**

Errors

- Send Request Errors
- `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

5. **Send a `keepAlive` command to the EGM with an empty host ID in the WSDL and `g2sBody`.**
{42647941-3e42-450f-801f-e6f2e5dad239}

Command

- `communications.keepAlive`
  Expect **G2S_MSX001**, **G2S_MSX004** or **G2S_MSX005**

Errors

- Send Request Errors
- `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

6. **Send a `keepAlive` command to the EGM with a wrong host ID in the WSDL and `g2sBody`.**
{ea3296a9-aa03-4e9e-004b-acf993381ba2}

Command

- `communications.keepAlive`
  Expect **G2S_MSX001**, **G2S_MSX003** or **G2S_MSX005**

Errors

- Send Request Errors
- `E-CM-00026` [Endpoint MUST validate egmId and hostId in ${location}.]

Test Case: CM-COR-00019

This case verifies that only qualified devices are included in descriptor lists.

Objective

Verify that EGM properly filters devices in the descriptor list.

Requirements Under Test

- 2.31.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Get a descriptor list that includes all devices.**
   {01e7df8d-e0b0-4e05-00fb-a1fa77954ceb}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Verify that only devices owned by the CVT are included in the descriptor list when `getDescriptor.includeOwner` is set to true and all other flags are set to false.**
   {02b560f8-6964-4e67-8081-69a2d8cc1efa}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CM-00047` [Device ${deviceClass}[${deviceId}] MUST not be included in descriptor list for ${permissionType} devices.]
   - `E-CM-00048` [Device ${deviceClass}[${deviceId}] MUST be included in descriptor list for ${permissionType} devices.]

3. **Verify that only devices that are configured by the CVT are included in the descriptor list when `getDescriptor.includeConfigs` is set to true and all other flags are set to false.**
{03b1a17f-fe71-4f9d-80d8-52f0fc185db2}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CM-00047` [Device ${deviceClass}[${deviceId}] MUST not be included in descriptor list for ${permissionType} devices.]
   - `E-CM-00048` [Device ${deviceClass}[${deviceId}] MUST be included in descriptor list for ${permissionType} devices.]

4. **Verify that only devices that the CVT is a guest of are included in the descriptor list when `getDescriptor.includeGuests` is set to true and all other flags are set to false.**
{04f7ea5e-c8b0-4b95-8040-1adfe8c4805c}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CM-00047` [Device ${deviceClass}[${deviceId}] MUST not be included in descriptor list for ${permissionType} devices.]
   - `E-CM-00048` [Device ${deviceClass}[${deviceId}] MUST be included in descriptor list for ${permissionType} devices.]

5. **Verify that all devices that the CVT is a not an owner, configurator or guest of are included in the descriptor list when `getDescriptor.includeOthers` is set to true and all other flags are set to false.**
{0528d08d-a88a-4f22-801b-8fce6ef5c477}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CM-00047` [Device ${deviceClass}[${deviceId}] MUST not be included in descriptor list for ${permissionType} devices.]
   - `E-CM-00048` [Device ${deviceClass}[${deviceId}] MUST be included in descriptor list for ${permissionType} devices.]

Test Case: CM-COR-00020

This case verifies that the EGM properly supports the `setKeepAlive` command.

Objectives

- Verify that EGM sets a keep alive timer as specified by the host.
- Verify that EGM stops sending keepAlive commands if the `setKeepAlive.interval` is set to zero (0).

Requirements Under Test

- 2.36.1
- 2.37.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner
G2S_meters	owner
G2S_eventHandler	owner

Test Procedure

1. **Disable event and meter subscriptions.**
   {01e8a673-7a9e-4b30-80b3-cee50b795191}

   Commands
   - `eventHandler.clearEventSub`
   - `meters.clearMeterSub`

   Error
   - Send Request Errors

2. **Set the keep alive interval to five (5) seconds and verify EGM sends a `keepAlive` within ten (10) seconds.**
   {0241ea3a-721c-4b70-0055-a192b3b883f1}

   Commands
   - `communications.setKeepAlive`

- `communications.keepAlive`

Errors

- Expected Request Errors
- Send Request Errors

3. **Attempt to set a keep alive timer to 100 milliseconds. Verify that EGM responds with a `setKeepAliveAck` or `G2S_CMX001 Invalid Interval Specified` error.**
{03abfea2-518c-404d-806c-a8b614748bf7}

Commands

- `communications.setKeepAlive`

  Expect **G2S_CMX001** or **Normal Response**
- `communications.keepAlive`

Errors

- Expected Request Errors
- Send Request Errors
- `E-CM-00049` [EGM MUST NOT send a keepAlive at this time.]

4. **Set the keep alive timer to 0 seconds.**
{04d9e73e-4ba8-4d68-80f0-d81408260e72}

Command

- `communications.setKeepAlive`

Error

- Send Request Errors

5. **Set the keep alive timer interval to zero (0). Verify that the EGM does not does not send a `keepAlive` command within ten (10) seconds.**
{05c7286e-dd1a-4505-00f3-b8c07a8ab8f6}

Command

- `communications.setKeepAlive`

Errors

- Send Request Errors
- `E-CM-00049` [EGM MUST NOT send a keepAlive at this time.]

6. **Reset the keepAlive interval.**
{06e984c6-224c-40ab-80d2-7063faf9233c}

Command

- `communications.setKeepAlive`

Error

- Send Request Errors

Test Case: CM-COR-00021

Deleted

This test case was deleted and replaced by test case CM-COR-00079.

Test Case: CM-COR-00023

This case verifies that disabling the communications device is handled properly.

Objectives

- Verify that the EGM retries the commsDisabled at the specified syncTimer rate.
- Verify that the EGM responds to host-originated requests while in the sync state.
- Verify that the EGM does not send EGM-originated requests while in the sync state except commsDisabled.
- Verify that the EGM generates G2S_CME004 after receiving an enabling setCommsState command.

Requirements Under Test

- 1.32.7
- 2.3.4
- 2.3.5
- 2.6.1
- 2.6.2
- 2.6.3
- 2.6.5
- 2.6.13
- 2.7.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Disable the communications device.**
   {01dff477-018b-400d-002e-e10af3314585}

   Commands
   - `communications.setCommsState`
   - `communications.commsDisabled`

   Errors

- Expected Request Errors
- Send Request Errors

2. **Wait for the commsDisabled to be resent.**
{02e18d8f-a01b-4c0d-0004-002e4d2418ea}

Command
- `communications.commsDisabled`

Error
- Expected Request Errors

3. **Verify that the EGM responds to host-oriented commands while in the SYNC state.**
{03ecf81e-dc37-4d8b-001c-efe45e7d596a}

Command
- `communications.getCommsProfile`

Error
- Send Request Errors

4. **Enable the communications device**
{04d9137e-180b-4ce6-00cb-3daa2d364e90}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})
- `G2S_CME003` [Comms Disabled by Host]
- `G2S_CME004` [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors

Test Case: CM-COR-00026

This case continuously tests that host ID values are greater than zero.

Objective

Continuously verify that `hostId` values in `g2sMessage` elements and G2S WSDL calls are greater than zero.

Requirements Under Test

- 1.6.7

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that `hostId` values in `g2sMessage` elements and G2S WSDL calls are greater than zero.**
   {936a4dda-647d-492f-9289-822d2afeaf9d}

   Error
   - `E-CM-00004` [Invalid host ID of ${hostId}.]

Test Case: CM-COR-00027

This case continuously tests the integrity of device owners and configurators after a `commsOnLine` command is received from the EGM.

Objective

Continuously verify the integrity of the device owners and configurators in the `descriptorList` response after a `commsOnLine` command is received from the EGM.

Requirements Under Test

- 1.8.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that each device has an owner.**
   {ad9b5385-319f-4b19-916d-1b4aacc55293}

   Error
   - `E-CM-00005` [Device {$deviceId} does not have an owner.]

2. **Assert that each device has a configurator of the EGM or a valid host.**
   {27635069-b862-4472-bd07-7e25520ef8e8}

   Error
   - `E-CM-00029` [Device ${device} has an invalid configurator of ${configuratorId}]

Test Case: CM-COR-00028

This case continuously tests that EGM ID values have GSA-assigned prefixes.

Objective

Continuously verify that the prefix of the EGM ID matches "[A-Z0-9]_".

Requirements Under Test

- 1.15.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the EGM ID has a GSA-assigned prefix followed by the underscore character.**
   {a0ed8154-8057-418f-9c81-5463b6e8c8b4}

   Error
   - `E-CM-00006` [An EGM ID of '${egmID}' is not valid.]

Test Case: CM-COR-00029

This case continuously tests that the `egmId` and the `hostId` are consistent between SOAP calls and `g2sBody` attributes.

Objective

Continuously verify that the `egmId` and the `hostId` from the SOAP call matches the `egmId` and the `hostId` in the `g2sBody` element.

Requirements Under Test

- 1.3.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the `egmId` and `hostId` are valid.**
   {1ed3201c-2044-480a-8c66-b22392ff0685}

   Error
   - `E-CM-00002` [Endpoint used the wrong end point ID ${endpointId}.]

Test Case: CM-COR-00030

This case continuously tests that command IDs are not reused.

Objective

Continuously verify that command IDs are not reused, except for resends.

Requirements Under Test

- 1.28.3

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that command IDs are not reused, except for resends.**
   {56895d89-f5c9-4148-8d52-e30399d7a71e}

   Error

   - `E-CM-00007` [Command ID of ${commandId} must be greater than ${lastCommandId}.]

Test Case: CM-COR-00031

This case continuously tests that the `dateTime` value in commands is always increasing.

Objective

Continuously verify that the `dateTime` value is always increasing, for each command, except for resends.

Requirements Under Test

- 1.28.3

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the `dateTime` value is always increasing, for each command, except for resends.**
   {87171ae8-1eb2-4e0d-a2c0-03b6ecdf261b}

   Error

   - `E-CM-00008` [Command dateTime of ${dateTime} must be greater than ${lastDateTime}.]

Test Case: CM-COR-00032

This case continuously tests that the device structure is preserved unless the `deviceReset` value is **true**.

Objective

Continuously verify that the device structure is identical to the previous device structure if `commsOnLine.deviceReset` is **false**. If the device structure has changed, that `deviceReset` value is set to **true**.

Requirements Under Test

- 1.39.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that the device structure is identical to the previous device structure, if `commsOnLine.deviceReset` is false. If the device structure has changed, that `deviceReset` is set to true.**
   {4220c1bc-e652-4662-b793-c78f9d4ae287}

   Error
   - `E-CM-00003` [Device ${deviceClass} ${deviceId} structure should not have changed.]

Test Case: CM-COR-00033

This case continuously tests that device identifiers for host-owned devices are correct.

Objective

Continuously verify that device identifiers for host-owned devices are correct.

Requirements Under Test

- 1.13.6

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that host-owned devices have the correct device ID.**
   {6f8e9955-ebc1-468d-8fc6-1cfaa418b261}

   Error

   - `E-CM-00027` [Host-owned devices MUST have the same device ID as the owner's host ID.]

Test Case: CM-COR-00034

This case verifies that the error code in a `g2sAck` response is either `G2S_none` or a message-level error.

Objective

Continuously verify that the `errorCode` in a `g2sAck` response is either **G2S_none** or starts with G2S_MSX.

Requirements Under Test

- 1.26.15

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that a `g2sAck` response has an `errorCode` attribute value of `G2S_none` or starts with G2S_MSX.**
   {51cb1565-29b1-414d-806a-a42c7fc28703}

   Error

   - `E-CM-00035` [The error code of ${errorCode} is not valid for a g2sAck. It MUST be G2S_none or start with G2S_MSX.]

Test Case: CM-COR-00035

This case continuously tests that class-level elements are correct.

Objectives

- Continuously verify that all messages are syntactically correct, according to the selected G2S schema.
- Continuously verify that the device identifier in each response matches the device identifier in the corresponding request.
- Continuously verify that application-level responses have an existing session ID.
- Continuously verify that the `sessionMore` value is set to **false.**
- Continuously verify that the `errorCode` value is **G2S_none** unless an error is being reported.
- Continuously verify that the errorText attribute is empty unless the `errorCode` attribute value is not **G2S_none**.
- Continuously verify that if the `errorCode` attribute value is not **G2S_none,** the command-level element is not present.

Requirements Under Test

- 1.27.1
- 1.27.3
- 1.27.9
- 1.27.26
- 1.27.27
- 1.27.28
- 1.27.29

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that that all messages are syntactically correct, according to the selected G2S schema.**
{013654b0-39f3-4b8b-8007-8f5c0f9c2237}

   Error
   - `E-CM-00018` [Message MUST be syntactically correct per the G2S schema.]

2. **Assert that the device identifier in each response matches the device identifier in the corresponding request.**
{02535165-6bfc-43b5-00f8-75def2ea4766}

   Error
   - `E-CM-00019` [Device ID of ${deviceId} in a response MUST match the corresponding request device ID.]

3. **Assert that application-level responses have an existing session ID.**
{034a382d-be60-41e0-8012-884f695646c0}

   Error
   - `E-CM-00020` [Application responses MUST use an existing session ID.]

4. **Assert that that the `sessionMore` attribute value is set to false.**
{04b36026-a241-4763-00a4-9fe3c32c6a7c}

   Error
   - `E-CM-00022` [The sessionMore attribute MUST be set to false.]

5. **Assert that the `errorCode` attribute value is `G2S_none` unless an error is being reported.**
{05dd0dd5-a6f2-4c7f-0024-fa35052ec5b3}

   Error
   - `E-CM-00015` [Error code MUST be G2S_none, except for application-level errors.]

6. **Assert that the `errorText` attribute value is empty unless the `errorCode` attribute value is not `G2S_none`.**
{0635473e-27b8-49c7-80fd-31bf9ca9e45f}

   Error
   - `E-CM-00016` [Error text MUST be empty if error code is G2S_none.]

7. **Assert that if the `errorCode` attribute value is not `G2S_none`, the command-level element is not present.**
{075eb5f6-1e84-4f0f-8021-2fa9107c3c32}

   Error
   - `E-CM-00015` [Error code MUST be G2S_none, except for application-level errors.]

Test Case: CM-COR-00036

This case verifies that EGM transitions to closing state when it receives a G2S_MSX003 error while in the sync state.

Objectives

- Verify that EGM transitions to the closing state when it receives a G2S_MSX003 Communications Not Online error while in the sync state.
- Verify that EGM generates a G2S_CME101 Comms Not Established event.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.4.2
- 2.5.11
- 2.6.6
- 2.6.13
- 2.7.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Disable the communications device.**
   {01245330-e3b5-4cb6-0005-d550acc217fc}

   Commands
   - `communications.setCommsState`
   - `communications.commsDisabled`

   Errors
   - Expected Request Errors
   - Send Request Errors
   - `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

2. **Send a G2S_MSX003 error when the commsDisabled is resent and wait for the commsClosing command.**
{02572e70-bf9b-4ae8-001d-0179c1f6daee}

   Commands
   - communications.commsDisabled
   - communications.commsClosing

   Error
   - Expected Request Errors

3. **Wait for the commsOnLine command**
{034a1ce5-cf8d-43ad-00db-5a59d2acd894}

   Command
   - communications.commsOnLine

   Error
   - Expected Request Errors

4. **Wait for the commsDisabled command.**
{04075fec-c363-4e3a-80e3-a377480c8439}

   Command
   - communications.commsDisabled

   Error
   - Expected Request Errors

5. **Enable the communications device and verify that the events are generated by the EGM.**
{059ad5e3-8c34-4dc9-00ae-84ddf97db235}

   Command
   - communications.setCommsState

   Events
   (Time out: ${event.timeOut})
   - G2S_CME003 [Comms Disabled by Host]
   - G2S_CME101 [Comms Not Established]
   - G2S_CME001 [Comms Disabled by EGM]
   - G2S_CME002 [Comms Enabled by EGM]
   - G2S_CME100 [Comms Established]
   - G2S_CME004 [Comms Enabled by Host]

   Errors

- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00037

This case verifies that EGM transitions to closing state when the host is unreachable while in the sync state.

Objectives

- Verify that EGM transitions to the closing state when the host is unreachable while in the sync state.
- Verify that EGM generates a G2S_CME120 Comms Host Unreachable.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.4.2
- 2.5.11
- 2.6.12
- 2.6.13
- 2.7.9
- 2.10.14
- 2.10.15
- 2.57.2
- 2.58.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the timeToLive and noResponseTimer for the cabinet device.**
   {01fa3c50-1d49-4e59-00d4-4f692540e225}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **Disable the communications device.**
{02246283-bb13-43b4-80a0-a4d2ca1cf403}

Commands
- `communications.setCommsState`
- `communications.commsDisabled`

Errors
- Expected Request Errors
- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

3. **Stop the CVT web server and wait for the EGM to transition to the closed state.**
{030ff21f-a746-48fc-8025-cd30dbb9ea5f}

4. **Restart the CVT web server.**
{04a6293e-fea5-4068-00f8-b87f6d0961ac}

5. **Wait for the commsOnLine command**
{056cf55b-9bde-4fce-8013-0db717755b7d}

Command
- `communications.commsOnLine`

Error
- Expected Request Errors

6. **Wait for the commsDisabled command.**
{06c9a5f4-ab1e-4f48-00ea-61bcb8bd6d17}

Command
- `communications.commsDisabled`

Error
- Expected Request Errors

7. **Enable the communications device and verify that the events are generated by the EGM.**
{07ddaa3b-0b84-4d01-0019-04e347837842}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})

- G2S_CME003 [Comms Disabled by Host]
- G2S_CME120 [Comms Host Unreachable]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME121 [Comms Transport Up]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00038

This case continuously tests that the EGM does not send EGM-originated requests unless allowed by the current communications state.

Objectives

- EGM does not send any EGM-originated request in the opening state except for commsOnLine.
- EGM does not send any EGM-originated request in the sync state except for commsDisabled.
- EGM does not send any EGM-originated request in the closing state except for commsClosing.

Requirements Under Test

- 2.3.3
- 2.5.3
- 2.5.4
- 2.6.4
- 2.9.4

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that only allowed EGM-originated requests are sent in each communications state.**
   {01397c37-a4cf-493c-8046-6bc5dd7192bb}

   Error
   - `E-CM-00051` [EGM-originated request ${request} is not allowed when the comms state is ${commsState}.]

2. **Assert the frequency of commsOnLine requests is not less than one second.**
   {02936801-37a9-4cf5-00a1-66ef8bfb4d54}

   Error
   - `E-CM-00052` [CommsOnLine commands cannot be sent with a frequency less than one second.]

Test Case: CM-COR-00039

This case verifies that the EGM transitions to the closing state when a configuration change is made while in the sync state.

Objectives

- Verify that EGM transitions to the closing state when a configuration change is made while in the sync state.
- Verify that EGM transitions to the closing state when a configuration change is made while in the online state.
- Verify that EGM generates a G2S_CME101 Comms Not Established event.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.4.2
- 2.5.11
- 2.6.8
- 2.6.13
- 2.7.5
- 2.7.9
- 2.24.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner
G2S_cabinet	guest or non-guest or owner

Test Procedure

1. **Determine the guest permissions of the cabinet device.**
   {01f4350a-a566-4203-80bd-e9f36c2b4a1d}

   Command
   - `communications.getDescriptor`

Error

- Send Request Errors

2. **Disable the communications device.**
{025badec-441d-49aa-00fe-ce8ec93c4b7a}

Commands

- `communications.setCommsState`
- `communications.commsDisabled`

Errors

- Expected Request Errors
- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

3. **If the CVT is a guest of the cabinet device.**

   a. **Instruct the user to remove the CVT as a guest of the cabinet device.**
   {030f2c14-da59-422f-0087-8532988d48ec}

   Command

   - `communications.commsClosing`

   Actions

   - Instruct the user to 'Please remove the CVT as a guest of the cabinet device using the administrative interface.'.

   Errors

   - Expected Request Errors
   - DUT Error
   - `E-CM-00058` [The user failed to change the device permission from the administrative interface.]

4. **If the CVT is not a guest of the cabinet device.**

   a. **Instruct the user to add the CVT as a guest of the cabinet device.**
   {047f4f10-9304-4d13-0064-149ffbab4d54}

   Command

   - `communications.commsClosing`

   Actions

   - Instruct the user to 'Please add the CVT as a guest of the cabinet device using the administrative interface.'.

Errors

- Expected Request Errors
- DUT Error
- E-CM-00058 [The user failed to change the device permission from the administrative interface.]

5. **Wait for the EGM to enter the SYNC state.**
{054c170d-1957-4772-809a-afd0cc515e66}

Commands

- communications.commsOnLine
- communications.commsDisabled

Errors

- Expected Request Errors
- E-CM-00053 [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]

6. **Enable the communications device and verify that the events are generated by the EGM.**
{061bc131-eb2b-43c3-003c-4fd1e15707c1}

Command

- communications.setCommsState

Events

(Time out: ${event.timeOut})

- G2S_CME003 [Comms Disabled by Host]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

7. **If the CVT was originally a guest of the cabinet device.**

   a. **Instruct the user to add the CVT as a guest of the cabinet device.**
   {076aee1e-ae51-4df0-009e-0af13f6dd000}

   Commands

- communications.commsClosing
- communications.commsOnLine
- communications.commsDisabled
- communications.setCommsState

Actions

- Instruct the user to 'Please add the CVT as a guest of the cabinet device using the administrative interface.'.

Events

(Time out: ${event.timeOut})

- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Expected Request Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]
- E-CM-00053 [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]
- E-CM-00058 [The user failed to change the device permission from the administrative interface.]

8. **If the CVT was not originally a guest of the cabinet device.**

   a. **Instruct the user to remove the CVT as a guest of the cabinet device.**
   {0885b06f-3bee-4816-0037-532972cb738c}

   Commands

   - communications.commsClosing
   - communications.commsOnLine
   - communications.commsDisabled
   - communications.setCommsState

   Actions

   - Instruct the user to 'Please remove the CVT as a guest of the cabinet device using the administrative interface.'.

   Events

(Time out: ${event.timeOut})

- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Expected Request Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]
- E-CM-00053 [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]
- E-CM-00058 [The user failed to change the device permission from the administrative interface.]

Test Case: CM-COR-00040

This case verifies that EGM transitions to closing state when it receives a G2S_MSX003 error while in the online state.

Objectives

- Verify that EGM transitions to the closing state when it receives a G2S_MSX003 Communications Not Online error while in the online state.
- Verify that EGM generates a G2S_CME101 Comms Not Established event.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.4.2
- 2.5.11
- 2.6.13
- 2.7.2
- 2.7.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a G2S_MSX003 error when the EGM responds with a commsProfile.**
   {0130cccd-4aef-4cef-80d6-ad28830eefd0}

   Commands
   - `communications.getCommsProfile`
   - `communications.commsClosing`

   Errors
   - Expected Request Errors
   - Send Request Errors

2. **Wait for the commsOnLine command**
   {0200973d-2a4d-43cd-805d-5b6d92333a00}

Command

- `communications.commsOnLine`

Error

- Expected Request Errors

3. **Wait for the commsDisabled command.**
{03e2adf1-d1c2-4150-00d1-8273fa0bdc5d}

Command

- `communications.commsDisabled`

Error

- Expected Request Errors

4. **Enable the communications device and verify that the events are generated by the EGM.**
{0425427b-ea90-434e-00cd-f6b8854571b0}

Command

- `communications.setCommsState`

Events
(Time out: ${event.timeOut})

- `G2S_CME101` [Comms Not Established]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME100` [Comms Established]
- `G2S_CME004` [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00041

This case verifies that EGM transitions to closing state when the host is unreachable while in the online state.

Objectives

- Verify that EGM transitions to the closing state when the host is unreachable while in the online state.
- Verify that EGM generates a G2S_CME120 Comms Host Unreachable.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.4.2
- 2.5.11
- 2.6.12
- 2.6.13
- 2.7.10
- 2.10.14
- 2.10.15
- 2.57.2
- 2.58.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the timeToLive and noResponseTimer for the cabinet device.**
   {010eb75d-bae6-4984-800d-0549f290c336}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **Stop the CVT web server.**
{02ad54b5-d038-4a01-803a-ba9a963708b7}

3. **Restart the CVT web server.**
{031f92a1-7713-4322-80e7-e8996c720ef0}

Commands
- `communications.commsOnLine`
- `communications.commsDisabled`

Errors
- Expected Request Errors
- `E-CM-00056` [EGM MUST transition to closing state when the host is unavailable.]

4. **Enable the communications device and verify that the events are generated by the EGM.**
{04f0560d-2fb1-4080-00a2-a1aaf678a4c6}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})
- `G2S_CME120` [Comms Host Unreachable]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME121` [Comms Transport Up]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME100` [Comms Established]
- `G2S_CME004` [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00042

This case verifies that closing the communications device is handled properly.

Objectives

- Verify that the EGM retries the commsClosing at the specified commsProfile.timeToLive rate.
- Verify that the EGM responds to host-originated requests while in the closing state.
- Verify that the EGM generates G2S_CME001 after receiving commsClosingAck response.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.5.1
- 2.5.11
- 2.6.1
- 2.6.13
- 2.9.2
- 2.9.3
- 2.9.5
- 2.9.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the timeToLive value for the communications device.**
   {018a3ade-a02d-49a3-00ab-b60c8d224a02}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response.**
{02942470-ba8a-456c-000c-b6e08c5e90cc}

   Commands
   - `communications.getCommsProfile`
   - `communications.commsClosing`

   Errors
   - Expected Request Errors
   - Send Request Errors

3. **Verify that the EGM responds to host-originated requests while in the closing state.**
{03126d7c-6087-451b-8031-44051a277e3d}

   Command
   - `communications.getCommsStatus`

   Errors
   - Send Request Errors
   - `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

4. **If the commsProfile.timeToLive is less than 25 seconds**

   a. **Wait for the commsClosing to be resent.**
   {044e706d-4c89-4ae3-8033-62834081ebf4}

      Command
      - `communications.commsClosing`

      Error
      - Expected Request Errors

5. **Wait for the EGM to enter the SYNC state.**
{0505f618-327a-455e-8071-cd617a078822}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Error
   - Expected Request Errors

6. **Enable the communications device and verify that the events are generated by the EGM.**
{065969b1-522c-4780-80d0-eb472cbc06ff}

Command

- `communications.setCommsState`

Events

(Time out: ${event.timeOut})

- `G2S_CME101` [Comms Not Established]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME100` [Comms Established]
- `G2S_CME004` [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00043

This case verifies that EGM transitions to closed state when it receives a G2S_MSX003 error while in the closing state.

Objectives

- Verify that the EGM generates G2S_CME101 when it receives the G2S_MSX003 error in the closing state.
- Verify that the EGM generates G2S_CME001 when it transitions to the closed state.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.5.1
- 2.5.11
- 2.6.1
- 2.6.13
- 2.9.1
- 2.9.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and then a G2S_MSX003 error on the commsClosing request.**
   {019b8182-76fe-4e60-009d-d2dd583b74e5}

   Commands
   - `communications.getCommsProfile`
   - `communications.commsClosing`

   Errors
   - Expected Request Errors
   - Send Request Errors

2. **Wait for the EGM to enter the SYNC state.**
{026403c6-578a-43f5-80e7-054c72a9b6ae}

Commands
- `communications.commsOnLine`
- `communications.commsDisabled`

Error
- Expected Request Errors

3. **Enable the communications device and verify that the events are generated by the EGM.**
{036c03f7-7442-40c7-801d-439eb1c39367}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})
- `G2S_CME101` [Comms Not Established]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME100` [Comms Established]
- `G2S_CME004` [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00044

This case verifies that EGM generates G2S_CME120 event when the host is unreachable while in the closing state.

Objectives

- Verify that the EGM generates G2S_CME120 when the host is unreachable in the closing state.
- Verify that the EGM generates G2S_CME001 when it transitions to the closed state.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.5.1
- 2.5.11
- 2.6.1
- 2.6.13
- 2.9.6
- 2.10.17
- 2.58.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Required Configurations

Option	Value
commsProfile.timeToLive	< 25000

Test Procedure

1. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and then stop the web server.**
   {0166f329-d841-4c0d-80b4-f938f12eab58}

   Commands
   - `communications.getCommsProfile`

- communications.commsClosing

Errors

- Expected Request Errors
- Send Request Errors

2. **Wait 90 seconds and then start the web server.**
{02c3ec36-ac06-4e92-00ca-983556665fc8}

3. **Wait for the EGM to enter the SYNC state.**
{03816f71-5be0-4bcc-8067-8bd94c4d6a2a}

Commands

- communications.commsOnLine
- communications.commsDisabled

Error

- Expected Request Errors

4. **Enable the communications device and verify that the events are generated by the EGM.**
{04a80ae5-ff4e-4a70-0050-0a59f6a96b9c}

Command

- communications.setCommsState

Events

(Time out: ${event.timeOut})

- G2S_CME101 [Comms Not Established]
- G2S_CME120 [Comms Host Unreachable]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME121 [Comms Transport Up]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00045

This case verifies that the opening of the communications device is handled properly.

Objectives

- Verify that the EGM places a commsOnLine command in the outbound queue.
- Verify that the EGM must receive a commsOnLineAck before transitioning to the sync state.
- Verify that if the EGM receives a G2S_MSX003 in the opening state that the EGM generates a G2S_CME101 event.
- Verify that the EGM sets the keepAlive interval to zero.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.5.1
- 2.5.2
- 2.5.4
- 2.5.5
- 2.5.11
- 2.6.1
- 2.6.13
- 2.9.6
- 2.42.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Set the keepAlive interval to 5 (five) seconds.**
   {01a4d304-bf9c-45e9-0022-c10e2b7d58b9}

   Commands
   - `communications.setKeepAlive`
   - `communications.keepAlive`

Errors

- Expected Request Errors
- Send Request Errors

2. **Wait for a keepAlive and respond with a G2S_MSX003 error.**
{02adeaa9-18a6-4c08-00d8-70dbdaf39282}

Commands

- `communications.keepAlive`
- `communications.commsClosing`
- `communications.commsOnLine`

Error

- Expected Request Errors

3. **Wait for the commsOnLine command to be sent and respond with a G2S_MSX003 error code.**
{0334b3fa-46bb-4cc6-0053-c271603f728a}

Command

- `communications.commsOnLine`

Errors

- Expected Request Errors
- Event Not Expected Error

4. **Wait for the commsOnLine command to be retried and bring the EGM to the sync state.**
{049f4b12-3580-4b65-8045-600311a321b7}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Error

- Expected Request Errors

5. **Enable the communications device and verify that the events are generated by the EGM.**
{0575b4b7-7cc4-4aa1-0094-2056e74ffd16}

Command

- `communications.setCommsState`

Events
(Time out: ${event.timeOut})

- G2S_CME101 [Comms Not Established]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

6. **Verify that a keepAlive is not sent within 10 (ten) seconds**
{06d1d8cc-f4bc-49ee-80f9-5f63779f5ffa}

Error

- E-CM-00049 [EGM MUST NOT send a keepAlive at this time.]

Test Case: CM-COR-00046

This case verifies that the EGM transitions to the closing state when a configuration change is made while in the opening state.

Objectives

- Verify that EGM transitions to the closing state when a configuration change is made while in the opening state.
- Verify that EGM transitions to the closing state when a configuration change is made while in the online state.
- Verify that EGM generates a G2S_CME101 Comms Not Established event.

Requirements Under Test

- 2.3.4
- 2.3.5
- 2.5.1
- 2.5.2
- 2.5.7
- 2.5.11
- 2.6.1
- 2.6.13
- 2.7.5
- 2.7.9
- 2.24.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner
G2S_cabinet	guest or non-guest or owner

Test Procedure

1. **Determine the guest permissions of the cabinet device.**
   {01ed7061-b011-45e0-8057-a6147cde5a38}

   Command

- communications.getDescriptor

Error

- Send Request Errors

2. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and leave the EGM in the opening state.**
{026195b4-760a-43f0-80d4-2af67b63eb8b}

Commands

- communications.getCommsProfile
- communications.commsClosing
- communications.commsOnLine

Errors

- Expected Request Errors
- Send Request Errors

3. **If the CVT is a guest of the cabinet device.**

   a. **Instruct the user to remove the CVT as a guest of the cabinet device.**
   {034ec334-9d4d-43d9-802a-a2cd5d613055}

   Command

   - communications.commsClosing

   Actions

   - Instruct the user to 'Please remove the CVT as a guest of the cabinet device using the administrative interface.'.

   Errors

   - Expected Request Errors
   - DUT Error
   - E-CM-00058 [The user failed to change the device permission from the administrative interface.]

4. **If the CVT is not a guest of the cabinet device.**

   a. **Instruct the user to add the CVT as a guest of the cabinet device.**
   {0425a8d3-15c6-4f90-00fe-0f9634afe83d}

   Command

   - communications.commsClosing

   Actions

- Instruct the user to 'Please add the CVT as a guest of the cabinet device using the administrative interface.'.

Errors

- Expected Request Errors
- DUT Error
- `E-CM-00058` [The user failed to change the device permission from the administrative interface.]

5. **Wait for the EGM to enter the SYNC state.**
{05992650-36f6-48da-00d1-23ccce13e6bb}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Errors

- Expected Request Errors
- `E-CM-00053` [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]

6. **Enable the communications device and verify that the events are generated by the EGM.**
{06e9d8e6-bcd3-4ce7-80b4-9c4e7d84041c}

Command

- `communications.setCommsState`

Events

(Time out: ${event.timeOut})

- `G2S_CME101` [Comms Not Established]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME001` [Comms Disabled by EGM]
- `G2S_CME002` [Comms Enabled by EGM]
- `G2S_CME100` [Comms Established]
- `G2S_CME004` [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

7. **If the CVT was originally a guest of the cabinet device.**

    a. **Instruct the user to add the CVT as a guest of the cabinet device.**
    {06a8e3ec-dba4-41f7-00d4-23dd628fd3b6}

    Commands
    - `communications.commsClosing`
    - `communications.commsOnLine`
    - `communications.commsDisabled`
    - `communications.setCommsState`

    Actions
    - Instruct the user to 'Please add the CVT as a guest of the cabinet device using the administrative interface.'.

    Events
    (Time out: ${event.timeOut})
    - `G2S_CME001` [Comms Disabled by EGM]
    - `G2S_CME002` [Comms Enabled by EGM]
    - `G2S_CME100` [Comms Established]
    - `G2S_CME004` [Comms Enabled by Host]

    Errors
    - Expected Request Errors
    - Send Request Errors
    - Event Checking Errors
    - DUT Error
    - `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]
    - `E-CM-00053` [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]
    - `E-CM-00058` [The user failed to change the device permission from the administrative interface.]

8. **If the CVT was not originally a guest of the cabinet device.**

    a. **Instruct the user to remove the CVT as a guest of the cabinet device.**
    {07e4ed26-3087-4a2c-8068-5307c0b38933}

    Commands
    - `communications.commsClosing`
    - `communications.commsOnLine`
    - `communications.commsDisabled`
    - `communications.setCommsState`

Actions

- Instruct the user to 'Please remove the CVT as a guest of the cabinet device using the administrative interface.'.

Events

(Time out: ${event.timeOut})

- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Expected Request Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]
- E-CM-00053 [CommsOnLine MUST have deviceChanged attribute set to true if owner, configurator, or guest permissions have changed for the CVT.]
- E-CM-00058 [The user failed to change the device permission from the administrative interface.]

Test Case: CM-COR-00047

This case verifies that the EGM does not process invalid XML.

Objective

Verify that EGM does not transition to the sync state when an invalid setCommsState is sent to the EGM.

Requirements Under Test

- 1.25.10
- 1.25.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a `setCommsState` command with an invalid `enable` attribute value.**
   {019ee47c-8657-4eac-802a-2975e2f3fe8f}

   Command
   - `communications.setCommsState`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

2. **Verify that the EGM did not transaction to the SYNC state.**
   {028dafe3-ec54-404f-80f6-a56e8a143664}

   Command
   - `communications.getCommsStatus`

   Errors
   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

Test Case: CM-COR-00048

This case verifies that the EGM handles unknown error codes.

Objective

Verify that EGM does not transition to the closed state when an unknown error is returned for the `commsClosing` command.

Requirements Under Test

- 1.26.1
- 1.42.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a G2S_MSX003 error code to the next keepAlive command and then a RBG_MSX048 error code to the commsClosing command.**
   {019254fc-cde7-4787-00ed-3a651d34f86b}

   Commands
   - `communications.setKeepAlive`
   - `communications.keepAlive`
   - `communications.commsClosing`

   Errors
   - Expected Request Errors
   - Send Request Errors

2. **Verify that the EGM is still in the closing state.**
   {02af0640-5f86-4a7e-00d9-1c4d9b0f9afb}

   Command
   - `communications.getCommsStatus`

   Errors

- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

3. **Bring the EGM back to the online state.**
{0343e516-f94b-483d-009d-be6dde4d38d9}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`
- `communications.setCommsState`

Errors

- Expected Request Errors
- Send Request Errors

Test Case: CM-COR-00049

This case verifies that the EGM handles command ID out of order error conditions.

Objectives

- Verify that EGM does not process a `setCommsState` command that is out of command ID order.
- Verify that EGM also returns a `G2S_MSX007 Commands Not In commandId Order` message level error or generates `G2S_APE001 At Least One Syntax/Semantic Command Error` or `G2S_APE006 Incorrect Use Of Response` event when a keepAliveAck is sent out of command ID order.

Requirements Under Test

- 1.30.5
- 1.30.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the EGM does not process a `setCommsState` command that has a command ID that is out of order.**
   {018af445-6cac-4bda-80dd-73ce0a3199a2}

   Command
   - `communications.setCommsState`

     Expect **G2S_APX013** or **G2S_MSX007**

   Errors
   - Send Request Errors
   - `E-CM-00054` [EGM MUST report a command ID out of order error.]

2. **Verify that the EGM reports a command ID out of order error when a keepAliveAck response is sent out of order.**
   {0206a1a6-38f0-45ab-00be-f9159d7f81a9}

Commands

- `communications.setKeepAlive`
- `communications.keepAlive`

Event

(Time out: 10000)

- `G2S_APE001` [At Least One Syntax/Semantic Command Error] or `G2S_APE006` [Incorrect Use Of Response]

Errors

- Expected Request Errors
- Send Request Errors
- Event Checking Errors
- `E-CM-00054` [EGM MUST report a command ID out of order error.]

Test Case: CM-COR-00050

This case verifies that the EGM generates G2S_APE003 or responds with the application level error G2S_APX014 when it receives an unknown namespace in a response or notification.

Objectives

- Verify that the EGM generates G2S_APE003 event or sends error G2S_APX014 for a response with an unknown namespace.
- Verify that the EGM generates G2S_APE003 event or sends error G2S_APX014 for a notification with an unknown namespace.

Requirements Under Test

- 1.41.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the EGM generates G2S_APE003 event or sends error G2S_APX014 for a response with an unknown namespace.**
   {0119a175-9658-42aa-0008-5b014a926145}

   Commands
   - `communications.setKeepAlive`
   - `communications.keepAlive`

   Event
   (Time out: 10000)
   - `G2S_APE003` [Unknown XML Component Encountered]

   Errors
   - Expected Request Errors
   - Send Request Errors
   - Event Checking Errors
   - `E-CM-00055` [EGM MUST report an unknown namespace at the class level error.]

2. **Disable the sending of keepAlive commands.**
{02e9e82f-15d4-429b-008e-b39a94cbe241}

Command
- `communications.setKeepAlive`

Error
- Send Request Errors

3. **Verify that the EGM generate G2S_APE003 event or sends error G2S_APX014 for a notification with an unknown namespace.**
{03645239-5f1b-4eca-00f1-88c84cea1b52}

Event
(Time out: 5000)
- `G2S_APE003` [Unknown XML Component Encountered]

Errors
- Send Request Errors
- Event Checking Errors
- `E-CM-00055` [EGM MUST report an unknown namespace at the class level error.]

Test Case: CM-COR-00051

Deleted

This test case was deleted and replaced by test case CM-COR-00093.

Test Case: CM-COR-00052

This case verifies that when the noResponseTimer is set to zero that the EGM does not transition to closing state when the host stops responding.

Objective

Verify that if the no-response timer is set to zero that the EGM does not transitions to G2S_ hostUnreachable when the host stops sending g2sAcks.

Requirements Under Test

- 2.10.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner

Required Configurations

Option	Value
commsProfile.noResponseTimer	0

Test Procedure

1. **Set the keepAlive frequency to five seconds and disable sending keepAliveAck responses to keepAlive commands.**
   {015a2aa3-2ac0-43d9-0043-bd2ed9dc639a}

   Command
   - `communications.setKeepAlive`

   Errors
   - Send Request Errors
   - Event Not Expected Error

2. **Verify that the communications device is still in the G2S_onLine state.**
   {02030c22-93c3-4afe-007b-73d7a9f7f457}

   Command
   - `communications.getCommsStatus`

Errors

- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00053

This case verifies that the EGM is disabled when a requiredForPlay communications device is disabled.

Objective

Verify that the EGM executes its disable logic when a requiredForPlay communications device is host disabled.

Requirements Under Test

- 2.21.3
- 2.23.2
- 3.3.3
- 3.3.16
- 3.3.18
- 3.5.16

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
commsProfile.requiredForPlay	true

Test Procedure

1. **Determine the current values for cabinetStatus.deviceClass and cabinetStatus.deviceId.**
   {01712b37-c1fc-4aa2-0008-56b602a5284f}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

2. **Disable the communications device.**
   {020eeabc-011d-4dc7-0080-47984b6954d9}

Command

- `communications.setCommsState`

Errors

- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

3. **Verify that the EGM is host disabled by the communications device.**
{036fd5a1-78dd-4c23-0014-02c487a67160}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

4. **Verify that the disabled text is displayed.**
{04ec223d-07a2-4760-0043-44f231a58a19}

Action

- Verify that text 'CM-COR-00053 Step #2' appears onscreen.

Errors

- DUT Error
- `E-CB-00014` [Required text is not displayed.]

5. **Disable the communications device with an empty disable text.**
{0555c448-5daf-446b-0090-80d3f44e5838}

Command

- `communications.setCommsState`

Actions

- Verify that text 'CM-COR-00053 Step #2' does NOT appear onscreen.

Errors

- Send Request Errors
- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]

6. **Enable the communications device.**
{061f8d98-9d81-4e3b-8097-508cd8ded3ba}

Command

- `communications.setCommsState`

Events

(Time out: ${event.timeOut})

- `G2S_CME003` [Comms Disabled by Host]
- `G2S_CBE204` [Host Command Disabled EGM]
- `G2S_CME004` [Comms Enabled by Host]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00054

This case verifies that the EGM transitions the egmState properly for a communications device that is required for play.

Objectives

- Verify that the EGM evaluates all devices before setting the egmState.
- Verify that the EGM follows the order or precedence for setting the egmState.

