Car Connectivity Consortium

MirrorLink®

Common API Test Specification

Version 1.1.4
(CCC-TS-039)

VERSION HISTORY

Version	Date	Comment
1.1.0	18 March 2014	Approved Version
1.1.1	14 May 2014	Approved Errata Version
1.1.2	22 May 2014	Approved Errata Version
1.1.3	29 October 2014	Approved Errata Version
1.1.4	27 August 2015	Approved Errata Version

LIST OF CONTRIBUTORS

Pichon, Ed (Editor) E-Qualus Partners, LLC
Brakensiek, Jörg (Editor) Car Connectivity Consortium LLC

TRADEMARKS

MirrorLink is a registered trademark of Car Connectivity Consortium LLC
Bluetooth is a registered trademark of Bluetooth SIG Inc.
RFB and VNC are registered trademarks of RealVNC Ltd.
UPnP is a registered trademark of UPnP Implementers Corporation.
Other names or abbreviations used in this document may be trademarks of their respective owners.

LEGAL NOTICE

The copyright in this Specification is owned by the Car Connectivity Consortium LLC ("CCC LLC"). Use of this Specification and any related intellectual property (collectively, the "Specification"), is governed by these license terms and the CCC LLC Limited Liability Company Agreement (the "Agreement").

Use of the Specification by anyone who is not a member of CCC LLC (each such person or party, a "Member") is prohibited. The legal rights and obligations of each Member are governed by the Agreement and their applicable Membership Agreement, including without limitation those contained in Article 10 of the LLC Agreement.

CCC LLC hereby grants each Member a right to use and to make verbatim copies of the Specification for the purposes of implementing the technologies specified in the Specification to their products ("Implementing Products") under the terms of the Agreement (the "Purpose"). Members are not permitted to make available or distribute this Specification or any copies thereof to non-Members other than to their Affiliates (as defined in the Agreement) and subcontractors but only to the extent that such Affiliates and subcontractors have a need to know for carrying out the Purpose and provided that such Affiliates and subcontractors accept confidentiality obligations similar to those contained in the Agreement. Each Member shall be responsible for the observance and proper performance by such of its Affiliates and subcontractors of the terms and conditions of this Legal Notice and the Agreement. No other license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of this Legal Notice, the Agreement and Membership Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement and other liability permitted by the applicable Agreement or by applicable law to CCC LLC or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS, AND COMPLIANCE WITH APPLICABLE LAWS.

Each Member hereby acknowledges that its Implementing Products may be subject to various regulatory controls under the laws and regulations of various jurisdictions worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Implementing Products. Examples of such laws and regulatory controls include, but are not limited to, road safety regulations, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Implementing Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Implementing Products related to such regulations within the applicable jurisdictions.

Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses.

NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS. ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST CCC LLC AND ITS MEMBERS RELATED TO USE OF THE SPECIFICATION.

CCC LLC reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

TABLE OF CONTENTS

TERMS AND ABBREVIATIONS

ACMS	Application Certification Management System
BT	Bluetooth
ML	MirrorLink
OCSP	Online Certificate Status Protocol
RFB	Remote Framebuffer
UPnP	Universal Plug and Play
USB	Universal Serial Bus
VNC	Virtual Network Computing

1 ABOUT

This document specifies all MirrorLink conformance test cases for the MirrorLink Common API[2]. MirrorLink Server devices are required to comply with the platform-specific Common API specification corresponding the mobile device platform of the device. A platform-specific test app will be made available to the device developers and testers for use in performing these tests.

There is no platform-specific Common API test specification, as this specification will be applicable to all device platforms.

These test cases only apply to MirrorLink servers. MirrorLink clients have no Common-API related test cases.

The specification lists a series of requirements, either explicitly or within the text, which are mandatory elements for a compliant solutions. Recommendations are given, to ensure optimal usage and to provide suitable performance. All recommendations are optional.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are following the notation as described in RFC 2119 [1].

1. MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
2. MUST NOT: This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
3. SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
4. SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
5. MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

2 DEFINITIONS

2.1 Execution of Test Cases

Every test case is uniquely identified by an identifier.

- A MirrorLink server MUST pass all test cases, starting with SR.
- A MirrorLink client MUST pass all test cases, starting with CL. Note: There are no client tests in this test specification.

Every test case description includes an entry, whether the test cases is considered mandatory or not.

- Test cases marked as MANDATORY, MUST be executed.
- Test cases marked as CONDITIONAL, MUST be executed if the given condition is met.
- Test cases marked as CONDITIONAL, MUST NOT be executed if the given condition is not met.
- Test cases marked as NONE, MUST NOT be executed.
- Note: All test cases in this specification are marked as Mandatory.

2.2 Optional Common API Methods

A number of methods defined in the Common API specification are listed as optional. On some platforms, those optional methods may need to be implemented to return defined default values. On other platforms, inherent OS mechanisms allow the applications to determine what capabilities are available, and do not require the implementation "dummy" methods. As such, for some platforms the entries in the PICS corresponding to the Common API methods do not impact which tests are run, but instead impact what the expected results of those tests are.

2.3 Test Application Description

A test application is required to perform the tests in this specification. The requirements for the test application can be found in the references [6]. The test application may be provided by a third party, or developed by the device manufacturer. If the test application code is modified from a third party application, a statement that the fundamental API-interactions of the application has not been modified must be provided to the Certification Body.

2.4 Server Definitions

The following definitions are frequently used in different server test cases. Usage is indicated by the given designator name.

2.4.1 Test App Launch

Launch the Test Application on the MirrorLink Server. Note: This is basically the VNC Server Launch step, with two modifications:

- Launch the Test App instead of any application in the last step.
- Perform an initial check on the status of the Test App. If it is running, and is in the foreground, the rest of the Test App Launch steps can be skipped.

Step	Name	Description	Expected Result
1	Check Test Application Running	If a UPnP connection is available, check if the Test Application is running on the DUT and is in the foreground.	• If UPnP connection is not available, skip to step 3. • If test application is not running, skip to step 3.
2	Check Test Application Usable	Check if the test app is usable.	• If test application is in the foreground and unblocked, skip remaining steps.