Requirements Under Test

- 3.3.7
- 3.3.8
- 3.3.10
- 3.3.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner
G2S_cabinet	owner

Required Configurations

Option	Value
commsProfile.requiredForPlay	true

Test Procedure

1. **Determine the current values for cabinetStatus.deviceClass and cabinetStatus.deviceId.**
   {01525491-c6a2-4817-0082-2f036df2ba39}

   Command
   - `cabinet.getCabinetStatus`

   Error
   - Send Request Errors

2. **Lock the cabinet device.**
   {029ba54b-7365-40a9-8052-93ce72f450b3}

Command
- `cabinet.setCabinetLockOut`

Events
(Time out: ${event.timeOut})
- `G2S_CBE009` [Host Locked Cabinet]
- `G2S_CBE211` [Host Action Locked EGM]

Errors
- Send Request Errors
- Event Checking Errors

3. **Disable the communications device.**
{03e18243-779c-4905-80ed-248eb6c85164}

Command
- `communications.setCommsState`

Errors
- Send Request Errors
- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

4. **Verify that the egmState is still G2S_hostLocked.**
{042bc7ce-f45c-450d-80b3-fedb2e8c1b6c}

Command
- `cabinet.getCabinetStatus`

Errors
- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

5. **Instruct user to open the operator menu and verify egmState is G2S_operatorMode.**
{05f2b302-cd8d-46e8-0070-87a7f5b8b5b0}

Command
- `cabinet.getCabinetStatus`

Actions
- Instruct the user to 'Open the operator menu'.

Errors

- Send Request Errors
- DUT Error
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

6. **Unlock the cabinet device and verify that egmState is still G2S_operatorMode.**
{061d399a-7dbf-4000-009a-531639cd94ce}

Command
- `cabinet.setCabinetLockOut`

Errors
- Send Request Errors
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

7. **Instruct user to close the operator menu and verify egmState is G2S_hostDisabled.**
{07905e31-4578-4a8b-8052-2570008894e6}

Command
- `cabinet.getCabinetStatus`

Actions
- Instruct the user to 'Close the operator menu'.

Errors
- Send Request Errors
- DUT Error
- `E-CB-00018` [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

8. **Enable the communications device.**
{083e335f-b488-4b60-0090-c2de07257cd9}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})
- `G2S_CME003` [Comms Disabled by Host]
- `G2S_CBE206` [Operator Menu Activated]
- `G2S_CBE010` [Host Unlocked Cabinet]

- G2S_CBE204 [Host Command Disabled EGM]
- G2S_CME004 [Comms Enabled by Host]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

9. **Disable the communications device.**
{101311f6-ac7b-419a-004d-e58be9b2feb2}

Command

- communications.setCommsState

Errors

- Send Request Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

10. **Lock the cabinet device and verify egmState is still G2S_hostDisabled.**
{119f256c-b9a5-4dcd-00e3-555cabcc4669}

Command

- cabinet.setCabinetLockOut

Errors

- Send Request Errors
- E-CB-00018 [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

11. **Enable the communications device.**
{12300c9b-3945-4dee-80ad-8ff603dc2ea1}

Command

- communications.setCommsState

Events

(Time out: ${event.timeOut})

- G2S_CME003 [Comms Disabled by Host]
- G2S_CME100 [Comms Established]
- G2S_CBE204 [Host Command Disabled EGM]
- G2S_CBE009 [Host Locked Cabinet]

- G2S_CME004 [Comms Enabled by Host]
- G2S_CBE211 [Host Action Locked EGM]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

12. **Unlock the cabinet device.**
{13f1a183-ebb4-40c0-0032-42718de5852e}

Command

- cabinet.setCabinetLockOut

Events

(Time out: ${event.timeOut})

- G2S_CBE010 [Host Unlocked Cabinet]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- E-CB-00018 [The EGM state of ${actual} is wrong it MUST be ${expected}.]
- E-CB-00020 [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: CM-COR-00055

This case verifies that the host sends a valid getCommsProfile command.

Objective

Verify the getCommsProfile command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct user to send getCommsProfile command from the host.**
   {011d8eaf-330b-4e6d-a2e9-734a2bc4a306}

   #### Command

   - `communications.getCommsProfile`

   #### Actions

   - Instruct the user to 'Send getCommsProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CM-COR-00056

This case verifies that the host sends a valid getCommsStatus command.

Objective

Verify the getCommsStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct user to send getCommsStatus command from the host.**
   {01550366-6b4a-4036-955c-882483cde536}

   #### Command

   - `communications.getCommsStatus`

   #### Actions

   - Instruct the user to 'Send getCommsStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CM-COR-00057

This case verifies that the host sends a valid getDescriptor command.

Objective

Verify the getDescriptor command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct user to send getDescriptor command from the host.**
   {014c2f38-bee8-4a47-b220-45025a556076}

   #### Command

   - `communications.getDescriptor`

   #### Actions

   - Instruct the user to 'Send getDescriptor to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CM-COR-00058

This case verifies that the host sends a valid setKeepAlive command.

Objective

Verify the setKeepAlive command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct user to send setKeepAlive command from the host.**
   {0164fe75-c09d-46d5-beaa-2b34037f5aca}

   #### Command

   - `communications.setKeepAlive`

   #### Actions

   - Instruct the user to 'Send setKeepAlive to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CM-COR-00059

This case verifies that the host sends a valid keepAlive command.

Objective

Verify the keepAlive command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct user to send keepAlive command from the host.**
   {01e8289d-f618-4f41-803c-0f8f4c3f79cd}

   #### Command

   - `communications.keepAlive`

   #### Actions

   - Instruct the user to 'Send keepAlive to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: CM-COR-00060

This case verifies that the host can process UTF-8 and UTF-16 messages.

Objective

Verify that the host can process UTF-8 and UTF-16 messages.

Requirements Under Test

- 1.1.1
- 1.1.2
- 1.2.1
- 1.2.4
- 1.2.8
- 1.2.9
- 1.27.33

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a keepAlive command to the host using UTF-8.**
   {01fa4351-7526-4b68-ad5f-02786d21da69}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

2. **Send a keepAlive command to the host using UTF-16.**
   {02246955-ddd6-49e2-a3eb-9f17937b297f}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

Test Case: CM-COR-00061

This case continuously tests that the EGM ID and host ID values are correct.

Objectives

- Continuously verify that the egmId in the SOAP call matches the egmId in the g2sBody element, and that they both match the egmId of the EGM involved in the test.
- Continuously verify that the hostId in the SOAP call matches the hostId in the g2sBody element, and that they both match the hostId of the host involved in the test.

Requirements Under Test

- 1.3.1
- 1.25.3
- 2.24.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the egmId and hostId from the SOAP call matches the egmId and hostId in the g2sBody element.**
   {01f741b8-3908-43e0-b22c-75219988b648}

Test Case: CM-COR-00062

This case continuously tests that commands from the host have the correct session type and that all requests from the EGM have host responses.

Objectives

- Continuously verify that commands from the host have the correct session type.
- Continuously verify that all requests from the EGM have host responses.

Requirements Under Test

- 1.4.1
- 1.4.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that commands from the host have the correct session type.**
   {019dd75f-d589-44ab-8178-2090c4bd4ece}
2. **Verify that all EGM requests have host responses.**
   {02dbf91b-1ade-46ab-8a8f-ee7b2b6951a4}

Test Case: CM-COR-00063

This case verifies that the host does not process commands with the wrong session type.

Objectives

- Verify that the host is still working after sending a cabinetStatus as a request.
- Verify that the host is still working after sending a keepAliveAck response to a setCommsState request.
- Verify that the host is still working after sending a commsDisabled as a response.
- Verify that the host is still working after sending a commsDisabled as a notification.

Requirements Under Test

- 1.4.4
- 1.4.5
- 1.4.6
- 1.4.7
- 2.6.1
- 2.28.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a commsStatus as a request and verify G2S_APX008 Command Not Supported error code.**
   {011a6f5e-4e12-4d29-8986-3afa9940997f}

   Command
   - `communications.commsStatus`

     Expect **G2S_APX008**

   Error
   - Send Request Errors

2. **Set the response manager to respond to setCommsState with keepAliveAck.**
{025a7fd8-1041-47ea-8992-020a89704e82}

3. **Reset communications to the host and wait for setCommsState command from the host.**
{0379110f-cf82-4e4b-9a44-5486527355a9}

   Command
   - `communications.setCommsState`

   Error
   - Expected Request Errors

4. **Send a commsDisabled to the host to verify that the host responds with a commsDisabledAck.**
{045e6d9e-d7b1-4c20-bba6-579797e0e076}

   Command
   - `communications.commsDisabled`

   Error
   - Send Request Errors

5. **Clear the response manager and wait for setCommsState from the host.**
{05caad1d-67b4-4c10-813e-e954e455bf4e}

   Command
   - `communications.setCommsState`

   Error
   - Expected Request Errors

6. **Send a commsDisabled as a response to the host and verify that no response is sent from the host.**
{0696acee-cf3f-4c3f-98a6-69dc4d81e4be}

   Command
   - `communications.commsDisabled`

     Expect **G2S_APX009**

   Errors
   - Send Request Errors
   - `E-XX-00019` [An unexpected response of ${response} was received.]

7. **Send a commsDisabled as a notification to the host and verify that no response is sent from the host.**
{078e8bc6-450e-428a-84ff-d09f75e9231c}

Command

- `communications.commsDisabled`

Errors

- Send Request Errors
- `E-XX-00019` [An unexpected response of ${response} was received.]

Test Case: CM-COR-00064

This case continuously tests that commands are valid from the host.

Objectives

- Continuously verify that the host does not send owner commands to a guest device.
- Continuously verify that g2sAcks from the host have an error code of G2S_none or start with G2S_MSX.
- Continuously verify that errorText must be empty unless an error is being reported.
- Continuously verify that device identifiers match between EGM requests and host responses.

Requirements Under Test

- 1.8.6
- 1.26.15
- 1.26.16
- 1.27.1
- 1.27.3

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the host does not send owner commands to a guest device.**
   {01f85f6e-24b0-4845-8808-16b1b8b440b4}
2. **Verify that g2sAcks from the host have an error code of G2S_none or start with G2S_MSX.**
   {023bbb2c-e9d5-4591-aa56-2c4b79125b6e}
3. **Verify that errorText must be empty unless an error is being reported.**
   {03207e50-d03e-4722-8761-7c768b31f72e}
4. **Verify that host commands are syntactically correct.**
   {046200e6-971f-44db-b976-c050762d99a0}

5. **Verify that device identifiers match between EGM requests and host responses.**
{05af51f4-a51b-4995-b189-fd5c49f4c219}

Test Case: CM-COR-00065

This case verifies error handling in the communications class.

Objectives

- Verify that the host responds with an error code for devices it does not own.
- Verify that the host responds appropriately to a keepAlive when the EGM is in the sync state.
- Verify that the host responds to commsOnLine command even when the EGM is in the sync state.

Requirements Under Test

- 1.8.7
- 1.9.9
- 1.44.2
- 2.6.1
- 2.24.3
- 2.28.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a commsDisabled for a communications device that the host does not own.**
   {013fed95-aa76-48cf-ae57-bff99a3a2c14}

   Command
   - `communications.commsDisabled`

     Expect **G2S_APX003** or **G2S_APX999**

   Error
   - Send Request Errors

2. **Set the response manager to ignore setCommsState commands.**
   {029dea00-eaba-4eee-8036-f117eb1701ab}

3. **Reset communications to the host.**
{03f06920-c6b5-4331-9be1-6bc5066a986b}

Command
- `communications.setCommsState`

Error
- Expected Request Errors

4. **Send a keepAlive to the host to verify that the host sends G2S_APX016 Command Not Processed, Device Disabled, G2S_APX019 Command Not Processed, communications Device In sync State, or G2S_APX999 Undefined Error.**
{043f8b35-8a79-446c-8115-149055c1e866}

Command
- `communications.keepAlive`
  Expect **G2S_APX016**, **G2S_APX019** or **G2S_APX999**

Error
- Send Request Errors

5. **Send a commsOnline to the host while comms are disabled and verify the host responds.**
{05ca795a-c8fe-4ff3-8432-5e0f8a60e6fd}

Command
- `communications.commsOnLine`
  Expect **G2S_APX016**, **G2S_APX019**, **G2S_APX999**, **G2S_MSX003** or **Normal Response**

Error
- Send Request Errors

6. **Clear the response manager and wait for setCommsState from the host.**
{0671fafa-4f03-4e0d-a9d1-9527b5e05bd0}

Command
- `communications.setCommsState`

Error
- Expected Request Errors

Test Case: CM-COR-00066

This case verifies that the host ID is configurable.

Objectives

- Verify that the host ID can be configured to the largest legal value of 2147483647.
- Verify that the host can respond to commands where the device ID is the largest legal value.

Requirements Under Test

- 1.14.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Close communications to the host.**
   {01f646cc-6aed-4a3e-843f-d4ffa900b5ab}

2. **Instruct user to set the host ID to 2147483647.**
   {021916c2-ff39-421e-b5aa-c15411d9d830}

   Action
   - Instruct the user to 'Set the host ID to 2147483647. Click the True button when the host is ready.'.

   Error
   - DUT Error

3. **Send a commsOnLine to host ID 2147483647.**
   {03119881-5909-4408-8e84-a2d916980f00}

   Command
   - `communications.commsOnLine`
     Expect **G2S_MSX003**

   Error

- Send Request Errors

4. **Send a second commsOnLine to host ID 2147483647 and verify commsOnLineAck response.**
{04a49e71-4884-4258-be27-80f95aed7733}

   Command
   - `communications.commsOnLine`

   Error
   - Send Request Errors

5. **Instruct user to reset host ID to 17.**
{05259758-fd3b-4364-8e51-ff531e1af025}

   Action
   - Instruct the user to 'Set the host ID to 17. Click the True button when the host is ready.'.

   Error
   - DUT Error

6. **Start communications with host ID 17.**
{06b4e6dd-b272-496f-97f9-c157de22dc1d}

   Command
   - `communications.setCommsState`

   Error
   - Expected Request Errors

Test Case: CM-COR-00067

This case verifies that the host verifies host ID and EGM ID.

Objectives

- Verify that the host validates the host ID in the SOAP call and g2sBody.
- Verify that the host validates the EGM ID in the SOAP call and g2sBody.

Requirements Under Test

- 1.14.11
- 1.25.5
- 1.25.7
- 1.26.6
- 1.26.7
- 1.26.8
- 1.26.13
- 1.26.14

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Close communications to the host.**
   {018c1a33-d585-42a1-9bf4-2cab86ad2d5f}

2. **Send a keepAlive with with the wrong hostId in the WSDL call.**
   {0214ff84-7b58-47f7-96dd-95ad1288f771}

   Command
   - `communications.keepAlive`

     Expect **G2S_MSX001** or **G2S_MSX003**

   Error
   - Send Request Errors

3. **Send a keepAlive with the wrong hostId in the g2sBody.**
{03e4e2dc-77ce-4d5c-a79d-0da798a9d2f5}

Command

- `communications.keepAlive`

  Expect **G2S_MSX001** or **G2S_MSX003**

Error

- Send Request Errors

4. **Send a keepAlive with the wrong egmId in the SOAP WSDL call.**
{0412864f-428b-400b-bb37-f9a747dd1949}

Command

- `communications.keepAlive`

  Expect **G2S_MSX002** or **G2S_MSX003**

Error

- Send Request Errors

5. **Send a keepAlive with the wrong egmId in the g2sBody.**
{05746201-48a2-42b0-a265-1da5334a6ded}

Command

- `communications.keepAlive`

  Expect **G2S_MSX002** or **G2S_MSX003**

Error

- Send Request Errors

6. **Start communications with host ID 17.**
{06087e16-351e-4f14-865f-d2d1134e1329}

Command

- `communications.setCommsState`

Error

- Expected Request Errors

Test Case: CM-COR-00068

This case verifies that the host can process time stamps with 15 digits of precision.

Objective

Verify that the host can process keepAlive commands with 15 digits of precision in the dateTime attribute.

Requirements Under Test

- 1.17.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a keepAlive with 15 digits of precision in the dateTime attribute.**
   {018c64bf-2c19-4d4f-abc2-4c425bc0292d}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

Test Case: CM-COR-00069

This case verifies that the host updates the egmLocation per commsOnLine.

Objectives

- Verify that the host updates the egmLocation per commsOnLine.
- Verify that the host keeps an EGM in the offline state until the g2sAck for a commsOnLineAck is received.

Requirements Under Test

- 1.25.8
- 1.26.9
- 1.26.11
- 1.42.2
- 2.26.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Close communications to host.**
   {01a618f6-b225-4e5a-990d-3a17ad684a8c}

2. **Send commsOnLine with wrong EGM location.**
   {02d6cd8b-af06-457c-8071-a607c4fda2e8}

   Command
   - `communications.commsOnLine`

     Expect **G2S_MSX003**

   Error
   - Send Request Errors

3. **Send commsOnLine the second time to guarantee G2S_MSX003 Communications Not Online error.**
   {03a16804-d88a-4f24-ae1c-4968066e6dff}

Command

- `communications.commsOnLine`

  Expect **G2S_MSX003**

Error

- Send Request Errors

4. **Set the response manager to ignore commsOnLineAck commands.**
   {04f7443e-8cc3-4f4c-91fe-846f469ca25c}

5. **Send commsOnLine with the correct EGM location and verify G2S_MSX003 error code.**
   {053d2a33-9554-4a01-b606-50378a8ab0ab}

   Command

   - `communications.commsOnLine`

     Expect **G2S_MSX003**

   Error

   - Send Request Errors

6. **Send commsDisabled and verify G2S_MSX003 error code.**
   {061d36ba-0cc9-4f81-9135-994bf8206ef8}

   Command

   - `communications.commsDisabled`

     Expect **G2S_MSX003**

   Error

   - Send Request Errors

7. **Set the response manager to response with 'CVT_error!' to a commsOnLineAck.**
   {07db5d85-e204-4370-87bc-369c49f2fb28}

8. **Sends a commsOnLine command to the host.**
   {0846e839-c36b-4a7d-9794-f356a173a446}

   Command

   - `communications.commsOnLine`

   Error

   - Send Request Errors

9. **Reset the response manager.**
   {090d03ab-31dd-49fa-9100-cdf38b592577}

10. **Start communications to the host and wait for setCommsState.**
    {1072f21a-4e57-4027-addc-8bef4b46d60d}

Command

- `communications.setCommsState`

Error

- Expected Request Errors

Test Case: CM-COR-00070

This case verifies that the host validates communications commands against the G2S schema.

Objective

Verify that the host validates communications commands.

Requirements Under Test

- 1.25.10
- 1.25.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send the keepAlive command with an additional g2s:testing attribute.**
   {01e7071d-961a-4a49-bac3-ee34cc2c9351}

   #### Command

   - `communications.keepAlive`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   #### Error

   - Send Request Errors

Test Case: CM-COR-00071

This case tests that the host does not process commands that have a message level error.

Objective

Verify that the host does not process commsClosingAck with a message level error.

Requirements Under Test

- 1.26.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Set the response manager to send an unknown error code of '000_err !%' to a commsClosingAck command.**
   {011515e0-ecfc-44c5-b3d4-677e58d488c4}

2. **Send the commsClosing command.**
   {026c71df-6f9b-42c1-a08c-3a00a7739ffa}

   Command
   - `communications.commsClosing`

   Error
   - Send Request Errors

3. **Reset the response manager.**
   {03768d36-27e8-4203-ae2d-366fb4e71d0e}

4. **Send the commsClosing the second time and verify the host sends commsClosingAck.**
   {043045a6-2621-4d0b-9d3c-4db150a6f643}

   Command
   - `communications.commsClosing`

   Error
   - Send Request Errors

5. **Start communications to the host.**
{051ff4e7-97e7-4aa0-bf93-a937ef48ea14}

Command

- `communications.setCommsState`

Error

- Expected Request Errors

Test Case: CM-COR-00072

This case continuously tests semantic rules on the G2S class-level element.

Objectives

- Continuously verify that command IDs are increasing by one.
- Continuously verify that notifications do not have a response.
- Continuously verify that all requests have a response.
- Continuously verify that sessionMore is false.
- Continuously verify that if a command level element is present that errorCode is G2S_none.
- Continuously verify that if errorCode is G2S_none that errorText is empty.
- Continuously verify that if errorText is not empty that errorCode is not G2S_none.

Requirements Under Test

- 1.27.7
- 1.27.8
- 1.27.9
- 1.27.26
- 1.27.27
- 1.27.28
- 1.27.29
- 2.2.2
- 2.2.5
- 2.2.8

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that command IDs are increasing by one.**
   {012bd698-3bc6-4cae-9b14-c811e44cd2d5}

2. **Verify that notifications do not have a response.**
   {028b53d9-bc4c-4230-bec4-17852126b64f}
3. **Verify verify that all requests have a response.**
   {03a2e9c2-63a2-4cad-9a1d-105a8b325f3c}
4. **Verify that sessionMore is false.**
   {0413c8b6-6222-4110-ab2a-c390a614e03d}
5. **Verify that if a command level element is present that errorCode is G2S_none.**
   {0514b982-9b83-4c57-a242-984f57e6233e}
6. **Verify that if errorCode is G2S_none that errorText is empty.**
   {061d3eee-e54e-42be-b795-025e7ad743c1}

Test Case: CM-COR-00073

This case tests that the host processes duplicate communications commands.

Objective

Verify that the host processes duplicate commands.

Requirements Under Test

- 1.28.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send five commsDisabled commands in one G2S message. Verify the host sends five responses.**
   {019e42c4-0bcf-40eb-bf69-23a1e2a5125a}

   Command
   - `communications.commsDisabled`

   Error
   - Send Request Errors

Test Case: CM-COR-00074

This case verifies that the host can process commands that are four MiB in size.

Objective

Verify that the host processes four MiB sized commands.

Requirements Under Test

- 1.29.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a four MiB keepAlive command to the host.**
   {01ff0667-0617-4af6-82aa-abfaea644aee}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

Test Case: CM-COR-00075

This case verifies that the host checks for command IDs out of order.

Objective

Verify that the host sends G2S_MSX007 Commands Not In commandId Order message level error or G2S_APX013 Command Not In commandId Order error code when command IDs are out of order.

Requirements Under Test

- 1.30.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a keepAlive command to the host.**
   {0108685c-779e-4132-807c-232cb570926a}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

2. **Send a keepAlive command to the host with a command ID that is out of order.**
   {02dcd85b-d883-4b9e-80c5-92cd19d71cdb}

   Command
   - `communications.keepAlive`

     Expect **G2S_APX013** or **G2S_MSX007**

   Error
   - Send Request Errors

3. **Send a keepAlive command to the host.**
   {03130b79-b397-4f0b-8020-ab4d1c934dfc}

Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: CM-COR-00076

This case verifies that the host validate unknown classes and commands.

Objectives

- Verify that the host sends G2S_MSX004 Incomplete/Malformed XML, G2S_MSX005 Invalid Data Type Encountered or G2S_APX004 Incomplete/Malformed XML error for schema invalid communications command.
- Verify that the host sends G2S_APX007 Class Not Supported or G2S_APX014 Unknown Class Encountered error code for an unknown class.
- Verify that if the host sends a G2S_APX014 error that the device is the communications device, sessionId is zero and sessionRetry is false.
- Verify that the host sends G2S_APX008 Command Not Supported or G2S_AXP015 Unknown Command Encountered error for an unknown command.
- Verify that if the host sends G2S_AXP015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.41.2
- 1.41.6
- 1.41.13
- 1.41.14
- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a schema invalid commsDisabled and verify G2S_MSX004, G2S_MSX005 or G2S_APX004 error code.**
   {0112e200-8b8a-4588-a18f-6e4c8ad3f84d}

   Command

- `communications.commsDisabled`
  Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

Error

- Send Request Errors

2. **Set the response manager to send an unknown response to setCommsState command.**
{02d0f1fb-b783-4371-9b71-46fc655a3177}

3. **Reset communications to the host and wait for the setCommsState command.**
{034cc532-f589-483e-b143-f226c5616c8a}

Command

- `communications.setCommsState`

Errors

- Expected Request Errors
- `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

4. **Reset the response manager.**
{0418a56a-d211-4ce3-a12a-086296bf6f66}

5. **Send a command from an unknown class and verify G2S_APX007 or G2S_APX014 error.**
{051b86ab-b880-4b46-8982-d6319e56f198}

Command

- `cvt.testing`
  Expect **G2S_APX007** or **G2S_APX014**

Errors

- Send Request Errors
- `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

6. **Send a unknown command and verify G2S_APX008 or G2S_APX015 error.**
{06ad8144-7079-420b-adb7-049f78591daf}

Command

- `communications.testing`
  Expect **G2S_APX008** or **G2S_APX015**

Errors

- Send Request Errors
- `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: CM-COR-00078

This case tests that the host can handle a delayed g2sAck in the communications class.

Objective

Verify that the host is still working after delaying the g2sAck in the communications request.

Requirements Under Test

- 1.30.7
- 2.6.1
- 2.28.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Set the response manager to send an delay the g2sAck for setCommsState command.**
   {01233179-dacb-4cdd-b6b6-faf636b001aa}

2. **Reset communications to the host and wait for a setCommsState command.**
   {02e0d41f-3af9-410d-b7a5-f9f50ea60a9d}

   Command
   - `communications.setCommsState`

   Error
   - Expected Request Errors

3. **Reset the response manager.**
   {03996169-a74a-433c-9a09-abfec355107f}

4. **Send keepAlive command.**
   {04300018-ab9e-4307-9c52-b34bb3477b02}

   Command
   - `communications.keepAlive`

   Error

- Send Request Errors

Test Case: CM-COR-00079

This case tests that the EGM persists the device structure.

Objective

Verify that EGM does not change the device structure across EGM power-cycles.

Requirements Under Test

- 1.5.1
- 1.5.4

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Get the descriptor list.**
   {012b93e4-8ebd-4bcf-a046-2a969b415904}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Instruct user to power cycle EGM and wait for EGM to reconnect.**
   {0213876b-a4b9-4d97-94cf-3ddfb1a1b8ae}

   Command
   - `communications.commsOnLine`

   Actions
   - Instruct the user to 'Power cycle the EGM.'.

   Errors
   - Expected Request Errors
   - DUT Error
   - `E-CM-00057` [The user failed to power cycle the EGM.]

3. **Wait for EGM to retry commsOnLine.**
{03f9aa7e-af0d-42b0-b355-cb9dd9925515}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Error
   - Expected Request Errors

4. **Assert that the device structure has not changed after the EGM power cycle.**
{041f8b14-2587-4abc-94a5-f3a674aa8f09}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

5. **Enable the communications device.**
{051dfe38-aa76-4b51-b3a8-85fd62633f28}

   Command
   - `communications.setCommsState`

   Error
   - Send Request Errors

Test Case: CM-COR-00080

This case verifies that the EGM can process time stamps with 15 digits of precision.

Objective

Verify that the EGM can process keepAlive commands with 15 digits of precision in the `dateTime` attribute.

Requirements Under Test

- 1.17.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Send a keepAlive with 15 digits of precision in the dateTime attribute.**
   {01329bbb-ce4a-4aa0-8e98-db4ce2f1949c}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

Test Case: CM-COR-00081

This case continuously tests that transaction IDs are increasing.

Objectives

- That transactions IDs in EGM commands that set transaction IDs are always increasing.
- That transactions IDs for event reports that report the start of a transaction are always increasing.

Requirements Under Test

- 1.19.6

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that transaction IDs in commands that set the transaction ID are always increasing.**
   {017b2803-514c-4007-b635-86e679e73e2c}

   Error
   - `E-CM-00031` [Transaction ID ${transactionId} MUST be greater than ${lastTransactionId}.]

2. **Assert that transaction IDs for event reports that start a transaction are always increasing.**
   {0212f6fe-a038-4db9-a612-9ece6b43d3a7}

   Error
   - `E-CM-00031` [Transaction ID ${transactionId} MUST be greater than ${lastTransactionId}.]

Test Case: CM-COR-00082

This case verifies that the EGM populates the `commsOnLine.deviceChanged` attribute after the device structure changes.

Objectives

- Verify that the EGM sets `commsOnLine.deviceChanged` attribute to true if a new host is added.
- Verify that the EGM creates a communications device when a host is registered.
- Verify that the EGM removes the communication device when a host is unregistered.

Requirements Under Test

- 1.39.3
- 2.1.2
- 2.1.3

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Get the list of active devices.**
   {01f6e1d3-aa56-4732-92dc-daed26bf864b}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Instruct the user to register a new host.**
   {0293fbdf-1820-4f0c-97b9-55c1b5f348a7}

   Command
   - `communications.commsClosing`

Actions

- Instruct the user to 'Please register a new host.'.

Errors

- Expected Request Errors
- DUT Error
- `E-CM-00058` [The user failed to change the device permission from the administrative interface.]

3. **Wait for EGM to reconnect.**
{03e6d7be-f084-4b61-8f9d-e0397121d447}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Errors

- Expected Request Errors
- `E-CM-00061` [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

4. **Get descriptor list to verify a new communications device was created.**
{04f6913c-77c2-4b92-8f22-c6cde772a2d2}

Command

- `communications.getDescriptor`

Errors

- Send Request Errors
- `E-CM-00062` [Registered hosts MUST have a communications device.]

5. **If a new host was registered.**

   a. **Instruct the user to un-register the newly registered host.**
   {057e5b19-5212-41d2-b187-eb3cb5a9385e}

   Command

   - `communications.commsClosing`

   Actions

   - Instruct the user to 'Please unregister host ${g_newDevice.deviceId}.'.

   Errors

   - Expected Request Errors
   - DUT Error

- E-CM-00058 [The user failed to change the device permission from the administrative interface.]

b. **Wait for EGM to reconnect.**
{06a4b13c-e4ba-4288-bff4-00233dacce75}

Commands
- communications.commsOnLine
- communications.commsDisabled

Errors
- Expected Request Errors
- E-CM-00061 [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

c. **Get descriptor list to verify the communications device was removed.**
{0724c9ac-ca05-40cf-abe8-8714f401e7cc}

Command
- communications.getDescriptor

Errors
- Send Request Errors
- E-CM-00063 [Unregistered hosts MUST NOT have a communications device.]

6. **Enable the communications device.**
{08172d48-7950-4280-a767-a00c850fe20d}

Command
- communications.setCommsState

Error
- Send Request Errors

Test Case: CM-COR-00083

This case verifies that the EGM transitions to the closing state when a device is activate or deactivated in the opening state.

Objectives

- Verify that the EGM transitions to the closing state when a device is deactivated in the opening state.
- Verify that the EGM transitions to the closing state when a device is activated in the opening state.

Requirements Under Test

- 2.5.6

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the current active devices.**
   {0163e601-b952-4795-8777-b4db12ffea92}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and leave the EGM in the opening state.**
   {02d2bdfc-41af-4a05-8b94-14d0616bfc3c}

   Commands
   - `communications.getCommsProfile`
   - `communications.commsClosing`
   - `communications.commsOnLine`

Errors

- Expected Request Errors
- Send Request Errors

3. **Instruct user to deactivate a device.**
{03a10dac-4770-4748-9bfc-f8382bf2760b}

Command

- `communications.commsClosing`

Actions

- Instruct the user to 'Please deactivate a device other than the communications device ${deviceUnderTest.deviceId}.'.

Errors

- Expected Request Errors
- DUT Error
- `E-CM-00064` [The user failed to deactivate a device.]

4. **Wait for the EGM to enter the SYNC state.**
{0407fef1-8f33-4e72-91dc-a91123e70409}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Errors

- Expected Request Errors
- `E-CM-00061` [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

5. **Determine which device was deactivated.**
{05d29300-afb1-47d9-9bdb-03074f1be277}

Command

- `communications.getDescriptor`

Errors

- Send Request Errors
- `E-CM-00064` [The user failed to deactivate a device.]

6. **Enable the communications channel and verify the proper events are generated.**
{06ed7089-9333-4848-b25d-34a113c5fa7d}

Command

- communications.setCommsState

Events
(Time out: ${event.timeOut})
- G2S_CME101 [Comms Not Established]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

7. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and leave the EGM in the opening state.**
{07fd5248-06f5-4dd4-a139-685496414403}

Commands
- communications.getCommsProfile
- communications.commsClosing
- communications.commsOnLine

Errors
- Expected Request Errors
- Send Request Errors
- E-CM-00066 [CommsOnLine MUST have deviceChanged attribute set to false if no device changes have occurred for the EGM.]

8. **If the user deactivated a device.**

   a. **Instruct user to active a device.**
   {08e1e162-fcd8-471d-be16-50ccc1eefefc}

   Command
   - communications.commsClosing

   Actions
   - Instruct the user to 'Please activate device ${g_devices[0].toDisplayable()}.'.

   Errors
   - Expected Request Errors
   - DUT Error

- E-CM-00065 [The user failed to activate a device.]

b. **Wait for the EGM to enter the SYNC state.**
{098c75f7-74af-483e-99af-1f23dc5cce0e}

Commands
- communications.commsOnLine
- communications.commsDisabled

Errors
- Expected Request Errors
- E-CM-00061 [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

9. **Enable the communications channel and verify the proper events are generated.**
{10c9951a-0cb6-4d31-bee1-d2db31975b1d}

Command
- communications.setCommsState

Events
(Time out: ${event.timeOut})
- G2S_CME101 [Comms Not Established]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00084

This case verifies that the EGM transitions to the closing state when a device is activate or deactivated in the sync state.

Objectives

- Verify that the EGM transitions to the closing state when a device is deactivated in the sync state.
- Verify that the EGM transitions to the closing state when a device is activated in the sync state.

Requirements Under Test

- 2.6.7

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the current active devices.**
   {01d38ab0-3fb5-4bd2-9c5e-8569c8a98cd0}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and leave the EGM in the opening state.**
   {022ca298-e983-43ad-8048-d6b0c176a7c4}

   Commands
   - `communications.getCommsProfile`
   - `communications.commsClosing`
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Errors

- Expected Request Errors
- Send Request Errors

3. **Instruct user to deactivate a device.**
{0354a527-3b05-4723-81ee-84091b3e49a3}

Command
- `communications.commsClosing`

Actions
- Instruct the user to 'Please deactivate a device other than the communications device ${deviceUnderTest.deviceId}.'.

Errors
- Expected Request Errors
- DUT Error
- `E-CM-00064` [The user failed to deactivate a device.]

4. **Wait for the EGM to enter the SYNC state.**
{04bc8415-586e-4676-a1d2-f22290ffc9b0}

Commands
- `communications.commsOnLine`
- `communications.commsDisabled`

Errors
- Expected Request Errors
- `E-CM-00061` [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

5. **Determine which device was deactivated.**
{05d22212-7c2a-4866-8444-c380fc88b105}

Command
- `communications.getDescriptor`

Errors
- Send Request Errors
- `E-CM-00064` [The user failed to deactivate a device.]

6. **Enable the communications channel and verify the proper events are generated.**
{06388446-165d-48fe-8845-a2fa6dbfba3a}

Command
- `communications.setCommsState`

Events
(Time out: ${event.timeOut})

- G2S_CME101 [Comms Not Established]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

7. **Send a G2S_MSX003 error to the EGM when it sends a commsProfile response and leave the EGM in the sync state.**
{07d2d23f-4390-4aec-9847-5958d51b16d2}

Commands

- communications.getCommsProfile
- communications.commsClosing
- communications.commsOnLine

Errors

- Expected Request Errors
- Send Request Errors
- E-CM-00066 [CommsOnLine MUST have deviceChanged attribute set to false if no device changes have occurred for the EGM.]

8. **If the user deactivated a device.**

   a. **Instruct user to active a device.**
   {089d661a-e1aa-45e8-8d00-c2412a03c485}

   Command

   - communications.commsClosing

   Actions

   - Instruct the user to 'Please activate device ${g_devices[0].toDisplayable()}.'.

   Errors

   - Expected Request Errors
   - DUT Error
   - E-CM-00065 [The user failed to activate a device.]

b. **Wait for the EGM to enter the SYNC state.**
{09711631-fc3b-4e50-9f1d-9ce52588acff}

Commands
- communications.commsOnLine
- communications.commsDisabled

Errors
- Expected Request Errors
- E-CM-00061 [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

9. **Enable the communications channel and verify the proper events are generated.**
{10aad306-6ef3-4214-a3c9-dcf7c1372f16}

Command
- communications.setCommsState

Events
(Time out: ${event.timeOut})
- G2S_CME101 [Comms Not Established]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00085

This case verifies that the EGM transitions to the closing state when a device is activate or deactivated in the online state.

Objectives

- Verify that the EGM transitions to the closing state when a device is deactivated in the online state.
- Verify that the EGM transitions to the closing state when a device is activated in the online state.

Requirements Under Test

- 2.7.4

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine the current active devices.**
   {013c54a0-2ecb-4eb5-9e20-0489a3acd2ab}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Instruct user to deactivate a device.**
   {02397a63-f74c-4a71-98c5-c0c73860d113}

   Command
   - `communications.commsClosing`

   Actions
   - Instruct the user to 'Please deactivate a device other than the communications device ${deviceUnderTest.deviceId}.'.

Errors

- Expected Request Errors
- DUT Error
- E-CM-00064 [The user failed to deactivate a device.]

3. **Wait for the EGM to enter the SYNC state.**
{03d5102b-6528-4861-9dc5-6910159e7350}

Commands

- communications.commsOnLine
- communications.commsDisabled

Errors

- Expected Request Errors
- E-CM-00061 [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

4. **Determine which device was deactivated.**
{044ead32-fb3c-4570-ad27-f9f1e2af1d36}

Command

- communications.getDescriptor

Errors

- Send Request Errors
- E-CM-00064 [The user failed to deactivate a device.]

5. **Enable the communications channel and verify the proper events are generated.**
{05ee2f20-4f08-432a-80bd-e51e65f2c9cf}

Command

- communications.setCommsState

Events

(Time out: ${event.timeOut})

- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

6. **If the user deactivated a device.**

    a. **Instruct user to active a device.**
    {06a7eb64-4558-4e92-8865-c73fee3f80c8}

    Command
    - `communications.commsClosing`

    Actions
    - Instruct the user to 'Please activate device ${g_devices[0].toDisplayable()}.'.

    Errors
    - Expected Request Errors
    - DUT Error
    - `E-CM-00065` [The user failed to activate a device.]

    b. **Wait for the EGM to enter the SYNC state.**
    {0728ee18-b183-409b-b9f7-2ed35478dcab}

    Commands
    - `communications.commsOnLine`
    - `communications.commsDisabled`

    Errors
    - Expected Request Errors
    - `E-CM-00061` [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

7. **Enable the communications channel and verify the proper events are generated.**
{08b03288-721a-4812-9539-debae1ea78b5}

    Command
    - `communications.setCommsState`

    Events
    (Time out: ${event.timeOut})
    - `G2S_CME001` [Comms Disabled by EGM]
    - `G2S_CME002` [Comms Enabled by EGM]
    - `G2S_CME100` [Comms Established]
    - `G2S_CME004` [Comms Enabled by Host]

    Errors
    - Send Request Errors
    - Event Checking Errors

- `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00086

Deleted

This test case was deleted.

Test Case: CM-COR-00087

This case verifies that the EGM generates `G2S_CME101 Comms Not Established` event when the communications device is reset.

Objective

Verify that the EGM generates `G2S_CME101` event when the communications device is reset.

Requirements Under Test

- 2.20.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Instruct the user to reset the comms device.**
   {0142a8c5-ba04-4470-be30-59e1b0ca9f9f}

   #### Action

   - Instruct the user to 'Please reset the comms device ${deviceUnderTest.deviceId}.'.

   #### Events

   (Time out: ${reconnect.timeOut})

   - `G2S_CME101` [Comms Not Established]
   - `G2S_CME100` [Comms Established]

   #### Errors

   - Event Checking Errors
   - DUT Error
   - `E-CM-00067` [The user failed to reset the commsDevice.]

Test Case: CM-COR-00088

This case continuously tests `commsOnLine.subscriptionLost` and `commsOnLine.metersReset` attributes are set correctly.

Objectives

- That communications event subscriptions have not been lost unless `commsOnLine.subscriptionLost` is true.
- That cabinet meter subscriptions have not been lost unless `commsOnLine.subscriptionLost` is true.
- That meters have not been reset unless `commsOnLine.metersReset` is true.

Requirements Under Test

- 2.24.6
- 2.24.7

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: communications
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Assert that subscriptions have not been lost unless commsOnLine.subscriptionLost is true.**
   {01ac3270-9163-47f1-b210-ea94d89c54d2}

   Error
   - `E-CM-00031` [Transaction ID ${transactionId} MUST be greater than ${lastTransactionId}.]

2. **Assert that ever increasing meters are always increasing unless commsOnLine.metersReset is true.**
   {0259e320-aa02-48d6-80c8-395f88c84a60}

   Error
   - `E-CM-00031` [Transaction ID ${transactionId} MUST be greater than ${lastTransactionId}.]

Test Case: CM-COR-00089

This case verifies that the EGM properly supports `commsStatus.reenrollmentFailed` attribute.

Objectives

- Verify that the EGM does not report `G2S_unknown` for `commsStatus.reenrollmentFailed`.
- Verify that the EGM generates `G2S_CME150 Certificate Reenrollment Failure` event when the EGM certificate is revoked.
- Verify that the EGM generates `G2S_CME151 Certificate Reenrollment Failure Cleared` event when the EGM certificate is .

Requirements Under Test

- 2.22.3
- 2.65.1
- 2.66.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Set the response manager to ignore eventHandler.eventReports and communications.commsDisabled.**
   {01db7c98-31be-4062-9dd4-35bdaf05e746}

2. **Instruct the user to revoke the EGM's certificate.**
   {029a9392-a1d0-4c95-bece-8012bf6b4d51}

   #### Actions

   - Instruct the user to 'Please revoke the EGM's certificate in the certificate authority.'.
   - Instruct the user to 'Please power cycle the EGM.'.

   #### Errors

- DUT Error
- E-CM-00057 [The user failed to power cycle the EGM.]
- E-CM-00070 [The user failed to revoke the EGM's certificate.]

3. **Reset the response manager.**
{03f370e5-14d6-4cbd-8116-adc08dbb1f34}

4. **Wait for the EGM to enter the SYNC state.**
{04187feb-6d55-4551-946f-4259e85e6136}

Commands
- communications.commsOnLine
- communications.commsDisabled

Errors
- Expected Request Errors
- E-CM-00061 [CommsOnLine MUST have deviceChanged attribute set to true if the device structure has changed for the EGM.]

5. **Enable the communication channel and instruct user to approve EGM certificate.**
{0562f760-8957-4219-954b-a6354020219f}

Command
- communications.setCommsState

Actions
- Instruct the user to 'Please approve the EGM's certificate in the certificate authority.'.

Events
(Time out: ${event.timeOut})
- G2S_CME150 [Certificate Reenrollment Failure]
- G2S_CME151 [Certificate Reenrollment Failure Cleared]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00090

This case verifies that the EGM generates `G2S_CME006 Device Configuration Changed by Operator` event.

Objective

Verify that the EGM generates `G2S_CME006` event when the communication device is changed by the operator.