3	UPnP Connect	Preparing the UPnP connection by making an initialization, registering the client and waiting for the device to announce itself.	• Device announce itself in time
4	UPnP Device Description	Test the device description for parseable XML formatting and availability of service types and their control and event URLs.	• the device description can be parsed • it indicates support for TmApplicationServer:1 service • it indicates support for TmClientProfile:1 service
5	UPnP Action GetMaxNum Profiles	Received the number of supported client profiles	• UPnP action returns successfully with maxNumProfiles
6	UPnP Action SetClientProfile	Set client profile id to a number out of [0; maxNumProfiles-1] This test step MUST be skipped, if the maxNumProfiles is 1.[1]	• UPnP action returns successfully with resulted client profile
7	UPnP Action Launch Test Application	Launch the Test Application on the MirrorLink server device.	• UPnP action returns successfully with valid URL

1 Table 1: Test App Launch – Test Steps

2

[1] Set client Profile is tested separately.

3 SERVER FEATURE TEST CASES

3.1 Common API Info

3.1.1 SR/API/API/Version

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Common_API_Info

This test examines the ML Common API Version Get methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: This test can be run without the CTS being involved.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Common API Version	Test engineer invokes the Get ML Common API Version method using the Test App.	• ML Common API level displayed by the Test App matches that reported by the DUT manufacturer in the PIXIT (16000).

Table 2: ML Common API Info – Test Steps

3.1.2 SR/API/API/ModuleAvailability

Requirement: MANDATORY

Condition: Platform Common API Implementation Supports Capability (Android)

Features Tested: FEAT_SERVER_API_Common_API_Info

This test examines the Common API Module Available method for all known modules to check that the support matches that reported by the device manufacturer in the PICS.

Note: This test can be run without the CTS being involved.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Get Module Availability	Test engineer invokes the Get API Module Availability method using the Test App. The Test App invokes the method for each module reference: • Common API Info 0x00 • Device Info 0x01 • Certification Info 0x02 • Connection Info 0x03 • Display Info 0x04 • Event Info 0x05 • Virtual Keyboard 0x06 • Key Event Listing 0x07 • Context Info 0x08	• The supported status matches that reported in the server device's PICS.

Step	Name	Description	Expected Result
		• Device Status Info 0x09 • Data Services 0x0A • Notifications 0x0B • Web Applications 0x0C	

Table 3: Common API Module Availability – Test Steps

3.2 MirrorLink Device Info

3.2.1 SR/API/DI/MLVersion

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Device_Info

This test examines the ML Version Get methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: Checking availability of the ML Version on connection setup is tested in the connection status tests.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Set CTS ML Version	Set the CTS to advertise it's ML version as 1.1	
3	Execute Get ML Server Version	Test engineer invokes the Get ML Version Info method using the Test App.	• ML Version major and minor version displayed by the Test App matches that reported by the DUT manufacturer in the PIXIT (16010), or 1.1 (whichever is lower).
4	Set CTS ML Version	Set the CTS to advertise its ML version as 1.0.	
5	Execute Get ML Server Version	Test engineer invokes the Get ML Version Info method using the Test App.	• ML Version major and minor version displayed by the Test App is 1.0.

Table 4: MirrorLink Version Info – Test Steps

3.2.2 SR/API/DI/ClientInfo

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Device_Info

This test examines the ML Client Version, ML Client Manufacturer and Model Information Get methods to ensure that the Common API implementation responds to those methods, and provides the correct information. This test also tests that the ML Client Manufacturer and Model Information Callback method works properly.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	

Step	Name	Description	Expected Result
2	Execute Get ML Client Manufacturer and Model Information	Test engineer invokes the Get ML Client Manufacturer and Model Information method using the Test App.	• The Client Identifier, Friendly Name, Manufacturer, Model Name and Model Number as displayed by the Test App correspond to the values provided by the CTS. • The Success Boolean flag as displayed by the Test App indicates a value of TRUE.
3	Update Client Profile	Re-set client profile id to the original value, with updated entries in the Client Profile – append "-2" to the Client Identifier, Friendly Name, Manufacture, Model Name and Model Number values.	• The ML Client Manufacturer and Model Information Callback indicator indicates TRUE.
4	Execute Get ML Client Manufacturer and Model Information	Test engineer invokes the Get ML Client Manufacturer and Model Information method using the Test App.	• The Client Identifier, Friendly Name, Manufacturer, Model Name and Model Number as displayed by the Test App correspond to the values provided to the CTS in the ResultProfile. • The Success Boolean flag as displayed by the Test App indicates a value of TRUE.

Table 5: Client Info – Test Steps

3.2.3 SR/API/DI/ServerKeyboard

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Device_Info

This test examines the Server Device Virtual Keyboard Support Get methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: Checking availability of the ML Version on connection setup is tested in the connection status tests.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Server Device Virtual Keyboard Version	Test engineer invokes the Get Server Device Virtual Keyboard Info method using the Test App.	• The Available, Touch Support, Knob Support and Drive Mode indicators match as reported in the DUT's PICS.

Table 6: Server Device Virtual Keyboard Support – Test Steps

3.3 Certificate Information

3.3.1 SR/API/CI/CertificateInfo

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Certificate_Info

This test examines the Application Certification Status, Application Certifying Entities and Application Certification Information Get methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: The CTS does not have to be involved in this test.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Application Certification Status	Test engineer invokes the Get Application Certification Status method using the Test App.	• The Certificate Available and Advertised as Certified App flag indicators displayed by the Test App display “TRUE”.
3	Execute Get Application Certifying Entities	Test engineer invokes the Get Application Certifying Entities method using the Test App.	• Entity list as displayed by the Test App matches the list provided in the Application Certificate.
4	Execute Get Application Certification Information	Test engineer selects the first of the entities provided in Step 3, and invokes the Get Application Certification Information method for that certifying entity using the Test App.	• The values displayed by the Test App for the certified flag, and restricted and non-restricted strings correspond to the values provided in the application certificate.
5	Repeat Execute Get Application Certification Information	Test engineer repeats Step 4 for each of the certifying entities listed in Step 3.	As step 4.

Table 7: Certificate Info – Test Steps

3.4 Connection Information

3.4.1 SR/API/CI/MLConnectionsSetClientProfile

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Connection_Info

This test examines the Established ML Connection Get method and the Established ML Connection Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Disconnect	Disconnect the DUT from the CTS.	
2	Launch Test Application	Test engineer launches the Test Application on the DUT	• Test application in foreground on DUT • Test application registered to MirrorLink services
3	Check ML Connection status	Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of FALSE.

Step	Name	Description	Expected Result
4	UPnP Connect	Preparing the UPnP connection by making an initialization, registering the client and waiting for the device to announce itself.	• Device announce itself in time
5	UPnP Device Description	Test the device description for parseable XML formatting and availability of service types and their control and event URLs.	• Device description can be parsed • Indicates support for TmApplicationServer:1 service • Indicates support for TmClientProfile:1 service
6	UPnP Set Client Profile	Send UPnP `SetClientProfile` action for profile 0.	• UPnP action returns successfully with resulted client profile
7	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
8	Check ML connection status	Test engineer clears ML connection callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of TRUE.
9	Disconnect	Disconnect the DUT from the CTS.	
10	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
11	Check ML Connection status	Test engineer clears ML connection callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of FALSE.
12	Reconnect	Reconnect the DUT to the CTS (steps 4-6).	
13	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
14	Check ML connection status	Test engineer clears callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of TRUE.

Table 8: MirrorLink Connections – Set Client Profile

3.4.2 SR/API/CI/MLConnectionsGetApplicationList

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Connection_Info

This test examines the Established ML Connection Get method and the Established ML Connection Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Disconnect	Disconnect the DUT from the CTS.	
2	Launch Test Application	Test engineer launches the Test Application on the DUT	• Test application in foreground on DUT • Test application registered to MirrorLink services
3	Check ML Connection status	Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of FALSE.
4	UPnP Connect	Preparing the UPnP connection by making an initialization, registering the client and waiting for the device to announce itself.	• Device announce itself in time
5	UPnP Device Description	Test the device description for parseable XML formatting and availability of service types and their control and event URLs.	• Device description can be parsed • Indicates support for TmApplicationServer:1 service • Indicates support for TmClientProfile:1 service
6	UPnP Get Application List	Send UPnP `GetApplicationList` action for profile 0.	• UPnP action returns successfully with resulted app list
7	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
8	Check ML connection status	Test engineer clears ML connection callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of TRUE.
9	Disconnect	Disconnect the DUT from the CTS.	
10	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
11	Check ML Connection status	Test engineer clears ML connection callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of FALSE.
12	Reconnect	Reconnect the DUT to the CTS (steps 4-6).	
13	Check ML connection callback	Test engineer checks the ML Connection callback indicator	• Callback Fired indicates that callback was triggered
14	Check ML connection status	Test engineer clears callback indicator. Test engineer invokes the Get Established ML Connections method using the Test App.	• The test app displays a connection status of TRUE.

Table 9: MirrorLink Connections – Get Application List

3.4.3 SR/API/CI/AudioConnections

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Connection_Info

This test examines the Established Audio Connections Get method and the Established Audio Connections Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Established Audio Connections	Test engineer invokes the Get Established Audio Connections method using the Test App.	• The Media Audio Out, Media Audio In, Voice Control, Phone Audio and RTP Payload types as displayed by the Test App correspond to the active connections as known by the CTS.
3	Establish New Audio Connection	The Test Engineer creates a new audio connection as described in the DUT's PIXIT (16020). Possible methods include: • Placing a phone call. • Invoking voice control • Start the music player *OR* CTS opens a new audio connection.	• The Callback Fired indicator in the test app indicates that the callback was triggered.
4	Execute Get Established Audio Connections	Test engineer invokes the Get Established Audio Connections method using the Test App.	• The Media Audio Out, Media Audio In, Voice Control, Phone Audio and RTP Payload types as displayed by the Test App correspond to the active connections as known by the CTS.

Table 10: Audio Connections – Test Steps

3.4.4 SR/API/CI/RemoteDisplayConnection

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Connection_Info

This test examines the Established Display Remote Connection Get method and the Established Remote Display Connection Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Established Remote Display Connection	Test engineer invokes the Get Established Remote Display Connection method using the Test App.	• The connection identifier as displayed by the Test App indicates that VNC is the remote display type.
3	Intentional VNC Server Cleanup	See Definitions in [7]. Note: Other steps may be sufficient, such as physically disconnecting the DUT from the CTS.	• VNC session ended with the ML Client (CTS) no longer replicating the display on the DUT.

Step	Name	Description	Expected Result
4	Examine Test App	The Test Engineer brings the Test App to the foreground (if necessary).	• The Callback Fired indicator in the test app indicates that the callback was triggered.
5	Execute Get Established Remote Display Connection	Test engineer invokes the Get Established Remote Display Connection method using the Test App.	• The connection identifier as displayed by the Test App indicates that "No connection established" is the remote display type.

Table 11: Remote Display Connection – Test Steps

3.5 Display Related Features

3.5.1 SR/API/DRF/ClientDisplay

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Display_Info

This test examines the Client Display Configuration and Client Pixel Format Get methods, and the Client Display Configuration and Client Pixel Format Callbacks to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Client Display Configuration	Test engineer invokes the Get Client Display Configuration method using the Test App.	• The Horizontal Resolution, Vertical Resolution, Width, Height and Distance values as displayed by the Test App correspond to the values provided to the DUT by the CTS in the VNC Client Display Configuration message. • The Success flag as displayed by the Test App indicates "TRUE".
3	Modify Client Display Configuration	CTS sends a new VNC Client Display Configuration message with modified values (each value is increased by a value of 10).	• The Callback Fired indicator in the test app indicates that the callback was triggered.
4	Execute Get Client Display Configuration	Test engineer invokes the Get Client Display Configuration method using the Test App.	• The Client Horizontal Resolution, Client Vertical Resolution, Width, Height and Distance values as displayed by the Test App correspond to the values provided to the DUT by the CTS in the VNC Client Display Configuration message. • The Server Vertical Resolution, Server Horizontal Resolution, Server Pad Rows and Server Pad Columns match those displayed on the screen.

Step	Name	Description	Expected Result
			• The App Horizontal Resolution and App Vertical Resolution match those displayed on the screen. • The Success flag as displayed by the Test App indicates "TRUE".
5	Execute Get Client Pixel Format	Test engineer invokes the Get Client Pixel Format method using the Test App.	• The Pixel Format value as displayed by the Test App corresponds to the pixel format of the framebuffer data being sent to the CTS. • The Success flag as displayed by the Test App indicates "TRUE".
6	Set a New Pixel Format	CTS sends a new Set Pixel Format to a new format as supported by the DUT. Note: As per the VNC Specification, the CTS should send one Framebuffer Update Request message at a time, and only send the new Set Pixel Format after a Framebuffer Update has been received and before the next Framebuffer Update Request message is sent.	• The Callback Fired indicator in the test app indicates that the callback was triggered.
7	Execute Get Client Pixel Format	Test engineer invokes the Get Client Pixel Format method using the Test App.	• The Pixel Format value as displayed by the Test App corresponds to the pixel format of the framebuffer data being sent to the CTS. • The Success flag as displayed by the Test App indicates "TRUE".