Requirements Under Test

- 2.46.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Determine if the EGM has a comm config device.**
   {01debcb8-eccb-4158-8038-acda72d74731}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **If the EGM supports commConfig and the CVT has permission to get the commHostList.**

   a. **Get the comm host list to determine if a local change is allowed.**
   {02983779-9329-49eb-962c-9c8efbdea6f0}

   Command
   - `commConfig.getCommHostList`

   Error
   - Send Request Errors

3. **If the EGM supports local changes and the CVT has permission to receive event G2S_CME006**

a. **Instruct user to make a local change to a communication device.**
{0344029b-2698-436b-a836-134a43b6cfee}

Action

- Instruct the user to 'Please make a configuration change to communications device ${commsDevice.deviceId}'.

Event

(Time out: ${event.timeOut})

- `G2S_CME006` [Comms Configuration Changed by EGM]

Errors

- Event Checking Errors
- DUT Error

Test Case: CM-COR-00091

This case verifies that the EGM persisted the affected data set for communications events.

Objective

Verify that the EGM persists the cabinet status for `G2S_CME003 Device Disabled by Host` in the event handler log.

Requirements Under Test

- 4.21.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner
G2S_eventHandler	owner

Test Procedure

1. **Get the event handler log status.**
   {01de0861-f37d-45ea-9e81-5d75e9832623}

   Command
   - `eventHandler.getEventHandlerLogStatus`

   Error
   - Send Request Errors

2. **Disable the communications device.**
   {02516751-c5b3-473c-8dd5-03a823002605}

   Command
   - `communications.setCommsState`

   Error
   - Send Request Errors

3. **Get the event handler log status to determine the range of events to retrieve.**
   {03d0f1a8-0e63-4613-b95e-1d294926d891}

Command

- `eventHandler.getEventHandlerLogStatus`

Error

- Send Request Errors

4. **Get the event handler log and verify that the G2S_CME003 log record has the communications device in the affected device list.**
{049411cc-1fe9-47a0-b79b-a5f693dfaa5f}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00059` [Event ${eventCode} should have an entry in the event handler log list.]
- `E-EH-00060` [Event ${eventCode} MUST have an affected device entry for ${device}.]

5. **Enable the communications channel.**
{05083026-478f-4b44-90ce-89e2b4908300}

Command

- `communications.setCommsState`

Error

- Send Request Errors

Test Case: CM-COR-00092

This case continuously tests that EGM IDs and device IDs are correct.

Objectives

- Continuously verify that the EGM ID is unchanged during the test session.
- Continuously verify that device IDs are positive integer values unless an invalid device class or device ID is being reported.
- Continuously verify that all device IDs are valid descriptors for the EGM.

Requirements Under Test

- 1.7.1
- 1.7.3

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that EGM ID and device ID are valid.**
   {01c97321-1fac-40c6-ad0c-9562598bed72}

   Errors
   - `E-CM-00006` [An EGM ID of '${egmID}' is not valid.]
   - `E-CM-00012` [Invalid device ID for ${deviceClass} ${deviceId}.]

2. **Verify that device IDs are positive integer values.**
   {02fbfa0c-72ae-45e6-90d5-3b9311a3d093}

   Error
   - `E-CM-00012` [Invalid device ID for ${deviceClass} ${deviceId}.]

Test Case: CM-COR-00093

This test verifies that the EGM uses the no-response timer to determine the transport state of the communications device.

Objectives

- Verify that if the no-response timer is set to a non-zero value that the EGM transitions to G2S_hostUnreachable when the host stops sending g2sAcks.
- Verify that if the no-response timer is set to a non-zero value that the EGM transitions to G2S_hostUnreachable when the host stops sending application responses.

Requirements Under Test

- 2.10.6
- 2.10.11
- 2.58.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: EGM
- **Required Hosts**: 1

Required Devices

Device	Permissions
G2S_communications	owner
G2S_eventHandler	owner

Required Configurations

Option	Value
commsProfile.noResponseTimer	!= 0

Test Procedure

1. **Determine the noResponseTimer value.**
   {01dfd8c6-8978-414f-9592-a79afccc3b76}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **Set the keepAlive frequency to five seconds and disable sending g2sAcks to keepAlive commands.**
{024b269a-cb82-43ca-ab82-ea9b7cdc439d}

Commands
- communications.setKeepAlive
- communications.commsClosing
- eventHandler.setEventSub

Errors
- Expected Request Errors
- Send Request Errors

3. **Wait for the EGM to enter the SYNC state.**
{03e67a9d-d038-4cb6-a58a-28358d54dcb5}

Commands
- communications.commsOnLine
- communications.commsDisabled

Error
- Expected Request Errors

4. **Enable the communications device and verify that the events are generated by the EGM.**
{0412ba90-5963-4023-b74f-968de578d9aa}

Command
- communications.setCommsState

Events
(Time out: ${event.timeOut})
- G2S_CME120 [Comms Host Unreachable]
- G2S_CME003 [Comms Disabled by Host]
- G2S_CME001 [Comms Disabled by EGM]
- G2S_CME121 [Comms Transport Up]
- G2S_CME002 [Comms Enabled by EGM]
- G2S_CME100 [Comms Established]
- G2S_CME004 [Comms Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- E-CM-00050 [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

5. **Set the keepAlive frequency to five seconds and disable sending keepAliveAck responses to keepAlive commands.**
{057f0019-573a-42e7-a90d-41d13d3f009f}

   Commands
   - `communications.setKeepAlive`
   - `communications.commsClosing`

   Errors
   - Expected Request Errors
   - Send Request Errors

6. **Wait for the EGM to enter the SYNC state.**
{06f78896-5f06-4dcd-93d9-7de203e07b9e}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Error
   - Expected Request Errors

7. **Enable the communications device and verify that the events are generated by the EGM.**
{079c5201-c6a8-44d7-95fc-69e62236df64}

   Command
   - `communications.setCommsState`

   Events
   (Time out: ${event.timeOut})
   - `G2S_CME120` [Comms Host Unreachable]
   - `G2S_CME003` [Comms Disabled by Host]
   - `G2S_CME001` [Comms Disabled by EGM]
   - `G2S_CME121` [Comms Transport Up]
   - `G2S_CME002` [Comms Enabled by EGM]
   - `G2S_CME100` [Comms Established]
   - `G2S_CME004` [Comms Enabled by Host]

   Errors
   - Send Request Errors
   - Event Checking Errors
   - `E-CM-00050` [CommStatus attribute ${attribute} of ${actual} is wrong it should be ${expected}.]

Test Case: CM-COR-00094

This case continuously tests that the syncTimer is not less than 15 seconds

Objectives

- Continuously verify that the commsOnLineAck.syncTimer is not less than 15 seconds.
- Continuously verify that the commsDisabledAck.syncTimer is not less than 15 seconds.

Requirements Under Test

- 2.6.2
- 2.26.8
- 2.28.3

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the syncTimer is not less than 15 seconds.**
   {0130ff21-eca9-441d-a34a-ff9516230109}

   Error
   - `E-CM-00073` [The syncTimer MUST NOT be less than 15 seconds.]

Test Case: CM-COR-00095

This case continuously tests that the host responds with a commsClosingAck to a commsClosing request.

Objective

Continuously verify that the host sends a commsClosingAck to a commsClosing request.

Requirements Under Test

- 2.30.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: communications

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that the host sends a commsClosingAck to a commsClosing request.**
   {01fc8b5f-6fc4-47de-b426-c2d18cafe72e}

   Error
   - `E-CM-00010` [Endpoint did not send a valid response to ${command}.]

Test Case: CM-COR-00096

This case continuously tests that the host send commands in command ID order and that they are always increasing by one.

Objectives

- Continuously verify that commands are sent in command ID order.
- Continuously verify that commands in one g2s message are in command ID order.
- Continuously verify that commands IDs increment by one unless the command IDs are reset.
- Continuously verify that the retried commands have new command ID and dateTime values when sent by the host.

Requirements Under Test

- 1.28.3
- 1.30.3
- 1.30.4
- 2.2.10

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: communications
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_communications	owner

Test Procedure

1. **Verify that command IDs are in order.**
   {01f3b4fa-7b4a-467f-8431-198bdc1c4775}

   Error
   - `E-CM-00007` [Command ID of ${commandId} must be greater than ${lastCommandId}.]

2. **Verify that commands are increasing by one and that resent commands have new dateTime value.**
   {0232d0d9-a31d-4a5d-89fb-41e2d6d22da8}

   Errors

- `E-CM-00042` [Retried commands MUST have new dateTimeSent values.]
- `E-CM-00043` [Command IDs MUST strictly increase by one. Command ID ${commandId} MUST be one greater than ${lastCommandId}.]

Event Handler
Functional Groups

- Core Functionality (COR)

eventHandler

Test Case: EH-COR-00001

This case continuously tests that event reports are properly generated.

Objectives

- Continuously verify that `eventReport` commands are in the host's subscription or forced subscription.
- Continuously verify that `eventReport` commands do not contain additional `statusInfo` elements in a known namespace.
- Continuously verify that `eventReport` commands do not contain additional meters for a device that was not affected by the event.
- Continuously verify that `eventReport` commands do not contain additional `transactionInfo` elements in a known namespace.
- Continuously verify that `eventReport` commands contain all the affected data, based on the subscription.
- Continuously verify that the event payload device class is appropriate for the event code.
- Continuously verify that `eventReport` commands have the correct session type, based on subscription.

Requirements Under Test

- 1.30.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Assert that `eventReport` commands are in the host's subscriptions or forced subscription.**
{8c627207-5530-4143-00ec-5e53447fe09e}

   Error
   - `E-EH-00002` [Event ${eventCode} was not expected based on host subscription and forced subscription.]

2. **Assert that `eventReport` commands do not contain additional `statusInfo` elements in known namespaces.**
{68bb742e-7b0e-424b-8037-73acb9b5a896}

   Error
   - `E-EH-00003` [Event ${eventCode} contains additional statusInfo elements for known namespaces.]

3. **Assert that `eventReport` commands do not contain additional known meters sets that are not affected by the event.**
{8a363bf4-7d19-41e8-0031-6fae0ddf8364}

   Error
   - `E-EH-00004` [Event ${eventCode} contains additional known meter set that is not affected by the event.]

4. **Assert that `eventReport` commands do not contain additional `transactionInfo` elements in known namespaces.**
{249eec97-612b-4149-0020-3768501f59a4}

   Error
   - `E-EH-00005` [Event ${eventCode} contains additional transactionInfo elements for a known namespace.]

5. **Assert that `eventReport` commands include all affected data specified by the protocol.**
{7525c543-7040-4cdf-801b-6a667d5d9b20}

   Error
   - `E-EH-00006` [Event ${eventCode} is missing affected data.]

6. **Assert that the device class in the event payload is appropriate for the event code.**
{f50f9bb8-eb28-469f-abf3-05fe5d8ed7a3}

   Error
   - `E-EH-00011` [Event ${eventCode} has the wrong device class of ${actual} it should be ${expected}.]

7. **Assert that the `eventReport` has the correct session type, based on the event subscription.**
{fbfd3ded-373e-4eb2-adbe-e6406caa33cb}

Error

- `E-EH-00012` [Event ${eventCode} has the wrong session type of ${actual} it should be ${expected}.]

Test Case: EH-COR-00003

This case verifies that the EGM does not accept commands from non-guest hosts.

Objective

Verify that the EGM does not accept `eventHandler` commands from non-guest hosts.

Requirements Under Test

- 1.8.5
- 1.8.14
- 1.21.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	non-guest

Test Procedure

1. **Send a `setEventSub` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
   {013e5061-110d-4d66-a891-caba28d7b680}

   Command
   - `eventHandler.setEventSub`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

2. **Send a `clearEventSub` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
   {026adce7-b1a0-4e45-940f-98b9dbda416c}

   Command
   - `eventHandler.clearEventSub`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

3. **Send a `setEventHandlerState` command, and verify that the EGM responds with `G2S_APX010` or `G2S_APX012`.**
{0342d00b-bba4-43e5-b17a-a0ee5df30013}

   Command
   - `eventHandler.setEventHandlerState`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

4. **Send a `getEventHandlerProfile` command, and verify that the EGM responds with `G2S_APX012`.**
{048bd8d3-010a-4a77-91e7-3adb50b3dee5}

   Command
   - `eventHandler.getEventHandlerProfile`

     Expect **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

5. **Send a `getEventSub` command, and verify that the EGM responds with `G2S_APX012`.**
{057333d4-e6f0-4dde-92a3-e49270103689}

   Command
   - `eventHandler.getEventSub`

     Expect **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

6. **Send a `getEventHandlerStatus` command, and verify that the EGM responds with `G2S_APX012`.**
{065123e8-a39a-4a00-b99a-222b9e1dc18f}

   Command
   - `eventHandler.getEventHandlerStatus`

     Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

7. **Send a `getEventHandlerLogStatus` command, and verify that the EGM responds with `G2S_APX012`.**
{07d87504-a560-453f-9631-3b2d9458ff8d}

Command

- `eventHandler.getEventHandlerLogStatus`
  Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

8. **Send a `getEventHandlerLog` command, and verify that the EGM responds with `G2S_APX012`.**
{080e4f04-9cc7-436e-be98-2b3ebf222fbf}

Command

- `eventHandler.getEventHandlerLog`
  Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

9. **Send a `getSupportedEvents` command, and verify that the EGM responds with `G2S_APX012`.**
{09f02363-cbb8-4ad0-a1a4-c0b6c6810c43}

Command

- `eventHandler.getSupportedEvents`
  Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

Test Case: EH-COR-00004

This case verifies that the EGM does not accept control commands from a guest host.

Objectives

- Verify that the EGM does not accept `eventHandler` control commands from a guest host.
- Verify that the EGM accepts `eventHandler` non-control commands from a guest host.

Requirements Under Test

- 1.8.14
- 4.5.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	guest

Test Procedure

1. **Send a `setEventSub` command, and verify that the EGM responds with `G2S_APX010`.**
   {019ba91a-662b-4eb0-88dc-88b801760a93}

   Command
   - `eventHandler.setEventSub`

     Expect **G2S_APX010**

   Errors
   - Send Request Errors
   - `E-CM-00037` [The EGM must not allow control commands from a guest host.]

2. **Send a `clearEventSub` command, and verify that the EGM responds with `G2S_APX010`.**
   {026a0777-5353-4a29-96ee-996d63768451}

   Command
   - `eventHandler.clearEventSub`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors

- Send Request Errors
- E-CM-00037 [The EGM must not allow control commands from a guest host.]

3. **Send a `setEventHandlerState` command, and verify that the EGM responds with `G2S_APX010`.**
{0318c24c-36df-4762-9a8f-70c5ea72df3a}

Command
- eventHandler.setEventHandlerState

  Expect **G2S_APX010**

Errors
- Send Request Errors
- E-CM-00037 [The EGM must not allow control commands from a guest host.]

4. **Send a `getEventHandlerProfile` command, and verify that the EGM responds properly.**
{04a325fc-7bb2-4570-b33a-ac30255a171f}

Command
- eventHandler.getEventHandlerProfile

Errors
- Send Request Errors
- E-CM-00038 [The EGM must generate a response to a request from the host.]

5. **Send a `getEventSub` command, and verify that the EGM responds properly.**
{05e84752-8d91-4ce1-86d9-a46d8f95a993}

Command
- eventHandler.getEventSub

Errors
- Send Request Errors
- E-CM-00038 [The EGM must generate a response to a request from the host.]

6. **Send a `getEventHandlerStatus` command, and verify that the EGM responds properly.**
{062ab56e-e32f-4cd6-9161-6309a43ea80d}

Command
- eventHandler.getEventHandlerStatus

Errors
- Send Request Errors
- E-CM-00038 [The EGM must generate a response to a request from the host.]

7. **Send a `getEventHandlerLogStatus` command, and verify that the EGM responds properly.**
{07ea09b0-9909-4721-bd02-87c28f9091ea}

   Command
   - `eventHandler.getEventHandlerLogStatus`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

8. **Send a `getEventHandlerLog` command, and verify that the EGM responds properly.**
{08039325-24a7-476a-9fa3-2256937996c2}

   Command
   - `eventHandler.getEventHandlerLog`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

9. **Send a `getSupportedEvents` command, and verify that the EGM responds properly.**
{092767bc-6161-4591-8f36-4552692d8bae}

   Command
   - `eventHandler.getSupportedEvents`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

Test Case: EH-COR-00005

This test verifies `eventHandler` log processing.

Objectives

- Verify that the EGM returns a `G2S_APX003 Invalid Device Identifier` error when device ID zero (0) is used as a class-level attribute.
- Verify that the `eventHandler` log sequence numbers are contiguous.
- Verify that the EGM returns the entire event log when the `lastSequence` and `totalEntries` attribute values are zero (0).
- Verify that the EGM returns an empty event log when the requested `logSequence` number is not in the log.
- Verify that the EGM returns the specified log entries when `totalEntries` attribute value is not zero (0).
- Verify that the EGM does not send `eventReport` commands when the `eventHandler` device is disabled.

Requirements Under Test

- 1.7.4
- 1.19.12
- 1.19.13
- 1.21.2
- 1.21.3
- 1.21.4
- 1.21.5
- 1.21.6
- 1.21.7
- 4.3.6
- 4.5.5
- 4.5.6
- 4.7.1
- 4.7.3
- 4.7.4
- 4.26.1
- 4.29.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM

- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Send a `getEventHandlerLog` command to device ID zero (0), and verify the EGM responds with a `G2S_APX003 Invalid Device Identifier` error.**
   {010822ec-8c7d-4344-9158-71f5098cfe94}

   Command
   - `eventHandler.getEventHandlerLog`

     Expect **G2S_APX003**

   Errors
   - Send Request Errors
   - `E-XX-00015` [EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).]

2. **Clear the event subscription.**
   {0204329d-9e13-475f-9b08-7a49f93288c2}

   Commands
   - `eventHandler.clearEventSub`
   - `eventHandler.getEventSub`

   Error
   - Send Request Errors

3. **Determine the size of the `eventHandler` log.**
   {03021d8e-db49-4684-9ae8-849d6c3d5ea0}

   Command
   - `eventHandler.getEventHandlerProfile`

   Error
   - Send Request Errors

4. **Determine the `lastSequence` and `totalEntries` values in the `eventHandler` log.**
   {040222b5-677f-44ad-8aea-8f027515a1c5}

   Command
   - `eventHandler.getEventHandlerLogStatus`

Error

- Send Request Errors

5. **Verify that the `getEventHandlerLog` command, with `lastSequence` and `totalEntries` attribute values of zero (0), retrieves the entire log.**
{050662f6-3337-43dc-8f3c-12430f803738}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00043` [Event log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-EH-00044` [Event log MUST start with logSequence number ${sequenceNumber}.]
- `E-EH-00045` [Event log MUST have ${logEntries} entries.]

6. **Verify that requesting the `logSequence` less than the smallest `logSequence` number returns an empty log.**
{06a75f5e-5d45-401a-b6e3-5fc52873c160}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00046` [Event log MUST be empty when requesting a lastSequence number not in the log.]

7. **Verify that requesting the `logSequence` greater than the last `lastSequence` number returns an empty log.**
{07ab5ea8-3105-407e-8889-3aacfcd8b640}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00046` [Event log MUST be empty when requesting a lastSequence number not in the log.]

8. **Verify that the EGM returns an `eventHandler` log with the correct number of entries.**
{081baa93-43fe-4f45-bb9a-05dc52dfab86}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00044` [Event log MUST start with logSequence number ${sequenceNumber}.]
- `E-EH-00045` [Event log MUST have ${logEntries} entries.]

9. **Verify that the EGM returns an `eventHandler` log with one entry if the `totalEntries` equals 1.**
{09ec2fd4-ef6b-4993-b8bc-4bd7c76c3ba8}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00045` [Event log MUST have ${logEntries} entries.]

10. **Set the event subscription for `G2S_EHE003`, `G2S_EHE004` and `G2S_EHE101`.**
{101b9146-a6aa-426a-84ef-7f3d5f5665f2}

Commands

- `eventHandler.setEventSub`
- `eventHandler.getEventSub`

Event

(Time out: ${event.timeOut})

- `G2S_EHE101` [Event Subscription Changed]

Errors

- Send Request Errors
- Event Checking Errors

11. **Disable the `eventHandler` device.**
{11116569-ae93-4506-9801-34ca6ff5580a}

Command

- `eventHandler.setEventHandlerState`

Errors

- Send Request Errors
- Event Not Expected Error

12. **Enable the `eventHandler` device.**
{1276e2cc-6868-4116-a4da-527f3ebbc513}

Command
- `eventHandler.setEventHandlerState`

Event
(Time out: ${event.timeOut})
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors

13. **Verify the `eventHandler` log is correct.**
{130439a0-371f-42ab-924d-18012390088b}

Command
- `eventHandler.getEventHandlerLog`

Errors
- Send Request Errors
- `E-EH-00043` [Event log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-EH-00044` [Event log MUST start with logSequence number ${sequenceNumber}.]
- `E-EH-00045` [Event log MUST have ${logEntries} entries.]

14. **Reset event subscription.**
{147d745c-cc37-4990-8f31-3b1a78021799}

Commands
- `eventHandler.setEventSub`
- `eventHandler.getEventSub`

Error
- Send Request Errors

Test Case: EH-COR-00007

This case verifies that event subscriptions associated with disabling the `eventHandler` device are correct.

Objectives

- Verify that event subscriptions are evaluated at the time an event is generated.
- Verify that affected data is collected when the `eventReport` command is sent.
- Verify that `eventHandler` events have the correct affected data.

Requirements Under Test

- 1.9.6
- 1.23.3
- 4.3.1
- 4.3.3
- 4.3.7
- 4.4.4
- 4.5.5
- 4.5.6
- 4.7.1
- 4.7.3
- 4.7.4
- 4.26.1
- 4.29.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager to suppress responses to `G2S_EHE004` `eventReport` commands.**
   {0125a6e6-f91a-48d1-97bd-85379bc576b9}
2. **Set the event subscription for `G2S_EHE101` to be not persisted.**
   {0283d9bf-ba29-4941-8580-149879c68219}

Commands
- `eventHandler.getEventSub`
- `eventHandler.clearEventSub`
- `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Errors
- Send Request Errors
- Event Checking Errors

3. **Set the event subscription for `G2S_EHE004` to not include the device status.**
{03e30b59-2d31-476b-89ec-10afc87ad836}

Commands
- `eventHandler.getEventSub`
- `eventHandler.clearEventSub`
- `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Errors
- Send Request Errors
- Event Checking Errors

4. **Host disable the `eventHandler` device.**
{04de2c04-79b9-45e1-86f2-76b6d5ee3e08}

Command
- `eventHandler.setEventHandlerState`

Errors
- Send Request Errors
- Event Not Expected Error

5. **Clear the event subscription for `G2S_EHE102`.**
{05e21f02-b8ea-4c08-b3bc-8e06d36b0de9}

Commands
- `eventHandler.clearEventSub`
- `eventHandler.getEventSub`

Errors

- Send Request Errors
- Event Not Expected Error

6. **Host enable the `eventHandler` device.**
{06e4aee5-1259-4779-9076-c73c77f34502}

Command

- `eventHandler.setEventHandlerState`

Event

(Time out: ${event.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- Event Not Expected Error

7. **Set the event subscription for `G2S_EHE004` to include device statuses.**
{073d945e-d822-4a4c-864c-1a45af4fb794}

Commands

- `eventHandler.getEventSub`
- `eventHandler.setEventSub`

Events

(Time out: ${retry.timeOut})

- `G2S_EHE101` [Event Subscription Changed]
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors
- Event Not Expected Error

8. **Wait for the EGM to retry the `G2S_EHE004` `eventReport` command.**
{080a05b4-27c1-48d1-8d84-3f561ca24233}

Event

(Time out: ${retry.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Event Checking Errors
- Event Not Expected Error

9. **Set the response manager to respond to `G2S_EHE004` `eventReport` commands.**
{09893d63-7234-4afc-a52d-bbfd2984f9b3}

Event
(Time out: ${retry.timeOut})
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors
- Event Checking Errors
- Event Not Expected Error

10. **Host disable the `eventHandler` device.**
{10bd5a96-caa1-43ff-8f47-8c45606b3a41}

Command
- `eventHandler.setEventHandlerState`

Errors
- Send Request Errors
- Event Not Expected Error

11. **Host enable the `eventHandler` device.**
{11787a93-5765-4aab-8080-13f8d0af5ff2}

Command
- `eventHandler.setEventHandlerState`

Event
(Time out: ${event.timeOut})
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- Event Not Expected Error

12. **Reset event subscription.**
{12124e8a-2af0-4f1a-a119-5db5d77c09b5}

Commands
- `eventHandler.getEventSub`
- `eventHandler.setEventSub`

Error

- Send Request Errors

Test Case: EH-COR-00013

This case verifies the EGM handles supported events properly.

Objectives

- Verify that existing event subscriptions are ORed with a new subscription.
- Verify that wildcards are applied to all devices.
- Verify that event subscriptions are only for devices of which the CVT is the owner or guest.
- Verify that event subscription lists do not have wildcards.

Requirements Under Test

- 4.13.1
- 4.13.2
- 4.14.1
- 4.14.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	guest or owner
G2S_communications	owner

Test Procedure

1. **Determine active devices.**
   {011f3422-e73d-41c4-868c-f0d6540a11b0}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Verify that the EGM returns supported events for all active devices with events.**
   {027ba7d7-4c2c-4424-bf4e-b229edf8c63b}

   Command

- `eventHandler.getSupportedEvents`

Errors

- Send Request Errors
- `E-EH-00048` [Wild cards are not allowed in supportEvents responses.]
- `E-EH-00049` [Supported events list has an unknown device ${deviceClass}[${deviceId}].]
- `E-EH-00050` [Supported events list MUST include events for ${deviceClass}[${deviceId}].]

3. **Verify that the EGM returns an empty supported events list or `G2S_APX003 Invalid Device Identifier` error for an invalid device ID.**
{032a546e-7495-4a86-84c9-79496bd132fe}

Command

- `eventHandler.getSupportedEvents`

  Expect **G2S_APX003** or **Normal Response**

Errors

- Send Request Errors
- `E-EH-00047` [EGM MUST return an empty supported event list for ${deviceClass} [${deviceId}].]

4. **Verify that the EGM returns an empty supported events list or `G2S_APX007 Class Not Supported` error for an invalid device class.**
{0495a1af-afb6-4b01-a127-8108c006490e}

Command

- `eventHandler.getSupportedEvents`

  Expect **G2S_APX007** or **Normal Response**

Errors

- Send Request Errors
- `E-EH-00047` [EGM MUST return an empty supported event list for ${deviceClass} [${deviceId}].]

5. **Verify that the EGM returns supported events for a specified device ID.**
{050ad8e6-3cf1-41ad-a4de-9bea5408903b}

Command

- `eventHandler.getSupportedEvents`

Errors

- Send Request Errors
- `E-EH-00049` [Supported events list has an unknown device ${deviceClass}[${deviceId}].]
- `E-EH-00050` [Supported events list MUST include events for ${deviceClass}[${deviceId}].]

6. **Verify that the EGM returns an empty supported events list for device ID zero (0) or `G2S_APX003 Invalid Device Identifier`.**
{0681b18a-1890-4554-92ae-f754f61a2ed3}

   Command
   - `eventHandler.getSupportedEvents`

     Expect **G2S_APX003** or **Normal Response**

   Errors
   - Send Request Errors
   - `E-EH-00047` [EGM MUST return an empty supported event list for ${deviceClass} [${deviceId}].]

7. **Verify that the EGM returns the supported events for a specific device.**
{074f28a6-1347-4a1b-9154-5f496e6bd04c}

   Command
   - `eventHandler.getSupportedEvents`

   Errors
   - Send Request Errors
   - `E-EH-00050` [Supported events list MUST include events for ${deviceClass}[${deviceId}].]

Test Case: EH-COR-00014

This case continuously tests that meter info definitions are not included in event reports, and that wild cards are not in `eventSubList` responses.

Objectives

- Continuously verify that meter info definitions are not in event reports.
- Continuously verify that wild cards are not in `eventSubList` responses from the EGM.

Requirements Under Test

- 4.18.1
- 4.21.5
- 4.32.1

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Assert that there are no meter definitions attributes in event reports.**
   {9b034ba0-f336-4d0c-80d1-b272be142640}
2. **Assert that wild cards are not in `eventSubList` responses.**
   {f8d6bf52-6415-4059-807b-97ae222031e8}

Test Case: EH-COR-00015

This case verifies that the EGM manages event subscriptions properly.

Objectives

- Verify that the EGM handles the `clearEventSub` command.
- Verify that the EGM processes wildcard parameters in the `setEventSub`, `clearEventSub` and `getEventSub` commands.
- Verify that the EGM ORs new event subscriptions with previous event subscriptions.
- Verify that the EGM returns an empty event subscription list for a device that does not exist.

Requirements Under Test

- 4.15.1
- 4.15.2
- 4.15.3
- 4.15.6
- 4.15.7
- 4.15.9
- 4.17.1
- 4.17.2
- 4.17.3
- 4.17.4
- 4.18.1
- 4.19.1
- 4.19.4
- 4.19.5
- 4.19.6
- 4.29.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Get the list of active devices.**
   {013d5898-ba8b-43e5-a5b0-bf2476857a8f}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Get the list of supported events.**
   {022d91de-2e3c-44d3-ba57-4ac86538540f}

   Command
   - `eventHandler.getSupportedEvents`

   Error
   - Send Request Errors

3. **Clear all event subscriptions.**
   {03596a21-b4bf-42dd-b2d2-8067071fb0b2}

   Command
   - `eventHandler.clearEventSub`

   Error
   - Send Request Errors

4. **Verify that there are no host event subscriptions.**
   {04360f21-ae2d-4301-b3cd-5cc1b563ad8f}

   Command
   - `eventHandler.getEventSub`

   Errors
   - Send Request Errors
   - `E-EH-00055` [EGM MUST return an empty event subscription list for ${deviceClass} [${deviceId}], event code ${eventCode}.]

5. **Set event subscription for the `G2S_EHE004` event on all devices.**
   {0584e476-be28-48a9-9040-0d25c0ac491e}

   Command
   - `eventHandler.setEventSub`

   Error

- Send Request Errors

6. **Verify that there is a host subscription for the `G2S_EHE004` event, for all `eventHandler` devices for which the CVT owns or is a guest.**
{066ad7b8-23d0-4b66-a7b2-5c21f4cdb781}

   Command
   - `eventHandler.getEventSub`

   Errors
   - Send Request Errors
   - `E-EH-00052` [EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]
   - `E-EH-00053` [Event subscription for ${deviceClass}[${deviceId}] ${eventCode} MUST have ${attribute} set to ${expected}.]

7. **Set the event subscription for `G2S_EHE003`, and include sending the device status for the `G2S_EHE004` event for CVT's `eventHandler` device.**
{07204d95-4d3e-4633-8ba0-9d2b7ea4a254}

   Command
   - `eventHandler.setEventSub`

   Error
   - Send Request Errors

8. **Verify that there is a host subscription for `G2S_EHE003` and `G2S_EHE004`, and that the CVT's `eventHandler` device's `G2S_EHE004` includes sending the device status.**
{088ddb75-3b32-41b0-b3ab-3012a93eb574}

   Command
   - `eventHandler.getEventSub`

   Errors
   - Send Request Errors
   - `E-EH-00052` [EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]
   - `E-EH-00053` [Event subscription for ${deviceClass}[${deviceId}] ${eventCode} MUST have ${attribute} set to ${expected}.]

9. **Set event subscription for `G2S_APE001` on all devices.**
{09431dc1-d213-448b-813a-b4da9c6ab150}

   Command
   - `eventHandler.setEventSub`

Error

- Send Request Errors

10. **Verify that all devices that support `G2S_APE001` and have the proper host permissions are included in the event subscription list.**
{107fda53-7317-4be8-a1a4-e7152f349ae4}

    Command

    - `eventHandler.getEventSub`

    Errors

    - Send Request Errors
    - `E-EH-00052` [EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]
    - `E-EH-00054` [EGM must not have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]

11. **Verify the EGM returns an empty subscription list for the `CVT_testing` device class.**
{11abc78d-8bce-419c-bac6-170fd74baba5}

    Command

    - `eventHandler.getEventSub`

    Errors

    - Send Request Errors
    - `E-EH-00052` [EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]

12. **Reset event subscription.**
{12838a3a-c1d5-41c0-b7f7-0dbd55168e0a}

    Commands

    - `eventHandler.getEventSub`
    - `eventHandler.setEventSub`

    Error

    - Send Request Errors

Test Case: EH-COR-00016

This case tests that the EGM supports the required `deviceClass` attribute in the `getSupportedEvents` command.

Objective

Verify that the EGM validates the `getSupportedEvents` command.

Requirements Under Test

- 1.27.33

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	guest or owner

Test Procedure

1. **Send a `getSupportedEvents` command to the EGM without a `deviceClass` attribute.**
   {0115a2b7-e309-44a9-8e43-db11949432de}

   #### Command

   - `eventHandler.getSupportedEvents`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   #### Errors

   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

Test Case: EH-COR-00017

This case tests that the EGM rejects unknown commands.

Objective

Verify that the EGM sends the proper error for unknown commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	guest or owner

Test Procedure

1. **Send a `cvtTesting` command to the EGM.**
   {019b9049-c69f-4092-a09b-63c7e17d101d}

   Command
   - `eventHandler.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   Errors
   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: EH-COR-00018

This case tests `eventHandler` device identifiers and ownership.

Objectives

- Verify that `eventHandler` device IDs match their owner host ID.
- Verify that `eventHandler` devices are associated with registered hosts.
- Verify that `eventHandler` devices are not owned by the EGM.

Requirements Under Test

- 4.1.1
- 4.1.2
- 4.1.3
- 4.1.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	guest or non-guest or owner
G2S_commConfig	guest or owner

Test Procedure

1. **Send a `getCommHostList` command to the EGM, and verify `eventHandler` devices have the correct ownership and permissions.**
   {01066d43-4a34-4429-a98a-a8e1bee5b519}

   Command
   - `commConfig.getCommHostList`

   Errors
   - Send Request Errors
   - `E-CC-00001` [Device identifier for host-oriented devices MUST be equal to the host identifier of the owner host.]
   - `E-CC-00002` [EGM MUST have device class ${deviceClass} for host ID ${hostId}.]
   - `E-CC-00003` [Host-oriented devices MUST not be owned by unregistered hosts.]
   - `E-CC-00004` [EGM MUST not own host-oriented devices.]

Test Case: EH-COR-00019

This case tests that standard configuration options are available in `optionConfig`, and that they are consistent with the `eventHandlerProfile`.

Objectives

- Verify that the `optionConfig` parameters for option ID `G2S_protocolOptions` are configured properly.
- Verify that the `optionConfig` parameter definitions for option ID `G2S_forcedSubscriptionTable` are configured properly.
- Verify that the `eventHandlerProfile` command values match the `optionConfig` values.

Requirements Under Test

- 1.38.1
- 4.12.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the option list for the group ID `G2S_eventHandlerOptions`.**
   {0101ccfd-199b-4c17-aaa1-a99abcb6d360}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

2. **Assert that `G2S_protocolOptions` has the correct parameter configuration.**
   {02c833b0-c247-4911-b536-f74393306d00}

   Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]

3. **Assert that `G2S_forcedSubscriptionTable` has the correct parameter configuration.**
{03e1757b-8827-4ce4-a00f-6ad024c4b891}

   Error
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]

4. **Assert that the `eventHandler` profile values match the `optionConfig` values.**
{04864b3e-75a2-4a09-8d99-bc7ae72806d5}

   Command
   - `eventHandler.getEventHandlerProfile`

   Errors
   - Send Request Errors
   - `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: EH-COR-00020

This case tests that `eventReport` commands are retried at `eventProfile.timeToLive` frequency.

Objective

Verify that the `eventReport` commands are retried at the `timeToLive` frequency specified in the `eventHandler` profile.

Requirements Under Test

- 1.28.8
- 1.28.10
- 1.28.13
- 4.3.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Get the `eventHandler` profile to determine the retry frequency.**
   {0163a198-67a3-49ed-83a4-bae9b8ace843}

   Command
   - `eventHandler.getEventHandlerProfile`

   Error
   - Send Request Errors

2. **Set the response manager to ignore `G2S_EHE101` events.**
   {02cb4839-1e49-45bd-831b-090e31b7066c}

3. **Clear the event subscription for `G2S_EHE102` event.**
   {03f51835-eafe-4f85-a89b-98af0829bbe6}

   Commands
   - `eventHandler.clearEventSub`
   - `eventHandler.getEventSub`

Event
(Time out: ${event.timeOut})

- G2S_EHE101 [Event Subscription Changed]

Errors

- Send Request Errors
- Event Checking Errors

4. **Wait for the G2S_EHE101 event to be retried.**
{041320be-5da0-4be9-8686-5508f3ad681d}

Event
(Time out: ${g_retryTimeout})

- G2S_EHE101 [Event Subscription Changed]

Error

- Event Checking Errors

5. **Set the response manager to respond to G2S_EHE101 events.**
{05986117-1099-4b3f-bae3-cdbae8e9974e}

Test Case: EH-COR-00021

This case verifies that the EGM returns a `G2S_EHX001` error when the event subscription contains an error.

Objectives

- Verify that setting an event subscription for an unsupported class returns an error.
- Verify that setting an event subscription for a device that do not exists returns an error.
- Verify that setting an event subscription for an inactive device returns an error.
- Verify that setting an event subscription for a device that the CVT is not the owner or guest returns an error.

Requirements Under Test

- 1.5.5
- 4.15.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Get the list of devices in the EGM.**
   {0145d67d-5065-4720-9393-a750a1e83dc7}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Clear the event subscriptions.**
   {02a66eb9-1481-4a81-9b60-e8181c754c8c}

   Commands

- `eventHandler.clearEventSub`
- `eventHandler.getEventSub`

Error

- Send Request Errors

3. **Attempt to set an event subscription for an unsupported class.**
{033ba460-c7d4-4fec-b35e-fe9bb3e32cd5}

Command

- `eventHandler.setEventSub`

  Expect **G2S_EHX001**

Errors

- Send Request Errors
- `E-EH-00051` [EGM MUST NOT accept an event subscription for ${deviceClass} [${deviceId}].]

4. **Attempt to set an event subscription for a device that does not exist.**
{04abf891-6125-4b9c-a135-7ea17dd35464}

Command

- `eventHandler.setEventSub`

  Expect **G2S_EHX001**

Errors

- Send Request Errors
- `E-EH-00051` [EGM MUST NOT accept an event subscription for ${deviceClass} [${deviceId}].]

5. **If there is an inactive device in the descriptor list:**

   a. **Attempt to set an event subscription for an inactive device.**
   {05d609fb-9870-4e60-927e-be3212e5a549}

   Command

   - `eventHandler.setEventSub`

     Expect **G2S_EHX001**

   Errors

   - Send Request Errors
   - `E-EH-00051` [EGM MUST NOT accept an event subscription for ${deviceClass} [${deviceId}].]

6. **If there is a device the CVT does not own or is not a guest of:**

    a. **Attempt to set an event subscription for a device the CVT is not the owner or guest.**
    {06475494-a1e7-42f6-8946-d202bff5e8be}

    Command
    - `eventHandler.setEventSub`

      Expect **G2S_EHX001**

    Errors
    - Send Request Errors
    - `E-EH-00051` [EGM MUST NOT accept an event subscription for ${deviceClass} [${deviceId}].]

7. **Verify that there are no host event subscriptions.**
{07385708-8293-4663-8b14-a00a64a8e361}

    Command
    - `eventHandler.getEventSub`

    Errors
    - Send Request Errors
    - `E-EH-00055` [EGM MUST return an empty event subscription list for ${deviceClass} [${deviceId}], event code ${eventCode}.]

Test Case: EH-COR-00022

This case verifies that the EGM supports the `clearEventSub` command.

Objectives

- Verify that the `G2S_EHX001` event is not used with the `clearEventSub` command.
- Verify that the EGM does not generate a `G2S_EHE101` event when the event subscription does not change.

Requirements Under Test

- 4.19.1
- 4.19.3
- 4.19.4
- 4.19.5
- 4.19.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Verify that the EGM has a host event subscription for the `G2S_EHE004` event.**
   {0176e370-0eed-469a-9fdf-27630c04a5ff}

   Command
   - `eventHandler.getEventSub`

   Errors
   - Send Request Errors
   - `E-EH-00052` [EGM must have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]

2. **Clear the event subscription for the `G2S_EHE004` event.**
   {0231a940-8356-4e95-abff-a7a00565cc09}

   Command
   - `eventHandler.clearEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Errors
- Send Request Errors
- Event Checking Errors

3. **Verify that the EGM does not have a host event subscription for the `G2S_EHE004` event.**
{03fca9f4-01ff-46ac-8638-c6417450529e}

Command
- `eventHandler.getEventSub`

Errors
- Send Request Errors
- `E-EH-00051` [EGM MUST NOT accept an event subscription for ${deviceClass} [${deviceId}].]

4. **Clear the event subscription for the `G2S_EHE004` event, and verify that the `G2S_EHE101` event is not generated.**
{043e8e47-8998-48d3-b286-4c20f170491c}

Command
- `eventHandler.clearEventSub`

Errors
- Send Request Errors
- Event Not Expected Error

5. **Reset event subscription.**
{05f6794e-21fb-4181-ab85-f76305523b6a}

Commands
- `eventHandler.setEventSub`
- `eventHandler.getEventSub`

Error
- Send Request Errors

Test Case: EH-COR-00023

This case verifies that the host sends a valid getEventHandlerProfile command.

Objective

Verify the getEventHandlerProfile command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getEventHandlerProfile command from the host.**
   {01d84c30-7329-47f0-b429-c43f8f131838}

   #### Command

   - `eventHandler.getEventHandlerProfile`

   #### Actions

   - Instruct the user to 'Send getEventHandlerProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00024

This case verifies that the host sends a valid setEventSub command.

Objective

Verify the setEventSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send setEventSub command from the host.**
   {01217b63-018b-47fa-a50b-2482ca36eac5}

   #### Command

   - `eventHandler.setEventSub`

   #### Actions

   - Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00025

This case verifies that the host sends a valid getEventSub command.

Objective

Verify the getEventSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getEventSub command from the host.**
   {01304d25-f1a2-4954-966f-288bec9c379f}

   #### Command

   - `eventHandler.getEventSub`

   #### Actions

   - Instruct the user to 'Send getEventSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00026

This case verifies that the host sends a valid clearEventSub command.

Objective

Verify the clearEventSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send clearEventSub command from the host.**
   {01d91497-9909-4319-9394-e104cb1094cc}

   #### Command

   - `eventHandler.clearEventSub`

   #### Actions

   - Instruct the user to 'Send clearEventSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00027

This case verifies that the host sends a valid getEventHandlerStatus command.

Objective

Verify the getEventHandlerStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getEventHandlerStatus command from the host.**
   {013245cc-cd7c-4334-84cd-15899503e899}

   #### Command

   - `eventHandler.getEventHandlerStatus`

   #### Actions

   - Instruct the user to 'Send getEventHandlerStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00028

This case verifies that the host sends a valid setEventHandlerState command.

Objective

Verify the setEventHandlerState command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send setEventHandlerState command from the host.**
   {014c55b4-9f59-4bd0-91d9-8ed9c293d853}

   #### Command

   - `eventHandler.setEventHandlerState`

   #### Actions

   - Instruct the user to 'Send setEventHandlerState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00029

This case verifies that the host sends a valid getEventHandlerLogStatus command.

Objective

Verify the getEventHandlerLogStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getEventHandlerLogStatus command from the host.**
   {015b600e-ac43-4a15-9637-205247707309}

   #### Command

   - `eventHandler.getEventHandlerLogStatus`

   #### Actions

   - Instruct the user to 'Send getEventHandlerLogStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00030

This case verifies that the host sends a valid getEventHandlerLog command.

Objective

Verify the getEventHandlerLog command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getEventHandlerLog command from the host.**
   {01ceb247-db69-423c-917f-5f7278285300}

   #### Command

   - `eventHandler.getEventHandlerLog`

   #### Actions

   - Instruct the user to 'Send getEventHandlerLog to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00031

This case verifies that the host sends a valid getSupportedEvents command.

Objective

Verify the getSupportedEvents command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Instruct user to send getSupportedEvents command from the host.**
   {0152a56d-3a2b-4fb3-923a-0ad3ac6c04b3}

   #### Command

   - `eventHandler.getSupportedEvents`

   #### Actions

   - Instruct the user to 'Send getSupportedEvents to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: EH-COR-00032

This case verifies that the host does not process event handler commands with the wrong session type.

Objectives

- Verify that the host does not process a response sent as a request.
- Verify that the host sends G2S_APX008 Command Not Supported or G2S_AXP015 Unknown Command Encountered error for an unknown cabinet command.
- Verify that if the host sends G2S_AXP015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.4.4
- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Send a eventSubList as a request and verify G2S_APX008 Command Not Supported error code.**
   {01481648-f8cc-45d2-ab02-72846588bd06}

   Command
   - `eventHandler.eventSubList`

     Expect **G2S_APX008**

   Error
   - Send Request Errors

2. **Send a eventReport.cvtTesting command and verify G2S_APX008 Command Not Supported or G2S_APX015 Unknown Command Encountered error.**
   {02622211-0629-4ee7-953c-41bbc4657179}

Command

- `eventHandler.cvtTesting`

  Expect **G2S_APX008** or **G2S_APX015**

Errors

- Send Request Errors
- `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: EH-COR-00033

This case verifies that the host handles error responses to host eventHandler requests.

Objectives

- Verify that the host is still working after responding with a eventSubList to a setEventSub command.
- Verify that the host is still working after responding with a setEventSub response to a setEventSub command.
- Verify that the host is still working after responding with a setEventSubAck notification to a setEventSub command.
- Verify that the host is still working after responding with a unknown error code to a setEventSub command.

Requirements Under Test

- 1.4.5
- 1.4.6
- 1.4.7
- 4.5.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: HOST

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager to respond with a eventSubList response to a setEventSub request.**
   {0137eed2-688d-41fc-aa5d-17dbf41a446f}
2. **Instruct user to send setEventSub command from the host.**
   {0271b15a-d02f-4f0d-bc8a-5b115285647e}

   Command
   - `eventHandler.setEventSub`

Actions

- Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId} (1 of 4).'.

Errors

- Expected Request Errors
- DUT Error

3. **Set the response manager to respond with a setEventSub response to a setEventSub request.**
{03115ed7-d544-4039-8c57-3815311554b7}

4. **Instruct user to send setEventSub command from the host.**
{04c9fc9c-4296-4dd3-b245-c4597101f86d}

Command

- `eventHandler.setEventSub`

Actions

- Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId} (2 of 4).'.

Errors

- Expected Request Errors
- DUT Error

5. **Set the response manager to respond with a eventSubAck notification to a setEventSub request.**
{054d390b-9160-48e4-881a-d81a4e9f9234}

6. **Instruct user to send setEventSub command from the host.**
{06fa4ce4-f072-4898-8e02-4e63bd1cad75}

Command

- `eventHandler.setEventSub`

Actions

- Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId} (3 of 4).'.

Errors

- Expected Request Errors
- DUT Error

7. **Set the response manager to respond with 'CVT_ error' error code to a setEventSub request.**
{07190843-4aeb-47c8-99f3-6595dc909019}

8. **Instruct user to send setEventSub command from the host.**
{08e0ec0a-6d48-4bde-a1df-b16129b27d34}

   Command
   - `eventHandler.setEventSub`

   Actions
   - Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId} (4 of 4).'.

   Errors
   - Expected Request Errors
   - DUT Error

9. **Reset the response manager.**
{095f1662-37da-4c6e-9b9d-8c9a7f0095ac}

10. **Send an eventReport to the host to verify that it is still working.**
{10878670-ba4f-45cb-9e80-2375eccbe879}

    Command
    - `eventHandler.eventReport`

    Error
    - Send Request Errors

Test Case: EH-COR-00034

This case verifies that the host sends the proper error code for an invalid eventReport request.

Objective

Verify that the host sends G2S_MSX004 Incomplete/Malformed XML, G2S_MSX005 Invalid Data Type Encountered or G2S_APX004 Incomplete/Malformed XML error for an invalid eventReport.

Requirements Under Test

- 1.27.33
- 1.41.2
- 1.41.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Send an invalid eventReport with a missing transactionId attribute and verify G2S_MSX004, G2S_MSX005 or G2S_APX004 error code.**
   {01a9e41d-e4fe-4b93-bbd4-c6605e3c6661}

   Command
   - `eventHandler.eventReport`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Error
   - Send Request Errors

Test Case: EH-COR-00035

This case tests that the host processes duplicate eventReport commands and eventReports for invalid device IDs.

Objectives

- Verify that the host processes duplicate eventReport commands.
- Verify that the host sends five eventAcks for five duplicate eventReports.
- Verify that the host sends an eventAck for an eventReport for an invalid device ID.

Requirements Under Test

- 1.28.9
- 4.5.1
- 4.21.3
- 4.21.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Send five duplicate eventReports commands in one G2S message. Verify the host sends five eventAcks.**
   {0124fba8-6d97-40f6-b739-f80abc1743f3}

   Command
   - `eventHandler.eventReport`

   Error
   - Send Request Errors

2. **Send eventReport for an unknown device ID and verify that the host responds with eventAck.**
   {02b24338-814f-4c9f-8e3f-21d375ef090a}

   Command

- `eventHandler.eventReport`

Error

- Send Request Errors

Test Case: EH-COR-00036

This case tests that the host can handle a delayed g2sAck in the event handler class.

Objective

Verify that the host is still working after delaying the g2sAck in the event handler request.

Requirements Under Test

- 1.30.7
- 4.5.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager delay the g2sAck for setEventSub command.**
   {0168b812-a8c4-4715-bc54-b0e6d3fe27f6}

2. **Instruct user to send setEventSub command from the host.**
   {02686c1d-de72-475d-bfe1-b6de9d7fce0c}

   #### Command

   - `eventHandler.setEventSub`

   #### Actions

   - Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {036c7fb5-b9d8-46cc-a9ef-adeb75c3be23}

4. **Send a eventReport to the host to verify that it is still working.**
   {04b10319-1362-40da-b632-eccd0e6072f4}

   #### Command

- `eventHandler.eventReport`

Error

- Send Request Errors

Test Case: EH-COR-00037

This case tests that the host handles unknown error codes in eventHandler class.

Objective

Verify that the host is still working after sending an unknown error code in the eventHandler class.

Requirements Under Test

- 1.42.3
- 4.5.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: eventHandler

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager to send 'CVT_ error' for setEventSub command.**
   {019087e2-3a2a-4100-80cb-62ee85789c00}

2. **Instruct user to send setEventSub command from the host.**
   {0233c759-1ed3-4493-9474-dc97dc0009f0}

   Command
   - `eventHandler.setEventSub`

   Actions
   - Instruct the user to 'Send setEventSub to device ${deviceUnderTest.deviceId}.'.

   Errors
   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {036e13ea-8944-432d-888f-ada35181fdf5}

4. **Send a eventReport to the host to verify that it is still working.**
   {046288a8-c609-4ce5-b635-f0fa1507b49f}

   Command

- `eventHandler.eventReport`

Error

- Send Request Errors

Test Case: EH-COR-00038

This case verifies that the EGM handles forced subscriptions properly.

Objectives

- Verify that the EGM treats forced subscriptions as if set by the host.
- Verify that the EGM "ORs" host subscriptions and forced subscriptions.
- Verify that the EGM generates `G2S_EHE005 Device Configuration Changed by Host` when a forced subscription is set.

Requirements Under Test

- 1.23.25
- 1.23.26
- 4.27.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_optionConfig	owner
G2S_cabinet	guest or owner

Test Procedure

1. **Clear the event subscription for G2S_CBE307 Cabinet Door Open and G2S_CBE316 Cash-Out Button Pressed.**
   {01289355-7bcc-4d0f-8a5e-463cf3bf289c}

   Command
   - `eventHandler.clearEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Error
   - Event Checking Errors

2. **Set a host subscription for G2S_CBE307 to send device meters.**
{0201ae93-aced-441e-a068-0082f1729337}

Command
- `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

3. **Set a forced subscription for G2S_CBE307 to send device status and G2S_CBE316.**
{03add903-fd13-418f-931e-9f8d6db10a34}

Command
- `optionConfig.setOptionChange`

Event
(Time out: ${event.timeOut})
- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

4. **Instruct the user to press the cash out button and verify G2S_CBE316 event is generated.**
{04b74845-1cd2-4b78-af28-f236589348c0}

Action
- Instruct the user to 'Please press the cash out button.'.

Event
(Time out: ${event.timeOut})
- `G2S_CBE316` [Cash-Out Button Pressed]

Errors
- Event Checking Errors
- DUT Error

5. **Instruct user to open the cabinet door and verify that G2S_CBE307 has device status and device meters.**
{05415f34-8469-4412-a6ce-58ca348c83e9}

Action

- Request CABINET door OPEN.

Event
(Time out: ${event.timeOut})

- `G2S_CBE307` [Cabinet Door Open]

Errors

- Event Checking Errors
- DUT Error

6. **Instruct the user to close the cabinet door.**
{0692fbaf-1818-4d6a-9d1f-0beb4a694de1}

Action

- Request CABINET door CLOSE.

Event
(Time out: ${event.timeOut})

- `G2S_CBE308` [Cabinet Door Closed]

Errors

- Event Checking Errors
- DUT Error

Test Case: EH-COR-00039

This case verifies that the EGM sends the prescribed Event Handler response to each Event Handler request.

Objectives

- Verify that the EGM sends a `eventHandlerProfile` response to a `getEventHandlerProfile` request.
- Verify that the EGM sends a `setEventSubAck` response to a `setEventSub` request.
- Verify that the EGM sends a `eventSubList` response to a `getEventSub` request.
- Verify that the EGM sends a `clearEventSubAck` response to a `clearEventSub` request.
- Verify that the EGM sends a `eventHandlerStatus` response to a `getEventHandlerStatus` request.
- Verify that the EGM sends a `eventHandlerStatus` response to a `setEventHandlerState` request.
- Verify that the EGM sends a `eventHandlerLogStatus` response to a `getEventHandlerLogStatus` request.
- Verify that the EGM sends a `eventHandlerLogList` response to a `getEventHandlerLog` request.
- Verify that the EGM sends a `supportedEvents` response to a `getSupportedEvents` request.

Requirements Under Test

- 4.5.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Send a getEventHandlerProfile request to the EGM and verify eventHandlerProfile response.**
{01e07dd9-d6db-4670-a557-1af7b96e2a06}

   Command
   - `eventHandler.getEventHandlerProfile`

   Error
   - Send Request Errors

2. **Send a clearEventSub request to the EGM and verify clearEventSubAck response.**
{02f7758f-a65f-4684-b521-6402eec52c6d}

   Command
   - `eventHandler.clearEventSub`

   Error
   - Send Request Errors

3. **Send a setEventSub request to the EGM and verify setEventSubAck response.**
{030565a5-d86a-402f-b7d2-ffb1be3c0e9f}

   Command
   - `eventHandler.setEventSub`

   Error
   - Send Request Errors

4. **Send a getEventSub request to the EGM and verify eventSubList response.**
{041c8d09-2daf-4d1c-8339-d73a631b8bd8}

   Command
   - `eventHandler.getEventSub`

   Error
   - Send Request Errors

5. **Send a getEventHandlerStatus request to the EGM and verify eventHandlerStatus response.**
{05b4edd9-2f87-4458-aad2-7c2800ebc6ed}

   Command
   - `eventHandler.getEventHandlerStatus`

   Error
   - Send Request Errors

6. **Send a setEventHandlerState request to the EGM and verify eventHandlerStatus response.**
{0621bc20-8b76-4e1b-9e9f-6b59150a8a07}

Command
- `eventHandler.setEventHandlerState`

Error
- Send Request Errors

7. **Send a getEventHandlerLogStatus request to the EGM and verify eventHandlerLogStatus response.**
{078e079d-cad6-441b-9be5-c6f1ab56dbe7}

Command
- `eventHandler.getEventHandlerLogStatus`

Error
- Send Request Errors

8. **Send a getEventHandlerLog request to the EGM and verify eventHandlerLog response.**
{0862990a-aa51-4761-8c72-70ccc2baaa04}

Command
- `eventHandler.getEventHandlerLog`

Error
- Send Request Errors

9. **Send a getSupportedEvents request to the EGM and verify supportedEvents response.**
{09de3c12-37f3-437b-8a55-e67ccfd3bd54}

Command
- `eventHandler.getSupportedEvents`

Error
- Send Request Errors

Test Case: EH-COR-00040

This case continuously tests that only associated data that should be sent to the host are sent to the host in event data.

Objectives

- Continuously verify that `eventReport`s are for devices that the CVT is an owner or guest of.
- Continuously verify that affected `statusInfo` elements in `eventReport`s are for devices that the CVT is an owner or guest of.
- Continuously verify that affected `transactionInfo` elements in `eventReport`s are for devices that the CVT is an owner or guest of.

Requirements Under Test

- 4.15.8

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Assert that the CVT is a guest or owner of the device that generated the event.**
   {01abf29a-5151-4a70-b4aa-bdba76f876e9}
2. **Assert that the CVT is a guest or owner of devices in the statusInfo elements.**
   {025014bd-a8bd-499d-964f-07648009fa3c}
3. **Assert that the CVT is a guest or owner of devices in the transactionInfo elements.**
   {0351a9af-a129-481f-96ad-f5aef6d3f1ba}

Test Case: EH-COR-00041

This case verifies that the EGM generates `G2S_EHE006 Device Configuration Changed by Operator` event.

Objective

Verify that the EGM generates `G2S_EHE006` event when the event handler device is changed by the operator.

Requirements Under Test

- 4.28.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Determine if the EGM has a option config device.**
   {0119e5c0-2331-46f9-aca9-464815712bb9}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **If the EGM has an option config device.**

   a. **Determine if the EGM supports local changes to the event handler device.**
   {0201569d-a0e8-431c-b837-48c46c6d5b31}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

3. **If the EGM supports local changes.**

    a. **Instruct user to make a local change to the event handler device and verify G2S_EHE006 event is generated.**
    {0316ee55-0755-4b61-a9aa-e2d9358bc6b1}

    Action

    - Instruct the user to 'Please make a configuration change to event handler device ${deviceUnderTest.deviceId}'.

    Event

    (Time out: ${event.timeOut})

    - `G2S_EHE006` [Event Handler Configuration Changed by Operator]

    Errors

    - Event Checking Errors
    - DUT Error

Test Case: EH-COR-00042

This case verifies that the EGM persists event subscriptions.

Objective

Verify that the EGM has the same event subscription after a restart.

Requirements Under Test

- 1.23.29

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Clear the event subscription.**
   {01f00806-7c0f-422f-a136-08eb275cca5d}

   Command
   - `eventHandler.clearEventSub`

2. **Set the event subscription for a G2S_EHE001 Device Disabled by EGM.**
   {02dc1bc2-48df-480d-8480-2d6a0b66a73b}

   Command
   - `eventHandler.setEventSub`

3. **Instruct the user to power cycle the EGM.**
   {0396a12b-b1a9-448a-9c5b-b51d745f4cf0}

   Actions
   - Instruct the user to 'Please power cycle the EGM.'.

   Errors
   - DUT Error
   - `E-CM-00057` [The user failed to power cycle the EGM.]

4. **Wait for the EGM to reconnect to the host.**
{04957a5c-32e1-4aa8-b19b-a2d32333ff77}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Error
   - Expected Request Errors

5. **Verify that the event subscription was persisted.**
{05c32616-69be-440b-ab75-a5cba3f96d74}

   Command
   - `eventHandler.getEventSub`

   Errors
   - Send Request Errors
   - `E-EH-00063` [Event subscriptions must be persisted.]

Test Case: EH-COR-00043

This case verifies that the EGM supports required for play properly for Event Handler devices.

Objectives

- Verify that if the Event Handler device is `requiredForPlay`, the EGM is disabled when the event handler device is disabled.
- Verify that games are not playable if a `requiredForPlay` event handler device is disabled.

Requirements Under Test

- 1.37.1
- 4.12.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
eventHandlerProfile.requiredForPlay	true

Test Procedure

1. **Instruct the user to fund the EGM.**
   {01c843c8-ae0e-4940-b3f3-fe26198def90}

   Action
   - Instruct the user to 'Please fund the EGM.'.

   Errors
   - DUT Error
   - `E-XX-00022` [The user failed to fund the EGM.]

2. **Host disable the event handler device and verify that the event handler device is disabled.**
   {02a4db35-980e-4446-99b6-deb9924cce54}

Command

- `eventHandler.setEventHandlerState`

Errors

- Send Request Errors
- Event Not Expected Error
- `E-EH-00064` [Event handler status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Verify that the EGM is disabled.**
{0378a307-8365-4bbc-80c3-e58ed56cfef9}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

4. **Instruct user to verify that game play has been disabled.**
{04413d43-2957-48ef-9a11-9bfb01c792dc}

Action

- Instruct the user to 'Please verify that game play has been disabled. Select 'False' if game is playable.'.

Errors

- DUT Error
- `E-CB-00033` [While host locked, the EGM must not allow ${activity}.]

5. **Host enable the event handler device and verify expected events are generated.**
{05bfcbd7-c86b-476e-9e38-57b29ee9dbda}

Command

- `eventHandler.setEventHandlerState`

Events
(Time out: ${event.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: EH-COR-00044

This case verifies that non-persisted events are not stored in the event log.

Objective

Verify that non-persisted events are not stored in the event log.

Requirements Under Test

- 4.4.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Clear the event subscription.**
   {011cb62d-beeb-4afe-96dc-e953835c50bb}

   Command
   - `eventHandler.clearEventSub`

2. **Set the event subscription for a G2S_EHE004 Device Not Disabled by Host.**
   {02989b1d-09ce-4646-b0ce-79b3257e8f83}

   Command
   - `eventHandler.setEventSub`

3. **Get the latest event from the event handler log.**
   {030b9734-c5af-4448-a2ad-61b35d8b33ef}

   Command
   - `eventHandler.getEventHandlerLog`

   Error
   - Send Request Errors

4. **Host disable the event handler device.**
   {04f209f4-ddef-4ac0-b9e8-8b80686b4ab3}

Command

- `eventHandler.setEventHandlerState`

Error

- Send Request Errors

5. **Host enable the event handler device.**
{05ae5214-d988-42c8-a3ed-e40f1ed341fb}

Command

- `eventHandler.setEventHandlerState`

Event

(Time out: ${event.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors

6. **Verify that the last event report has not changed in the event handler log.**
{062992b7-d4f3-4d35-8cae-77bb4845bb4c}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00065` [Non-persisted events MUST NOT be stored in the event handler log.]

Test Case: EH-COR-00045

This case verifies that EGM continues to send events when the event queue is full.

Objective

Verify that EGM sends event reports to the host even if the event queue is full.

Requirements Under Test

- 4.4.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_cabinet	guest or owner

Test Procedure

1. **Determine the event handler queue size and queue behavior.**
   {013b5c05-3314-4a9d-bd2b-134b780b0401}

   Command
   - `eventHandler.getEventHandlerProfile`

   Error
   - Send Request Errors

2. **Set the response manager to suppress responses to eventReport commands.**
   {02c3c1c8-d600-45e6-a748-e508d8a83f54}

3. **While the event queue is not full.**

   1. **Clear the event subscription for G2S_EHE003.**
      {03266ef4-db1d-4568-a6a7-f399ef066832}

      Command
      - `eventHandler.clearEventSub`

   2. **Set an event subscription for G2S_EHE003.**
      {0469751e-ef5b-4533-8c51-77d2719467d6}

Command
- `eventHandler.setEventSub`

3. **Check to see if the event handler queue is full.**
{056d54bd-bf4a-423f-99be-4de3d969a35f}

Command
- `eventHandler.getEventHandlerLog`

Error
- Send Request Errors

4. **Verify event queue is full.**
{067a934a-334a-44dc-bdd9-05f73a62d97d}

Command
- `eventHandler.getEventHandlerLog`

Errors
- Send Request Errors
- `E-EH-00066` [Event queue is not full.]

5. **Verify that new events are sent to the host.**
{07983ef0-3f8e-421f-ad64-9c799bfe5d16}

Action
- Cash Out.

Event
(Time out: ${event.timeOut})
- `G2S_CBE316` [Cash-Out Button Pressed]

Errors
- Event Checking Errors
- DUT Error

6. **Set the response manager to respond to eventReport commands.**
{08575d32-4e2a-4b97-a3fb-cf20c3569ba1}

7. **While the queue has unack'd event reports.**

1. **Check for unack'd event reports.**
{0975af39-5bf3-41ce-a27f-29d521d6e8c9}

Command
- `eventHandler.getEventHandlerLog`

Error

- Send Request Errors

Test Case: EH-COR-00046

This case verifies that EGM persists `minLogEntries` in the event log.

Objective

Verify that EGM persists `minLogEntries` in the event log.

Requirements Under Test

- 4.12.5

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Determine the event handler queue size.**
   {0110e2a2-f515-4343-a8cf-bafc2b8a5b21}

   Command
   - `eventHandler.getEventHandlerProfile`

   Error
   - Send Request Errors

2. **Check to see if the event handler queue is full.**
   {0215eece-22e4-4165-8858-eace14e86883}

   Command
   - `eventHandler.getEventHandlerLogStatus`

   Error
   - Send Request Errors

3. **While the event queue is not full.**

   1. **Clear the event subscription for G2S_EHE003.**
      {0347d12c-a96d-4792-9050-e39a3b77ef06}

Command

- `eventHandler.clearEventSub`

2. **Set an event subscription for G2S_EHE003.**
{04b47715-c0a3-4932-93e2-3c6db741f973}

Command

- `eventHandler.setEventSub`

3. **Check to see if the event handler queue is full.**
{05911519-cf79-4c1f-9223-5bbf071d65e2}

Command

- `eventHandler.getEventHandlerLogStatus`

Error

- Send Request Errors

4. **Clear the event subscriptions**
{066d13bf-4ef5-4879-8dab-6efd08b754ec}

Command

- `eventHandler.clearEventSub`

5. **Verify event queue is full.**
{07501efb-758d-4f7f-bebd-501bc3d04998}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00066` [Event queue is not full.]

6. **Instruct user to power cycle the EGM.**
{080f08f8-469a-4aa5-a6f4-9c9a01d400be}

Actions

- Instruct the user to 'Please power cycle the EGM'.

Errors

- DUT Error
- `E-CM-00057` [The user failed to power cycle the EGM.]

7. **Wait for the EGM to reconnect to the host.**
{095e299c-746d-4f81-b0e6-1154d2b916b6}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Error

- Expected Request Errors

8. **Verify that the event log was persisted.**
{10a31b06-fbff-4d30-be23-0684e448542f}

Command

- `eventHandler.getEventHandlerLogStatus`

Errors

- Send Request Errors
- `E-EH-00067` [Event log MUST be persisted.]

Test Case: EH-COR-00047

This case verifies that `clearEventSub` does not clear forced event subscriptions.

Objective

Verify that EGM does not clear forced event subscription when it receives `clearEventSub`.

Requirements Under Test

- 4.19.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_optionConfig	owner

Test Procedure

1. **Set a forced event subscription for G2S_EHE101 Event Subscription Changed.**
   {01d91d45-b754-4ca1-8615-3f5265a3b1d1}

   Command
   - `optionConfig.setOptionChange`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE005` [Event Handler Configuration Changed by Host]

   Errors
   - Event Checking Errors
   - `E-XX-00002` [CVT timed out waiting for the response.]
   - `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
   - `E-XX-00005` [An unexpected error of ${errorCode} was received.]

2. **Clear all of the host subscriptions.**
   {021f0293-c0c4-4a0c-8607-13788eed0b58}

   Command
   - `eventHandler.clearEventSub`

Event

(Time out: ${event.timeOut})

- `G2S_EHE101` [Event Subscription Changed]

Error

- Event Checking Errors

Test Case: EH-COR-00048

This case verifies that EGM sends `G2S_EHE001 Device Disabled by EGM` and `G2S_EHE002 Device Not Disabled by EGM` events.

Objectives

- Verify that EGM sends `G2S_EHE001` when the event queue is full and the `queueBehavior` is `G2S_disable`.
- Verify that EGM sends `G2S_EHE002` when the event queue is no longer full and the `queueBehavior` is `G2S_disable`.

Requirements Under Test

- 4.23.1
- 4.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Required Configurations

Option	Value
eventHandlerProfile.queueBehavior	G2S_ disable

Test Procedure

1. **Clear all of the host subscriptions.**
   {01f679b6-9a9f-4c76-8d60-462d01b86766}

   Command
   - `eventHandler.clearEventSub`

2. **Set a host event subscription for G2S_EHE001, G2S_EHE002, and G2S_EHE101**
   {0206dee7-15a2-465c-a6b5-f2b42310b50a}

   Command
   - `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

3. **Determine the event handler queue size.**
{03f5c6b7-0667-460c-b387-281a1cd78a3d}

Command
- `eventHandler.getEventHandlerProfile`

Error
- Send Request Errors

4. **Set the response manager to suppress responses to eventReport commands.**
{040ea511-967b-4faf-a0f2-5d498cef59be}

5. **While the event queue is not full.**

    1. **Clear the event subscription for G2S_EHE003.**
    {0519eec2-c1cd-4b8d-8a63-c3501ee20fff}

    Command
    - `eventHandler.clearEventSub`

    2. **Set an event subscription for G2S_EHE003.**
    {06c8d807-caa1-4641-abd6-23e7d25cb15c}

    Command
    - `eventHandler.setEventSub`

    3. **Check to see if the event handler queue is full.**
    {07f5f9ab-2e56-4a80-9b41-6eff5195b60f}

    Command
    - `eventHandler.getEventHandlerLogStatus`

    Error
    - Send Request Errors

6. **Verify that the event handler device is EGM disabled and G2S_EHE001 was sent.**
{087d2d66-dcb7-4b47-ba38-44a075d16813}

Command
- `eventHandler.getEventHandlerStatus`

Errors

- Send Request Errors
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]
- `E-EH-00064` [Event handler status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

7. **Set the response manager to respond to eventReport commands.**
{0920ad95-ce62-49e0-b5dc-0ed7a8ca2faa}

8. **While the queue has unack'd event reports.**

    1. **Check for unack'd event reports.**
    {100291f7-b97e-40ed-bea5-82926efb86e1}

    Command

    - `eventHandler.getEventHandlerLog`

    Error

    - Send Request Errors

9. **Get the latest event report and verify that it is G2S_EHE002.**
{113b335c-b3d8-4f2a-843c-c8f810ae9173}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]

Test Case: EH-COR-00049

This case verifies that the EGM does not allow the setting of an event subscription for devices from a non-guest host.

Objectives

- Verify that EGM does not have a host event subscription for a device that the CVT does not have permission for.
- Verify that EGM sends `G2S_EHX001 Invalid Subscription` error code for an event subscription for a device that the CVT does not have permission for.

Requirements Under Test

- 1.8.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Get the descriptor list to determine if there is a device that the CVT does not have permission to access.**
   {0151b270-48ab-4d6e-97f4-1e2a16b42162}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **If there is a device that the CVT does not have permission to access.**

   a. **Verify that there is no host subscription for a device that the CVT does not have access to.**
   {02a2fcfd-e284-4dd0-b667-cc55bbb1f533}

   Command

- `eventHandler.getEventSub`

Errors

- Send Request Errors
- `E-EH-00054` [EGM must not have a subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]

b. **Set an host event subscription and verify that the EGM responds with G2S_EHX001 error code.**
{034783bf-465f-4b52-8093-7bd9dcb4944c}

Command

- `eventHandler.setEventSub`

  Expect **G2S_EHX001**

Error

- Send Request Errors

Test Case: EH-COR-00050

This case verifies that the EGM persists the log sequence number for the event handler log.

Objective

Verify that EGM persists the log sequence number for the the event handler log.

Requirements Under Test

- 1.19.10

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner

Test Procedure

1. **Get the last sequence number of the event handler log.**
   {01a6b6ce-8398-4de4-8003-4ac7a6e2a93b}

   Command
   - `eventHandler.getEventHandlerLogStatus`

   Error
   - Send Request Errors

2. **Instruct the user to power cycle the EGM.**
   {02647618-c0c1-4ecc-bf8c-9bc21270c904}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Actions
   - Instruct the user to 'Please power cycle the EGM.'.

   Errors
   - Expected Request Errors
   - DUT Error

- `E-CM-00057` [The user failed to power cycle the EGM.]

3. **Verify that the EGM persisted the log sequence number for the event handler log.**
{03af3418-5c04-4a9a-a10a-89966cef02f6}

Command

- `eventHandler.getEventHandlerLogStatus`

Errors

- Send Request Errors
- `E-XX-00024` [Log sequence number for ${deviceClass} MUST be persisted.]

Test Case: EH-COR-00051

This case verifies that the event handler log has a separate sequence number.

Objective

Verify that the event handler log has a separate sequence number.

Requirements Under Test

- 1.19.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_gamePlay	guest or owner

Test Procedure

1. **Get the last sequence number of the event handler log.**
   {01a94810-5b96-4e2b-ac78-1bdfbc7528ca}

   Command
   - `eventHandler.getEventHandlerLogStatus`

   Error
   - Send Request Errors

2. **Get the last sequence number of the game play recall log.**
   {02c74f6a-94e2-426a-8ef7-93b853e2eca3}

   Command
   - `gamePlay.getRecallLogStatus`

   Error
   - Send Request Errors

3. **Clear the host event subscription for G2S_EHE103.**
   {03d58ead-115f-4cd3-8325-887ce504851a}

   Command

- `eventHandler.clearEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

4. **Verify that the event handler log sequence number has incremented by one.**
{04708d63-3b0d-4d0e-8803-f2722df5df45}

Command
- `eventHandler.getEventHandlerLogStatus`

Errors
- Send Request Errors
- `E-XX-00021` [The ${deviceClass} log sequence number MUST increment by one for a new log entry.]

5. **Verify that the recall log sequence number has not changed.**
{051fe958-da3f-4b3c-8380-de156dd94941}

Command
- `gamePlay.getRecallLogStatus`

Errors
- Send Request Errors
- `E-XX-00025` [Each log MUST have its own sequence number.]

Test Case: EH-COR-00052

This case verifies that the EGM evaluates the event subscription at the time the event is generated.

Objectives

- Verify that the EGM evaluates the event subscription at the time the event is generated.
- Verify that the EGM persists the affected data when the event is generated.

Requirements Under Test

- 1.23.1
- 1.23.23

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_cabinet	guest or owner

Test Procedure

1. **Clear the event subscription for cabinet door open.**
   {01a026ff-7172-410d-9587-5ad790ae8efb}

   Command
   - `eventHandler.clearEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Error
   - Event Checking Errors

2. **Set the event subscription to include the cabinet status in the cabinet door open event.**
   {0255938f-d272-4d27-83d3-bc44cf9d5f3b}

   Command
   - `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

3. **Instruct the user to open the cabinet door.**
{03fc72f3-cede-45c0-bbee-0c02481c1aef}

Action
- Request CABINET door OPEN.

Event
(Time out: ${event.timeOut})
- `G2S_CBE307` [Cabinet Door Open]

Errors
- Event Checking Errors
- DUT Error
- `E-XX-00011` [An unknown exception ${exceptionMessage} has caused the step to fail (${exceptionId}).]

4. **Instruct the user to close the cabinet door.**
{0487e18a-d6fa-4d11-9e98-023a915f51a7}

Action
- Request CABINET door CLOSE.

Error
- DUT Error

5. **Clear the event subscription for cabinet door open.**
{058f0623-24c0-45d6-990d-cae2fafcd8a6}

Command
- `eventHandler.clearEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Error
- Event Checking Errors

6. **Set the event subscription to include the cabinet device meter set in the cabinet door open event.**
{0664c97a-fb84-4e84-9334-1d6cc2123bab}

   Command
   - `eventHandler.setEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Error
   - Event Checking Errors

7. **Instruct the user to open the cabinet door.**
{07df1e4b-7185-474e-afbe-7f269bd4ebc1}

   Action
   - Request CABINET door OPEN.

   Event
   (Time out: ${event.timeOut})
   - `G2S_CBE307` [Cabinet Door Open]

   Errors
   - Event Checking Errors
   - DUT Error
   - `E-XX-00011` [An unknown exception ${exceptionMessage} has caused the step to fail (${exceptionId}).]

8. **Instruct the user to close the cabinet door.**
{08090ad1-a611-4242-bad4-ff538cb8a445}

   Action
   - Request CABINET door CLOSE.

   Error
   - DUT Error

9. **Verify that the first G2S_CBE307 has the cabinet device in the affected device list and the second G2S_CBE307 has the cabinet device in the affected meter list.**
{098cff81-df2a-4a82-a9fa-7f6c2366360f}

   Command
   - `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-EH-00006` [Event ${eventCode} is missing affected data.]

Test Case: EH-COR-00053

This case verifies that the EGM persists forced event subscriptions when the event handler device is host disabled.

Objectives

- Verify that the EGM persisted forced event subscriptions that have forcePersist set to true when the event handler is host disabled.
- Verify that the EGM sends the forced event when the event handler is host enabled.

Requirements Under Test

- 4.7.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_cabinet	guest or owner
G2S_optionConfig	owner

Test Procedure

1. **Set a persisted forced event subscription for cash-out button pressed event.**
   {018f82e1-c9a9-46ee-99de-3b74c62dce4a}

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE005` [Event Handler Configuration Changed by Host]

   Error
   - Event Checking Errors

2. **Host disable the event handler device.**
   {021d5170-273c-4375-9899-ed918c3140cf}

   Command
   - `eventHandler.setEventHandlerState`

Error

- Send Request Errors

3. **Instruct user to press the cash-out button.**
{03889a80-8355-45b1-a3ae-aaf4c6e5b52d}

Action

- Cash Out.

Errors

- Event Not Expected Error
- DUT Error
- `E-XX-00026` [The user failed to press the cash-out button.]

4. **Host enable the event handler device and verify that the G2S_CBE316 Cash-Out Button Pressed event is sent to the host.**
{04348c78-49c3-466b-a33b-539c58d57eaf}

Command

- `eventHandler.setEventHandlerState`

Events

(Time out: ${event.timeOut})

- `G2S_CBE316` [Cash-Out Button Pressed]
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: EH-COR-00054

This case verifies that the EGM is disabled when a required for play event handler device is disabled.

Objectives

- Verify that the EGM is disabled when a required for play event handler device is host disabled.
- Verify that the EGM is disabled when a required for play event handler is disabled by the EGM.

Requirements Under Test

- 4.8.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
eventHandlerProfile.requiredForPlay	true

Test Procedure

1. **Host disable the event handler device.**
   {01fd07b3-846d-4183-9155-fb2dbbe71e66}

   Command
   - `eventHandler.setEventHandlerState`

   Errors
   - Send Request Errors
   - Event Not Expected Error

2. **Verify that the EGM is host disabled by the event handler device.**
   {02dc090c-2bc9-4659-80e8-9cd0ffb12f0e}

   Command
   - `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Host enable the event handler device.**
{035f78b7-eb08-4914-b75d-8e054aaf5c9e}

Command

- `eventHandler.setEventHandlerState`

Events

(Time out: ${event.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors

4. **Get the event handler profile.**
{04ad0a93-81ba-48a0-9757-07ce560c3264}

Command

- `eventHandler.getEventHandlerProfile`

Error

- Send Request Errors

5. **If the event handler queue behavior is G2S_disable.**

   a. **Clear all of the host subscriptions.**
   {05a213a1-b3de-43ea-b0e7-aedcb65c21c2}

   Command

   - `eventHandler.clearEventSub`

   b. **Set a host event subscription for G2S_EHE001, G2S_EHE002, and G2S_EHE101**
   {06ed6318-f20a-422d-b275-6e73cf03a4c9}

   Command

   - `eventHandler.setEventSub`

   Event

   (Time out: ${event.timeOut})

   - `G2S_EHE101` [Event Subscription Changed]

Error

- Event Checking Errors

c. **Set the response manager to suppress responses to eventReport commands.**
{0706cadf-c336-419e-8638-2b6c1b40464d}

6. **While the event queue is not full.**

1. **Clear the event subscription for G2S_EHE003.**
{08f341a6-18aa-4840-b663-a47cc4bb8d8e}

Command

- `eventHandler.clearEventSub`

2. **Set an event subscription for G2S_EHE003.**
{09149977-5b1a-4598-b6db-113acbd75038}

Command

- `eventHandler.setEventSub`

3. **Check to see if the event handler queue is full.**
{102278c1-0309-4e8a-933f-88b2bbdaf5c4}

Command

- `eventHandler.getEventHandlerLogStatus`

Error

- Send Request Errors

7. **If the event handler queue behavior is G2S_disable.**

a. **Verify that the EGM is host disabled by the event handler device.**
{11ef7616-a124-412f-9256-a312ffb35ae5}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: EH-COR-00055

This case verifies that the EGM processes eventAck responses even when the event handler device is disabled.

Objective

Verify that an eventAck responses is processed when the event handler device is disabled.

Requirements Under Test

- 4.8.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Set the response manager to stop responding to eventReport requests.**
   {01f9c5cc-111e-4462-9797-aaa51ddd350f}

2. **Clear the event subscription for host disabled event handler device and wait for G2S_EHE101 event.**
   {02996ccb-e7af-461c-a5a8-d86bf081ed24}

   #### Command

   - `eventHandler.clearEventSub`

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_EHE101` [Event Subscription Changed]

   #### Errors

   - Event Checking Errors
   - `E-XX-00011` [An unknown exception ${exceptionMessage} has caused the step to fail (${exceptionId}).]

3. **Host disable the event handler device.**
   {03811fbd-2d91-457c-bfe9-661661fedaa2}

Command
- `eventHandler.setEventHandlerState`

Errors
- Send Request Errors
- Event Not Expected Error

4. **Verify that the latest G2S_EHE101 log entry has eventAck set to false.**
{04268141-e627-45c1-a033-aa787b7e3cc5}

Command
- `eventHandler.getEventHandlerLog`

Errors
- Send Request Errors
- `E-EH-00069` [Event log attribute eventAck must be false for unacknowledged event reports.]

5. **Send the eventAck response to the G2S_EHE101 event.**
{05c1401b-d55d-40d1-a554-2c1dd7990bbe}

Command
- `eventHandler.eventAck`

Error
- Send Request Errors

6. **Verify that the latest G2S_EHE101 log entry has eventAck set to true.**
{062f8911-5f4c-4fee-b2dd-e9bb226ec337}

Command
- `eventHandler.getEventHandlerLog`

Errors
- Send Request Errors
- `E-EH-00070` [Event log attribute eventAck must be true for acknowledged event reports.]

7. **Host enable the event handler device.**
{07208d04-7ec0-4d6f-8992-843d7fbe29ca}

Command
- `eventHandler.setEventHandlerState`

Event
(Time out: ${event.timeOut})

- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors

8. **Reset the response manager.**
{088b555c-940f-4b49-a7f6-2d10d8ab3dd7}

Test Case: EH-COR-00056

This case verifies that the EGM can set a forced event subscription for all devices and all events.

Objectives

- Verify that a forced event subscription for `deviceClass` of `G2S_all` sets an event subscription for all classes.
- Verify that a forced event subscription for `deviceId` of `-1` sets an event subscription for all devices.
- Verify that a forced event subscription for `eventCode` of `G2S_all` sets an event subscription for all events.

Requirements Under Test

- 4.12.7
- 4.12.8
- 4.12.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the list of supported events.**
   {01ec252e-e94e-4974-a766-3859a2cb312d}

   Command
   - `eventHandler.getSupportedEvents`

   Error
   - Send Request Errors

2. **Set a persisted forced event subscription for all events.**
   {025abcb0-8deb-4652-b432-38a470ace825}

   Command

- `optionConfig.setOptionChange`

Event
(Time out: ${event.timeOut})

- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors

- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

3. **Verify that all events have a forced event subscription.**
{03c9379a-ffd3-4047-88a2-c105dd9230dc}

Command

- `eventHandler.getEventSub`

Errors

- Send Request Errors
- `E-EH-00071` [EGM MUST have a forced subscription for ${deviceClass}[${deviceId}], event code ${eventCode}.]

Test Case: EH-COR-00057

This case verifies that the EGM removes event subscription when the host is no longer allowed to have event subscriptions.

Objective

Verify that when the device ownership of cabinet device is changed such that the CVT is no longer the owner or guest of the cabinet device that cabinet event subscriptions are cleared.

Requirements Under Test

- 4.15.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_communications	owner
G2S_cabinet	guest or owner

Test Procedure

1. **Instruct user to change the ownership of the cabinet device.**
   {0153b9e9-8bf6-46a0-9ec5-2e2ddde67cc6}

   #### Commands

   - `communications.commsOnLine`
   - `communications.commsDisabled`

   #### Actions

   - Instruct the user to 'Please change the ownership of the cabinet device. The CVT MUST not be an owner or a guest of the cabinet device.'.

   #### Errors

   - Expected Request Errors
   - DUT Error
   - `E-CM-00072` [The user failed to change the device permission for ${device}.]

2. **Verify that the host event subscription for the cabinet device has been cleared.**
   {02ef6767-adc3-47d9-a4fe-84172864c40d}

Command

- `eventHandler.getEventSub`

Errors

- Send Request Errors
- `E-EH-00072` [EGM MUST clear the event subscriptions for a device when the host no longer owns nor is it a guest of that device.]

3. **Instruct user to reset the cabinet ownership and guest permissions to its original value.**
{036990f7-b8ca-41a2-a66a-24fcfe1cf681}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Actions

- Instruct the user to 'Please change the ownership of the cabinet device back to the original value.'.