Table 12: Client Display – Test Steps

3.5.2 SR/API/DRF/Orientation

Requirement: CONDITIONAL

Condition: DUT supports the orientation change switch.

Features Tested: FEAT_SERVER_API_Display_Info

This test examines the Set Framebuffer Orientation Support method, and the Switch Framebuffer Orientation Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: This test does not require the CTS to be involved.

Note: This test assumes that the DUT's OS supports dynamic changing of orientation support. If it does not, these features cannot be tested.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	

Step	Name	Description	Expected Result
2	Set Orientation	Test engineer physically rotates the device into landscape orientation.	
3	Change Orientation	CTS sends an orientation change command (to portrait mode) to the server device. Note: The default mode is "Landscape Only"	• The application does not rotate into portrait mode. • The Callback Fired indicator indicates TRUE.
4	Change Orientation	CTS sends an orientation change command (to landscape mode) to the server device.	•
5	Configure Orientation Support	Test engineer configures the DUT to support Landscape and Portrait modes using the Test App. (Note: This should trigger the Callback Fired indicator to clear.)	• The Server reports that the app supports portrait and landscape? • The success indicator flag displayed by the Test App indicates TRUE.
6	Change Orientation	CTS sends an orientation change command (to portrait mode) to the server device.	• The application rotates into portrait mode. • The Callback Fired indicator indicates TRUE.
7	Configure Orientation Support	Test engineer configures the DUT to support Portrait mode only using the Test App. (Note: This should trigger the Callback Fired indicator to clear.)	• The Server reports that the app portrait? • The success indicator flag displayed by the Test App indicates TRUE.
8	Change Orientation	CTS sends an orientation change command (to landscape mode) to the server device.	• The Callback Fired indicator indicates TRUE.

Table 13: Orientation – Test Steps

3.6 Event Related Features

3.6.1 SR/API/ERF/EventConfiguration

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Events_Info

This test examines the Client Event Configuration Get method, and the Client Event Configuration Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Client Event Configuration	Test engineer invokes the Get Client Event Configuration method using the Test App. Note: The CTS should NOT advertise support for Rotary Knob	• The Knob Support, Device Key Support, Multimedia Key Support, Function Key Support, ITU Key Support, Touch Event Support and Pressure Mask values as displayed by the Test App

Step	Name	Description	Expected Result
		by default for this test, but report support for everything else (Device Keys, Multimedia Keys, Function keys, ITU Keys, Touch Events, and a Pressure Mask of 8.	correspond to the values provided to the DUT by the CTS in the VNC Client Event Configuration message and-ed with the values provided to the CTS by the DUT in the VNC Server Event Configuration Message. • The Success flag as displayed by the Test App indicates "TRUE".
3	Modify Client Event Configuration	Add advertised support for Rotary Knobs to the CTS Client Configuration. CTS resends the Client Configuration Message.	• The Callback Fired indicator in the test app indicates that the callback was triggered.
4	Execute Get Client Event Configuration	Test engineer invokes the Get Client Event Configuration method using the Test App.	• The Knob Support, Device Key Support, Multimedia Key Support, Function Key Support, ITU Key Support, Touch Event Support and Pressure Mask values as displayed by the Test App correspond to the values provided to the DUT by the CTS in the VNC Client Event Configuration message and-ed with the values provided to the CTS by the DUT in the VNC Server Event Configuration Message. • The Success flag as displayed by the Test App indicates "TRUE".

Table 14: Client Events – Test Steps

3.6.2 SR/API/ERF/EventMapping

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Events_Info

This test examines the Get Event Mapping Get method, and the Get Event Mapping Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Execute Get Event Mapping	Test engineer invokes the Get Event Mapping method using the Test App.	• The reported mapping matches that setup between the DUT and the CTS. • The Success flag as displayed by the Test App indicates "TRUE".

Step	Name	Description	Expected Result
3	Modify Client Event Configuration	The CTS sends a new Event Mapping Request message.	• The Callback Fired indicator in the test app indicates that the callback was triggered.
4	Execute Get Event Mapping	Test engineer invokes the Get Event Mapping method using the Test App.	• The reported mapping matches that setup between the DUT and the CTS. • The Success flag as displayed by the Test App indicates "TRUE".

Table 15: Event Mapping – Test Steps

3.7 Virtual Keyboard Related Features

3.7.1 SR/API/ERF/VirtualKeyboard

Requirement: CONDITIONAL

Condition: Support for Client Virtual Keyboard module

Features Tested: FEAT_SERVER_API_Virtual_Keyboard_Info

This test examines the Virtual Keyboard Support and Key Event List Support Get methods, the Show Virtual Keyboard and Key Event List Set methods, and the Virtual Keyboard Text Entry Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Send VKB Trigger Request	CTS sends a VKB Trigger request message, indicating support for text entry exchange, and an initial text length of 5.	
3	Execute Virtual Keyboard Support Get	Test engineer invokes the Virtual Keyboard Support get method using the Test App.	• The Virtual Keyboard Flag, Text Entry Flag, and Text Length values as displayed by the Test App correspond to the values provided to the CTS in the Server Event Configuration message. • The Success flag as displayed by the Test App indicates "TRUE".
4	Configure VKB Display	Test engineer configures the options for displaying the virtual keyboard. • Show keyboard set to "TRUE" • Key event list set to "FALSE". • If the DUT supports Server Cut Text, set the "Text Entry" value to "Mirr	• Test App allows these settings to be configured.
5	Show VKB	Test engineer selects the "Show Virtual Keyboard" button.	• The DUT sends the Virtual Keyboard Trigger – Show Keyboard message.

Step	Name	Description	Expected Result
			• If the DUT declares support for Server Cut Text, the CTS receives a Server Cut Text message of "Mirr".
8	Text Entry Callback	CTS sends a Client Cut Text message of "MirrorLink"	• The Text Entry field value displayed by the Test App reflects displays "MirrorLink". • The Callback Fired indicator in the test app indicates that the callback was triggered.
7	Hide VKB	Test Engineer configures the "Show Virtual Keyboard" message to hide the keyboard and invokes the method.	• The DUT sends the Virtual Keyboard Trigger – Hide Keyboard message.