Errors

- Expected Request Errors
- DUT Error
- `E-CM-00072` [The user failed to change the device permission for ${device}.]

Test Case: EH-COR-00058

This case verifies that the EGM generates `G2S_EHE003 Device Disabled by EGM` when the event handler device is host disabled.

Objective

Verify that when the event handler device is host disabled that the EGM generates G2S_EHE003 event.

Requirements Under Test

- 4.25.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: eventHandler
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner
G2S_optionConfig	owner

Test Procedure

1. **Instruct user to change the ownership of the cabinet device.**
   {01522606-0460-495e-ada0-aa72b62f5116}

   Command
   - `optionConfig.setOptionChange`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE005` [Event Handler Configuration Changed by Host]

   Errors
   - Event Checking Errors
   - `E-XX-00002` [CVT timed out waiting for the response.]
   - `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
   - `E-XX-00005` [An unexpected error of ${errorCode} was received.]

2. **Host disable the event handler device and verify that G2S_EHE003 is not sent while the event handler device is disabled.**
   {02eeb1b2-bafb-48fc-b484-c4259659d19f}

Command

- `eventHandler.setEventHandlerState`

Errors

- Send Request Errors
- Event Not Expected Error

3. **Host enable the event handler device and verify that G2S_EHE003 is sent.**
{03855b78-2db9-4e65-979d-5499ec6b9654}

Command

- `eventHandler.setEventHandlerState`

Events

(Time out: ${event.timeOut})

- `G2S_EHE003` [Event Handler Disabled by Host]
- `G2S_EHE004` [Event Handler Enabled by Host]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: EH-COR-00059

This case verifies that EGM sends `G2S_EHE102 Event Handler Queue Overflow` and `G2S_EHE103 Event Handler Queue Overflow Cleared` events.

Objectives

- Verify that EGM sends `G2S_EHE102` when the event queue is full and the `queueBehavior` is `G2S_overwrite`.
- Verify that EGM sends `G2S_EHE103` when the event queue is no longer full.

Requirements Under Test

- 4.30.1
- 4.31.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: eventHandler
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_eventHandler	owner

Test Procedure

1. **Clear all of the host subscriptions.**
   {01b63a6d-d668-4d35-b421-89e536091f66}

   Command
   - `eventHandler.clearEventSub`

2. **Set a host event subscription for G2S_EHE003, G2S_EHE101, G2S_EHE102, and G2S_EHE103**
   {02fc5f59-27a8-441d-816c-0e395ed71dae}

   Command
   - `eventHandler.setEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Error

- Event Checking Errors

3. **Determine the event handler queue behavior and queue size.**
{031a2183-d487-4362-a2aa-2406df14bf11}

Command
- `eventHandler.getEventHandlerProfile`

Error
- Send Request Errors

4. **Set the response manager to suppress responses to eventReport commands.**
{04b5df3e-d04e-42cd-b4f2-5fca69b9f2b6}

5. **While the event queue is not full.**

1. **Clear the event subscription for G2S_EHE003.**
{051b409d-143b-493b-8368-d0dcd06bcfc7}

Command
- `eventHandler.clearEventSub`

2. **Set an event subscription for G2S_EHE003.**
{069d783a-1ad5-488f-be04-e0550057c6b9}

Command
- `eventHandler.setEventSub`

3. **Check to see if the event handler queue is full.**
{072f9a3c-e32c-43a9-b114-97ea314e02a5}

Command
- `eventHandler.getEventHandlerLogStatus`

Error
- Send Request Errors

6. **If the eventHandlerProfile.queueBehavior is G2S_overwrite**

a. **Verify that the event handler device has eventHandlerOverflow set to true and G2S_EHE102 event was generated.**
{089272bb-1cd7-4324-ba4b-37f19552e7a8}

Command
- `eventHandler.getEventHandlerStatus`

Errors

- Send Request Errors
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]
- `E-EH-00064` [Event handler status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

7. **Set the response manager to respond to eventReport commands.**
{09c968e5-f42f-4cee-b4be-c0d77d81bdd3}

8. **While the queue has unack'd event reports.**

    1. **Check for unack'd event reports.**
    {1096395e-e48d-4ee8-ab0d-c03bf0079ab8}

    Command
    - `eventHandler.getEventHandlerLog`

    Error
    - Send Request Errors

9. **Get the latest event report and verify that it is G2S_EHE103.**
{117beb0b-c134-4fda-a3be-7d03852f4935}

Command
- `eventHandler.getEventHandlerLog`

Errors
- Send Request Errors
- `E-EH-00030` [Event ${eventCode} was expected, but it was not received.]

gamePlay

Game Play
Functional Groups

- Configure Accessible Games and Denominations (gtkGC) (AGD)
- Core Functionality (COR)
- Game Outcome Support (1k) (GOS)

Test Case: GP-AGD-00001

This case verifies that accessible game and denomination standard configuration options are available in `optionConfig`, and that they are consistent with the `gamePlayProfile` and `denomList`.

Objectives

- Verify that `gamePlayProfile.setAccessViaConfig` is consistent with `cabinetProfile.enhancedConfigMode`.
- Verify that the `optionConfig` parameter for option ID `G2S_setAccessViaConfig` is configured properly.
- Verify that the `optionConfig` parameter for option ID `G2S_accessibleGame` is configured properly.
- Verify that the `optionConfig` parameters for option ID `G2S_denomList` are configured properly.

Requirements Under Test

- 1.38.1
- 3.10.13
- 6.9.9
- 6.9.14
- 6.9.21

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_cabinet	guest or owner
G2S_optionConfig	owner

Test Procedure

1. **Get the `cabinet` profile values.**
   {01f49655-da94-4e4b-9e07-5bdfb7c97ef5}

   Command
   - `cabinet.getCabinetProfile`

   Error
   - Send Request Errors

2. **Get the `gamePlay` profile values.**
   {0232ba68-5ab4-4ca2-98b4-00f2666f7500}

   Command
   - `gamePlay.getGamePlayProfile`

   Errors
   - Send Request Errors
   - `E-CB-00021` [The ${class} profile attribute ${attribute} is wrong since cabinetProfile.enhancedConfigMode is false. It MUST be ${expected}, but it was ${actual}.]

3. **Get the `gamePlay` denomination list.**
   {03c42fd4-55c5-49b9-9de9-c45ef34e224d}

   Command
   - `gamePlay.getGameDenoms`

   Error
   - Send Request Errors

4. **Get the option list for the group ID `G2S_gamePlayOptions`.**
   {044ede78-779d-4532-bdb3-42b9cf5d68d5}

   Command
   - `optionConfig.getOptionList`

   Error
   - Send Request Errors

5. **Assert that `G2S_setAccessViaConfig` has the correct parameter configuration.**
   {05d0806c-fe60-406b-bc8a-7ab7eb6e5683}

   Error
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]

6. **Assert that `G2S_accessibleGame` has the correct parameter configuration.**
{0617f3b9-c806-43e3-8d38-102d6dea5d80}

   Error
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]

7. **Assert that `G2S_denomList` has the correct parameter configuration.**
{07f57267-bd95-42c3-9583-8a58d1e1d1a5}

   Errors
   - `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]
   - `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]

8. **Assert that the `gamePlay` profile values match the `optionConfig` values.**
{088d415b-8acf-4db6-ba72-ced3225193d8}

   Error
   - `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: GP-COR-00001

This case verifies that the EGM does not accept commands from non-guest hosts.

Objective

Verify that the EGM does not accept commands from non-guest hosts.

Requirements Under Test

- 1.8.5
- 6.5.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	non-guest

Test Procedure

1. **Send a `setGamePlayState` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
   {01042dd2-49de-4338-aad4-4a81485231fc}

   Command
   - `gamePlay.setGamePlayState`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

2. **Send a `setActiveDenoms` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
   {0284eae2-abdd-4756-8010-399e88a43ff4}

   Command
   - `gamePlay.setActiveDenoms`

Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

3. **Send a `getGamePlayStatus` command, and verify that the EGM responds with a `G2S_APX012` error.**
{0353efe8-3dc7-4741-0072-98b0de1f691a}

Command

- gamePlay.getGamePlayStatus

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

4. **Send a `getGamePlayProfile` command, and verify that the EGM responds with a `G2S_APX012` error.**
{041cda1e-d42e-4f0d-00bf-a8791e9262a4}

Command

- gamePlay.getGamePlayProfile

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

5. **Send a `getGameDenoms` command, and verify that the EGM responds with a `G2S_APX012` error.**
{0586b823-9aab-43f8-00fb-dba1abbc4849}

Command

- gamePlay.getGameDenoms

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

6. **Send a `getRecallLogStatus` command, and verify that the EGM responds with a `G2S_APX012` error.**
{0629aa30-e857-49e2-805e-b972ddbe2187}

Command

- `gamePlay.getRecallLogStatus`

  Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

7. **Send a `getRecallLog` command, and verify that the EGM responds with a `G2S_APX012` error.**
{07850082-d9df-48e8-8034-bf43184f3ab0}

Command

- `gamePlay.getRecallLog`

  Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

Test Case: GP-COR-00002

This case verifies that the EGM does not accept control commands from a guest host.

Objectives

- Verify that the EGM does not accept control commands from a guest host.
- Verify that the EGM accepts non-control commands from a guest host.

Requirements Under Test

- 1.8.14
- 6.5.1
- 6.5.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	guest

Test Procedure

1. **Send a `setGamePlayState` command, and verify that the EGM responds with a `G2S_APX010` error.**
   {01062026-221d-4f6b-9d9f-2a72a2afe68c}

   Command
   - `gamePlay.setGamePlayState`
     Expect **G2S_APX010**

   Errors
   - Send Request Errors
   - `E-CM-00037` [The EGM must not allow control commands from a guest host.]

2. **Send a `setActiveDenoms` command, and verify that the EGM responds with a `G2S_APX010`.**
   {029f4664-1d26-445f-ac92-d78068ba996b}

   Command
   - `gamePlay.setActiveDenoms`

Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00037` [The EGM must not allow control commands from a guest host.]

3. **Send a `getGamePlayStatus` command, and verify that the EGM responds properly.** {03a214b0-ca78-4f8b-8c69-67b2ef401f13}

   Command
   - `gamePlay.getGamePlayStatus`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

4. **Send a `getGamePlayProfile` command, and verify that the EGM responds properly.** {0481b5e7-7850-4d62-b54c-d25a0a82ce06}

   Command
   - `gamePlay.getGamePlayProfile`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

5. **Send a `getGameDenoms` command, and verify that the EGM responds properly.** {05a4e3bf-3c30-416a-a1eb-be4fc7cb9f88}

   Command
   - `gamePlay.getGameDenoms`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

6. **Send a `getRecallLogStatus` command, and verify that the EGM responds properly.** {06bacf1b-d4a4-48e9-82a7-db4830fa4b40}

   Command
   - `gamePlay.getRecallLogStatus`

   Errors
   - Send Request Errors
   - `E-CM-00038` [The EGM must generate a response to a request from the host.]

7. **Send a `getRecallLog` command, and verify that the EGM responds properly.**
{079e5c2f-3f60-4ea1-9dfc-f240ad0d8522}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

Test Case: GP-COR-00003

This case verifies that the EGM rejects unknown commands.

Objective

Verify that the EGM sends the proper error for unknown commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: gamePlay
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Send a cvtTesting command to the EGM.**
   {01940324-6ba1-4f7a-b787-1f856d241bbf}

   #### Command

   - `gamePlay.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: GP-COR-00004

This case verifies that standard configuration options are available in `optionConfig`, and that they are consistent with the `gamePlayProfile` command.

Objectives

- Verify that the `optionConfig` parameters for option ID `G2S_protocolOptions` are configured properly.
- Verify that the `optionConfig` parameters for option ID `G2S_gamePlayOptions` are configured properly.

Requirements Under Test

- 1.38.1
- 6.9.21

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_optionConfig	owner

Test Procedure

1. **Get the `gamePlay` profile values.**
   {01210f0c-d467-44cd-9891-192fc8f2a469}

   Command
   - `gamePlay.getGamePlayProfile`

   Error
   - Send Request Errors

2. **Get the option list for the group ID `G2S_gamePlayOptions`.**
   {02e84de3-ddf9-4b69-a356-090009f3605a}

   Command
   - `optionConfig.getOptionList`

Error

- Send Request Errors

3. **Assert that `G2S_protocolOptions` has the correct parameter configuration.**
{0342bc1d-a0de-4583-9b52-acf64536cad3}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]

4. **Assert that `G2S_gamePlayOptions` has the correct parameter configuration.**
{04391b42-aea3-455a-943a-ad5900051338}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]

5. **Assert that the `gamePlay` profile values match the `optionConfig` values.**
{05e5eb40-791c-4a22-8c87-0447d5f224ad}

Error

- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: GP-COR-00005

This case verifies that the EGM returns a `G2S_APX003 Invalid Device Identifier` error when device ID zero (0) is used.

Objective

Verify that the EGM returns a `G2S_APX003` error when device ID zero (0) is used as a class-level attribute.

Requirements Under Test

- 1.7.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Send a `getRecallLog` command to device ID zero (0), and verify that the EGM responds with a `G2S_APX003` error.**
   {014fa4e2-783c-4b4c-0074-2a7a5f762614}

   #### Command

   - `gamePlay.getRecallLog`

     Expect **G2S_APX003**

   #### Errors

   - Send Request Errors
   - `E-XX-00015` [EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).]

Test Case: GP-COR-00006

This case verifies that forced event subscriptions are treated as if they are set by the host.

Objective

Verify that the EGM sends `gamePlay` events when a forced event subscription exists.

Requirements Under Test

- 1.23.25
- 1.23.26
- 6.12.1
- 6.13.1
- 6.20.1
- 6.20.2
- 6.35.1
- 6.35.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_eventHandler	owner and configurator
G2S_optionConfig	owner

Test Procedure

1. **Determine if the `gamePlay` device support `G2S_GPE099 All Device Faults Cleared` and `G2S_GPE202 Device Fault` events.**
   {013d704b-c49a-4da5-80a8-ec45422b7f12}

   Command
   - `eventHandler.getSupportedEvents`

   Error
   - Send Request Errors

2. **If the `gamePlay` device supports `G2S_GPE099` and `G2S_GPE202`.**

   a. **Clear event subscriptions for the `gamePlay` device.**
   {02a1a29d-faa9-493b-8058-0106607617b0}

Commands

- `eventHandler.clearEventSub`
- `eventHandler.getEventSub`

Event

(Time out: ${event.timeOut})

- `G2S_EHE101` [Event Subscription Changed]

Errors

- Send Request Errors
- Event Checking Errors

b. **Set a forced event subscription for `gamePlay` events.**
{03b7c40c-e80d-4d9e-80d5-1f0c1539fd02}

Commands

- `optionConfig.setOptionChange`
- `eventHandler.getEventSub`

Event

(Time out: ${event.timeOut})

- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors

- Send Request Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

c. **Determine which active denominations are available for the game play device.**
{04cf382f-e8a6-4ebb-0033-57ae2c0124ac}

Command

- `gamePlay.getGameDenoms`

Error

- Send Request Errors

d. **Instruct user to cause a general tilt fault.**
{05c1b639-5c07-42a7-80b6-d8b3cdaac0d1}

Action

- Instruct the user to cause a device fault to generate 'G2S_GPE202'.

Events
(Time out: ${event.timeOut})

- G2S_GPE202 [Device Tilt]
- G2S_GPE001 [Device Disabled by EGM]

Errors

- Event Checking Errors
- DUT Error

e. **Instruct user to clear the device fault.**
{06a67e70-256f-4e90-002e-5c5296b57fec}

Action

- Instruct the user to clear a device fault to generate 'G2S_GPE099'.

Events
(Time out: ${event.timeOut})

- G2S_GPE099 [Device Tilts Cleared]
- G2S_GPE002 [Device Enabled by EGM]

Errors

- Event Checking Errors
- DUT Error

f. **Clear the forced event subscription.**
{0794bffc-3832-43e7-807a-e0b98443ce75}

Command

- optionConfig.setOptionChange

Event
(Time out: ${event.timeOut})

- G2S_EHE005 [Event Handler Configuration Changed by Host]

Errors

- Event Checking Errors
- E-XX-00002 [CVT timed out waiting for the response.]
- E-XX-00003 [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- E-XX-00005 [An unexpected error of ${errorCode} was received.]

Test Case: GP-COR-00007

This case verifies that the EGM does not process invalid XML.

Objectives

- Verify that the EGM validates the `getRecallLog` command.
- Verify that the EGM validates the `commandId` attribute in the `gamePlay` class element.

Requirements Under Test

- 1.41.2
- 1.41.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Send a `getRecallLog` command with an invalid `totalEntries` attribute.**
   {01526868-efbc-431a-80d8-438738aff036}

   Command
   - `gamePlay.getRecallLog`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

2. **Send a `getRecallLog` command with an invalid `commandId` attribute.**
   {026547fa-937f-4438-00f8-8fdc5ebdaf2f}

   Command
   - `gamePlay.getRecallLog`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors

- Send Request Errors
- E-CM-00039 [The EGM must validate the G2S request.]

Test Case: GP-COR-00008

This test verifies `gamePlay` log processing.

Objectives

- Verify that the `recallLog` sequence numbers are contiguous.
- Verify that the EGM returns an empty `recallLog` when the requested logSequence number is not in the log.
- Verify that the EGM returns the specified log entries when totalEntries attribute value is not zero (0).
- Verify that there is a `winLevelItem` when `initialWin` or `secondaryWin` is non-zero.
- Verify that the `winLevelItem` is present in the `gamePlayProfile`.

Requirements Under Test

- 1.7.4
- 1.19.2
- 1.19.13
- 1.21.2
- 1.21.3
- 1.21.4
- 1.21.5
- 1.21.6
- 1.21.7
- 1.21.8
- 6.4.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Determine the size of the `recallLog`.**
   {01f50a83-ab3b-4be5-00b7-045695760322}

Command

- `gamePlay.getGamePlayProfile`

Error

- Send Request Errors

2. **Determine the `lastSequence` and `totalEntries` values in the `recallLog`.**
{02be698d-03a2-434c-0064-af8ac8288689}

Command

- `gamePlay.getRecallLogStatus`

Error

- Send Request Errors

3. **Verify that the `getRecallLog` command, with `lastSequence` and `totalEntries` attribute values of zero (0), retrieves the entire log.**
{03443695-3df6-458d-804a-d1996c1e5b84}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00001` [Recall log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-GP-00002` [Recall log MUST start with logSequence number ${sequenceNumber}.]
- `E-GP-00003` [Recall log MUST have ${logEntries} entries.]

4. **Verify that requesting the `logSequence` less than the smallest `logSequence` number returns an empty log.**
{04e4494b-d426-4ac2-0092-5d692462a670}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00004` [Recall log MUST be empty when requesting a lastSequence number not in the log.]

5. **Verify that requesting the `logSequence` greater than the last `lastSequence` number returns an empty log.**
{05cf8afa-b27f-43eb-0091-80bfc9a35f08}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00004` [Recall log MUST be empty when requesting a lastSequence number not in the log.]

6. **Verify that the EGM returns an `recallLog` with the correct number of entries.**
{06b1ba0f-5d00-48fe-8058-09d6969f7fde}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00002` [Recall log MUST start with logSequence number ${sequenceNumber}.]
- `E-GP-00003` [Recall log MUST have ${logEntries} entries.]

7. **Verify that the EGM returns an `recallLog` with one entry if the `totalEntries` equals 1.**
{0705cf09-38ce-42de-00fc-5048a7724e31}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00003` [Recall log MUST have ${logEntries} entries.]

8. **Verify the `winLevelItem` entries are correct.**
{08112aa0-a165-467a-8026-308e68d16242}

Errors

- `E-GP-00012` [Recall Log MUST have a winLevelItem when initialWin or secondaryWin is non-zero.]
- `E-GP-00013` [Win level index of ${winLevelIndex} MUST exist in the gamePlayProfile.]

Test Case: GP-COR-00009

This case verifies that the EGM supports protocol specified error codes.

Objectives

- Verify that the EGM returns a `G2S_GPX001 Invalid Denomination Specified` error when an invalid denomination is specified.
- Verify that the EGM either returns a `G2S_GPX002 EGM Does Not Permit Operations That Would Cause A gamePlay Device Collision` error or allows activating a denomination that would cause the same theme and denomination or game combo to be active on two `gamePlay` devices.

Requirements Under Test

- 1.44.2
- 6.2.9
- 6.9.8
- 6.10.2
- 6.34.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
gamePlayProfile.setAccessViaConfig	false

Test Procedure

1. **Determine the `idleTimePeriod` for the `cabinet` device.**
   {0195cc62-f0a3-4b95-0028-3e2b7ef50531}

   Command
   - `cabinet.getCabinetProfile`

Error

- Send Request Errors

2. **Determine the active game play devices.**
{02247f1c-e7c9-4392-80a9-1b28553cbd2b}

Command

- `communications.getDescriptor`

Error

- Send Request Errors

3. **For each active `gamePlay` device.**

    1. **Determine the active game combos for `gamePlay` device.**
    {036fa95f-4c28-4fa4-8049-d8b12bf1c6de}

    Commands

    - `gamePlay.getGamePlayProfile`
    - `gamePlay.getGameDenoms`

    Error

    - Send Request Errors

4. **Determine if there is a potential game combo conflict.**
{044a55de-9f1c-489a-807f-3fc5942d364e}

5. **If potential game combo confict exists.**

    a. **Verify that the EGM either enables the game denom that could cause a game conflict or returns a `G2S_GPX002` error.**
    {0513d32f-c880-462c-006c-ef6bdf595979}

    Command

    - `gamePlay.setActiveDenoms`

      Expect **G2S_GPX002** or **Normal Response**

    Event

    (Time out: ${event.timeOut})

    - `G2S_GPE201` [Active Denomination Changed]

    Errors

    - Send Request Errors
    - Event Checking Errors
    - `E-GP-00005` [Game denom ${denomId} must be ${activeState}.]

6. **Verify that the EGM returns `G2S_GPX001` error for an invalid denomination.**
{06b67982-b477-4488-80b2-a16859422714}

   Command
   - `gamePlay.setActiveDenoms`

     Expect **G2S_GPX001**

   Error
   - Send Request Errors

7. **Verify that the EGM returns `G2S_GPX001` error for an invalid denomination range.**
{07635e0e-4212-4f3c-00f3-6225002ab3bb}

   Command
   - `gamePlay.setActiveDenoms`

     Expect **G2S_GPX001**

   Error
   - Send Request Errors

Test Case: GP-COR-00010

This case verifies `gamePlay` device configuration.

Objectives

- Verify that the `gamePlay` device has one theme.
- Verify that the `gamePlay` device has one paytable.
- Verify that the `gamePlay` device has at least one wager category.
- Verify that the `gamePlay` device has at least one denomination.
- Verify that if the `gamePlay` device is enabled that there is at least one active denomination.
- Verify that game combinations with a disabled denomination are not accessible.

Requirements Under Test

- 6.1.2
- 6.1.3
- 6.1.4
- 6.1.5
- 6.1.6
- 6.1.7
- 6.1.8
- 6.1.9
- 6.1.10
- 6.9.1
- 6.9.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner

Required Configurations

Option	Value
gamePlayProfile.setAccessViaConfig	false

Test Procedure

1. **Verify the `gamePlay` profile is configured properly.**
   {0146d67e-e846-422d-0088-3e643c1cf7e0}

   Command
   - `gamePlay.getGamePlayProfile`

   Error
   - Send Request Errors

2. **Verify that the active `gameDenomList` is properly configured.**
   {02138346-dff4-402c-0080-12cce0512436}

   Command
   - `gamePlay.getGameDenoms`

   Errors
   - Send Request Errors
   - `E-GP-00007` [Enabled game play devices MUST have at least one active game denom.]

3. **For each inactive denominations.**

   1. **Attempt to play inactive game combo.**
      {03762416-92d9-440f-805d-1da88b999ca5}

      Action
      - Play Game #${deviceUnderTest.deviceId}

      Errors
      - DUT Error
      - `E-GP-00006` [Inactive game denom ${denomId} MUST NOT be playable for game play device ${deviceId}.]

Test Case: GP-COR-00011

This case verifies the events and meters associated with a game play cycle.

Objectives

- Verify that the `gamePlay` events are sent in proper order.
- Verify that the `recallLog` record matches the `gamePlay` events.
- Verify that the last played game play attributes in the `cabinetStatus` are updated properly.

Requirements Under Test

- 3.62.1
- 3.71.1
- 6.23.1
- 6.23.2
- 6.23.3
- 6.24.1
- 6.24.2
- 6.24.3
- 6.25.1
- 6.25.2
- 6.26.1
- 6.29.1
- 6.29.2
- 6.29.3
- 6.30.1
- 6.31.1
- 6.31.2
- 6.32.1
- 6.33.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_noteAcceptor	guest or owner

Device	Permissions
G2S_meters	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
gamePlayProfile.centralAllowed	false

Test Procedure

1. **Determine the active game denoms.**
{01114cb6-7859-4041-0030-6dec3f99e5de}

   Command
   - `gamePlay.getGameDenoms`

   Errors
   - Send Request Errors
   - `E-GP-00007` [Enabled game play devices MUST have at least one active game denom.]

2. **Get the note acceptor profile.**
{02fede18-04f6-4b54-0088-c446b8c99d07}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

3. **Get device and currency meters.**
{036d4256-1426-438e-00c8-92764318af19}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

4. **Instruct the user to insert a note.**
{047ee1f4-116e-40de-00a4-695bd39d3608}

   Action
   - Insert a ${noteDenom} ${currency} note.

   Events
   (Time out: ${event.timeOut})

- G2S_NAE116 [Note In Escrow]
- G2S_NAE114 [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

5. **Get the cabinet and game play meters.**
{05a26106-ac87-4d66-8079-1ef4d25ad989}

Command

- meters.getMeterInfo

Error

- Send Request Errors

6. **Get the sequence number of the last game play recall log entry.**
{06aa08eb-899a-4125-8086-e204d9c47997}

Command

- gamePlay.getRecallLogStatus

Error

- Send Request Errors

7. **Instruct user to play a game.**
{073c8ada-a662-4d08-8099-f5c8713af6e7}

Action

- Play Game #${deviceUnderTest.deviceId}

Events

(Time out: ${event.timeOut})

- G2S_GPE103 [Primary Game Started] or G2S_GPE104 [Wager Changed]
- G2S_CBE314 [Game Combo Activated] or G2S_GPE105 [Primary Game Ended]
- G2S_GPE106 [Secondary Game Choice] or G2S_GPE109 [Secondary Game Started] or G2S_GPE110 [Secondary Game Ended] or G2S_GPE111 [Game Result]
- G2S_GPE112 [Game Ended]
- G2S_GPE113 [Game Idle]

Errors

- Event Checking Errors
- DUT Error
- E-GP-00011 [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]

8. **Verify game outcome with user.**
{083f4c10-1768-4827-8002-d9840b45e363}

Actions

- Instruct the user to 'Did you wager ${stateMachine.wager}?'.
- Instruct the user to 'Did you win ${stateMachine.win}?'.
- Instruct the user to 'Is the player credit meter ${stateMachine.creditMeter}?'.

Errors

- DUT Error
- `E-GP-00014` [User observed game play results do not match game play recall log.]

9. **Get the latest game recall log entry and verify content.**
{092797c5-42c9-41f8-00dd-e8b856eb0601}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- `E-GP-00003` [Recall log MUST have ${logEntries} entries.]
- `E-GP-00009` [Game recall log entry does not match game play event value. Attribute ${attribute} MUST be ${expected}, but it was ${actual}.]

10. **Assert the cabinet status last game played attributes are correct.**
{10230d15-d26c-4693-800d-e47639ca1492}

Command

- `cabinet.getCabinetStatus`

Errors

- Send Request Errors
- `E-CB-00020` [Cabinet status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

11. **If the EGM still has player credits**

    a. **Instruct user to cash out EGM.**
    {111fab32-e201-45f5-0061-9a28a0b9f8bd}

    Action

    - Cash Out.

    Event
    (Time out: ${event.timeOut})

- `G2S_CBE316` [Cash-Out Button Pressed]

Errors

- Event Checking Errors
- DUT Error

Test Case: GP-COR-00012

This case verifies that the EGM supports game play being disabled.

Objectives

- Verify that the EGM handles the `setGamePlayState` command.
- Verify that the EGM displays the disable text associated with the `setGamePlayState` command.

Requirements Under Test

- 3.5.12
- 6.9.8
- 6.14.1
- 6.15.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
gamePlayProfile.setAccessViaConfig	false
gamePlayProfile.requiredForPlay	true

Test Procedure

1. **Determine the `idleTimePeriod` for the `cabinet` device.**
   {017062d2-9999-4347-009c-ce8cb8c7fba1}

   Command
   - `cabinet.getCabinetProfile`

   Error
   - Send Request Errors

2. **Disable the game play device.**
{025127e3-1198-4b7f-803e-46db7530b2cd}

Command
- `gamePlay.setGamePlayState`

Actions
- Verify that text 'GP-COR-00012: Step 2' appears onscreen.

Event
(Time out: ${event.timeOut})
- `G2S_GPE003` [Device Disabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00014` [Required text is not displayed.]
- `E-GP-00010` [Game play status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

3. **Enable the game play device.**
{03c437f0-ad08-46e6-80f6-d68972f94056}

Command
- `gamePlay.setGamePlayState`

Actions
- Verify that text 'GP-COR-00012: Step 3' does NOT appear onscreen.

Event
(Time out: ${event.timeOut})
- `G2S_GPE004` [Device Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]
- `E-GP-00010` [Game play status attribute ${attribute} is wrong. It MUST be ${expected}, but it was ${actual}.]

Test Case: GP-COR-00013

This case verifies that the EGM supports setting of game denoms.

Objectives

- Verify that the EGM handles the `setActiveDenom` command.
- Verify that the EGM does not send `G2S_GPE201 Active Denominations Changed` event if the active denoms are not modified.

Requirements Under Test

- 6.2.4
- 6.9.8
- 6.10.1
- 6.10.4
- 6.12.1
- 6.13.1
- 6.16.1
- 6.34.1
- 6.34.2
- 6.34.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
gamePlayProfile.setAccessViaConfig	false

Test Procedure

1. **Determine the `idleTimePeriod` for the `cabinet` device.**
   {01bdfc5f-fdaa-4289-0053-0de83436598e}

   Command

- `cabinet.getCabinetProfile`

Error

- Send Request Errors

2. **Determine the active game denoms for the `gamePlay` device.**
{02515267-7e2f-4d4a-00f9-7fe14a1ac693}

Command

- `gamePlay.getGameDenoms`

Errors

- Send Request Errors
- `E-GP-00008` [EGM MUST only report denom ${denomId} once in game denom list.]

3. **Deactivate all game denoms.**
{03c76583-43ec-4c89-0011-11c15665fd06}

Command

- `gamePlay.setActiveDenoms`

Events

(Time out: ${event.timeOut})

- `G2S_GPE201` [Active Denomination Changed]
- `G2S_GPE005` [Device Configuration Changed by Host]
- `G2S_GPE001` [Device Disabled by EGM]

Errors

- Send Request Errors
- Event Checking Errors
- `E-GP-00005` [Game denom ${denomId} must be ${activeState}.]

4. **Enable game denoms.**
{04825125-557b-46ff-8039-d5a01fed76fa}

Command

- `gamePlay.setActiveDenoms`

Events

(Time out: ${event.timeOut})

- `G2S_GPE201` [Active Denomination Changed]
- `G2S_GPE005` [Device Configuration Changed by Host]
- `G2S_GPE002` [Device Enabled by EGM]

Errors

- Send Request Errors
- Event Checking Errors
- `E-GP-00005` [Game denom ${denomId} must be ${activeState}.]

5. **Verify that events are not generated when the game denom is not changed.**
{055a6a2b-e851-44d5-80d5-66f36cac3846}

Command

- `gamePlay.setActiveDenoms`

Errors

- Send Request Errors
- Event Not Expected Error
- `E-GP-00005` [Game denom ${denomId} must be ${activeState}.]

Test Case: GP-COR-00014

This case verifies that the EGM supports setting of game denoms from the operator menu.

Objectives

- Verify that the EGM handles disabling denoms from the operator menu.
- Verify that the EGM generates the appropriate events for denom state changes.

Requirements Under Test

- 6.9.8
- 6.10.1
- 6.10.4
- 6.12.1
- 6.13.1
- 6.16.1
- 6.17.1
- 6.34.1
- 6.34.2
- 6.34.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
gamePlayProfile.setAccessViaConfig	false

Test Procedure

1. **Determine the active game denoms for the `gamePlay` device.**
   {01388332-82a7-4922-0032-4dd548f76e15}

   Command
   - `gamePlay.getGameDenoms`

Errors

- Send Request Errors
- `E-GP-00008` [EGM MUST only report denom ${denomId} once in game denom list.]

2. **Instruct user to disable all denoms.**
{026370fa-cbc2-4c7b-00a5-8ab69fa78ccf}

Action

- Instruct the user to 'Disable all game denoms for device ID ${deviceUnderTest.deviceId}.'.

Events

(Time out: ${event.timeOut})

- `G2S_GPE201` [Active Denomination Changed]
- `G2S_GPE006` [Device Configuration Changed at EGM]
- `G2S_GPE001` [Device Disabled by EGM]

Errors

- Event Checking Errors
- DUT Error

3. **The EGM is in the operator or audit menu.**

    1. **Determine if the operator or audit menu is enabled.**
    {0306086d-7190-4afc-00cb-070cc3bc7ff0}

    Command

    - `cabinet.getCabinetStatus`

    Error

    - Send Request Errors

4. **Verify all game denoms are deactivated.**
{040bbd79-e0f5-46e4-80cf-d8d70d66b127}

Command

- `gamePlay.getGameDenoms`

Errors

- Send Request Errors
- `E-GP-00005` [Game denom ${denomId} must be ${activeState}.]

5. **Enable game denoms.**
{058c24f9-3218-46f9-8082-d835c9b4af2f}

Command

- `gamePlay.setActiveDenoms`

Events

(Time out: ${event.timeOut})

- G2S_GPE201 [Active Denomination Changed]
- G2S_GPE005 [Device Configuration Changed by Host]
- G2S_GPE002 [Device Enabled by EGM]

Errors

- Send Request Errors
- Event Checking Errors
- E-GP-00005 [Game denom ${denomId} must be ${activeState}.]

Test Case: GP-COR-00015

This case verifies that the host sends a valid setGamePlayState command.

Objective

Verify the setGamePlayState command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Instruct user to send setGamePlayState command from the host.**
   {01ae3007-f0d7-4b3f-aadc-55ebe4e0e7f7}

   #### Command

   - `gamePlay.setGamePlayState`

   #### Actions

   - Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00016

This case verifies that the host sends a valid getGamePlayStatus command.

Objective

Verify the getGamePlayStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getGamePlayStatus command from the host.**
   {01c0f633-19ee-4656-ab88-c69c66b3ec41}

   #### Command

   - `gamePlay.getGamePlayStatus`

   #### Actions

   - Instruct the user to 'Send getGamePlayStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00017

This case verifies that the host sends a valid getGamePlayProfile command.

Objective

Verify the getGamePlayProfile command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getGamePlayProfile command from the host.**
   {011044b0-b342-44e5-871d-267c5f2a392e}

   #### Command

   - `gamePlay.getGamePlayProfile`

   #### Actions

   - Instruct the user to 'Send getGamePlayProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00018

This case verifies that the host sends a valid setActiveDenoms command.

Objective

Verify the setActiveDenoms command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Instruct user to send setActiveDenoms command from the host.**
   {01f71b5f-4024-45fd-8be5-a51a9997efa3}

   #### Command

   - `gamePlay.setActiveDenoms`

   #### Actions

   - Instruct the user to 'Send setActiveDenoms to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00019

This case verifies that the host sends a valid getGameDenoms command.

Objective

Verify the getGameDenoms command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getGameDenoms command from the host.**
   {01393084-936f-495f-a669-3d3f026a5261}

   #### Command

   - `gamePlay.getGameDenoms`

   #### Actions

   - Instruct the user to 'Send getGameDenoms to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00020

This case verifies that the host sends a valid getRecallLogStatus command.

Objective

Verify the getRecallLogStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getRecallLogStatus command from the host.**
   {0195bf6e-d076-4167-8309-8d23ef09ec0c}

   #### Command

   - `gamePlay.getRecallLogStatus`

   #### Actions

   - Instruct the user to 'Send getRecallLogStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00021

This case verifies that the host sends a valid getRecallLog command.

Objective

Verify the getRecallLog command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getRecallLog command from the host.**
   {01260f61-6e30-405f-8daa-7e3c1dbb4850}

   #### Command

   - `gamePlay.getRecallLog`

   #### Actions

   - Instruct the user to 'Send getRecallLog to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-COR-00022

This case verifies that the host does not process gamePlay requests from the EGM.

Objective

Verify that the host does not process a gamePlayStatus sent as a request.

Requirements Under Test

- 1.4.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner

Test Procedure

1. **Send a gamePlayStatus as a request and verify G2S_APX008 Command Not Supported error code.**
   {012c06d5-36bc-4f88-8246-e73f1d431d7a}

   Command
   - `gamePlay.gamePlayStatus`
     Expect **G2S_APX008**

   Error
   - Send Request Errors

Test Case: GP-COR-00023

This case verifies that the host handles errors responses to gamePlay requests from the host.

Objectives

- Verify that the host is still working after responding with a gamePlayProfile to a setGamePlayState command.
- Verify that the host is still working after responding with a setGamePlayState response to a setGamePlayState command.
- Verify that the host is still working after responding with a gamePlayStatus notification to a setGamePlayState command.
- Verify that the host is still working after responding with a unknown error code to a setGamePlayState command.

Requirements Under Test

- 1.4.5
- 1.4.6
- 1.42.3

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to respond with a gamePlayProfile response to a setGamePlayState request.**
   {01bc65d2-216e-455f-b9e5-2fedaeb47326}

2. **Instruct user to send setGamePlayState command from the host.**
   {0202f510-e1fb-41cb-8697-96fac403b378}

   Command
   - `gamePlay.setGamePlayState`

Actions

- Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId} (1 of 4).'.

Errors

- Expected Request Errors
- DUT Error

3. **Set the response manager to respond with a setGamePlayState response to a setGamePlayState request.**
{0366561d-432f-4e6f-88b0-098a9f8c8ec5}

4. **Instruct user to send setGamePlayState command from the host.**
{04b186bc-d25e-439a-a164-e2fc426e413a}

Command

- `gamePlay.setGamePlayState`

Actions

- Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId} (2 of 4).'.

Errors

- Expected Request Errors
- DUT Error

5. **Set the response manager to respond with a gamePlayStatus notification to a setGamePlayState request.**
{05232e2b-7598-468a-8a01-e95ab8fa8727}

6. **Instruct user to send setGamePlayState command from the host.**
{06f41a05-57f5-413f-ae58-728b0497b0bf}

Command

- `gamePlay.setGamePlayState`

Actions

- Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId} (3 of 4).'.

Errors

- Expected Request Errors
- DUT Error

7. **Set the response manager to respond with 'CVT_ error' error code to a setGamePlayState request.**
{0767a565-086e-49be-90f1-03fd0847e4de}

8. **Instruct user to send setGamePlayState command from the host.**
{08bda38c-f70c-4f94-9d8e-bcaa2e7842bf}

   Command
   - `gamePlay.setGamePlayState`

   Actions
   - Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId} (4 of 4).'.

   Errors
   - Expected Request Errors
   - DUT Error

9. **Reset the response manager.**
{09cd7717-cda1-4f75-aadc-77b46195ecdf}

10. **Send a keepAlive to the host to verify that it is still working.**
{109180f5-694a-45f7-8246-4d34ec33a5af}

    Command
    - `communications.keepAlive`

    Error
    - Send Request Errors

Test Case: GP-COR-00024

This case verifies that the host still works after receiving a schema invalid gamePlayStatus response.

Objective

Verify that the host is still working after responding with a schema invalid gamePlayStatus to a setGamePlayState command.

Requirements Under Test

- 1.27.33

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to respond with a schema invalid gamePlayStatus to a setGamePlayState request.**
   {01b7e039-da2a-41ba-bcf3-6bf3693f679f}

2. **Instruct user to send setGamePlayState command from the host.**
   {02253362-5390-46cc-abc2-b79f5ed89e6c}

   #### Command
   - `gamePlay.setGamePlayState`

   #### Actions
   - Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors
   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03db59b6-6f4c-44eb-914b-c3e5ffa39e05}

4. **Send a keepAlive to the host to verify that it is still working.**
{042e7391-53bd-4084-a86c-e1990052bee5}

Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: GP-COR-00025

This case tests that the host processes delayed g2sAcks in gamePlay commands.

Objective

Verify that the host is still working after delaying the g2sAck in the setGamePlay request.

Requirements Under Test

- 1.30.7

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to delay the g2sAck for setGamePlayState command.**
   {016d2a03-0836-4913-8723-324aca4cafd6}

2. **Instruct user to send setGamePlayState command from the host.**
   {0222c27e-a20d-4673-8e53-7df52e185d77}

   #### Command

   - `gamePlay.setGamePlayState`

   #### Actions

   - Instruct the user to 'Send setGamePlayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03ffe9f4-5457-4f8d-b929-c8de47d9a95f}

4. **Send a keepAlive to the host to verify that it is still working.**
   {04208caa-74e3-41c1-8615-f00701fdcf21}

   #### Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: GP-COR-00026

This case tests that the host sends the appropriate error for an unknown gamePlay command.

Objectives

- Verify that the host sends G2S_APX008 Command Not Supported or G2S_AXP015 Unknown Command Encountered error for an unknown gamePlay command.
- Verify that if the host sends G2S_AXP015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Send an unknown gamePlay.cvtTesting command and verify G2S_APX008 or G2S_APX015 error.**
   {014d5e1d-3e98-4d02-8128-ef53fc31bb0a}

   #### Command

   - `gamePlay.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: GP-COR-00027

This case verifies that the game play device supports basic events and meters requirements.

Objectives

- Verify that the game play log sequence number increments by one.
- Verify that the game play log sequence number is persisted.
- Verify that the game play events are sent in the correct order.
- Verify that the game play events associated data is gather when the event is sent and retried.
- Verify that the game play meters are reported in balanced sets.
- Verify that pay out events are reported in atomic units.
- Verify that meter updates are done before the event is sent.

Requirements Under Test

- 1.19.9
- 1.19.10
- 1.22.2
- 1.23.1
- 1.23.3
- 5.9.1
- 5.9.2
- 5.9.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Test Procedure

1. **Determine the active game denoms.**
{012a577c-a341-488a-a627-9744d2ab54ca}

Command
- `gamePlay.getGameDenoms`

Errors
- Send Request Errors
- `E-GP-00007` [Enabled game play devices MUST have at least one active game denom.]