Table 16: Virtual Keyboard – Test Steps

3.8 Key Event Listing

3.8.1 SR/API/ERF/KeyEventListing

Requirement: Conditional

Condition: Key Event Listing Module Available

Features Tested: FEAT_SERVER_API_Key_Event_Listing_Info

This test examines the Virtual Keyboard Support and Key Event List Support Get methods, the Show Virtual Keyboard and Key Event List Set methods, and the Virtual Keyboard Text Entry Callback to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Send VKB Trigger Request	CTS sends a VKB Trigger request message, indicating support for Key Event Lists, and an initial text length of 5. CTS sends a Key Event Listing Request Message	
3	Execute Key Event List Support Get	Test engineer invokes the Key Event List Support get method using the Test App.	• The Key Event Listing flag indicates "TRUE". • The Success flag as displayed by the Test App indicates "TRUE".
4	Configure VKB Display	Test engineer configures the options for displaying the virtual keyboard. • Show keyboard set to "TRUE" • Key event list set to "TRUE". • If the DUT supports Server Cut Text, set the "Text Entry" value to "Mirr"	• Test App allows these settings to be configured.

Step	Name	Description	Expected Result
5	Show VKB	Test engineer selects the "Show Virtual Keyboard" button. Note: The Test App will automatically send a key event list of "a", "b", "c", "d", "e", "z" when the Key Event List is set to "TRUE".	• The DUT sends the Virtual Keyboard Trigger – Show Keyboard message. • The DUT sends a key event listing message with values of "a", "b", "c", "d", "e", "z". • If the DUT declares support for Server Cut Text, the CTS receives a Server Cut Text message of "Mirr".
6	Trigger Key Event	CTS sends a triggered key of "c".	• The Key Event Callback Key Selected value displayed by the Test App is "c". • The Callback Fired indicator in the test app indicates that the callback was triggered.

Table 17: Virtual Keyboard – Test Steps

3.9 Context Information Related Features

3.9.1 SR/API/CIRF/FramebufferInformation

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Context_Info

This test examines the Framebuffer Status and Framebuffer Context Information Set methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Check Framebuffer Context Info	CTS examines the Framebuffer context information provided by the DUT.	• The Application Category, Content Category and Framebuffer area provided by the DUT correspond to the information provided initially by the test app and contained in the application certificate (communicated to CTS in the UPnP advertisements). • The reported Application Identifier correctly identifies the application.
3	Set New Context Information	Test Engineer uses the Test App to configure the Framebuffer Context Information Set parameters: • Sets Application Category to Calendar - 0x000A, 0x0001 (see Table 6-2 in [8]). • Add new Video Content Category of "Image" (bit 2) (see Table 6-3 in [8]).	• The Application Category, Video Content Category and Framebuffer area reported to the CTS matches those set. • The reported Application Identifier correctly identifies the application.

Step	Name	Description	Expected Result
		• Sets the Framebuffer Area to a subset of the screen (can be fixed by Test App). • Handle blocking set to yes. Test Engineer selects the "Send Framebuffer Context Information" button.	
4	Block Content	CTS sends a blocking message to the DUT with a reason of "UI not in focus on remote display" (bit 8).	• The Framebuffer Blocking Information Callback values displayed by the Test App correspond to the blocked region and the blocking reason. • The Callback Fired indicator in the test app indicates that the callback was triggered.

Table 18: Framebuffer Information – Test Steps

3.9.2 SR/API/CIRF/AudioContext

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Context_Info

This test examines the Audio Context Set method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Begin Audio Playback	Test Engineer commands the Test App to begin playing an audio track.	• The DUT begins streaming audio data. • The reported Application Category in the RTP header packets correctly identifies the application category as set in the UPnP advertisements, and contained within the Test App's certificate OR the audio context info is set to 0x00000000, as indicated by PIXIT 16030. • The reported Application Identifier in the RTP header packets correctly identifies the application OR the audio identifier is set to 0x00000000, as indicated by PIXIT 16030.
3	Configure Audio Context Setting	Test Engineer sets the parameters of the Audio Context Information Set command using the Test App: • Audio Content – TRUE • Add a new Audio Content Category, such as "0x00030001" (see Table 6-2 in [8]).	• The reported Application category in the RTP header packets changes to the match the newly set category. • The reported Application Identifier in the RTP header packets correctly identifies the application.

Step	Name	Description	Expected Result
		• Handle Blocking set to TRUE The Test Engineer then causes the app to send the Audio Context Information Set command via the Test App.	• The reported Application category in the RTP header packets does not change for 10 seconds.
4	Block Content	CTS sends a blocking message to the DUT with a reason of "Global Audio Muted" (bit 3).	• The Audio Blocking Information Callback values displayed by the Test App correspond to the blocking reason provided by the CTS. • The Callback Fired indicator in the test app indicates that the callback was triggered.

Table 19: Audio Context – Test Steps

3.9.3 SR/API/CIRF/ContextReset

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Context_Info

This test examines the Context Information Reset Set method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Begin Audio Playback	Test Engineer commands the Test App to begin playing an audio track.	• The DUT begins streaming audio data.
3	Configure Audio Context Setting	Test Engineer sets the parameters of the Audio Context Information Set command using the Test App: • Audio Content – TRUE • A new Application Category, such as Music (0x00030001) (see Table 6-2 in [8]). The Test Engineer then causes the app to send the Audio Context Information Set command via the Test App.	• The reported Application category in the RTP header packets changes to the match the newly set category. • The reported Application Identifier in the RTP header packets correctly identifies the application.
4	Send Context Information Reset - Audio	Test Engineer configures the parameters for the Context Information Reset using the Test App to the following values: • Framebuffer Context Reset – FALSE • Audio Context Reset – TRUE Test Engineer then selects the "Send Context Information Reset" button.	• The reported Application Category in the RTP header packets correctly identifies the application category as set in the UPnP advertisements, and contained within the Test App's certificate, OR the audio context info is set to 0x00000000, as indicated by PIXIT 16030. • The reported Application Identifier in the RTP header packets

Step	Name	Description	Expected Result
			correctly identifies the application OR the audio identifier is set to 0x00000000, as indicated by PIXIT 16030.
5	Set New FB Context Information	Test Engineer uses the Test App to configure the Framebuffer Context Information Set parameters: • Sets Application Category to Calendar - 0x000A0001 (see Table 6-2 in [8]). • Sets Content Category to Image – 2 (see Table 6-3 in [8]). • Sets the Framebuffer Area to a subset of the screen (can be fixed by Test App). Test Engineer selects the "Send Framebuffer Context Information" button.	• The App Identifier, Application Category, Content Category and Framebuffer area reported to the CTS matches those set.
6	Send Context Information Reset - FB	Test Engineer configures the parameters for the Context Information Reset using the Test App to the following values: • Framebuffer Context Reset – TRUE • Audio Context Reset – FALSE Test Engineer then selects the "Send Context Information Reset" button.	• The reported Application Category matches the application category as set in the UPnP advertisements, and contained within the Test App's certificate. • The reported Application Identifier correctly identifies the application.