2. **Get the game play recall log to determine the next log sequence number.**
{0294a190-a855-4c40-bbef-5f00d54e3709}

Command
- `gamePlay.getRecallLog`

Error
- Send Request Errors

3. **Get the note acceptor profile.**
{03ee16d8-fc94-4519-b3e5-0706c699311d}

Command
- `noteAcceptor.getNoteAcceptorProfile`

Error
- Send Request Errors

4. **Get the note acceptor and cabinet meters.**
{048cbdb4-2f2a-4b89-a2e6-738314466e75}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

5. **Instruct the user to insert a note.**
{054d085e-4c8c-4880-b9f4-9d892d723d3d}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

6. **Get the EGM meters.**
{06690590-e4ec-4de2-aae8-699da9dfe5e3}

Command

- `meters.getMeterInfo`

Error

- Send Request Errors

7. **Stop responding to eventReports.**
{07748d7b-37f7-49bc-85e1-2bc3f1c7f65e}

8. **Instruct user to play a game and verify game play events are generated.**
{086c4dfa-fba0-495a-ae89-9259303da1f5}

Action

- Play Game #${deviceUnderTest.deviceId}

Events

(Time out: ${event.timeOut})

- `G2S_GPE103` [Primary Game Started] or `G2S_GPE104` [Wager Changed]
- `G2S_GPE105` [Primary Game Ended]
- `G2S_GPE106` [Secondary Game Choice] or `G2S_GPE109` [Secondary Game Started] or `G2S_GPE110` [Secondary Game Ended] or `G2S_GPE111` [Game Result]
- `G2S_GPE112` [Game Ended]
- `G2S_GPE113` [Game Idle]

Errors

- Event Checking Errors
- DUT Error
- `E-GP-00011` [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]

9. **Get the recall log and verify that the log sequence number incremented by one.**
{09b035ed-5015-41f6-a450-d891d9dbafe8}

Command

- `gamePlay.getRecallLog`

Errors

- Send Request Errors
- E-XX-00021 [The ${deviceClass} log sequence number MUST increment by one for a new log entry.]

10. **Start responding to eventReports and verify events are resent with the current log and meter values.**
{104e4a8f-63e0-48e2-89ac-4bad8537ffaf}

Events
(Time out: ${event.timeOut})
- G2S_GPE103 [Primary Game Started] or G2S_GPE104 [Wager Changed]
- G2S_GPE105 [Primary Game Ended]
- G2S_GPE106 [Secondary Game Choice] or G2S_GPE109 [Secondary Game Started] or G2S_GPE110 [Secondary Game Ended] or G2S_GPE111 [Game Result]
- G2S_GPE112 [Game Ended]
- G2S_GPE113 [Game Idle]

Errors
- Event Checking Errors
- E-EH-00062 [Event ${eventCode} MUST re-gather affected ${type} ${name} when it is resent.]

11. **Get the EGM meters to verify game play meters were incremented in balanced sets.**
{11a27616-a027-46fc-aca2-1833d95fd6ed}

Command
- meters.getMeterInfo

Errors
- Send Request Errors
- E-MT-00002 [Meter values are not in balance.]

12. **Instruct the user to power cycle the EGM.**
{12a680c1-9502-4879-9a9a-f80dada75daf}

Action
- Instruct the user to 'Please power cycle the EGM'.

Errors
- DUT Error
- E-CM-00057 [The user failed to power cycle the EGM.]

13. **Wait for EGM to reconnect.**
{1319e0d5-5c79-4531-905e-941bd27b0a27}

Commands

- communications.commsOnLine
- communications.commsDisabled
- communications.setCommsState

Errors

- Expected Request Errors
- Send Request Errors

14. **Instruct the user to insert a note.**
{1463b27b-dab3-402e-a47f-d3cca9381131}

Action

- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- G2S_NAE116 [Note In Escrow]
- G2S_NAE114 [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

15. **Get the cabinet and game play meters.**
{15d60105-ecd0-40b8-8763-d5b64e0260d5}

Command

- meters.getMeterInfo

Error

- Send Request Errors

16. **Instruct user to play a game and verify game play events.**
{16033ba4-b96e-40ed-a3a9-2ecf26a1c23a}

Action

- Play Game #${deviceUnderTest.deviceId}

Events
(Time out: ${event.timeOut})

- G2S_GPE103 [Primary Game Started] or G2S_GPE104 [Wager Changed]
- G2S_GPE105 [Primary Game Ended]
- G2S_GPE106 [Secondary Game Choice] or G2S_GPE109 [Secondary Game Started] or G2S_GPE110 [Secondary Game Ended] or G2S_GPE111 [Game Result]

- G2S_GPE112 [Game Ended]
- G2S_GPE113 [Game Idle]

Errors

- Event Checking Errors
- DUT Error
- E-GP-00011 [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]

17. **Get the recall log and verify that the log sequence number incremented by one.**
{17d54dc5-44f0-4265-86f1-a7391dd79e14}

Command

- gamePlay.getRecallLog

Errors

- Send Request Errors
- E-XX-00021 [The ${deviceClass} log sequence number MUST increment by one for a new log entry.]

Test Case: GP-COR-00028

This case verifies that the game play updates wager category and game denomination meters properly.

Objectives

- Verify the sum of wager category wagered amounts equals the actual amount wagered for the game cycle.
- Verify for each wager category that was actually wagered, played count meter for that wager category is incremented by one.
- Verify that at least one set of game denomination meters is reported at device level and class level.
- Verify that the sum of game denomination meters equals the corresponding performance meters for the device.
- Verify that the class-level game denomination meters are provided for the actual wager amount.
- Verify that the sum of game denomination meters equals the corresponding performance meters at the class level.
- Verify that the EGM reports total amount wagered, number of games played, and theoretical payback amount for each distinct game denomination.
- Verify that if the EGM reports only a single denomination that it responds to game denomination meter queries.
- Verify that if the EGM reports only a single denomination, that game denominations are integer multiples of the reported denomination.

Requirements Under Test

- 5.20.2
- 5.20.3
- 5.20.4
- 5.20.5
- 5.20.6
- 5.20.7
- 5.20.8
- 6.1.1
- 6.2.1
- 6.2.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner
G2S_cabinet	guest or owner

Test Procedure

1. **Get the note acceptor profile.**
   {01800b34-b983-4622-8ad9-b64ada19240b}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error
   - Send Request Errors

2. **Determine the active game denoms.**
   {02dbb572-a28b-43fb-be53-da17abfeff38}

   Command
   - `gamePlay.getGameDenoms`

   Errors
   - Send Request Errors
   - `E-GP-00007` [Enabled game play devices MUST have at least one active game denom.]

3. **Get the note acceptor and cabinet meters.**
   {0374012e-4b7d-401c-8349-389ebbce8d87}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

4. **Instruct the user to insert a note.**
   {04b6e215-f388-40f0-8ded-d042cbcde82b}

   Action
   - Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- G2S_NAE116 [Note In Escrow]
- G2S_NAE114 [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

5. **Get the EGM meters.**
{051f1f23-60f2-4a21-9adb-e879fc927b3e}

Command

- meters.getMeterInfo

Error

- Send Request Errors

6. **Instruct user to play a game and verify game play events are generated.**
{06c66dc5-4c26-4edb-ab77-36c6a17a429a}

Action

- Play Game #${deviceUnderTest.deviceId}

Events
(Time out: ${event.timeOut})

- G2S_GPE103 [Primary Game Started] or G2S_GPE104 [Wager Changed]
- G2S_GPE105 [Primary Game Ended]
- G2S_GPE106 [Secondary Game Choice] or G2S_GPE109 [Secondary Game Started] or G2S_GPE110 [Secondary Game Ended] or G2S_GPE111 [Game Result]
- G2S_GPE112 [Game Ended]
- G2S_GPE113 [Game Idle]

Errors

- Event Checking Errors
- DUT Error
- E-GP-00011 [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]

7. **Verify that sum of wager category's wagered amounts equals the actual amount wagered.**
{07a4f16a-6bb4-4757-af87-fe9149d6bcc8}

Command

- meters.getMeterInfo

Errors

- Send Request Errors
- `E-MT-00023` [The sum of wager category wagered amounts of MUST equal the actual amount wagered.]

8. **Verify that the played count meter incremented by one for each wager category that was played.**
{08d5d5a8-0c07-4d21-96ae-417b33368e3e}

Error

- `E-MT-00024` [Wager category ${wagerCategory} played count meter must be increment by one when a game is played for that wager category.]

9. **Verify that there is at least one game denomination meter at the device and class levels.**
{09cdd604-d6f0-402a-bd24-64329a1799d5}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-MT-00025` [The game play device MUST have at least one game denomination at the ${levelName} level.]

10. **Verify that the sum of game denomination meters equals the corresponding performance meters for the device.**
{10e32349-e35d-4021-b1c4-2d94e00afe06}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-MT-00026` [The sum of game denomination meter deltas of ${actual} for ${meterName} does not match device performance meter delta ${expected}.]

11. **Verify that the class-level game denomination meters are provided for the actual wager amount.**
{11f556b1-d2e7-44ee-9510-6e5c023b2490}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- E-MT-00008 [Meter ${meterName} value of ${actual} is wrong it should be ${expected}.]

12. **Verify that the sum of game denomination meters equals the corresponding performance meters at the class level.**
{12f4dfe2-5580-4688-9601-a9162c40da8a}

Command
- meters.getMeterInfo

Errors
- Send Request Errors
- E-MT-00026 [The sum of game denomination meter deltas of ${actual} for ${meterName} does not match device performance meter delta ${expected}.]

13. **Verify that the EGM reports total amount wagered, number of games played, and theoretical payback amount for each distinct game denomination.**
{13eee0e3-1304-4898-bd29-a26535d7281d}

Command
- meters.getMeterInfo

Errors
- Send Request Errors
- E-MT-00027 [EGM must report ${field} for each distinct reported game denomination.]

14. **If the EGM reports a single denomination**

    a. **Verify that if the EGM reports only a single denomination that it responds to game denomination meter queries.**
    {14a9f468-cf5c-4e2b-943b-fb0b854ddec1}

    Command
    - meters.getMeterInfo

    Errors
    - Send Request Errors
    - E-MT-00028 [EGM must respond to queries for game denom meters.]

    b. **Verify that if the EGM reports only a single denomination, that game denominations are integer multiples of the reported denomination.**
    {154a5461-7119-404d-867b-87cf274c2d85}

    Command
    - meters.getMeterInfo

Errors

- Send Request Errors
- `E-MT-00029` [Game denom wager amount of ${wageredAmt} is not a multiple of reported denomination ${denomValue}.]

Test Case: GP-COR-00029

This case verifies that the game play profile win levels are configured properly for progressives and secondary wagers.

Objectives

- Verify that if `gamePlayProfile.progAllowed` that there is at least `winLevelItem` with `progressiveAllowed` set to true.
- Verify that if `gamePlayProfile.progAllowed` is false that there is at least `winLevelItem` with `progressiveAllowed` set to false.
- Verify that if `gamePlayProfile.secondaryAllowed` that there is at least `winLevelItem` with `secondaryAllowed` set to true.

Requirements Under Test

- 6.9.3
- 6.9.4
- 6.9.5
- 6.9.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: gamePlay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Verify game play profile win levels are configured properly.**
   {0110c7b0-4be8-40f0-a229-559b64df5a07}

   #### Command
   - `gamePlay.getGamePlayProfile`

   #### Errors
   - Send Request Errors
   - `E-GP-00015` [Game play device that allows progressives MUST have a win-level that allows progressives.]

- E-GP-00016 [Game play device that allows secondary games MUST have a win-level that allows secondary games.]
- E-GP-00017 [Game play device that MUST have at least one win-level.]

2. **If game play device supports progressives**

    a. **Get the list of progressive devices.**
    {0231b8bf-ea2c-4d86-b275-8426bc580796}

    Command
    - communications.getDescriptor

    Error
    - Send Request Errors

3. **While there are progressive devices to check**

    1. **Verify that the game play win levels are linked at most to one progressive level.**
    {033c0882-6210-46ff-bc2b-bcd69c64924a}

    Command
    - progressive.getProgressiveProfile

    Errors
    - Send Request Errors
    - E-GP-00018 [A win level index, denomation and number of credits wagered (${winLevelIndex}/${denom}/${credits}) for a game play device MUST NOT be linked to more than one progressive level.]

Test Case: GP-COR-00030

This case tests that the host continues to work after sending a gameProfile as a notification.

Objective

Verify that the host continues to respond to keepAlive commands after sending a gameProfile as a notification.

Requirements Under Test

- 1.4.7

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Send a gamePlay.gamePlayProfile command as a notification.**
   {01ada057-c2fb-4a5d-8003-b0f11cbf5cb2}

   Command
   - `gamePlay.gamePlayProfile`

   Errors
   - Send Request Errors
   - `E-XX-00019` [An unexpected response of ${response} was received.]

2. **Send a keepAlive to verify that the host is still responding.**
   {0205a23c-0cf8-4804-b7e5-fbab817b9a7b}

   Command
   - `communications.keepAlive`

   Error
   - Send Request Errors

Test Case: GP-COR-00031

This case verifies that the host handles invalid XML in the response to a getGamePlayProfile.

Objective

Verify that the host is still working after receiving invalid XML in the response to a getGamePlayProfile command.

Requirements Under Test

- 1.41.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	owner
G2S_communications	owner

Test Procedure

1. **Set the response manager to respond with invalid XML in the gamePlayProfile response to a getGamePlayProfile request.**
   {010ff657-2c0b-4b29-9920-99edcc5ecc08}

2. **Instruct user to send getGamePlayProfile command from the host.**
   {02aa199a-4559-4100-97eb-673e95204a5b}

   #### Command

   - `gamePlay.getGamePlayProfile`

   #### Actions

   - Instruct the user to 'Send getGamePlayProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03de0919-f0a7-4623-b078-6a255e22b000}

4. **Send a keepAlive to the host to verify that it is still working.**
{0437211e-483d-4270-a5c5-1f1486424afb}

Command

- `communications.keepAlive`

Error

- Send Request Errors

Test Case: GP-GOS-00002

This case verifies that the host sends a valid getOutcomeLogStatus command.

Objective

Verify the getOutcomeLogStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getOutcomeLogStatus command from the host.**
   {016c33ce-8ed9-45c0-85f6-42c60d3f5c70}

   #### Command

   - `gamePlay.getOutcomeLogStatus`

   #### Actions

   - Instruct the user to 'Send getOutcomeLogStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: GP-GOS-00003

This case verifies that the host sends a valid getOutcomeLog command.

Objective

Verify the getOutcomeLog command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: gamePlay

Required Devices

Device	Permissions
G2S_gamePlay	guest or owner

Test Procedure

1. **Instruct user to send getOutcomeLog command from the host.**
   {0160ada5-6898-44d6-8dae-e93e253e6f52}

   #### Command

   - `gamePlay.getOutcomeLog`

   #### Actions

   - Instruct the user to 'Send getOutcomeLog to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

handpay

Handpay
Functional Groups

Test Case: JP-COR-00001

This case verifies that the EGM does not accept commands from non-guest hosts.

Objective

Verify that the EGM does not accept commands from non-guest hosts.

Requirements Under Test

- 1.8.5
- 1.8.14

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	non-guest

Test Procedure

1. **Send a `setHandpayState` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
   {0180fbcb-e23a-4011-803b-22b1fb79398f}

   Command
   - `handpay.setHandpayState`

     Expect **G2S_APX010** or **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

2. **Send a `setRemoteKeyOff` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
   {020146c1-00c2-4fff-80b0-d2328b646a5c}

   Command
   - `handpay.setRemoteKeyOff`

Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

3. **Send a `getHandpayProfile` command, and verify that the EGM responds with a `G2S_APX012` error.**
{03dec69f-cf43-417f-004c-c9937987629c}

Command

- handpay.getHandpayProfile

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

4. **Send a `getHandpayStatus` command, and verify that the EGM responds with a `G2S_APX012` error.**
{04b650c5-bff6-492f-0069-7cc3bf708989}

Command

- handpay.getHandpayStatus

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

5. **Send a `getHandpayLogStatus` command, and verify that the EGM responds with a `G2S_APX012` error.**
{05bbfbb5-3fc0-4c39-00fd-aae25a64db74}

Command

- handpay.getHandpayLogStatus

  Expect **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

6. **Send a `getHandpayLog` command, and verify that the EGM responds with a `G2S_APX012` error.**
{069853d5-b6e1-40bd-00fd-779a57944720}

Command

- `handpay.getHandpayLog`

  Expect **G2S_APX012**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

Test Case: JP-COR-00002

This case verifies that the EGM does not accept control commands from a guest host.

Objectives

- Verify that the EGM does not accept control commands from a guest host.
- Verify that the EGM accepts non-control commands from a guest host.

Requirements Under Test

- 1.8.14
- 11.3.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest

Test Procedure

1. **Send a `setHandpayState` command, and verify that the EGM responds with a `G2S_APX010` error.**
   {013bcbab-2c7b-4e44-80ba-5347a6ec3954}

   Command
   - `handpay.setHandpayState`

     Expect **G2S_APX010**

   Errors
   - Send Request Errors
   - `E-CM-00037` [The EGM must not allow control commands from a guest host.]

2. **Send a `setRemoteKeyOff` command, and verify that the EGM responds with a `G2S_APX010` error.**
   {0290e23a-983a-4c8e-8074-acc6d817d480}

   Command
   - `handpay.setRemoteKeyOff`

     Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00037` [The EGM must not allow control commands from a guest host.]

3. **Send a `getHandpayProfile` command, and verify that the EGM responds properly.**
{03b9094c-7b8a-428b-80b7-1170fbd8ee7d}

Command

- `handpay.getHandpayProfile`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

4. **Send a `getHandpayStatus` command, and verify that the EGM responds properly.**
{04db7828-ef50-47f5-005d-d1ce29b2601b}

Command

- `handpay.getHandpayStatus`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

5. **Send a `getHandpayLogStatus` command, and verify that the EGM responds with a `G2S_APX012` error.**
{05eb9899-bf93-43c5-0072-a0d98aa46567}

Command

- `handpay.getHandpayLogStatus`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

6. **Send a `getHandpayLog` command, and verify that the EGM responds with a `G2S_APX012` error.**
{067ff32b-259e-4ea7-000a-bf1fe1884df0}

Command

- `handpay.getHandpayLog`

Errors

- Send Request Errors
- `E-CM-00038` [The EGM must generate a response to a request from the host.]

Test Case: JP-COR-00003

This case verifies that the EGM rejects unknown commands.

Objective

Verify that the EGM sends the proper error for unknown commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Send a `cvtTesting` command, and verify that the EGM responds with a `G2S_APX008` or `G2S_APX015` error.**
   {01a1276a-afbb-4d95-8023-52f387eb5c76}

   #### Command

   - `handpay.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: JP-COR-00004

This case verifies that standard configuration options are available in `optionConfig`, and they are consistent with the `handpayProfile` command.

Objectives

- Verify that the optionConfig parameters for option ID `G2S_protocolOptions` are configured properly.
- Verify that the optionConfig parameters for option ID `G2S_handpayOptions` are configured properly.

Requirements Under Test

- 1.38.1
- 11.7.10
- 11.26.1
- 11.26.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest or owner
G2S_optionConfig	guest or owner
G2S_communications	owner

Test Procedure

1. **Determine if WAT and Voucher devices are available.**
   {013526f2-03d9-4b86-80d6-f3fe472cfc9a}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Get the `handpay` profile values.**
   {02b64d92-2b2e-4e1d-80cc-5a7f0e663351}

Command

- `handpay.getHandpayProfile`

Error

- Send Request Errors

3. **Get the option list for the group ID `G2S_handpayOptions`.**
{037d716b-538b-482e-8042-1d9b923dac70}

Command

- `optionConfig.getOptionList`

Error

- Send Request Errors

4. **Assert that `G2S_protocolOptions` has the correct parameter configuration.**
{04bd4707-0c82-4af4-00bb-613310636ff5}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]

5. **Assert that `G2S_handpayOptions` has the correct parameter configuration.**
{05166728-6816-4da6-0084-e59c9546db4e}

Errors

- `E-OC-00004` [Option config parameter ${paramId} is not configured properly. Parameter MUST be configurable, but has both canModLocal=false and canModRemote=false.]
- `E-OC-00005` [Option config parameter ${paramId} is not configured properly. Parameter MUST NOT be configurable, but has canModRemote=true.]

6. **Assert that the `handpay` profile values match the `optionConfig` values.**
{067d6309-bd40-4a98-0022-3eb1b5aed047}

Error

- `E-OC-00003` [Option config parameter ${paramId} of ${actual} does not match expected value of ${expected}.]

Test Case: JP-COR-00005

This case verifies that the EGM returns a `G2S_APX003 Invalid Device Identifier` error when device ID zero (0) is used.

Objective

Verify that the EGM returns a `G2S_APX003` error when device ID zero (0) is used as a class-level attribute.

Requirements Under Test

- 1.7.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Send a `getHandpayLog` command to device ID zero (0), and verify that the EGM responds with a `G2S_APX003` error.**
   {01c0eadf-b831-43ac-8026-bfef15082fa5}

   #### Command

   - `handpay.getHandpayLog`

     Expect **G2S_APX003**

   #### Errors

   - Send Request Errors
   - `E-XX-00015` [EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).]

Test Case: JP-COR-00006

This case verifies that forced event subscriptions are treated as if they are set by the host.

Objective

Verify that the EGM sends `handpay` events when a forced event subscription exists.

Requirements Under Test

- 1.23.25
- 1.23.26
- 11.16.1
- 11.17.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_eventHandler	owner and configurator
G2S_optionConfig	owner

Test Procedure

1. **Clear event subscriptions for the `handpay` device.**
   {01bf26c5-75e1-4410-006c-f637b0ab7fe0}

   Commands
   - `eventHandler.clearEventSub`
   - `eventHandler.getEventSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set a forced event subscription for `handpay` events.**
   {02c8d78a-cce6-47dd-00c4-944aee4ae1aa}

Commands
- `optionConfig.setOptionChange`
- `eventHandler.getEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors
- Send Request Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

3. **Verify that `G2S_JPE003 Device Disabled by Host` is generated when the device is host disabled.**
{030cb025-dfd3-4dc9-805e-89495f774875}

Command
- `handpay.setHandpayState`

Event
(Time out: ${event.timeOut})
- `G2S_JPE003` [Device Disabled by Host]

Errors
- Send Request Errors
- Event Checking Errors

4. **Verify that `G2S_JPE004 Device Not Disabled by Host` is generated when the device is host enabled.**
{044a46db-40b3-4f22-007f-80453f6ac086}

Command
- `handpay.setHandpayState`

Event
(Time out: ${event.timeOut})
- `G2S_JPE004` [Device Enabled by Host]

Errors
- Send Request Errors
- Event Checking Errors

5. **Clear the forced event subscription.**
{057a22eb-5b3f-49c4-0083-c689c495503f}

Command

- `optionConfig.setOptionChange`

Event

(Time out: ${event.timeOut})

- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors

- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

Test Case: JP-COR-00007

This case verifies that the EGM does not process invalid XML.

Objectives

- Verify that the EGM validates the `getHandpayLog` command.
- Verify that the EGM validates the `commandId` attribute in the `handpay` class element.

Requirements Under Test

- 1.41.2
- 1.41.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send a `getHandpayLog` command with an invalid `totalEntries` attribute.**
   {01038d7e-38fe-4697-0058-2b17e5855f22}

   Command
   - `handpay.getHandpayLog`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

2. **Send a `getHandpayLog` command with an invalid `commandId` attribute.**
   {02e1656f-d350-41f6-8011-61a722e80a64}

   Command
   - `handpay.getHandpayLog`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors

- Send Request Errors
- E-CM-00039 [The EGM must validate the G2S request.]

Test Case: JP-COR-00008

This case verifies `handpay` log processing.

Objectives

- Verify that the `handpayLog` sequence numbers are contiguous.
- Verify that the EGM returns an empty `handpayLog` when the requested logSequence number is not in the log.
- Verify that the EGM returns the specified log entries when totalEntries attribute value is not zero (0).

Requirements Under Test

- 1.19.2
- 1.19.13
- 1.21.2
- 1.21.3
- 1.21.4
- 1.21.5
- 1.21.6
- 1.21.7
- 1.21.8
- 11.2.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Determine the size of the `handpayLog`.**
   {018de4f4-7290-42fe-802c-96b9b52506a7}

   Command
   - `handpay.getHandpayProfile`

   Error

- Send Request Errors

2. **Determine the `lastSequence` and `totalEntries` values in the `handpayLog`.**
{026faa5b-4241-4ae3-0065-cbdbd3409aa7}

Command
- `handpay.getHandpayLogStatus`

Error
- Send Request Errors

3. **Verify that the `getHandpayLog` command, with `lastSequence` and `totalEntries` attribute values of zero (0), retrieves the entire log.**
{0360504a-1cd5-4ac9-802b-95b5270e405d}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00001` [Handpay log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Verify that requesting the `logSequence` less than the smallest `logSequence` number returns an empty log.**
{04759330-8307-44df-8083-ba0ec1589884}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00004` [Handpay log MUST be empty when requesting a lastSequence number not in the log.]

5. **Verify that requesting the `logSequence` greater than the last `lastSequence` number returns an empty log.**
{05ac8cda-d600-4c99-00bb-f193cff0c326}

Command
- `handpay.getHandpayLog`

Errors

- Send Request Errors
- `E-JP-00004` [Handpay log MUST be empty when requesting a lastSequence number not in the log.]

6. **Verify that the EGM returns an `handpayLog` with the correct number of entries.**
{066d9736-8e23-47e7-801e-ef91a5bc03d9}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

7. **Verify that the EGM returns an `handpayLog` with one entry if the `totalEntries` equals 1.**
{0730456c-ff30-4295-00a2-e8e62c16e7e0}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

Test Case: JP-COR-00009

This case verifies disabled text is shown properly on the EGM.

Objectives

- Verify that disable text is shown when a required for play handpay device is disabled.
- Verify that empty text is not shown on the EGM.
- Verify that the EGM returns the specified log entries when totalEntries attribute value is not zero (0).

Requirements Under Test

- 3.3.18
- 3.5.2
- 11.5.3
- 11.5.4
- 11.16.1
- 11.17.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner

Required Configurations

Option	Value
handpayProfile.requiredForPlay	true

Test Procedure

1. **Disable the handpay device with a known disabled text.**
   {01458dfe-e482-4f65-00d3-bf0c33f619e7}

   #### Command

   - `handpay.setHandpayState`

   #### Actions

- Verify that text 'JP-COR-00009 step #1' appears onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_JPE003` [Device Disabled by Host]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors
- Send Request Errors
- Event Checking Errors
- DUT Error
- `E-CB-00014` [Required text is not displayed.]

2. **Attempt to disable the handpay device with an empty disabled text.**
{02dadfbf-b8d1-4222-809d-d6fe3efc6104}

Command
- `handpay.setHandpayState`

Actions
- Verify that text 'JP-COR-00009 step #2' does NOT appear onscreen.

Errors
- Send Request Errors
- Event Not Expected Error
- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]

3. **Enable the handpay device.**
{0372fe39-26ef-4aca-808c-14801b73a687}

Command
- `handpay.setHandpayState`

Actions
- Verify that text 'JP-COR-00009 step #3' does NOT appear onscreen.

Events
(Time out: ${event.timeOut})
- `G2S_JPE004` [Device Enabled by Host]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

- DUT Error
- `E-CB-00013` [Disable text MUST NOT be displayed.]

Test Case: JP-COR-00010

This case verifies handpay device configuration.

Objective

Verify that the EGM only exposes one handpay device.

Requirements Under Test

- 11.1.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	guest or non-guest or owner
G2S_communications	owner

Test Procedure

1. **Verify that the descriptor list has only one active handpay device.**
   {0184a509-89ce-476b-005b-f0f837a57b38}

   Command
   - `communications.getDescriptor`

   Errors
   - Send Request Errors
   - `E-JP-00005` [EGM MUST only expose one handpay device.]

Test Case: JP-COR-00011

This case verifies that the messages, events and meters associated with a cancel-credit handpay are correct.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that `handpayLog` has a separate `logSequence` counter.

Requirements Under Test

- 1.28.10
- 11.1.7
- 11.2.1
- 11.2.2
- 11.2.4
- 11.2.5
- 11.2.6
- 11.7.4
- 11.8.1
- 11.8.9
- 11.12.3
- 11.20.3
- 11.21.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner

Device	Permissions
G2S_meters	owner
G2S_eventHandler	owner

Required Configurations

Option	Value
handpayProfile.enabledLocalHandpay	true

Test Procedure

1. **Get the handpay profile.**
   {01840919-3a63-4896-8097-7f5cd3c1a4f6}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {02869dae-1d4f-418c-00b2-740e7d7298c5}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {033f15d6-a920-4ae7-8094-f35a497a141e}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
   {0417a05c-9944-4f37-80ad-b397872e2d19}

Command

- `noteAcceptor.getNoteAcceptorProfile`

Error

- Send Request Errors

5. **Get device and currency meters.**
{05e9e767-0870-42ef-8095-38caa3088ddb}

Command

- `meters.getMeterInfo`

Error

- Send Request Errors

6. **Instruct the user to insert a note.**
{06f152a0-b4c6-4efd-8021-a135a258c48a}

Action

- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

7. **Set the response manager to not send g2sAcks to `handpayRequest` commands.**
{07947b62-4486-4ddd-8012-00288161b17e}

8. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{081520d3-5ae9-47e6-804f-9cf41a96e6b0}

Command

- `handpay.handpayRequest`

Actions

- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})

- G2S_JPE101 [Handpay Pending]
- G2S_CBE210 [Device Action Locked EGM]

Errors

- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- E-JP-00006 [Handpay type must be ${handpayType}.]
- E-JP-00007 [Handpay ${cashType} requested amount must be ${requestAmt}.]
- E-JP-00008 [Handpay source reference list must be empty for a cancel-credit handpay.]
- E-JP-00014 [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

9. **Get the latest handpay log entry.**
{09137f2a-0de5-4d6b-80de-c57e69a48630}

Command

- handpay.getHandpayLog

Errors

- Send Request Errors
- E-JP-00012 [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

10. **Reset the response manager and wait for the EGM to resend the `handpayRequest` command.**
{1059839c-76f3-4111-00c7-938971cea92a}

Command

- handpay.handpayRequest

Event

(Time out: ${event.timeOut})

- G2S_JPE102 [Handpay Request Acknowledged]

Errors

- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- E-JP-00009 [Retried handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

11. **Set the response manager to not send g2sAcks to `keyedOff` commands.**
{1187e455-99a0-4628-808e-90cae309621d}

12. **Instruct user to key off the handpay.**
{12cdb19b-fc36-425b-80a3-3fa657a87103}

Command
- `handpay.keyedOff`

Actions
- Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})
- `G2S_JPE104` [Handpay Keyed Off]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors
- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-JP-00010` [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

13. **Reset the response manager and wait for the EGM to resend the keyedOff command**
{139d7017-cc1a-472a-80c0-86a1d707625a}

Event
(Time out: ${event.timeOut})
- `G2S_JPE105` [Key-Off Acknowledged]

Errors
- Event Checking Errors
- Event Not Expected Error

14. **Verify that the handpay log is contiguous and that the log size is correct.**
{1491b2f9-c4bc-4b42-8095-6524ce6baf08}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00001` [Handpay log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

15. **Verify that the `handpayLog` has a separate `logSequence` counter.**
{15b2a8aa-aca2-477f-80cd-8e10fe01ef01}

Command

- `eventHandler.getEventHandlerLog`

Errors

- Send Request Errors
- `E-XX-00016` [Each log MUST have a separate counter.]

Test Case: JP-COR-00012

This case verifies that the messages, events and meters associated with cancelling a cancel-credit handpay are correct.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that the EGM overwrites cancelled cancel-credit handpay log entries.

Requirements Under Test

- 11.1.7
- 11.2.1
- 11.2.2
- 11.2.5
- 11.2.6
- 11.2.7
- 11.7.4
- 11.8.1
- 11.8.9
- 11.12.1
- 11.20.3
- 11.21.1
- 11.23.1
- 11.24.1
- 11.25.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.enabledLocalHandpay	true

Test Procedure

1. **Get the handpay profile.**
   {01d532ee-93bd-479f-00d7-bf839149b6d8}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {02556ba2-4fa0-496c-001c-2d62b3fcd872}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {0337b21e-4400-42e1-007b-2a71a3020711}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
   {04dccf05-cf53-4364-80b6-4e6627f86acd}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error

- Send Request Errors

5. **Get device and currency meters.**
{0523cc40-3581-480c-0040-94e9e2c19d44}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

6. **Instruct the user to insert a note.**
{060ed99a-a198-4cf0-005a-49d6af9d951d}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

7. **Set the response manager to not send g2sAcks to `handpayRequest` commands.**
{0734423f-728a-4dce-8008-018f55ad3dc4}

8. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{08ccd8dd-0072-4ae8-0078-3f47758a5624}

Command
- `handpay.handpayRequest`

Actions
- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})
- `G2S_JPE101` [Handpay Pending]
- `G2S_CBE210` [Device Action Locked EGM]

Errors
- Expected Request Errors
- Event Checking Errors

- DUT Error
- `E-JP-00006` [Handpay type must be ${handpayType}.]
- `E-JP-00007` [Handpay ${cashType} requested amount must be ${requestAmt}.]
- `E-JP-00008` [Handpay source reference list must be empty for a cancel-credit handpay.]
- `E-JP-00014` [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

9. **Get the latest handpay log entry.**
{090ee5db-4f68-49bf-00bd-eba373616362}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]
- `E-JP-00012` [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

10. **Wait for the EGM to resend the handpayRequest command.**
{1034f1c8-791f-4c5b-0083-e897c06e2585}

Command
- `handpay.handpayRequest`

Event
(Time out: ${event.timeOut})
- `G2S_JPE102` [Handpay Request Acknowledged]

Errors
- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- `E-JP-00009` [Retried handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

11. **Instruct user to cancel the cancel-credit handpay.**
{11a0cfb5-cf34-4d47-8008-233a2bc6c8ac}

Command
- `handpay.keyedOff`

Actions
- Cancel Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})

- G2S_JPE106 [Canceled-Credit Request Canceled]
- G2S_CBE205 [EGM Enabled and Playable]
- G2S_JPE105 [Key-Off Acknowledged]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- E-JP-00010 [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

12. **Instruct user to cause a second cancel-credit handpay**
{12169ab9-d89f-44ad-805a-920b6c30c358}

Command

- handpay.handpayRequest

Actions

- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})

- G2S_JPE101 [Handpay Pending]
- G2S_CBE210 [Device Action Locked EGM]
- G2S_JPE102 [Handpay Request Acknowledged]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- E-JP-00006 [Handpay type must be ${handpayType}.]
- E-JP-00007 [Handpay ${cashType} requested amount must be ${requestAmt}.]
- E-JP-00008 [Handpay source reference list must be empty for a cancel-credit handpay.]
- E-JP-00013 [Handpay requests MUST have unique transaction IDs, a handpay request for ${transactionId} already exists.]

13. **Get the latest handay log entry and verify that log sequence and transaction ID are correct.**
{13f611d2-da34-4d09-8085-bde4f7af65c8}

Command

- handpay.getHandpayLog

Errors

- Send Request Errors
- `E-JP-00011` [Cancel-credit handpays must overwrite the last handpay log entry if that entry is a cancelled cancel-credit handpay.]
- `E-JP-00012` [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

14. **Instruct user to key off the handpay.**
{14773a6b-3546-49a3-8058-593a7b2c800e}

Command

- `handpay.keyedOff`

Actions

- Key Off to 'HANDPAY'.

Events

(Time out: ${event.timeOut})

- `G2S_JPE104` [Handpay Keyed Off]
- `G2S_JPE105` [Key-Off Acknowledged]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- `E-JP-00010` [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

15. **Verify that the handpay log is contiguous and that the log size is correct.**
{1521d98a-16cb-4265-008a-48932550e90c}

Command

- `handpay.getHandpayLog`

Errors

- Send Request Errors
- `E-JP-00001` [Handpay log list MUST be contiguous. A skip was detected at ${sequenceNumber}.]
- `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
- `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

Test Case: JP-COR-00013

This case verifies that the messages, events and meters associated with a game win handpay keyed off to the credit meter are correct.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that game play meters are updated after the payment method has been determined.

Requirements Under Test

- 11.1.7
- 11.1.8
- 11.1.9
- 11.1.10
- 11.2.1
- 11.2.2
- 11.2.8
- 11.8.4
- 11.8.6
- 11.20.3
- 11.21.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	(guest or owner) and configurator
G2S_gamePlay	owner
G2S_optionConfig	owner
G2S_noteAcceptor	guest or owner

Device	Permissions
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.enabledLocalHandpay	true

Test Procedure

1. **Determine the cabinet large win limit.**
{01692bc7-25ac-4ae1-80fc-946cb479a383}

   Command
   - `cabinet.getCabinetProfile`

   Error
   - Send Request Errors

2. **Determine the smallest active denom and game play profile.**
{02ec9d9a-cc15-4212-00e7-b04ea0311db2}

   Commands
   - `gamePlay.getGamePlayProfile`
   - `gamePlay.getGameDenoms`

   Error
   - Send Request Errors

3. **Set the large win limit.**
{03d3219c-fe33-4cd2-001d-75839aea4e29}

   Commands
   - `optionConfig.enterOptionConfigMode`
   - `optionConfig.setOptionChange`

   Errors
   - Send Request Errors
   - `E-XX-00002` [CVT timed out waiting for the response.]
   - `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
   - `E-XX-00005` [An unexpected error of ${errorCode} was received.]

4. **Get the note acceptor profile.**
{04a0abaa-3c24-45c1-0061-c154bf34673f}

   Command

- `noteAcceptor.getNoteAcceptorProfile`

Error

- Send Request Errors

5. **Get the sequence number of the last handpay log entry.**
{0511e444-2152-4965-8018-72809f571b45}

Command

- `handpay.getHandpayLogStatus`

Error

- Send Request Errors

6. **Play a game until a large win occurs.**

1. **Get the cabinet and game play meters.**
{0609c3e9-eb93-451b-8007-a2b8c9cbfa0c}

Command

- `meters.getMeterInfo`

Error

- Send Request Errors

2. **Instruct the user to insert a note to fund the credit meter.**
{07583380-43ca-4879-80de-6212846da308}

Action

- Insert a ${noteDenom} ${currency} note.

Events

(Time out: ${event.timeOut})

- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

3. **Instruct user to play a game.**
{08d1db03-3e22-439b-8022-2f1b53be83b2}

Command

- `handpay.handpayRequest`

Actions

- Play Game #${gamePlayDevice.deviceId}

Events
(Time out: ${event.timeOut})

- G2S_GPE103 [Primary Game Started] or G2S_GPE104 [Wager Changed]
- G2S_CBE314 [Game Combo Activated] or G2S_GPE105 [Primary Game Ended]
- G2S_GPE106 [Secondary Game Choice] or G2S_GPE109 [Secondary Game Started] or G2S_GPE110 [Secondary Game Ended] or G2S_GPE111 [Game Result]
- G2S_GPE112 [Game Ended] or G2S_GPE113 [Game Idle] or G2S_JPE101 [Handpay Pending]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- E-GP-00011 [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]
- E-JP-00016 [Handpay source references MUST be included in non-cancelled-credit handpays.]
- E-JP-00017 [The total value of source references of ${sourceAmount} MUST equal EGM paid amount ${egmPaidAmount} plus request amount ${requestAmount}.]

7. **If there was a game win that caused a handpay request.**

   a. **Get the latest handpay log entry.**
   {091d4a92-6372-4468-00b2-1af582c50845}

   Command

   - handpay.getHandpayLog

   Errors

   - Send Request Errors
   - E-JP-00012 [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

   b. **Instruct user to key off the game win handpay.**
   {107d9ef0-15bf-4426-0012-a4048113e547}

   Command

   - handpay.keyedOff

   Actions

   - Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})

- G2S_JPE104 [Handpay Keyed Off]
- G2S_GPE112 [Game Ended]
- G2S_GPE113 [Game Idle]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- E-GP-00011 [Game play cycle event ${eventCode} MUST not be generated when the game is in ${eventState}.]

8. **If there are available player credits then cause a cancel credit handpay**

   a. **Instruct user to cause a cancel credit handpay request.**
   {117e2f2c-2e0a-41ed-00a7-db67d80c4a50}

   Command

   - handpay.handpayRequest

   Actions

   - Cancel Credit Handpay.

   Events
   (Time out: ${event.timeOut})

   - G2S_JPE101 [Handpay Pending]
   - G2S_JPE102 [Handpay Request Acknowledged]
   - G2S_CBE210 [Device Action Locked EGM]

   Errors

   - Expected Request Errors
   - Event Checking Errors
   - DUT Error

   b. **Instruct user to key off the cancel credit handpay.**
   {12a9b86a-ca42-43e6-0022-5740fbcf8339}

   Command

   - handpay.keyedOff

   Actions

   - Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})

- `G2S_JPE104` [Handpay Keyed Off]
- `G2S_JPE105` [Key-Off Acknowledged]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Send Request Errors
- Event Checking Errors
- DUT Error

9. **Reset the large win limit to the original value.**
{1399f1d4-f91a-4b05-003e-9b54b1c73cc6}

Commands

- `optionConfig.setOptionChange`
- `optionConfig.enterOptionConfigMode`

Errors

- Send Request Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

Test Case: JP-COR-00014

This case verifies that the EGM processes handpays when the handpay device is disabled.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that the EGM processes cancel-credit handpays when the `handpay` device is disabled.

Requirements Under Test

- 1.36.9
- 3.3.8
- 3.3.16
- 3.3.17
- 3.3.18
- 3.5.16
- 11.1.7
- 11.2.1
- 11.2.2
- 11.2.6
- 11.3.4
- 11.3.5
- 11.5.3
- 11.7.3
- 11.8.2
- 11.16.1
- 11.17.1
- 11.20.3
- 11.21.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.disabledLocalHandpay	true
handpayProfile.requiredForPlay	true

Test Procedure

1. **Get the handpay profile.**
   {01e9b52a-e0f7-413c-002c-a2d3b60bfe61}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {0261a3d5-6664-4d4f-00d4-f880aac47091}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {037a4ea9-7c9e-465a-007d-6accdaec325c}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
{04a54002-86b9-4681-805d-342dabee5b4b}

Command
- `noteAcceptor.getNoteAcceptorProfile`

Error
- Send Request Errors

5. **Get device and currency meters.**
{051f882a-d5ec-4e1b-804a-8a13d56d2934}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

6. **Instruct the user to insert a note.**
{064d78ff-3655-46ae-80b9-1c2f09acc781}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

7. **Disable the `handpay` device.**
{078ab6e3-d24a-4f2b-8002-411d43c1e6ff}

Command
- `handpay.setHandpayState`

Events
(Time out: ${event.timeOut})
- `G2S_JPE003` [Device Disabled by Host]
- `G2S_CBE204` [Host Command Disabled EGM]

Errors

- Send Request Errors
- Event Checking Errors

8. **Verify the disable text is displayed on the EGM.**
{082dfcb4-0659-4bdd-8067-0d32391f7671}

Action

- Verify that text 'JP-COR-00014 step #7' appears onscreen.