Table 20: Context Reset – Test Steps

3.10 Device Status Related Features

3.10.1 SR/API/DSRF/DeviceModes

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Device_Status

This test examines the Drive Mode and Night Mode Callback methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	View Drive Mode Status	Test Engineer, using the Test App, configures the app to indicate Drive Mode status.	
3	Activate Drive Mode	CTS sends the Device Status Request message, indicating that Drive Mode is enabled.	• The Drive Mode status indicator in the Test App displays TRUE. • The Callback Fired indicator in the test app indicates that the Drive Mode callback was triggered.

Step	Name	Description	Expected Result
4	De-Activate Drive Mode	CTS sends the Device Status Request message, indicating that Drive Mode is disabled.	• The Drive Mode status indicator in the Test App displays FALSE. • The Callback Fired indicator in the test app indicates that the Drive Mode callback was triggered.
5	View Night Mode Status	Test Engineer, using the Test App, configures the app to indicate Night Mode status.	
6	Activate Night Mode	CTS sends the Device Status Request message, indicating that Night Mode is enabled.	• The Night Mode status indicator in the Test App displays TRUE. • The Callback Fired indicator in the test app indicates that the Night Mode callback was triggered.
7	De-Activate Night Mode	CTS sends the Device Status Request message, indicating that Night Mode is disabled.	• The Night Mode status indicator in the Test App displays FALSE. • The Callback Fired indicator in the test app indicates that the Night Mode callback was triggered.

Table 21: Device Modes – Test Steps

3.10.2 SR/API/DSRF/Microphone

Requirement: MANDATORY

Condition: None

Features Tested: FEAT_SERVER_API_Device_Status

This test examines the Open Microphone Set and Open Microphone Callback methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Setup audio back channel	Setup the RTP audio backchannel.	If the DUT declares support for Voice Control or Phone Audio over RTP in the PICS • CTS launches the RTP Client on the DUT
3	Set Open Microphone	Test Engineer, using the Test App, configures the parameters of the Set Open Microphone message: • Mic Input – TRUE Test engineer then selects the "Send Set Open Microphone" button.	If the DUT declares support for Voice Control or Phone Audio in the PICS: • The DUT opens a microphone connection to the CTS.
4	Set Closed Microphone	Test Engineer, using the Test App, configures the parameters of the Set Open Microphone message:	If the DUT declares support for Voice Control or Phone Audio in the PICS:

Step	Name	Description	Expected Result
		• Mic Input – FALSE Test engineer then selects the "Send Set Open Microphone" button.	• The DUT closes the microphone connection to the CTS.
5	Client Open Microphone	CTS opens a microphone connection from the (simulated) ML Client.	If the DUT declares support for Voice Control or Phone Audio in the PICS: • The Min Input Status indicator in the Test App shows TRUE. • The Callback Fired indicator in the test app indicates that the Open Microphone callback was triggered. Else (the DUT does not declare support for Voice Control or Phone Audio in the PICS): • The Callback Fired indicator does not indicate the callback was triggered.
6	Client Close Microphone	CTS closes the microphone connection from the (simulated) ML Client.	If the DUT declares support for Voice Control or Phone Audio in the PICS: • The Min Input Status indicator in the Test App shows False. • The Callback Fired indicator in the test app indicates that the Open Microphone callback was triggered. Else (the DUT does not declare support for Voice Control or Phone Audio in the PICS): • The Callback Fired indicator does not indicate the callback was triggered.

Table 22: Microphone – Test Steps

3.11 Data Services

3.11.1 SR/API/DS/DataServices

Requirement: Conditional

Condition: Support for the Data Services Module

Features Tested: FEAT_SERVER_API_Data_Services

This test examines the Get Available Services Get method, Register to a Service and Unregister from a Service Set methods, and the Available Services Callback methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	CDB Identification	Identify the CDB server in the UPnP Application Listing, checking for CDB protocol ID	• Successful identification of CDB server

Step	Name	Description	Expected Result
			• Only 1 CDB server provided
3	Launch CDB Server	Call UPnP ApplicationServer:1 LaunchApplication action and receive the CDB server's URL.	• Receive URL, without getting an error message or a timeout
4	Get Available Services List	Test Engineer uses the test app to invoke the Get Available Services method.	• The Test App displays the list of available services, including the version information, service ID and service Name, as provided by the CTS and supported by the DUT (GPS and/or Location).
5	Add Additional Services	CTS sends a new ServicesSupported message, with an additional services advertised (can be a "dummy" service).	• The Callback Fired indicator does not indicate "TRUE".
6	Get Updated Available Services List	Test Engineer uses the test app to invoke the Get Available Services method.	• The Test App displays the updated list of available services, including the version information, service ID and service Name, as provided by the CTS and supported by the DUT (GPS and/or Location).
7	Register a Service	Test Engineer selects one of the available services (non-dummy) from the available list, and selects the "Register to Selected Service" button, which invokes the Register to a Service set method.	• The DUT sends a Start Service request to the CTS for the selected service ID. • The success flag indicator in the Test App displays TRUE.
8	Start Service	CTS Sends ServiceResponse message to DUT.	• The Test App displays the service ID indicated in the ServiceResponse message. • The success flag indicator in the Test App displays TRUE. • The callback fired indicator displays TRUE.
9	Unregister From the Service	Test Engineer selects the "Unregister From Selected Service" button, which invokes the Unregister from a Service command for the selected service.	• The DUT sends a Stop Service request to the CTS for the selected service ID. • The success flag indicator in the Test App displays TRUE.
10	CDB ByeBye	Send a CDB ByeBye Message	• DUT sends CDB ByeBye message.
11	CDB Terminate Application	Disconnect TCP Call UPnP ApplicationServer:1 TerminateApplication action	• No further CDB messages received • Receive TRUE response

Table 23: Data Services – Test Steps

3.11.2 SR/API/DS/DataObjectSinkLocation

Requirement: Conditional

Condition: Support for the Data Services Module AND

Location Data Service

Features Tested: FEAT_SERVER_API_Data_Services

This test examines the Subscribe to an Object and Unsubscribe to an Object Set methods, and the Subscribe to an Object Callback method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: This test uses the Location Object.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	CDB Identification	Identify the CDB server in the UPnP Application Listing, checking for CDB protocol ID	• Successful identification of CDB server • Only 1 CDB server provided
3	Launch CDB Server	Call UPnP ApplicationServer:1 LaunchApplication action and receive the CDB server's URL.	• Receive URL, without getting an error message or a timeout
4	Get Available Services List	Test Engineer uses the test app to invoke the Get Available Services method.	• The Test App displays the list of available services, including the version information, service ID and service Name, as provided by the CTS and supported by the DUT.
5	Get Update	Test Engineer invokes the Get an Object data configured for a Location object of the registered service.	• The displayed values of the Location object (lat, long, altitude, accuracy, altitude accuracy, timestamp, etc.) match the values as provided by the CTS.
6	Subscribe to Object	Test Engineer selects an object from the Available Services list, and selects the "Subscribe to Object" button, which invokes the Subscribe to an Object method with the appropriate Location ObjectID.	• The CTS receives a Subscribe message for the selected Service ID, and an appropriate object ID. • The success indicator in the Test App displays TRUE.
7	Complete Subscription	CTS and DUT complete object subscription.	• The Service ID and Object ID values displayed within the Test App correspond to the data object settings provided via SBP interaction. • The success flag indicator in the Test App displays TRUE. • The callback fired indicator displays TRUE.
8	Show Updates	Allow the SBP object subscription to run for at least 10 intervals as reported in the Subscribe to an Object Callback (step 6). The CTS adjusts the reported Location altitude by 100 meters every half second.	• Test App displays that the Get Object callback has been invoked every Interval ms (±100 ms) or whenever the CTS adjusts the Location object. • The displayed values of the Location object (lat, long, altitude, accuracy, altitude accuracy, timestamp, etc.) match the values as provided by the CTS. • The displayed Altitude increases with each step.

Step	Name	Description	Expected Result
9	Unsubscribe to Object	Test Engineer selects the "Unsubscribe from Location Object", which invokes the Unsubscribe from an Object method	• The Test App displays the service ID indicated in the ServiceResponse message. • The success flag indicator in the Test App displays TRUE. • The callback fired indicator displays TRUE.
10	Unregister From the Service	Test Engineer selects the "Unregister From Selected Service" button, which invokes the Unregister from a Service command for the selected service.	• The DUT sends a Cancel message to the CTS for the selected service ID and object ID. • The success flag indicator in the Test App displays TRUE.

Table 24: Data Object Sink Location – Test Steps

3.11.3 SR/API/DS/DataObjectSinkGPS

Requirement: Conditional

Condition: Support for the Data Services Module AND

GPS Data Service

Features Tested: FEAT_SERVER_API_Data_Services

This test examines the Subscribe to an Object and Unsubscribe to an Object Set methods, and the Subscribe to an Object Callback method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Note: This test uses the GPS Object.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	CDB Identification	Identify the CDB server in the UPnP Application Listing, checking for CDB protocol ID	• Successful identification of CDB server • Only 1 CDB server provided
3	Launch CDB Server	Call UPnP ApplicationServer:1 LaunchApplication action and receive the CDB server's URL.	• Receive URL, without getting an error message or a timeout
4	Get Available Services List	Test Engineer uses the test app to invoke the Get Available Services method.	• The Test App displays the list of available services, including the version information, service ID and service Name, as provided by the CTS and supported by the DUT.
5	Get Update	Test Engineer invokes the Get an Object data configured for the NMEA_description object of the registered service.	• The displayed values of the NMEA_Description object (supportedSentences) match the values as provided by the CTS.
6	Subscribe to Object	Test Engineer selects an object from the Available Services list, and selects the "Subscribe to Object" button, which invokes the Subscribe to the NMEA Object method with the appropriate ObjectID.	• The CTS receives a Subscribe message for the selected Service ID, and an appropriate object ID. • The success indicator in the Test App displays TRUE.

Step	Name	Description	Expected Result
7	Complete Subscription	CTS and DUT complete object subscription.	• The Service ID and Object ID values displayed within the Test App correspond to the data object settings provided via SBP interaction. • The success flag indicator in the Test App displays TRUE. • The callback fired indicator displays TRUE.
8	Show Updates	Allow the SBP object subscription to run for at least 10 intervals as reported in the Subscribe to an Object Callback (step 6).	• Test App displays that the Get Object callback has been invoked every Interval ms (±100 ms) or whenever the CTS adjusts the NMEA object. • The displayed values of NMEA object match the values as provided by the CTS.
9	Unsubscribe to Object	Test Engineer selects the "Unsubscribe from NMEA Object", which invokes the Unsubscribe from an Object method	• The Test App displays the service ID indicated in the ServiceResponse message. • The success flag indicator in the Test App displays TRUE. • The callback fired indicator displays TRUE.
10	Unregister From the Service	Test Engineer selects the "Unregister From Selected Service" button, which invokes the Unregister from a Service command for the selected service.	• The DUT sends a Cancel message to the CTS for the selected service ID and object ID. • The success flag indicator in the Test App displays TRUE.

Table 25: Data Object Sink GPS– Test Steps

3.12 Notifications

3.12.1 SR/API/NTF/NotificationSupport

Requirement: CONDITIONAL

Condition: Support for Notifications Module

Features Tested: FEAT_SERVER_API_Notifications

This test examines the Notifications Enabled Get method, the Notifications Supported Set method, and the Notifications Enabled Callback method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Check Notification Enabled Status	Test Engineer selects the "Get Notification Enabled Status" button, which invokes the Notifications Enabled method.	• The Notifications Enabled flag as displayed by the Test App indicates FALSE.