Errors

- DUT Error
- `E-CB-00014` [Required text is not displayed.]

9. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{094e0294-0008-4d17-004b-c25739c7563b}

Command

- `handpay.handpayRequest`

Actions

- Cancel Credit Handpay.

Events

(Time out: ${event.timeOut})

- `G2S_JPE101` [Handpay Pending]
- `G2S_JPE102` [Handpay Request Acknowledged]

Errors

- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- `E-JP-00006` [Handpay type must be ${handpayType}.]
- `E-JP-00007` [Handpay ${cashType} requested amount must be ${requestAmt}.]
- `E-JP-00008` [Handpay source reference list must be empty for a cancel-credit handpay.]
- `E-JP-00014` [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

10. **Get the latest handpay log entry.**
{101900cf-3218-403e-0024-743ea5beb135}

Command

- `handpay.getHandpayLog`

Errors

- Send Request Errors
- E-JP-00012 [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

11. **Instruct user to key off the handpay.**
{115cf676-c8d7-45ae-807f-8df82d43cd0b}

Command
- handpay.keyedOff

Actions
- Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})
- G2S_JPE104 [Handpay Keyed Off]
- G2S_JPE105 [Key-Off Acknowledged]

Errors
- Expected Request Errors
- Event Checking Errors
- Event Not Expected Error
- DUT Error
- E-JP-00010 [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

12. **Enable the handpay device.**
{124e88d6-9da1-41f1-80c4-97a0dcc8c0af}

Command
- handpay.setHandpayState

Events
(Time out: ${event.timeOut})
- G2S_JPE004 [Device Enabled by Host]
- G2S_CBE205 [EGM Enabled and Playable]

Errors
- Send Request Errors
- Event Checking Errors

Test Case: JP-COR-00015

This case verifies that the EGM processes handpays when the handpay device is disabled and is not required for play.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that the EGM processes cancel-credit handpays when the `handpay` device is disabled.

Requirements Under Test

- 1.36.9
- 3.3.8
- 3.3.16
- 11.1.7
- 11.2.1
- 11.2.6
- 11.3.4
- 11.3.5
- 11.5.3
- 11.8.2
- 11.16.1
- 11.17.1
- 11.20.3
- 11.21.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner

Device	Permissions
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.disabledLocalHandpay	true
handpayProfile.requiredForPlay	false

Test Procedure

1. **Get the handpay profile.**
   {01e70fd1-0225-4522-80f5-2890a1ed1c36}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {02585ee7-675d-41b3-80ff-78cab17c534d}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {03a12239-c90c-4823-00fb-3a598e660904}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
   {048acbf1-6472-42ef-00b1-921f18fd7699}

Command

- `noteAcceptor.getNoteAcceptorProfile`

Error

- Send Request Errors

5. **Get device and currency meters.**
{05299c08-9660-4049-80f5-7a6e48bce864}

Command

- `meters.getMeterInfo`

Error

- Send Request Errors

6. **Instruct the user to insert a note.**
{064dffd2-6b15-44af-80f4-82fe37c6231e}

Action

- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})

- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors

- Event Checking Errors
- DUT Error

7. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{07f92ebc-24f9-4d8b-8078-82f2b3ef08b9}

Command

- `handpay.handpayRequest`

Actions

- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})

- `G2S_JPE101` [Handpay Pending]
- `G2S_JPE102` [Handpay Request Acknowledged]
- `G2S_CBE210` [Device Action Locked EGM]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- `E-JP-00006` [Handpay type must be ${handpayType}.]
- `E-JP-00007` [Handpay ${cashType} requested amount must be ${requestAmt}.]
- `E-JP-00008` [Handpay source reference list must be empty for a cancel-credit handpay.]
- `E-JP-00014` [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

8. **Get the latest handpay log entry.**
{083ea993-da36-4769-8078-93dfaaa59544}

Command

- `handpay.getHandpayLog`

Errors

- Send Request Errors
- `E-JP-00012` [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

9. **Instruct user to key off the handpay.**
{09f04a85-daa3-42e7-8071-fb5b41a41039}

Command

- `handpay.keyedOff`

Actions

- Key Off to 'HANDPAY'.

Events

(Time out: ${event.timeOut})

- `G2S_JPE104` [Handpay Keyed Off]
- `G2S_JPE105` [Key-Off Acknowledged]
- `G2S_CBE205` [EGM Enabled and Playable]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- `E-JP-00010` [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

Test Case: JP-COR-00016

This case verifies that the messages, events and meters associated with a cancel-credit handpay are correct when owner host authorizes a local handpay.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that when the owner host authorizes a local handpay that the handpay log entry is updated properly.

Requirements Under Test

- 11.1.7
- 11.2.1
- 11.2.2
- 11.2.6
- 11.8.1
- 11.10.2
- 11.10.7
- 11.10.9
- 11.20.3
- 11.21.1
- 11.22.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.enabledLocalHandpay	true

Test Procedure

1. **Get the handpay profile.**
   {01bccdac-88fa-4bbc-8086-ff534c083b55}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {02ab4986-f6bc-4b1c-00c3-4f3f2c094cef}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {030bb30d-6c85-48a6-80b9-cbc4c0731632}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
   {04f6262e-bd7c-4fd1-0054-23fff3d36e53}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error

- Send Request Errors

5. **Get device and currency meters.**
{05d7f981-58a2-4903-8057-a8c802ed6087}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

6. **Instruct the user to insert a note.**
{0669a9cc-00b3-452b-001e-27e6d448712d}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

7. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{079f797f-9a36-4fae-0046-c853f53ce2b1}

Command
- `handpay.handpayRequest`

Actions
- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})
- `G2S_JPE101` [Handpay Pending]
- `G2S_CBE210` [Device Action Locked EGM]
- `G2S_JPE102` [Handpay Request Acknowledged]

Errors
- Expected Request Errors
- Event Checking Errors
- DUT Error

- `E-JP-00006` [Handpay type must be ${handpayType}.]
- `E-JP-00007` [Handpay ${cashType} requested amount must be ${requestAmt}.]
- `E-JP-00008` [Handpay source reference list must be empty for a cancel-credit handpay.]
- `E-JP-00014` [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

8. **Get the latest handpay log entry.**
{083a6073-1a0e-4f1a-009c-db4cd99d071e}

Command
- `handpay.getHandpayLog`

Errors
- Send Request Errors
- `E-JP-00012` [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

9. **Remotely authorize a local handpay.**
{09e23e1b-4ba1-42ee-8055-4714c05a6fd4}

Command
- `handpay.setRemoteKeyOff`

Event
(Time out: ${event.timeOut})
- `G2S_JPE103` [Remote Key-Off Initiated]

Errors
- Send Request Errors
- Event Checking Errors
- `E-JP-00015` [Handpay remote key-off ack MUST set transactionId to ${expected}, but was ${actual}.]

10. **Instruct user to key off the handpay.**
{1043228f-cc62-4e7c-808f-522f946f21c5}

Command
- `handpay.keyedOff`

Actions
- Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})

- G2S_JPE104 [Handpay Keyed Off]
- G2S_JPE105 [Key-Off Acknowledged]
- G2S_CBE205 [EGM Enabled and Playable]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error
- E-JP-00010 [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

11. **Attempt to send a second setRemoteKeyOff to the EGM for the same transaction.**
{1120d663-3098-44ce-002b-8c72228c6cbb}

Command

- handpay.setRemoteKeyOff

  Expect **G2S_JPX003**

Errors

- Send Request Errors
- Event Not Expected Error
- E-JP-00018 [EGM MUST return a G2S_JPX003 Transaction Is Not Currently Pending error for an transaction that has already been keyed-off.]

Test Case: JP-COR-00017

This case verifies that the messages, events and meters associated with a cancel-credit handpay are correct when owner host remotely keys off the handpay.

Objectives

- Verify that the `handpay` events are correct.
- Verify that the `handpayLog` records match the `handpay` events.
- Verify that the locked device attributes in the `cabinetStatus` are updated properly.
- Verify that when the owner host remotely keys-off a handpay that the handpay log entry is updated properly.

Requirements Under Test

- 11.1.7
- 11.2.1
- 11.2.2
- 11.2.6
- 11.8.1
- 11.10.2
- 11.10.7
- 11.10.9
- 11.20.3
- 11.21.1
- 11.22.1
- 11.23.1
- 11.24.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_cabinet	guest or owner
G2S_noteAcceptor	guest or owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.enabledRemoteHandpay	true

Test Procedure

1. **Get the handpay profile.**
   {0108916c-13ba-40e7-8078-a1c6617787c9}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Get the sequence number of the last handpay log entry.**
   {027d3c6e-61c0-4427-8086-efb7b2f8fe4d}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

3. **If the handpay log is not empty.**

   a. **Get the last handpay log entry to determine if it was a cancelled cancel-credit handpay.**
   {038b8b7c-a23a-4e58-8072-72cda287d896}

   Command
   - `handpay.getHandpayLog`

   Errors
   - Send Request Errors
   - `E-JP-00002` [Handpay log MUST start with logSequence number ${sequenceNumber}.]
   - `E-JP-00003` [Handpay log MUST have ${logEntries} entries.]

4. **Get the note acceptor profile.**
   {046eb09c-d13b-4393-00f6-aa57ff840f93}

   Command
   - `noteAcceptor.getNoteAcceptorProfile`

   Error

- Send Request Errors

5. **Get device and currency meters.**
{058ed9ab-a146-4304-8050-8c5851bb0b9d}

Command
- `meters.getMeterInfo`

Error
- Send Request Errors

6. **Instruct the user to insert a note.**
{06e5f88c-b00e-4ccf-808b-2caa3241a915}

Action
- Insert a ${noteDenom} ${currency} note.

Events
(Time out: ${event.timeOut})
- `G2S_NAE116` [Note In Escrow]
- `G2S_NAE114` [Note Stacked]

Errors
- Event Checking Errors
- DUT Error

7. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
{07bcded3-4832-4245-00cc-9a57b5798f26}

Command
- `handpay.handpayRequest`

Actions
- Cancel Credit Handpay.

Events
(Time out: ${event.timeOut})
- `G2S_JPE101` [Handpay Pending]
- `G2S_CBE210` [Device Action Locked EGM]
- `G2S_JPE102` [Handpay Request Acknowledged]

Errors
- Expected Request Errors
- Event Checking Errors
- DUT Error

- E-JP-00006 [Handpay type must be ${handpayType}.]
- E-JP-00007 [Handpay ${cashType} requested amount must be ${requestAmt}.]
- E-JP-00008 [Handpay source reference list must be empty for a cancel-credit handpay.]
- E-JP-00014 [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

8. **Get the latest handpay log entry.**
{08a66667-ec18-4168-00ac-778857807fc2}

Command
- handpay.getHandpayLog

Errors
- Send Request Errors
- E-JP-00012 [Handpay log entry MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

9. **Remotely key-off the handpay.**
{09b6ce95-8f1c-49dd-809d-4cec92587abd}

Commands
- handpay.setRemoteKeyOff
- handpay.keyedOff

Events
(Time out: ${event.timeOut})
- G2S_JPE103 [Remote Key-Off Initiated]
- G2S_JPE104 [Handpay Keyed Off]
- G2S_JPE105 [Key-Off Acknowledged]
- G2S_CBE205 [EGM Enabled and Playable]

Errors
- Expected Request Errors
- Send Request Errors
- Event Checking Errors
- E-JP-00010 [Handpay keyed-off request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]
- E-JP-00015 [Handpay remote key-off ack MUST set transactionId to ${expected}, but was ${actual}.]

10. **Attempt to send a second setRemoteKeyOff to the EGM for the same transaction.**
{107b663b-e15e-48d9-802b-c31bc4d757d3}

Command
- handpay.setRemoteKeyOff

Expect **G2S_JPX003**

Errors

- Send Request Errors
- Event Not Expected Error
- `E-JP-00018` [EGM MUST return a G2S_JPX003 Transaction Is Not Currently Pending error for an transaction that has already been keyed-off.]

Test Case: JP-COR-00018

This case verifies that the host sends a valid getHandpayProfile command.

Objective

Verify the getHandpayProfile command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Instruct user to send getHandpayProfile command from the host.**
   {0152326c-3013-432f-829c-f26d4b6fddac}

   #### Command

   - `handpay.getHandpayProfile`

   #### Actions

   - Instruct the user to 'Send getHandpayProfile to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00019

This case verifies that the host sends a valid setHandpayState command.

Objective

Verify the setHandpayState command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Instruct user to send setHandpayState command from the host.**
   {01e462cf-4d98-4236-b5f9-6c9dd6200411}

   #### Command

   - `handpay.setHandpayState`

   #### Actions

   - Instruct the user to 'Send setHandpayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00020

This case verifies that the host sends a valid getHandpayStatus command.

Objective

Verify the getHandpayStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Instruct user to send getHandpayStatus command from the host.**
   {01f9e51f-bec1-4636-b9f0-0a94d0c913f7}

   #### Command

   - `handpay.getHandpayStatus`

   #### Actions

   - Instruct the user to 'Send getHandpayStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00021

This case verifies that the host sends a valid setRemoteKeyOff command.

Objective

Verify the setRemoteKeyOff command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Instruct user to send setRemoteKeyOff command from the host.**
   {010a036b-5268-4a17-83a2-62086b5ef500}

   #### Command

   - `handpay.setRemoteKeyOff`

   #### Actions

   - Instruct the user to 'Send setRemoteKeyOff to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00022

This case verifies that the host sends a valid getHandpayLogStatus command.

Objective

Verify the getHandpayLogStatus command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Instruct user to send getHandpayLogStatus command from the host.**
   {01321a9b-50af-4add-b67e-06b4883eca41}

   #### Command

   - `handpay.getHandpayLogStatus`

   #### Actions

   - Instruct the user to 'Send getHandpayLogStatus to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00023

This case verifies that the host sends a valid getHandpayLog command.

Objective

Verify the getHandpayLog command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	guest or owner

Test Procedure

1. **Instruct user to send getHandpayLog command from the host.**
   {01a5b17e-2366-4881-99ab-360c5eecb0ed}

   #### Command

   - `handpay.getHandpayLog`

   #### Actions

   - Instruct the user to 'Send getHandpayLog to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: JP-COR-00024

This case verifies that the host does not process handpay commands with the wrong session type.

Objectives

- Verify that the host does not process a handpayAck sent as a request.
- Verify that the host does not process a handpayRequest sent as a response.
- Verify that the host does not process a handpayRequest sent as a notification.

Requirements Under Test

- 1.4.4
- 1.4.6
- 1.4.7

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send a handpayAck as a request and verify G2S_APX008 Command Not Supported error code.**
   {01962ec5-4a7c-4332-ac90-3560274e101b}

   Command
   - `handpay.handpayAck`

     Expect **G2S_APX008**

   Error
   - Send Request Errors

2. **Send a handpayRequest as a response to the host and verify that no response is sent from the host.**
   {027c5842-59ee-44ee-b48c-700e7ae373d8}

   Command
   - `handpay.handpayRequest`

Expect **G2S_APX009** or **G2S_APX014**

Errors

- Send Request Errors
- `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]
- `E-XX-00019` [An unexpected response of ${response} was received.]

3. **Send a handpayRequest as a notification to the host and verify that no response is sent from the host.**
{035acb6a-f9d0-4f99-ab94-dc48cf954f3c}

Command

- `handpay.handpayRequest`

Errors

- Send Request Errors
- `E-CM-00011` [Endpoint MUST NOT send application-level response for a notification.]
- `E-XX-00019` [An unexpected response of ${response} was received.]

Test Case: JP-COR-00025

This case verifies that the host still works after receiving the wrong response to setHandpayState.

Objective

Verify that the host is still working after receiving the wrong response to setHandpayState.

Requirements Under Test

- 1.4.5

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Set the response manager to respond with a handpayProfile response to a setHandpayState request.**
   {012756a2-3f5f-4c19-8377-c91045159ab2}

2. **Instruct user to send setHandpayState command from the host.**
   {0259c4f8-e6cf-48cb-9391-c1361db09abd}

   #### Command

   - `handpay.setHandpayState`

   #### Actions

   - Instruct the user to 'Send setHandpayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {0363c362-5e61-4c6d-bc60-ae48b4b239aa}

4. **Send a keyedOff to the host to verify that it is still working.**
   {040e0218-01de-437e-be4d-9ad37235f4c7}

   #### Command

- `handpay.keyedOff`

Error

- Send Request Errors

Test Case: JP-COR-00026

This case verifies that the host send the proper error code for a schema invalid handpayRequest command.

Objective

Verify that the host sends G2S_MSX004 Incomplete/Malformed XML, G2S_MSX005 Invalid Data Type Encountered or G2S_APX004 Incomplete/Malformed XML error for a schema invalid handpayRequest.

Requirements Under Test

- 1.27.33
- 1.41.2
- 1.41.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send an invalid handpayRequest with a missing handpayDateTime attribute and verify G2S_MSX004, G2S_MSX005 or G2S_APX004 error code.**
   {01b5f95a-abee-43a6-8c1a-9ce9214fc913}

   Command
   - `handpay.handpayRequest`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Error
   - Send Request Errors

Test Case: JP-COR-00027

This case verifies that the host handle multiple handpayRequest commands.

Objectives

- Verify that the host sends five handpayAcks for five handpayRequest commands.
- Verify that the host sends a handpayAck for an already keyed off handpay.

Requirements Under Test

- 1.28.9
- 11.3.1
- 11.8.14

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send five handpayRequest commands and verify that the host sends five handpayAcks.**
   {013c2b9a-94cb-4157-955d-a67460a9b558}

   Command
   - `handpay.handpayRequest`

   Error
   - Send Request Errors

2. **Send keyedOff command to the host and verify a keyedOffAck response.**
   {02b42ccc-b346-44b6-bc0a-6d563532a083}

   Command
   - `handpay.keyedOff`

   Error
   - Send Request Errors

3. **Send handpayRequest command to the host with the same transaction ID and verify a handpayAck response.**
{032910eb-fa66-4d1d-ae85-cd1ded73c098}

Command

- `handpay.handpayRequest`

Error

- Send Request Errors

Test Case: JP-COR-00028

This case tests that the host sends the appropriate error for an unknown handpay command.

Objectives

- Verify that the host sends G2S_APX008 Command Not Supported or G2S_APX015 Unknown Command Encountered error for an unknown handpay command.
- Verify that if the host sends G2S_APX015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_communications	owner

Test Procedure

1. **Send a unknown handpay command and verify G2S_APX008 or G2S_APX015 error.**
   {019802cf-ea4b-4ac7-b105-08c64e86d9d4}

   #### Command

   - `handpay.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: JP-COR-00029

This case tests that the host can handle duplicate remoteKeyOffAck responses.

Objective

Verify that the host still works after receiving multiple remoteKeyOffAck response for one setRemoteKeyOff command.

Requirements Under Test

- 11.3.1
- 11.11.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send a handpay request and prompt user to send a setRemoteKeyOff.**
   {01aad907-af61-4058-a1d2-62acea77bbcb}

   Command
   - `handpay.setRemoteKeyOff`

   Actions
   - Instruct the user to 'Send setRemoteKeyOff to device ${deviceUnderTest.deviceId}.'.

   Errors
   - Expected Request Errors
   - DUT Error

2. **Send a duplicate remoteKeyOffAck response.**
   {02c4d76e-bb6a-4f60-8297-803562c85eb2}

   Command
   - `handpay.remoteKeyOff`

   Errors

- Send Request Errors
- `E-XX-00019` [An unexpected response of ${response} was received.]

3. **Verify that the host is still working by sending a keyedOff command.**
{03ba4f4e-2f7c-43f8-81d1-5847dc862d0f}

Command
- `handpay.keyedOff`

Error
- Send Request Errors

4. **If the handpay is still pending.**

   a. **Key off the handpay.**
   {04d03934-20c9-48a4-8215-e6fe155c9473}

Test Case: JP-COR-00030

This case tests that the host does not use handpay application errors for invalid keyedOff commands.

Objective

Verify that the host still works after receiving multiple incorrect keyedOff responses for one setRemoteKeyOff command.

Requirements Under Test

- 11.3.1
- 11.12.2
- 11.12.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_noteAcceptor	guest or non-guest or owner

Test Procedure

1. **Send a handpay request to the host.**
   {018db053-56b4-453c-952b-c4fc9c8b27be}

2. **Send a keyOff with a key off type of CVT_localKey.**
   {0216c41f-0844-4f02-8d27-1f35810a48a1}

   Command
   - `handpay.keyedOff`

   Error
   - Send Request Errors

3. **Send a keyedOff with a key off for the maximum amount.**
   {03bdca3d-d642-4672-8bad-e9891baef61b}

   Command
   - `handpay.keyedOff`

   Error

- Send Request Errors

4. **Send a keyedOff with a key off for the correct amount.**
{04aefd44-9f76-4434-b277-18e25214cba0}

Command

- `handpay.keyedOff`

Error

- Send Request Errors

Test Case: JP-COR-00031

This case tests that the host handles unknown error codes in handpay class.

Objectives

- Verify that the host is still working after sending an unknown error code in the handpay class.
- Verify that the host is does not use handpay application errors for invalid keyedOff commands.

Requirements Under Test

- 1.42.3
- 11.3.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: handpay

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Set the response manager to send 'CVT_ error' for setHandpayState command.**
   {012063ba-6ac4-4e76-a009-2426d3ec467d}

2. **Instruct user to send setHandpayState command from the host.**
   {029803f3-5413-4477-b8d8-8c6996239d11}

   #### Command
   - `handpay.setHandpayState`

   #### Actions
   - Instruct the user to 'Send setHandpayState to device ${deviceUnderTest.deviceId}.'.

   #### Errors
   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {033e1f87-1cb9-4cf1-b35f-8784ada4be4d}

4. **Send a keyedOff command to the host to verify that it is still working.**
   {04919b9f-79b2-42c2-8512-db7218d9eeb9}

Command

- `handpay.keyedOff`

Error

- Send Request Errors

Test Case: JP-COR-00032

This case verifies that the EGM sends the prescribed Handpay response to each Handpay request.

Objectives

- Verify that the EGM sends a `handpayProfile` response to a `getHandpayProfile` request.
- Verify that the EGM sends a `handpayStatus` response to a `setHandpayState` request.
- Verify that the EGM sends a `handpayStatus` response to a `getHandpayStatus` request.
- Verify that the EGM sends a `G2S_JPX003 Transaction Is Not Currently Pending` error code to a `setRemoteKeyOff` request.
- Verify that the EGM sends a `handpayLogStatus` response to a `getHandpayLogStatus` request.
- Verify that the EGM sends a `handpayLogList` response to a `getHandpayLog` request.

Requirements Under Test

- 1.27.33
- 11.3.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Send a getHandpayProfile request to the EGM and verify a handpayProfile response.**
   {01f2b9c2-8e22-4cfe-ac02-aaf0a1b10cfb}

   Command
   - `handpay.getHandpayProfile`

   Error
   - Send Request Errors

2. **Send a setHandpayState request to the EGM and verify a handpayStatus response.**
   {029b3283-2f14-47f3-8d33-3cad38e94964}

Command

- `handpay.setHandpayState`

Error

- Send Request Errors

3. **Send a getHandpayStatus request to the EGM and verify a handpayStatus response.**
{036c950b-a403-441c-b4bd-25948db3ae09}

Command

- `handpay.getHandpayStatus`

Error

- Send Request Errors

4. **Send a setRemoteKeyOff request to the EGM and verify a G2S_JPX003 error response.**
{04db9ee5-217a-4349-9ace-e45d3fbf8a65}

Command

- `handpay.setRemoteKeyOff`
  Expect **G2S_JPX003**

Error

- Send Request Errors

5. **Send a getHandpayLogStatus request to the EGM and verify a handpayLogStatus response.**
{05e25bdb-7769-44df-bf6f-eebcecc61d18}

Command

- `handpay.getHandpayLogStatus`

Error

- Send Request Errors

6. **Send a getHandpayLog request to the EGM and verify a handpayLogList response.**
{06c9f966-dbd0-42df-b8ea-656aa731fc45}

Command

- `handpay.getHandpayLog`

Error

- Send Request Errors

Test Case: JP-COR-00033

This case continuously tests that `handpayRequest.requestPromoAmt` and `handpayRequest.requestNonCashAmt` are zero.

Objective

Continuously verify that `handpayRequest.requestPromoAmt` and `handpayRequest.requestNonCashAmt` are zero.

Requirements Under Test

- 11.8.11

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Assert that requestPromoAmt and requestNonCashAmt attributes are set to zero in handpay request commands.**
   {01481397-ad6c-4507-bec9-4156ffe1ec8d}

   Error

   - `E-JP-00020` [In G2S 2.1, handpay request commands MUST set requestPromoAmt and requestNonCashAmt to zero.]

Test Case: JP-COR-00034

This case continuously tests that partial payments are recorded in the `egmPaidCashableAmt`, `egmPaidPromoAmt`, and/or `egmPaidNonCashAmt` attribute of the `handpayRequest`.

Objective

Continuously verify that partial payments are recorded in the `egmPaidCashableAmt`, `egmPaidPromoAmt`, and/or `egmPaidNonCashAmt` attribute of the `handpayRequest`.

Requirements Under Test

- 11.8.5

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: handpay
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner

Test Procedure

1. **Assert that partial payments are recorded in the egmPaidCashableAmt, egmPaidPromoAmt, and/or egmPaidNonCashAmt attribute of the handpayRequest.**
   {019023df-2178-47c5-98e9-4558c7fad564}

   Error

   - `E-JP-00017` [The total value of source references of ${sourceAmount} MUST equal EGM paid amount ${egmPaidAmount} plus request amount ${requestAmount}.]

Test Case: JP-COR-00035

This case verifies that the EGM supports partial handpays when `handpayProfile.partialHandpays` is set to true.

Objectives

- Verify that if `handpayProfile.partialHandpays` is set to true that the EGM accepts partial handpays.
- Verify that the EGM commits a partial handpay and generates a new handpay request for the remaining amount.
- Verify that the EGM responds with G2S_JPX004 Invalid Key-off Amount error when the key off amounts are invalid.

Requirements Under Test

- 11.7.9
- 11.10.4
- 11.10.5
- 11.10.6
- 11.10.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: EGM
- **GSA Class**: handpay
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.partialHandpays	true

Test Procedure

1. **Get the handpay profile to determine which key off types are allowed.**
   {015654ef-31a7-47d7-8ab2-6fc1cbc2acb4}

   Command

- `handpay.getHandpayProfile`

Error

- Send Request Errors

2. **If the EGM supports local or remote handpay key off type.**

   a. **Get the sequence number of the last handpay log entry.**
   {02c7e04f-6ba5-47ba-a2cd-aa6ad1392ed1}

   Command

   - `handpay.getHandpayLogStatus`

   Error

   - Send Request Errors

   b. **Instruct user to fund the EGM.**
   {03b7bb47-4629-4d97-aa92-487417090661}

   Action

   - Instruct the user to 'Please fund the EGM.'.

   Errors

   - DUT Error
   - `E-XX-00022` [The user failed to fund the EGM.]

   c. **Get device meters.**
   {047210fd-e3c3-41d7-b74f-479b3e1474cb}

   Command

   - `meters.getMeterInfo`

   Error

   - Send Request Errors

   d. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
   {053c6a40-c909-4dbb-b0cf-de1d087d3c8a}

   Command

   - `handpay.handpayRequest`

   Actions

   - Cancel Credit Handpay.

   Events
   (Time out: ${event.timeOut})

- G2S_JPE101 [Handpay Pending]
- G2S_JPE102 [Handpay Request Acknowledged]

Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error

e. **Verify that the EGM responds with G2S_JPX004 error when the key off amounts are invalid.**
{06c8c4e2-abe0-4d41-85bb-5345a62fa16f}

Command

- handpay.setRemoteKeyOff

  Expect **G2S_JPX004**

Error

- Send Request Errors

3. **Set the response manager to suppress responses to handpayRequest commands.**
{073abca3-ef24-46cb-b10c-25ab5e9ce39b}

Error

- DUT Error

4. **If remote key off is selected**

a. **Verify that the EGM supports a partial remote handpay.**
{082848bc-c755-452d-a070-e9bdf77fe7c7}

Command

- handpay.setRemoteKeyOff

Events

(Time out: ${event.timeOut})

- G2S_JPE103 [Remote Key-Off Initiated]
- G2S_JPE104 [Handpay Keyed Off]
- G2S_JPE105 [Key-Off Acknowledged]

Errors

- Send Request Errors
- Event Checking Errors

5. **If local key off is selected**

    a. **Verify that the EGM supports a partial local handpay.**
    {09f1f0e5-d799-40b1-9bca-37bf880339f7}

    Command
    - `handpay.setRemoteKeyOff`

    Event
    (Time out: ${event.timeOut})
    - `G2S_JPE103` [Remote Key-Off Initiated]

    Errors
    - Send Request Errors
    - Event Checking Errors

    b. **Instruct the user to key-off the handpay.**
    {1055c296-a416-4240-863d-433686a50ca8}

    Action
    - Key Off to 'HANDPAY'.

    Events
    (Time out: ${event.timeOut})
    - `G2S_JPE104` [Handpay Keyed Off]
    - `G2S_JPE105` [Key-Off Acknowledged]

    Errors
    - Event Checking Errors
    - DUT Error
    - `E-JP-00019` [The user failed to create a handpay.]

6. **Get the device meters.**
{11a3cab9-98df-4916-a035-f5e94b0d2f94}

    Command
    - `meters.getMeterInfo`

    Error
    - Send Request Errors

7. **If the EGM supports local or remote handpay key off type.**

    a. **Reset the response manager and wait for the second handpayRequest.**
    {12166628-c499-4c30-a1a6-5d381d1c0453}

    Command

- handpay.handpayRequest

Errors

- Expected Request Errors
- E-JP-00014 [Handpay request MUST set attribute ${attribute} to ${expected}, but was ${actual}.]

b. **Instruct user to key off the handpay.**
{137a2b4e-4f40-4305-8ebe-e555a750b91a}

Action

- Key Off to 'HANDPAY'.

Events
(Time out: ${event.timeOut})

- G2S_JPE104 [Handpay Keyed Off]
- G2S_JPE105 [Key-Off Acknowledged]

Errors

- Event Checking Errors
- DUT Error
- E-EH-00030 [Event ${eventCode} was expected, but it was not received.]
- E-JP-00019 [The user failed to create a handpay.]

Test Case: JP-COR-00036

This case verifies that the EGM does not support partial handpays when `handpayProfile.partialHandpays` is set to false.

Objectives

- Verify that if `handpayProfile.partialHandpays` is set to false that the EGM does not accept partial handpays.
- Verify that the EGM responds with G2S_JPX004 Invalid Key-off Amount error when the key off amounts are invalid.

Requirements Under Test

- 11.7.9
- 11.10.6
- 11.10.11
- 11.10.12
- 11.10.13

Test Type: QUICK

Criteria

- **Protocol**: G2S 2.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_meters	owner

Required Configurations

Option	Value
handpayProfile.partialHandpays	false

Test Procedure

1. **Get the handpay profile to determine which key off types are allowed.**
   {0110bff8-b083-4358-b056-e77137cb5a15}

   Command
   - `handpay.getHandpayProfile`

   Error

- Send Request Errors

2. **If the EGM supports local or remote handpay key off type.**

   a. **Get the sequence number of the last handpay log entry.**
   {0245bb29-e4e7-4e42-922a-07ed995fb711}

   Command
   - `handpay.getHandpayLogStatus`

   Error
   - Send Request Errors

   b. **Instruct user to fund the EGM.**
   {033b1fef-9368-48a4-9a65-d1a7a377e0ac}

   Action
   - Instruct the user to 'Please fund the EGM.'.

   Errors
   - DUT Error
   - `E-XX-00022` [The user failed to fund the EGM.]

   c. **Get device meters.**
   {04b4ac0d-5a9e-4c54-b727-8e490c89df8f}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

   d. **Instruct user to cause a cancel-credit handpay and wait for a `handpayRequest`.**
   {05ccc45c-b6c3-4275-ad31-af7462b04d36}

   Command
   - `handpay.handpayRequest`

   Actions
   - Cancel Credit Handpay.

   Events
   (Time out: ${event.timeOut})
   - `G2S_JPE101` [Handpay Pending]
   - `G2S_JPE102` [Handpay Request Acknowledged]

   Errors

- Expected Request Errors
- Event Checking Errors
- DUT Error

e. **Verify that the EGM responds with G2S_JPX004 error when the key off amounts are invalid.**
{06d8f737-366b-4041-aa1e-0175e6245dc7}

Command
- `handpay.setRemoteKeyOff`

  Expect **G2S_JPX004**

Error
- Send Request Errors

f. **Verify that the EGM does not supports a partial remote handpay.**
{07ca3b11-408e-423e-bf2f-33fcd879e6c6}

Command
- `handpay.setRemoteKeyOff`

  Expect **G2S_JPX004**

Errors
- Send Request Errors
- Event Not Expected Error

3. **If the EGM supports remote handpay key off type.**

a. **Send a setRemoteKeyOff command with keyOffType=G2S_remoteHandpay.**
{08bb8e27-79f4-4a47-ae28-b7021274d4af}

Command
- `handpay.setRemoteKeyOff`

Events
(Time out: ${event.timeOut})
- `G2S_JPE103` [Remote Key-Off Initiated]
- `G2S_JPE104` [Handpay Keyed Off]
- `G2S_JPE105` [Key-Off Acknowledged]

Errors
- Send Request Errors
- Event Checking Errors

4. **If the EGM supports local handpay key off type.**

    a. **Send a setRemoteKeyOff command with keyOffType=G2S_localHandpay.**
    {092c1296-118a-4d86-bd60-395343ab4614}

    Command
    - `handpay.setRemoteKeyOff`

    Event
    (Time out: ${event.timeOut})
    - `G2S_JPE103` [Remote Key-Off Initiated]

    Errors
    - Send Request Errors
    - Event Checking Errors

    b. **Instruct user to key off the handpay.**
    {108f82c9-ea81-462c-9797-e2bea46bcbb5}

    Action
    - Key Off to 'HANDPAY'.

    Events
    (Time out: ${event.timeOut})
    - `G2S_JPE104` [Handpay Keyed Off]
    - `G2S_JPE105` [Key-Off Acknowledged]

    Errors
    - Event Checking Errors
    - DUT Error
    - `E-JP-00019` [The user failed to create a handpay.]

Test Case: JP-COR-00037

This case verifies that the EGM does not send `handpayRequest` commands after the handpay log entry has been committed.

Objectives

- Verify that after the handpay log entry is no longer `G2S_handpayRequest` that the EGM does not send handpayRequest commands even if no handpayRequest commands were sent to the host.
- Verify that after the handpay log entry is no longer `G2S_handpayRequest` that the EGM does not send handpayRequest commands for unacknowledged handpay requests.

Requirements Under Test

- 11.8.10

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: handpay
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_handpay	owner
G2S_communications	owner

Test Procedure

1. **Get the comms profile to determine if the device is required for play.**
   {01a6a55f-65f9-475c-b809-7a0be1bf3307}

   Command
   - `communications.getCommsProfile`

   Error
   - Send Request Errors

2. **If the communications device is not required for play.**

   a. **Set the response manager to respond with G2S_MSX003 error to handpayProfile and commsOnLine commands.**
   {023f63c7-601d-4498-8862-dc8a3c8e367b}

b. **Send a getHandayProfile command and respond with a G2S_MSX003 to force the EGM to disconnect.**
{03a1fd46-29a3-4157-bc99-1a0f24bc5cf6}

Command
- `handpay.getHandpayProfile`

Error
- Send Request Errors

c. **Instruct user to fund the EGM.**
{04cb25a1-1e92-419b-9568-6bc396f6ed96}

Action
- Instruct the user to 'Please fund the EGM.'.

Errors
- DUT Error
- `E-XX-00022` [The user failed to fund the EGM.]

d. **Instruct the user to cause a cancel-credit handpay.**
{058e8a0b-bbaa-4253-8f13-c82de28a0949}

Action
- Cancel Credit Handpay.

Errors
- DUT Error
- `E-JP-00019` [The user failed to create a handpay.]

e. **Instruct user to key off the handpay.**
{061a9b78-f8ef-47e5-be29-dc78e097f273}

Action
- Key Off to 'HANDPAY'.

Error
- DUT Error

f. **Reset the response manager and wait for the EGM to reconnect.**
{07ad22ee-968f-4358-902a-2bd47739e054}

Commands
- `communications.commsOnLine`
- `communications.commsDisabled`

Error

- Expected Request Errors

g. **Enable communications and verify no handpayRequests are sent before the keyedOff command.**
{08094004-e336-4c2b-ad46-4a9705f966fe}

Commands

- `communications.setCommsState`
- `handpay.keyedOff`

Errors

- Expected Request Errors
- Send Request Errors
- `E-JP-00021` [EGM MUST NOT send a handpayRequest when the handpay is no longer in the G2S_handpayRequest state.]

3. **Set the response manager to not respond to handpay.handpayRequest commands.**
{09dcf9b6-9f76-4dac-b6cb-0a2e1cef1179}

4. **Instruct user to fund the EGM.**
{10ffbcf3-1113-4afe-9d29-dddb5ca55251}

Action

- Instruct the user to 'Please fund the EGM.'.

Errors

- DUT Error
- `E-XX-00022` [The user failed to fund the EGM.]

5. **Instruct the user to cause a cancel-credit handpay.**
{110b7ce3-f18b-4c43-8916-0515f775b57f}

Command

- `handpay.handpayRequest`

Actions

- Cancel Credit Handpay.

Errors

- Expected Request Errors
- DUT Error
- `E-JP-00019` [The user failed to create a handpay.]

6. **Instruct user to key off the handpay.**
{123f0d25-ea5c-4914-a751-4978e3c2b125}

Command

- `handpay.keyedOff`

Actions

- Key Off to 'HANDPAY'.

Errors

- Expected Request Errors
- DUT Error
- `E-JP-00021` [EGM MUST NOT send a handpayRequest when the handpay is no longer in the G2S_handpayRequest state.]

meters

Meters
Functional Groups

- Audit Meter Support (g2sAM) (AUD)
- Core Functionality (COR)

Test Case: MT-AUD-00001

This case verifies that the host sends a valid getMeterSub with meter subscription type of G2S_onAudit.

Objective

Verify the getMeterSub with meterSubType=G2S_onAudit from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Instruct user to send getMeterSub with meterSubType=G2S_onAudit from the host.**
   {015e32db-d901-4fb1-84da-9916be3f4b0a}

   #### Command

   - `meters.getMeterSub`

   #### Actions

   - Instruct the user to 'Send getMeterSub with meterSubType=G2S_onAudit to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-AUD-00002

This case tests that the host can handle receiving onPeriodic meters when onAudit is requested.

Objective

Verify that the host is still working after receiving onPeriodic meters when the host requested onAudit meters.

Requirements Under Test

- 5.14.1
- 5.16.8

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set the response manager to respond with a periodic meters for getMeterInfo.**
   {01c5580c-1b69-424e-9da0-ef99570445dc}

2. **Instruct user to send getMeterInfo for onAudit meters from the host.**
   {023bb4e8-dfe9-4d62-9060-825d0a8c2417}

   #### Command

   - `meters.getMeterInfo`

   #### Actions

   - Instruct the user to 'Send getMeterInfo for G2S_onAudit to device ${deviceUnderTest.deviceId} (1 of 2).'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03159af4-d3ed-428e-93e7-e17b39252788}

4. **Instruct user to send getMeterInfo for onAudit meters from the host.**
   {04651c3b-c514-4f3d-8b81-98ad4bcc42fc}

Command

- `meters.getMeterInfo`

Actions

- Instruct the user to 'Send getMeterInfo for G2S_onAudit to device ${deviceUnderTest.deviceId} (2 of 2).'.

Errors

- Expected Request Errors
- DUT Error

5. **Send a meterInfo to the host to verify that it is still working.**
{05f8bbdf-d68f-45f2-8280-4a278e455dc6}

Command

- `meters.meterInfo`

Error

- Send Request Errors

Test Case: MT-AUD-00003

This case verifies that the host sends a getMeterSub to get the audit meters.

Objective

Verify the getMeterSub with meterSubType=G2S_onAudit from the host is valid.

Requirements Under Test

- 5.5.9

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 2.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Instruct user to send getMeterSub with meterSubType=G2S_onAudit from the host.**
   {0132b161-c3a6-4f71-8583-3765378fbfa6}

   #### Command

   - `meters.getMeterSub`

   #### Actions

   - Instruct the user to 'Send getMeterSub with meterSubType=G2S_onAudit to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-COR-00001

This case continuously tests that `meters` commands are correct.

Objectives

- Continuously verify that there are no wildcards in `meters` commands from the EGM.
- Continuously verify that the correct session type is used for each meter info type.

Requirements Under Test

- 5.3.10
- 5.3.11
- 5.4.9
- 5.4.10
- 5.18.2

Test Type: CONTINUOUS

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Assert that there are no wildcards in the `meterSubList` or `meterInfo` commands.**
   {01f27a6c-e16e-4f39-989f-75e58ea76ecc}
2. **Assert that the correct session type is used for known meter info types in `meterInfo` commands.**
   {0273ec37-6e68-4c44-9f6b-29890bf26819}

Test Case: MT-COR-00002

This case verifies that the EGM returns a `G2S_APX003 Invalid Device Identifier` error when device ID zero (0) is used.

Objective

Verify that the EGM returns a `G2S_APX003` error when device ID zero (0) is used as a class-level attribute.

Requirements Under Test

- 1.7.4
- 1.21.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send a `getMeterInfo` command to device ID zero (0), and verify that the EGM responds with a `G2S_APX003` error.**
   {01c0ea90-0afc-4238-bb90-a35d4d3c5477}

   #### Command

   - `meters.getMeterInfo`

     Expect **G2S_APX003**

   #### Errors

   - Send Request Errors
   - `E-XX-00015` [EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).]

Test Case: MT-COR-00003

This case verifies that the EGM does not accept commands from non-guest hosts.

Objective

Verify that the EGM does not accept `meters` commands from non-guest hosts.

Requirements Under Test

- 1.8.5
- 1.8.14

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	non-guest

Test Procedure

1. **Send a `getMeterInfo` command, and verify that the EGM responds with a `G2S_APX012 Command Restricted to Owner and Guests` error.**
   {018aebb6-faeb-4256-842a-d04cae27ad31}

   Command
   - `meters.getMeterInfo`

     Expect **G2S_APX012**

   Errors
   - Send Request Errors
   - `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

2. **Send a `getMeterSub` command, and verify that the EGM responds with a `G2S_APX012` error.**
   {02abb541-25ea-49b7-8995-78c2947cc8ab}

   Command
   - `meters.getMeterSub`

     Expect **G2S_APX012**

   Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

3. **Send a `setMeterSub` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
{0378794a-58bb-4c53-8206-98fb424ed273}

Command

- meters.setMeterSub

  Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

4. **Send a `clearMeterSub` command, and verify that the EGM responds with a `G2S_APX010` error or a `G2S_APX012` error.**
{04610b07-92ba-4526-8ff3-86f31e2f8a92}

Command

- meters.clearMeterSub

  Expect **G2S_APX010** or **G2S_APX012**

Errors

- Send Request Errors
- E-CM-00036 [The EGM must not allow commands from the non-guest host.]

Test Case: MT-COR-00004

This case verifies that the EGM does not accept control commands from a guest host.

Objectives

- Verify that the EGM **does not** accept `meters` control commands from a guest host.
- Verify that the EGM accepts `meters` non-control commands from a guest host.

Requirements Under Test

- 1.8.14
- 5.14.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	guest

Test Procedure

1. **Send a `getMeterInfo` command, and verify that the EGM responds properly.**
   {016cc1b7-a602-4b65-bf1f-572c22903175}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

2. **Send a `getMeterSub` command, and verify that the EGM responds properly.**
   {02231ef1-e83e-45b8-8ece-9fef1c193736}

   Command
   - `meters.getMeterSub`

   Error
   - Send Request Errors

3. **Send a `setMeterSub` command, and verify that the EGM responds with a `G2S_APX010 Command Restricted to Owner` error.**

{03e413fe-d1a2-4175-a025-aed88bc4a90a}

Command

- `meters.setMeterSub`

  Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

4. **Send a `clearMeterSub` command, and verify that the EGM responds with a `G2S_APX010` error.**
{04c4ff09-6d24-445a-8a28-8097f1b17cac}

Command

- `meters.clearMeterSub`

  Expect **G2S_APX010**

Errors

- Send Request Errors
- `E-CM-00036` [The EGM must not allow commands from the non-guest host.]

Test Case: MT-COR-00005

This case verifies the events associated with meter subscriptions.

Objective

Verify that the EGM sends `meter` events.

Requirements Under Test

- 5.21.1
- 5.22.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set a `meter` subscription, and verify that the `G2S_MTE100` event is generated.**
   {0128f1e8-67f7-4ec1-8f6b-7a2ce681fe66}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Clear a `meter` subscription, and verify that the `G2S_MTE101` event is generated.**
   {0292a2a5-0012-4b84-92a2-9d5b0d05f3ff}

   Command
   - `meters.clearMeterSub`

   Event
   (Time out: ${event.timeOut})

- `G2S_MTE101` [Meter Subscription Cleared]

Errors

- Send Request Errors
- Event Checking Errors

Test Case: MT-COR-00006

This case verifies that forced event subscriptions are treated as if they are set by the host.

Objectives

- Verify that the EGM sends `meters` events when a forced event subscription exist.
- Verify that the EGM combines the host event subscription with forced event subscriptions.

Requirements Under Test

- 1.23.25
- 1.23.26
- 5.21.1
- 5.22.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_eventHandler	owner and configurator
G2S_optionConfig	owner

Test Procedure

1. **Clear event subscriptions for `meters` devices.**
   {01671a00-2463-4a22-91da-b64938e95647}

   #### Commands
   - `eventHandler.getEventSub`
   - `eventHandler.clearEventSub`

   #### Event
   (Time out: ${event.timeOut})
   - `G2S_EHE101` [Event Subscription Changed]

   #### Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set a forced event subscription that is not persisted for `G2S_MTE100` and `G2S_MTE101` events.**
{02e5b0e9-680b-4799-bcfe-322d7d3af04c}

Command
- `optionConfig.setOptionChange`

Event
(Time out: ${event.timeOut})
- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

3. **Set an event subscription that is persisted for the `G2S_MTE100` event.**
{034a6f9b-a183-4022-a93d-1f46d9b434c0}

Commands
- `eventHandler.getEventSub`
- `eventHandler.setEventSub`

Event
(Time out: ${event.timeOut})
- `G2S_EHE101` [Event Subscription Changed]

Errors
- Send Request Errors
- Event Checking Errors

4. **Set an end-of-day meter subscription.**
{045419ea-69f2-4de4-83ef-28735bf4ab85}

Command
- `meters.setMeterSub`

Event
(Time out: ${event.timeOut})
- `G2S_MTE100` [Meter Subscription Set]

Errors
- Send Request Errors
- Event Checking Errors

5. **Clear the end-of-day meter subscription.**
{05639ba8-df35-40c6-9501-2bf472849d30}

Command
- `meters.clearMeterSub`

Event
(Time out: ${event.timeOut})
- `G2S_MTE101` [Meter Subscription Cleared]

Errors
- Send Request Errors
- Event Checking Errors

6. **Clear the forced event subscription.**
{0610419a-6f33-481e-bd49-02796a0a9908}

Command
- `optionConfig.setOptionChange`

Event
(Time out: ${event.timeOut})
- `G2S_EHE005` [Event Handler Configuration Changed by Host]

Errors
- Event Checking Errors
- `E-XX-00002` [CVT timed out waiting for the response.]
- `E-XX-00003` [The wrong response was received. It MUST be ${expected}, but was ${actual}.]
- `E-XX-00005` [An unexpected error of ${errorCode} was received.]

Test Case: MT-COR-00007

This case verifies that the EGM rejects unknown commands.

Objective

Verify that the EGM rejects `meters.cvtTesting` commands.

Requirements Under Test

- 1.41.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Send a `cvtTesting` command to the EGM.**
   {01b62a1d-50ce-4fb9-bb41-f105e7c3b613}

   #### Command

   - `meters.cvtTesting`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00034` [G2S_APX015 MUST have a session ID of zero (0) and the session retry set to false.]

Test Case: MT-COR-00008

This case verifies that the EGM does not process invalid XML.

Objectives

- Verify that the EGM validates the `getMeterSub` command.
- Verify that the EGM validates the `commandId` attribute in the `meters` class element.

Requirements Under Test

- 1.41.2
- 1.41.5

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Send a `getMeterSub` command with an invalid `meterSubType` attribute.**
   {01b1fe63-59b7-47f1-a4aa-157cde5f1ba0}

   Command
   - `meters.getMeterSub`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors
   - Send Request Errors
   - `E-CM-00039` [The EGM must validate the G2S request.]

2. **Send a `getMeterSub` command with an invalid `commandId` attribute.**
   {028ecf7b-5fd1-4d28-9520-0f64d25378f7}

   Command
   - `meters.getMeterSub`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Errors

- Send Request Errors
- E-CM-00039 [The EGM must validate the G2S request.]

Test Case: MT-COR-00009

This case verifies that the EGM handles `meterInfo` command retries.

Objectives

- Verify that the EGM retries `meterInfo` commands.
- Verify that the EGM gathers new meter values when retrying `meterInfo` commands.

Requirements Under Test

- 1.28.8
- 5.4.11

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set an end-of-day meter subscription for one minute in the future.**
   {01dc558b-6509-44d5-b45d-d0ddafd23f2b}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Wait for the end-of-day `meterInfo` command, and determine time-to-live for the meter device and cabinet door open count.**
   {022fe957-e822-446f-b76c-22638da46d9c}

   Command
   - `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00005` [Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.]
- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

3. **Set the response manager to not respond to `meterInfo` commands.**
{030fd851-0143-405e-a8f4-3e0edee98847}

4. **Set an end-of-day meter subscription for one minute in the future.**
{045219e0-8d76-4a4b-a84f-4e6e53fd49dd}

Command

- `meters.setMeterSub`

Event

(Time out: ${event.timeOut})

- `G2S_MTE100` [Meter Subscription Set]

Errors

- Send Request Errors
- Event Checking Errors

5. **Wait for the end-of-day `meterInfo` command.**
{054f563e-a872-4d9b-9ad5-1e3ee289ce1b}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00005` [Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.]

6. **Wait for the `meterInfo` command to be retried.**
{063ee7d9-0564-43da-910c-13bc105abc46}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00005` [Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.]

7. **Instruct the user to open the cabinet door.**
{07aa04af-cc11-4b01-8131-f3e8913ea978}

   Action
   - Request CABINET door OPEN.

   Error
   - DUT Error

8. **Set the response manager to respond to `meterInfo` commands.**
{0822f4de-a69b-406d-a3a2-8ced54b80db5}

9. **Wait for the `meterInfo` command to be retried, and verify that the cabinet door open count was incremented.**
{0986c740-9dd7-4481-9a1b-db9dd5be8664}

   Command
   - `meters.meterInfo`

   Errors
   - Expected Request Errors
   - `E-MT-00005` [Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.]
   - `E-MT-00008` [Meter ${meterName} value of ${actual} is wrong it should be ${expected}.]

10. **Instruct the user to close the cabinet door.**
{102ec4eb-6f31-4c49-a4fb-99002866e49e}

    Action
    - Request CABINET door CLOSE.

    Error
    - DUT Error

Test Case: MT-COR-00010

This case verifies that the EGM supports the protocol-specified error codes.

Objective

Verify that the EGM returns an `G2S_MTX001` error for an unknown `meterSubType` attribute.

Requirements Under Test

- 1.44.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Attempt to set the `CVT_testing` meter subscription.**
   {01a445dc-3e48-4192-8583-ec7391050a97}

   Command
   - `meters.setMeterSub`

     Expect **G2S_MTX001**

   Errors
   - Send Request Errors
   - `E-MT-00003` [EGM MUST return a G2S_MTX001 Invalid meterSubType Specified error for an unknown meterSubType.]

2. **Attempt to clear the `CVT_testing` meter subscription.**
   {02add681-ea26-40ea-ba21-b90be1c8f5cd}

   Command
   - `meters.clearMeterSub`

     Expect **G2S_MTX001**

   Errors
   - Send Request Errors
   - `E-MT-00003` [EGM MUST return a G2S_MTX001 Invalid meterSubType Specified error for an unknown meterSubType.]

Test Case: MT-COR-00011

This case verifies meter device identifiers and ownership.

Objectives

- Verify that `meters` class device IDs match their owner host ID.
- Verify that `meters` class devices are associated with registered hosts.
- Verify that `meters` class devices are not owned by the EGM.

Requirements Under Test

- 5.1.1
- 5.1.2
- 5.1.3
- 5.1.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	guest or non-guest or owner
G2S_commConfig	guest or owner

Test Procedure

1. **Send a `getCommHostList` command to the EGM, and verify that the `meters` class devices have the correct ownership and permissions.**
   {01c857c0-e2ec-48f1-9aba-88b1c43d8b3d}

   #### Command

   - `commConfig.getCommHostList`

   #### Errors

   - Send Request Errors
   - `E-CC-00001` [Device identifier for host-oriented devices MUST be equal to the host identifier of the owner host.]
   - `E-CC-00002` [EGM MUST have device class ${deviceClass} for host ID ${hostId}.]
   - `E-CC-00003` [Host-oriented devices MUST not be owned by unregistered hosts.]
   - `E-CC-00004` [EGM MUST not own host-oriented devices.]

Test Case: MT-COR-00012

This case verifies that the EGM handles periodic meter subscriptions.

Objectives

- Verify that the EGM sends periodic meter subscriptions.
- Verify that the EGM gathers new meter values when sending `meterInfo` commands.

Requirements Under Test

- 5.3.1
- 5.3.2
- 5.3.3
- 5.3.10

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set a periodic meter subscription for 30 seconds in the past for every one minute.**
   {010c4fb6-3f76-45b6-a2c1-fd89df7f2d03}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Wait for a `meterInfo` command. Then, verify that the command is sent as a notification, and determine the cabinet door open count.**
   {0213ccde-f55c-4603-8375-b80e84877fa8}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00005` [Wrong number of devices meters are being reported in the meterInfo command. Expected ${expected} device(s), but received ${actual}.]
- `E-MT-00006` [Meter ${meterName} was expected for the ${device} device but was not sent in the meterInfo command.]

3. **Verify that a `meterInfo` command is sent within one minute +/- 5 seconds.**
{0313e7e0-97f0-438b-b244-f9575e958851}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00015` [Periodic meter subscription should have a frequency of ${expected}, but it was ${actual}.]

4. **Instruct the user to open the cabinet door.**
{046efedb-3187-4c85-8a98-62caf88f3301}

Action

- Request CABINET door OPEN.

Error

- DUT Error

5. **Verify that the cabinet door open count incremented on the next periodic `meterInfo` command.**
{058d0478-d6b0-4263-92ef-671caaec8b3b}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00008` [Meter ${meterName} value of ${actual} is wrong it should be ${expected}.]

6. **Set a meter subscription for the `cabinet` class-level meters for 30 seconds in the future for every one minute.**
{0697dd3f-72c1-462c-8bbb-9d6261577508}

Command

- `meters.setMeterSub`

Event

(Time out: ${event.timeOut})

- `G2S_MTE100` [Meter Subscription Set]

Errors

- Send Request Errors
- Event Checking Errors

7. **Verify that the `meterInfo` command has the `cabinet` class-level meters.**
{076cffb2-0a99-4b36-a801-021099036c62}

Command

- `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00008` [Meter ${meterName} value of ${actual} is wrong it should be ${expected}.]

8. **Instruct the user to close the cabinet door.**
{08971363-3794-4f82-af41-ff7fdfa47c01}

Action

- Request CABINET door CLOSE.

Error

- DUT Error

Test Case: MT-COR-00014

This case verifies `meterInfo` responses to `getMeterInfo` requests.

Objectives

- Verify that the EGM includes `meterType`, `meterRollover` and `meterIncreasing` attributes when meter definitions are requested.
- Verify that the EGM includes device meters for all active devices.
- Verify that the EGM supports wildcards in meter selection.
- Verify that game play denomination and wager category meters are consistent.

Requirements Under Test

- 5.9.6
- 5.9.7
- 5.11.1
- 5.12.1
- 5.12.2
- 5.12.3
- 5.16.1
- 5.16.2
- 5.16.6
- 5.16.7

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	guest or owner
G2S_communications	owner

Test Procedure

1. **Send a `getDescriptor` command to the EGM to determine the available devices.**
   {01791da8-6fe2-417c-8348-715af23571ba}

   Command
   - `communications.getDescriptor`

Error

- Send Request Errors

2. **Verify that class-level meters include meter definitions when requested.**
{0276cacb-840a-4ee1-a8da-f572048a749a}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-MT-00009` [Device ${device} was not expected in ${meterType} meter list.]
- `E-MT-00010` [Device ${device} was expected in ${meterType} meter list.]
- `E-MT-00011` [Meter ${meterName} in ${device} was expected to have meter definitions.]
- `E-MT-00012` [Meter ${meterName} in ${device} was not expected to have meter definitions.]
- `E-MT-00013` [Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.]
- `E-MT-00014` [Meter ${meterName} in ${device} sub values of ${subValue} for ${type} does not equal the device value of ${deviceValue}.]

3. **Verify that device-level meters include all active devices.**
{030c7343-2fbd-4234-b76f-77527b588ff4}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-MT-00009` [Device ${device} was not expected in ${meterType} meter list.]
- `E-MT-00010` [Device ${device} was expected in ${meterType} meter list.]
- `E-MT-00011` [Meter ${meterName} in ${device} was expected to have meter definitions.]
- `E-MT-00012` [Meter ${meterName} in ${device} was not expected to have meter definitions.]
- `E-MT-00013` [Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.]

4. **Verify that game denomination meters sum to the game play device meters.**
{04676b21-cfe1-4d42-a9d7-9da0dba843f3}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- E-MT-00009 [Device ${device} was not expected in ${meterType} meter list.]
- E-MT-00010 [Device ${device} was expected in ${meterType} meter list.]
- E-MT-00011 [Meter ${meterName} in ${device} was expected to have meter definitions.]
- E-MT-00012 [Meter ${meterName} in ${device} was not expected to have meter definitions.]
- E-MT-00013 [Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.]
- E-MT-00014 [Meter ${meterName} in ${device} sub values of ${subValue} for ${type} does not equal the device value of ${deviceValue}.]

5. **Verify that wager category meters sum to the game play device meters.**
{05cf9697-c76f-4f72-b574-db5da42e1ea3}

Command
- meters.getMeterInfo

Errors
- Send Request Errors
- E-MT-00009 [Device ${device} was not expected in ${meterType} meter list.]
- E-MT-00010 [Device ${device} was expected in ${meterType} meter list.]
- E-MT-00011 [Meter ${meterName} in ${device} was expected to have meter definitions.]
- E-MT-00012 [Meter ${meterName} in ${device} was not expected to have meter definitions.]
- E-MT-00013 [Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.]
- E-MT-00014 [Meter ${meterName} in ${device} sub values of ${subValue} for ${type} does not equal the device value of ${deviceValue}.]

6. **Verify that currency meters sum to the device meters.**
{065ccf39-d5b1-42dc-b430-d96985bfe62e}

Command
- meters.getMeterInfo

Errors
- Send Request Errors
- E-MT-00009 [Device ${device} was not expected in ${meterType} meter list.]
- E-MT-00010 [Device ${device} was expected in ${meterType} meter list.]
- E-MT-00011 [Meter ${meterName} in ${device} was expected to have meter definitions.]
- E-MT-00012 [Meter ${meterName} in ${device} was not expected to have meter definitions.]
- E-MT-00013 [Meter ${meterName} in ${device} has the wrong meter type of ${actual}, it should be ${expected}.]

- `E-MT-00014` [Meter ${meterName} in ${device} sub values of ${subValue} for ${type} does not equal the device value of ${deviceValue}.]

7. **Verify that the EGM supports redundant and contradictory meter selections.**
{07151bc3-00a7-4543-adce-da449e5f0bed}

Command

- `meters.getMeterInfo`

Error

- Send Request Errors

Test Case: MT-COR-00015

This case verifies that the host sends a valid getMeterInfo command.

Objective

Verify the getMeterInfo command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Instruct user to send getMeterInfo command from the host.**
   {01eb29d8-71d5-496c-8cc1-c8906dfcaf31}

   #### Command

   - `meters.getMeterInfo`

   #### Actions

   - Instruct the user to 'Send getMeterInfo to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-COR-00016

This case verifies that the host sends a valid setMeterSub command.

Objective

Verify the setMeterSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Instruct user to send setMeterSub command from the host.**
   {016c7d8f-b9b8-4d6a-aaf7-e034ef749a10}

   #### Command

   - `meters.setMeterSub`

   #### Actions

   - Instruct the user to 'Send setMeterSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-COR-00017

This case verifies that the host sends a valid getMeterSub command.

Objective

Verify the getMeterSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	guest or owner

Test Procedure

1. **Instruct user to send getMeterSub command from the host.**
   {01a90f83-f321-4f6a-84e2-e24b0795f57c}

   #### Command

   - `meters.getMeterSub`

   #### Actions

   - Instruct the user to 'Send getMeterSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-COR-00018

This case verifies that the host sends a valid clearMeterSub command.

Objective

Verify the clearMeterSub command from the host is valid.

Requirements Under Test

- 1.999.999

Test Type: COVERAGE

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Instruct user to send clearMeterSub command from the host.**
   {01af3343-b6bb-467c-abc9-a8c88ff4ba2d}

   #### Command

   - `meters.clearMeterSub`

   #### Actions

   - Instruct the user to 'Send clearMeterSub to device ${deviceUnderTest.deviceId}.'.

   #### Errors

   - Expected Request Errors
   - DUT Error

Test Case: MT-COR-00019

This case verifies that the host does not process meters commands with the wrong session type.

Objectives

- Verify that the host does not process a meterSubList sent as a request.
- Verify that the host does not process a meterInfo response with an invalid session ID.

Requirements Under Test

- 1.4.4
- 1.4.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send a meterSubList as a request and verify G2S_APX008 Command Not Supported error code.**
   {017ab210-c1af-4c61-8c10-20c4ea88f1e4}

   Command
   - `meters.meterSubList`

     Expect **G2S_APX008**

   Error
   - Send Request Errors

2. **Send a meterInfo as a response to the host with an invalid session ID.**
   {02043198-e84b-49e9-9457-7a5016e87aea}

   Command
   - `meters.meterInfo`

   Errors
   - Send Request Errors
   - `E-XX-00019` [An unexpected response of ${response} was received.]

Test Case: MT-COR-00020

This case verifies that the host handles errors responses to host meters requests.

Objectives

- Verify that the host is still working after responding with a meterInfo to a clearMeterSub command.
- Verify that the host is still working after responding with a meterSubList notification to a clearMeterSub command.

Requirements Under Test

- 1.4.5
- 1.4.7
- 5.14.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set the response manager to send a meterInfo response to a clearMeterSub command.**
   {01ea5cd7-aa87-4b9c-a887-97edf8683da5}
2. **Instruct user to send clearMeterSub command from the host.**
   {026197fa-8dc4-417c-b7f2-84f60f1ab57c}

   #### Command
   - `meters.clearMeterSub`

   #### Actions
   - Instruct the user to 'Send clearMeterSub to device ${deviceUnderTest.deviceId} (1 of 2).'.

   #### Errors
   - Expected Request Errors
   - DUT Error

3. **Set the response manager to send a meterSubList as a notification in response to a clearMeterSub command.**
   {03a75b0b-c685-437a-85b0-271287379d5c}

4. **Instruct user to send clearMeterSub command from the host.**
   {04b7d34c-92f6-48a6-abfb-7f47e78db101}

   Command
   - `meters.clearMeterSub`

   Actions
   - Instruct the user to 'Send clearMeterSub to device ${deviceUnderTest.deviceId} (2 of 2).'.

   Errors
   - Expected Request Errors
   - DUT Error

5. **Reset the response manager.**
   {0594baa2-0c33-4810-9dcd-04bc6fe61129}

6. **Send a meterInfo request to the host to verify it is still working.**
   {062a7239-40b9-4ac5-8733-3617126afa7a}

   Command
   - `meters.meterInfo`

   Error
   - Send Request Errors

Test Case: MT-COR-00021

This case verifies that the host sends the proper error code for an invalid meterInfo request.

Objective

Verify that the host sends G2S_MSX004 Incomplete/Malformed XML, G2S_MSX005 Invalid Data Type Encountered or G2S_APX004 Incomplete/Malformed XML error for an invalid meterInfo.

Requirements Under Test

- 1.27.33
- 1.41.2
- 1.41.6

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send an invalid meterInfo with a missing meterDateTime attribute and verify G2S_MSX004, G2S_MSX005 or G2S_APX004 error code.**
   {01c8032f-767c-40b8-a054-547426b00aef}

   Command
   - `meters.meterInfo`

     Expect **G2S_APX004**, **G2S_MSX004** or **G2S_MSX005**

   Error
   - Send Request Errors

Test Case: MT-COR-00022

This case tests that the host processes duplicate meterInfo commands.

Objective

Verify that the host processes duplicate meterInfo commands.

Requirements Under Test

- 1.28.9
- 5.14.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send five duplicate meterInfo(meterInfoType=CVT_testing) commands in one G2S message. Verify the host sends five meterInfoAcks.**
   {01c6a3b4-4f2d-4ab3-8a57-2d9969ec1a9e}

   Command
   - `meters.meterInfo`

   Error
   - Send Request Errors

Test Case: MT-COR-00023

This case tests that the host processes delayed g2sAcks in meters commands and unknown error codes in meter responses.

Objectives

- Verify that the host is still working after delaying the g2sAck for a clearMeterSub request.
- Verify that the host is still working after sending an unknown error code in response to a clearMeterSub request.

Requirements Under Test

- 1.30.7
- 1.42.3
- 5.14.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set the response manager to delay the g2sAck for clearMeterSub command.**
   {01a9ee25-b25c-4e40-8010-1bb7edb59a72}

2. **Instruct user to send clearMeterSub command from the host.**
   {02a9bef8-e173-4ee0-9b10-cd7432735748}

   #### Command

   - `meters.clearMeterSub`

   #### Actions

   - Instruct the user to 'Send clearMeterSub to device ${deviceUnderTest.deviceId} (1 of 2).'.

   #### Errors

   - Expected Request Errors
   - DUT Error

3. **Set the response manager to send 'CVT_ error' error code for clearMeterSub command.**
{030970ff-cbe4-4cfd-a623-153b81807d31}

4. **Instruct user to send clearMeterSub command from the host.**
{04538a85-9625-47c7-9c42-7cbd88b50e58}

   Command
   - `meters.clearMeterSub`

   Actions
   - Instruct the user to 'Send clearMeterSub to device ${deviceUnderTest.deviceId} (2 of 2).'.

   Errors
   - Expected Request Errors
   - DUT Error

5. **Reset the response manager.**
{0530f7f3-f9db-47dc-b7cb-430d749fbd33}

6. **Send a meterInfo to the host to verify that it is still working.**
{0615b4e0-0f27-4ddd-a993-12520c815e67}

   Command
   - `meters.meterInfo`

   Error
   - Send Request Errors

Test Case: MT-COR-00024

This case tests that the host handles a G2S_APX003 Invalid Device Identifier error code in response to a getMeterInfo command.

Objective

Verify that the host is still working after receiving a G2S_APX003 error code in response to a getMeterInfo command.

Requirements Under Test

- 5.7.4
- 5.14.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set the response manager to respond with a G2S_APX003 error for getMeterInfo command.**
   {018b64f6-d87f-4528-9a3e-7ef9082c7184}

2. **Instruct user to send getMeterInfo command from the host.**
   {025370fa-0a45-4997-94f4-c94788c20e35}

   #### Command
   - `meters.getMeterInfo`

   #### Actions
   - Instruct the user to 'Send getMeterInfo to device ${deviceUnderTest.deviceId}.'.

   #### Errors
   - Expected Request Errors
   - DUT Error

3. **Reset the response manager.**
   {03b66a86-ba58-4e7e-b56a-f83dfd933b33}

4. **Send a meterInfo to the host to verify that it is still working.**
{041b9a5a-409b-4bb4-8f2a-ebf52579904e}

Command

- `meters.meterInfo`

Error

- Send Request Errors

Test Case: MT-COR-00025

This case tests that the host sends the appropriate error for an unknown meter command.

Objectives

- Verify that the host sends G2S_APX008 Command Not Supported or G2S_AXP015 Unknown Command Encountered error for an unknown meters command.
- Verify that if the host sends G2S_AXP015 that the device is the communications device, sessionId is zero and sessionRetry is false.

Requirements Under Test

- 1.41.15

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: HOST
- **GSA Class**: meters

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Send a unknown meters command and verify G2S_APX008 or G2S_APX015 error.**
   {0108f9e7-2c56-4475-b5fa-9955b4ee7cae}

   #### Command

   - `meters.testing`

     Expect **G2S_APX008** or **G2S_APX015**

   #### Errors

   - Send Request Errors
   - `E-CM-00032` [The endpoint MUST return an error for commands from unknown classes.]

Test Case: MT-COR-00026

This case verifies that the EGM calculates the trigger point for periodic reports properly.

Objective

Verify EGM increments the trigger point when the trigger point is set in the past.

Requirements Under Test

- 5.3.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set a periodic meter subscription for one minute past midnight.**
   {013892a9-b896-4a1e-9483-d4dfc11925d1}

   ### Command
   - `meters.setMeterSub`

   ### Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   ### Errors
   - Send Request Errors
   - Event Checking Errors

2. **Verify that the meterInfo is sent as a notification within one minute +/- 5 seconds.**
   {025082a1-1e26-44e8-8ce4-86ac041b5841}

   ### Command
   - `meters.meterInfo`

   ### Error
   - Expected Request Errors

Test Case: MT-COR-00027

This case verifies that the EGM uses the `eodBase` when determining the trigger point for end-of-day reports.

Objective

Verify EGM uses the `eodBase` as the trigger point for end-of-day reports.

Requirements Under Test

- 5.4.2

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set an end-of-day meter subscription for one minute in the future.**
   {01d9bcf8-f643-4e80-8bfd-887d3129e25e}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Verify that the meterInfo is sent as a request within one minute +/- 5 seconds.**
   {0245465f-b048-4ee8-aeb8-97ada0476c52}

   Command
   - `meters.meterInfo`

   Error
   - Expected Request Errors

Test Case: MT-COR-00028

This case verifies that the EGM does not use device ID zero as a device identifier.

Objectives

- Verify EGM does not use device ID zero as a device identifier.
- Verify EGM reports class level meters with device ID zero.

Requirements Under Test

- 5.11.4

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send a getMeterInfo to device ID zero and verify error G2S_APX003 Invalid Device Identifier is sent.**
   {0135236e-cd44-4d23-901a-1dcc8d2a0a2d}

   Command
   - `meters.getMeterInfo`

     Expect **G2S_APX003**

   Errors
   - Send Request Errors
   - `E-XX-00015` [EGM MUST return G2S_APX003 application error for an invalid device ID of 0 (zero).]

2. **Verify the EGM send meters for device ID zero.**
   {023be1e7-b615-4c54-be1e-4c50dd747705}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

Test Case: MT-COR-00029

This case verifies that the EGM sends the prescribed Meters response to each Meters request.

Objectives

- Verify that the EGM sends a `meterInfo` response to a `getMeterInfo` request.
- Verify that the EGM sends a `meterSubList` response to a `setMeterSub` request.
- Verify that the EGM sends a `meterSubList` response to a `getMeterSub` request.
- Verify that the EGM sends a `meterSubList` response to a `clearMeterSub` request.

Requirements Under Test

- 5.14.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send a getMeterInfo request to the EGM.**
   {011f0838-3c30-421c-ba74-abea7f5363ef}

   Command
   - `meters.getMeterInfo`

   Error
   - Send Request Errors

2. **Send a setMeterSub request to the EGM.**
   {0276b5a8-1fe3-4141-bb53-be501d2ddded}

   Command
   - `meters.setMeterSub`

   Error
   - Send Request Errors

3. **Send a getMeterSub request to the EGM.**
{0326f950-5ecf-4c61-9655-802f658cc05a}

Command
- `meters.getMeterSub`

Error
- Send Request Errors

4. **Send a clearMeterSub request to the EGM.**
{0443c347-dab7-4586-946e-d68f39086df6}

Command
- `meters.clearMeterSub`

Error
- Send Request Errors

Test Case: MT-COR-00030

This case verifies that the EGM expands wildcard meter subscriptions.

Objective

Verify that the EGM expands wildcard meter subscriptions when responding with a `meterSubList`.

Requirements Under Test

- 5.19.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set a wildcard end-of-day meter subscription and verify that there are no wildcards in the meterSubList.**
   {01f8cd30-14e5-44bb-89e4-1bdb304d1169}

   #### Command

   - `meters.setMeterSub`

   #### Event

   (Time out: ${event.timeOut})

   - `G2S_MTE100` [Meter Subscription Set]

   #### Errors

   - Send Request Errors
   - Event Checking Errors
   - `E-MT-00017` [Wilcard meter subscriptions must be expanded in the meterSubList.]

Test Case: MT-COR-00031

This case verifies that the EGM reports meters for devices that are owned by a G2S host.

Objective

Verify EGM reports meters that are available to a host.

Requirements Under Test

- 5.18.1

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Get the descriptor list to determine which devices are owned by a g2s host.**
   {017c87ed-e200-4afd-84f9-95d83400c927}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Get the device meters and verify expected device meters are available.**
   {029f4b10-05b0-4b0e-86c1-aaed3316c1ca}

   Command
   - `meters.getMeterInfo`

   Errors
   - Send Request Errors
   - `E-MT-00010` [Device ${device} was expected in ${meterType} meter list.]

Test Case: MT-COR-00032

This case verifies that the EGM retrieves a meter subscription when the get meter subscription command has additional unknown attributes.

Objective

Verify EGM reports send a `meterSubList` when the `getMeterSub` contains additional unknown attributes.

Requirements Under Test

- 1.27.33

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send a getMeterSub command with an additional attributes in the 'cvt:Testing' namespace.**
   {01e5c08f-6543-406d-8793-6013f5342001}

   Command
   - `meters.getMeterSub`

   Errors
   - Send Request Errors
   - `E-MT-00018` [Wrong meter subscription type of ${actual} was sent in the meterSubList, it should be ${expected}.]

Test Case: MT-COR-00033

This case verifies that the EGM processes duplicate setMeterSub commands.

Objective

Verify the EGM processes duplicate setMeterSub commands.

Requirements Under Test

- 1.28.9

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Send five setMeterSub commands in one G2S message. Verify the EGM sends five meterSubList responses.**
   {015b6919-2eb2-416d-ba20-199564485d7a}

   Command
   - `meters.setMeterSub`

   Error
   - Send Request Errors

Test Case: MT-COR-00034

This case verifies that the EGM handles unknown error responses to `meterInfo` requests.

Objective

Verify the EGM resends meterInfo when it receives a CVT_ERROR application level error code.

Requirements Under Test

- 1.42.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set an end-of-day meter subscription for one minute in the future.**
   {015c4eed-ac39-4416-b30b-99587bcc4840}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set the response manager to respond to meterInfo commands with a CVT_ERROR error code.**
   {025fbda8-3287-49ed-9238-00506a73cf2f}

3. **Wait for the end-of-day meter subscription and send a CVT_ERROR application error code.**
   {0311caf5-2013-4d8a-b95e-b1cdba8cafc6}

   Command

- `meters.meterInfo`

Error

- Expected Request Errors

4. **Clear the response manager and wait for the end-of-day meter subscription to be retried.**
{0443ebb5-ec02-4b68-a6fa-b3c036ba04c4}

Command

- `meters.meterInfo`

Error

- Expected Request Errors

Test Case: MT-COR-00035

This case verifies that the EGM persists meter subscriptions.

Objectives

- Verify that the EGM has the same `periodicInterval` for periodic meter subscription after a restart.
- Verify that the EGM has the same `periodicBase` for end-of-day meter subscription after a restart.

Requirements Under Test

- 5.3.5
- 5.3.6
- 5.3.9
- 5.6.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Set a periodic meter subscription for five minutes apart.**
   {01a4a789-e5bf-4ed3-9e94-f09f64e7378a}

   #### Command
   - `meters.setMeterSub`

   #### Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   #### Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set an end-of-day meter subscription for three minutes past midnight.**
{02fc2754-7cb1-464a-8a13-d53da74d9d23}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

3. **Instruct the user to power off the EGM.**
{0316bc8c-740c-4d8d-b752-e80a3e9727f9}

   Action
   - Instruct the user to 'Please power off the EGM. Press 'TRUE' when the EGM has been powered off.'.

   Errors
   - DUT Error
   - `E-CM-00057` [The user failed to power cycle the EGM.]

4. **Wait for one periodic meter subscription period to pass.**
{04962cd5-ab0e-4646-a60f-a88b3042b15a}

5. **Instruct the user to power on the EGM and wait for the EGM to reconnect to the host.**
{05004ceb-758e-40c8-b18b-90801174785d}

   Commands
   - `communications.commsOnLine`
   - `communications.commsDisabled`

   Actions
   - Instruct the user to 'Please power on the EGM.'.

   Errors
   - Expected Request Errors
   - DUT Error

6. **Verify that the EGM has persisted the periodic subscription.**
{0606da99-d2c3-4654-906c-8fca417a620b}

   Command

- meters.getMeterSub

Errors

- Send Request Errors
- E-MT-00019 [Meter subscription ${meterSubType} MUST be persisted, the ${attribute} value of ${actual} is wrong, is should be ${expected}.]

7. **Verify that the EGM has persisted the end-of-day subscription.**
{072ad087-f179-4669-a3e5-1a7d02c72b62}

Command

- meters.getMeterSub

Errors

- Send Request Errors
- E-MT-00019 [Meter subscription ${meterSubType} MUST be persisted, the ${attribute} value of ${actual} is wrong, is should be ${expected}.]

8. **Enable communications and verify a meterInfo is sent within one minute.**
{08b7618c-827a-4287-8feb-c932af677e92}

Commands

- communications.setCommsState
- meters.meterInfo

Errors

- Expected Request Errors
- Send Request Errors

9. **Wait for the next periodic subscription and verify that the meterInfo is within one minute of periodicInterval.**
{09a131d8-136a-4ecb-8210-19e7b437d126}

Command

- meters.meterInfo

Errors

- Expected Request Errors
- E-MT-00020 [Periodic meter subscription date/time of ${meterDateTime} MUST be sent at intervals of ${periodicInterval} from an offset of ${periodicBase} seconds from midnight.]

Test Case: MT-COR-00036

This case verifies that the EGM sends an empty `meterInfo` list for `getMeterInfo` commands that specify an unsupported device class or invalid device.

Objectives

- Verify that the EGM send an empty `meterInfo` list for an unsupported device class.
- Verify that the EGM send an empty `meterInfo` list for an invalid device.

Requirements Under Test

- 5.7.3

Test Type: QUICK

Criteria

- **Protocol**: G2S 1.1+
- **GSA Class**: meters
- **Endpoint**: EGM
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Get the descriptor list to determine an invalid device ID.**
   {01039242-ec0e-414e-acae-3779d964d71a}

   Command
   - `communications.getDescriptor`

   Error
   - Send Request Errors

2. **Verify that the EGM responds with an empty meterInfo list for an unknown class.**
   {0211c310-e864-4c71-b143-13ca55c39135}

   Command
   - `meters.getMeterInfo`

   Errors
   - Send Request Errors
   - `E-MT-00021` [Meter info list for unknown device class or invalid device MUST be empty.]

3. **Verify that the EGM responds with an empty meterInfo list for an invalid device.**
{03ddc728-dd3b-475f-acd8-efc8d5030bdb}

Command

- `meters.getMeterInfo`

Errors

- Send Request Errors
- `E-MT-00021` [Meter info list for unknown device class or invalid device MUST be empty.]

Test Case: MT-COR-00037

This case verifies that the EGM sends end-of-day meters after becoming online if the EGM was offline during the EOD trigger point.

Objective

Verify that the EGM sends end-of-day meters after it becomes online if the EGM was in a comms outage during an EOD trigger point.

Requirements Under Test

- 5.6.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Set a end-of-day meter subscription for three minutes in the future.**
   {015c9766-5389-4348-990d-90f589de3799}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set the response manager to respond with G2S_MSX003 error to commsProfile and commsOnLine commands.**
   {02a7e1bd-6fdb-42cb-b055-4f33d5f29f3e}

3. **Send a getCommsProfile command and respond with a G2S_MSX003 to force the EGM to reconnect.**

{03debe43-6f53-4fdc-8b18-f2c4bd227765}

Command

- `communications.getCommsProfile`

Error

- Send Request Errors

4. **Verify that the EGM is in the opening state.**
{040be4f8-9e2f-4d12-a346-5294dd2cd278}

Command

- `communications.commsOnLine`

Error

- Expected Request Errors

5. **Wait for end-of-day trigger to pass.**
{057151e0-ba6d-425e-8e52-900ef69dacd8}

6. **Reset the response manager.**
{06a51c51-e208-47de-83e0-cd20b0c26745}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Error

- Expected Request Errors

7. **Enable communications and verify a meterInfo is sent within one minute.**
{07eb0369-740a-4b23-b418-fc8063c18b2e}

Commands

- `communications.setCommsState`
- `meters.meterInfo`

Errors

- Expected Request Errors
- Send Request Errors

Test Case: MT-COR-00038

This case verifies that the end-of-day meter snapshot is sent to the host if the EGM was powered down during the end-of-day trigger.

Objective

Verify that the EGM sends end-of-day meters after it becomes online if the EGM was in a power outage during an EOD trigger point.

Requirements Under Test

- 5.7.1

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner
G2S_communications	owner

Test Procedure

1. **Set an end-of-day meter subscription for five minutes in the future.**
   {0150af2f-4440-4f9a-9fe6-0cfab8e54a33}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

2. **Instruct the user to power off the EGM.**
   {023acf31-da68-4988-8664-2065c957fba2}

   Action

- Instruct the user to 'Please power off the EGM. Press 'TRUE' when the EGM has been powered off.'.

Errors

- DUT Error
- `E-CM-00057` [The user failed to power cycle the EGM.]

3. **Wait for one periodic meter subscription period to pass.**
{03cf6319-4651-4f7d-9480-95c523ca99e8}

4. **Instruct the user to power on the EGM and wait for the EGM to reconnect to the host.**
{04013ad8-3d54-4f6e-8029-ad0ea8cec262}

Commands

- `communications.commsOnLine`
- `communications.commsDisabled`

Actions

- Instruct the user to 'Please power on the EGM.'.

Errors

- Expected Request Errors
- DUT Error

5. **Enable communications and verify a meterInfo is sent within one minute.**
{057cfd81-d980-45d1-bf00-c8fb9d794625}

Commands

- `communications.setCommsState`
- `meters.meterInfo`

Errors

- Expected Request Errors
- Send Request Errors

Test Case: MT-COR-00039

This case verifies that the EGM retries end-of-day meters until the next EOD trigger point.

Objective

Verify that the EGM retries end-of-day meters until the next EOD trigger point.

Requirements Under Test

- 5.17.2

Test Type: SUFFICIENT

Criteria

- **Protocol**: G2S 1.1+
- **Endpoint**: EGM
- **GSA Class**: meters
- **Required Hosts**: 1+

Required Devices

Device	Permissions
G2S_meters	owner

Test Procedure

1. **Set a end-of-day meter subscription for one minutes in the future.**
   {013fd6fd-9c7d-4d6c-80ce-0520ba9c638a}

   ### Command
   - `meters.setMeterSub`

   ### Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   ### Errors
   - Send Request Errors
   - Event Checking Errors

2. **Set the response manager to respond with G2S_MSX003 error to commsProfile and commsOnLine commands.**
   {0201c7a5-ae4f-4aae-b3e7-603debbbd560}

3. **Verify that the EGM sends a meterInfo at the end-of-day trigger point.**
   {03a85f91-0dd6-47e6-92ed-479c384a04ab}

   ### Command
   - `meters.meterInfo`

Errors

- Expected Request Errors
- `E-MT-00022` [EGM MUST send end-of-day meters after the eodBase trigger point of ${eodBase}.]

4. **Verify a meterInfo is retried within one minute.**
{0422b203-3d34-4707-a60b-2dd6ce42573a}

   Command
   - `meters.meterInfo`

   Error
   - Expected Request Errors

5. **Verify a meterInfo is retried within one minute.**
{0594c26a-b0ff-4e4e-87bf-4f745b815000}

   Command
   - `meters.meterInfo`

   Error
   - Expected Request Errors

6. **Set a new end-of-day trigger for one minute in the future.**
{063fa69f-39bf-423c-bb7d-948fc7b3dc4c}

   Command
   - `meters.setMeterSub`

   Event
   (Time out: ${event.timeOut})
   - `G2S_MTE100` [Meter Subscription Set]

   Errors
   - Send Request Errors
   - Event Checking Errors

7. **While the EGM has not sent new end-of-day meters values.**

   1. **Wait for the EGM to send a meterInfo command.**
   {07fc69cd-f80d-4c95-8053-979789d6df45}

      Command
      - `meters.meterInfo`

      Errors

- Expected Request Errors
- E-MT-00022 [EGM MUST send end-of-day meters after the eodBase trigger point of ${eodBase}.]

8. **Verify that a new end-of-day meter values were sent to the host.**
{08398fe7-4c4b-4fad-8647-2b38b69044f9}

Error

- E-MT-00022 [EGM MUST send end-of-day meters after the eodBase trigger point of ${eodBase}.]