Step	Name	Description	Expected Result
3	Announce Support for Notifications	Test Engineer configures the parameters for the Notifications Supported set command using the Test App: • Notifications support – TRUE The Test Engineer then selects the "Set Notifications Supported" button to invoke the Notifications Supported Set command.	• The CTS receives a NotiAppListUpdate event from the DUT, indicating notification support is enabled. • The success indicator in the Test App displays TRUE.
4	Enable Notifications for Test App	CTS Sends a SetAllowedApplications message to the DUT, enabling notifications for the Test App.	
5	Check Notification Enabled Status	Test Engineer selects the "Get Notification Enabled Status" button, which invokes the Notifications Enabled method.	• The Notifications Enabled flag as displayed by the Test App indicates TRUE.
6	Change Notification Status	CTS Sends a SetAllowedApplications message to the DUT, disabling notifications for the Test App.	• The callback fired indicator displays TRUE.
7	Check Notification Enabled Status	Test Engineer selects the "Get Notification Enabled Status" button, which invokes the Notifications Enabled method.	• The Notifications Enabled flag as displayed by the Test App indicates FALSE.

Table 26: Notification Support – Test Steps

3.12.2 SR/API/NTF/NotificationConfiguration

Requirement: CONDITIONAL

Condition: Support for Notifications Module

Features Tested: FEAT_SERVER_API_Notifications

This test examines the Notification Configuration Get method and Notification Configuration Callback method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Get Notification Configuration	Test Engineer selects the "Get Notification Configuration" button, which invokes the Notification Configuration Get method.	• The Notification UI support, Max Actions, Max Action Name Length, Max Notification Title Length and Max Body Length values displayed by the Test App match values provided by the CTS to the DUT in the Client Profile. • The success indicator in the Test App displays TRUE.
3	Modify Notification Status	CTS modifies the client profile with modified values for the	• The callback fired indicator displays TRUE.

Step	Name	Description	Expected Result
		MaxActions, Max Action Name Length, Max Notification Title Length and Max Body Length	
4	Get Notification Configuration	Test Engineer selects the "Get Notification Configuration" button, which invokes the Notification Configuration Get method.	• The Notification UI support, Max Actions, Max Action Name Length, Max Notification Title Length and Max Body Length values displayed by the Test App match the updated values provided by the CTS to the DUT in the Client Profile. • The success indicator in the Test App displays TRUE.

Table 27: Notification Configuration – Test Steps

3.12.3 SR/API/NTF/ClientNotification

Requirement: CONDITIONAL

Condition: Support for Notifications Module

Features Tested: FEAT_SERVER_API_Notifications

This test examines the Send Notification for client-based Notification UI and Cancel Notification Set methods, and the Notification Pending and Receive Action Callback methods to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Get Notification Configuration	Test Engineer selects the "Get Notification Configuration" button, which invokes the Notification Configuration Get method.	• The Notification UI support, Max Actions, Max Action Name Length, Max Notification Title Length and Max Body Length values displayed by the Test App match values provided by the CTS to the DUT in the Client Profile. • The success indicator in the Test App displays TRUE.
3	Configure Notification	Test Engineer sets the parameter values of the Send Notification command, as constrained by the Get Notification Configuration information and prompted by the CTS: • Notification ID • Notification Title • Notification Body Test • Icon URL (choose from list provided by test app/chosen by test app) • Action List is defined by the test app, and includes	• The DUT sends a notification message to the CTS that matches the values selected in the Test App except icon URL field. • Correct icon made available • The notification ID as displayed by the test app matches that sent by the CTS.

Step	Name	Description	Expected Result
		○ Max Actions actions. ○ Icons for each action. ○ Titles for each action (test 1, test 2, test 3). Test Engineer selects the "Show Client-Notification" button, invoking the Send Notification for Client-Based Notification UI.	
4	Receive Action Callback	CTS sends InvokeNotiAction for the Notification ID for the first action in the action list.	• The Notification ID and action ID values displayed by the Test App match the notiID and actionID provides in the InvokeNotiAction by the CTS. • The callback fired indicator displays TRUE.
5	Re-Send Configure Notification	Test Engineer selects the "Show Client-Notification" button, using the same settings as previously, invoking the Send Notification for Client-Based Notification UI.	• The DUT sends a notification message to the CTS that matches the values selected in the Test App except icon URL field. • Correct icon made available • The success indicator in the Test App displays TRUE.
6	Cancel Notification	Test Engineer selects the "Cancel Notification" button, which invokes the Cancel Notification method.	• The CTS receives an ActiveNotiEvent message either as an empty string (no other notifications pending) or a notification for another app ID.

Table 28: Notification Configuration – Test Steps

3.12.4 SR/API/NTF/VNCNotification

Requirement: CONDITIONAL

Condition: Support for Notifications Module

Features Tested: FEAT_SERVER_API_Notifications

This test examines the Send Notification for VNC-based Notification UI Set method to ensure that the Common API implementation responds to those methods, and provides the correct information.

Step	Name	Description	Expected Result
1	Test App Launch	See Definitions	
2	Set VNC Notification	Test Engineer selects the "Send Notification for VNC-Based Notification UI" button, invoking the Send Notification for VNC-Based Notification UI set method.	• The DUT sends an ActiveNotiEvent message to the CTS. • The notification ID displayed by the test app matches the notification ID sent to the CTS.
3	Cancel Notification	Test Engineer selects the "Cancel Notification" button, which invokes the Cancel Notification method.	• The CTS receives an ActiveNotiEvent message either as an empty string (no other notifications pending) or a notification for another app ID.

1 Table 29: Notification Configuration – Test Steps

4 REFERENCES

[1] IETF, RFC 2119, "Keys words for use in RFCs to Indicate Requirement Levels", March 1997. http://www.ietf.org/rfc/rfc2119.txt

[2] Car Connectivity Consortium, "MirrorLink – Common API", Version 1.1, CCC-TS-038

[3] Car Connectivity Consortium, "MirrorLink – Interoperability Test Specification", Version 1.1, CCC-TS-033

[4] Car Connectivity Consortium, "MirrorLink – Handling of Application Certificates", Version 1.1.0; CCC-TS-036

[5] Car Connectivity Consortium, "MirrorLink – Application Base Certification Market Requirements Document", Version 0.5

[6] Car Connectivity Consortium" MirrorLink – Test Application Requirements", Version 0.3

[7] Car Connectivity Consortium, "MirrorLink – VNC Based Display and Control – Test", Version 1.1, CCC-TS-011

[8] Car Connectivity Consortium, "MirrorLink – UPnP Application Server Service", Version 1.1; CCC-TS-024