Basic Imaging Profile (BIP)

***Bluetooth*® Test Suite**

- **Revision:** BIP.TS.p17
- **Revision Date:** 2024-07-01
- **Prepared By:** BTI
- **Published during TCRL:** TCRL.2024-1

Contents

1 Scope

This Bluetooth document contains the Test Suite Structure (TSS) and test cases to test the implementation of the Bluetooth Basic Imaging Profile (BIP) with the objective to provide a high probability of air interface interoperability between the tested implementation and other manufacturers' Bluetooth devices.

2 References, definitions, and abbreviations

2.1 References

This document incorporates provisions from other publications by dated or undated reference. These references are cited at the appropriate places in the text and the publications are listed hereinafter. Additional definitions and abbreviations can be found in [1], [2], and [7].

[1] Bluetooth Core Specification, Version 2.0 or later

[2] Basic Imaging Profile Specification

[3] Serial Port Profile (SPP) Test Suite

[4] Specification of IrDA Object Exchange Protocol, Version 1.2

[5] Implementation Conformance Statement for Basic Imaging Profile (BIP)

[6] Generic Access Profile (GAP) Test Suite

[7] Test Strategy and Terminology Overview

[8] Generic Object Exchange Profile Specification

[9] BIP Implementation eXtra Information for Testing (IXIT)

[10] SDP Test Suite, SDP.TS

[11] GOEP Test Suite, GOEP.TS

2.2 Definitions

In this Bluetooth document, the definitions from [1], [2], and [7] apply.

2.3 Acronyms and abbreviations

In this Bluetooth document, the definitions, acronyms, and abbreviations from [1], [2], and [7] apply.

3 Test Suite Structure (TSS)

3.1 Overview

The Test Suite is divided into three plus seven test groups:

- Fundamental Functional Components group
- Feature Specific Functions group
- Multi-Function Sequences group
- Image Push
- Image Pull
- Advanced Image Printing
- Automatic Archive
- Remote Camera
- Remote Display

3.2 Test grouping

The test groups are organized in two or three levels. The first level defines a basic grouping in test objectives. When more refinement is necessary in the grouping, a second level separates the basic groups into functional modules. The last level in each branch contains the standard ISO subgroups BV and BI.

3.2.1 Test subgroups

The conformance Test Suite is further divided into subgroups that target one function or a class of functions with similar or complementary roles in the profile.

4 Test cases (TC)

4.1 Introduction

4.1.1 Test case identification conventions

Test cases are assigned unique identifiers per the conventions in [7]. The convention used here is: **<spec abbreviation>/<IUT role>/**<class>/**<feat>**/<func>/<subfunc>/<cap>/**<xx>-<nn>-<y>**.

Testing of this specification includes a set of tests from the GOEP Test Suite [11]; when used, the GOEP tests are referred to in the TCMT using the following convention:
<spec abbreviation>/<IUT role>/GOEP/<GOEP TC Identifier>.

Additionally, testing of this specification includes tests from the SDP Test Suite [10] referred to as Generic SDP Integrated Tests (GSIT); when used, the test cases in GSIT are referred to through a TCID string using the following convention:
<spec abbreviation>/<IUT role>/<GSIT test group>/< GSIT class >/<xx>-<nn>-<y>.

Identifier Abbreviation	**Spec Identifier <spec abbreviation>**
BIP	Basic Imaging Profile
Identifier Abbreviation	**Role Identifier <IUT role>**
AAI	Automatic Archive Initiator role
AAR	Automatic Archive Responder role
AIPI	Advanced Image Printing Initiator role
AIPR	Advanced Image Printing Responder role
IPLI	Image Pull Initiator role
IPLR	Image Pull Responder role
IPSI	Image Push Initiator role
IPSR	Image Push Responder role
RCI	Remote Camera Initiator role
RCR	Remote Camera Responder role
RDI	Remote Display Initiator role
RDR	Remote Display Responder role
SR	Server role (used in Service Discovery testing)
Identifier Abbreviation	**Reference Identifier <GSIT test group>**
CGSIT	Client Generic SDP Integrated Tests
SGSIT	Server Generic SDP Integrated Tests
Identifier Abbreviation	**Reference Identifier <GSIT class>**
ATTR	Attribute
OFFS	Attribute ID Offset String
SERR	Service Record
SFC	SDP Future Compatibility
Identifier Abbreviation	**Feature Identifier <feat>**
ACH	Automatic Archive Feature
ADP	Advanced Image Printing Feature
FFC	Fundamental Functional Component

Identifier Abbreviation	Spec Identifier <spec abbreviation>
FSF	Feature Specific Function
GOEP	Generic Object Exchange Profile
MFS	Multi-Function Sequences
PLL	Image Pull Feature
PSH	Image Push Feature
RMC	Remote Camera Feature
RMD	Remote Display Feature

Table 4.1: BIP TC feature naming conventions

4.1.2 Conformance

When conformance is claimed for a particular specification, all capabilities are to be supported in the specified manner. The mandated tests from this Test Suite depend on the capabilities to which conformance is claimed.

The Bluetooth Qualification Program may employ tests to verify implementation robustness. The level of implementation robustness that is verified varies from one specification to another and may be revised for cause based on interoperability issues found in the market.

Such tests may verify:

- That claimed capabilities may be used in any order and any number of repetitions not excluded by the specification
- That capabilities enabled by the implementations are sustained over durations expected by the use case
- That the implementation gracefully handles any quantity of data expected by the use case
- That in cases where more than one valid interpretation of the specification exists, the implementation complies with at least one interpretation and gracefully handles other interpretations
- That the implementation is immune to attempted security exploits

A single execution of each of the required tests is required to constitute a Pass verdict. However, it is noted that to provide a foundation for interoperability, it is necessary that a qualified implementation consistently and repeatedly pass any of the applicable tests.

In any case, where a member finds an issue with the test plan generated by the Bluetooth SIG qualification tool, with the test case as described in the Test Suite, or with the test system utilized, the member is required to notify the responsible party via an erratum request such that the issue may be addressed.

4.1.3 Configurations

The following notations are used to represent the process of putting the equipment in the proper initial condition.

In cases where the test case only makes sense in a primary connection, the representation is:

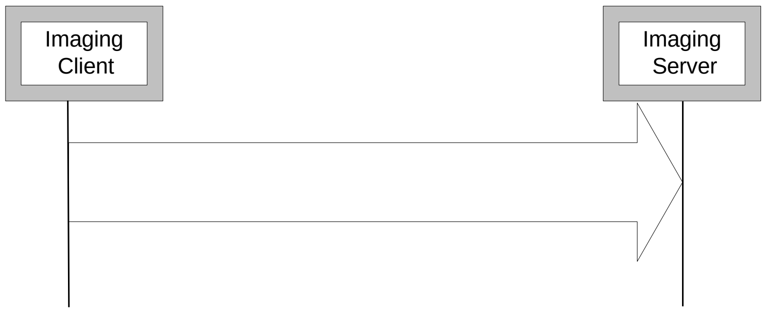


Figure 4.1: Configuration, in cases where the test case only makes sense in a primary connection

In cases where the test case could make sense either in a primary or a secondary connection depending on the Basic Imaging Profile feature, the representation is:

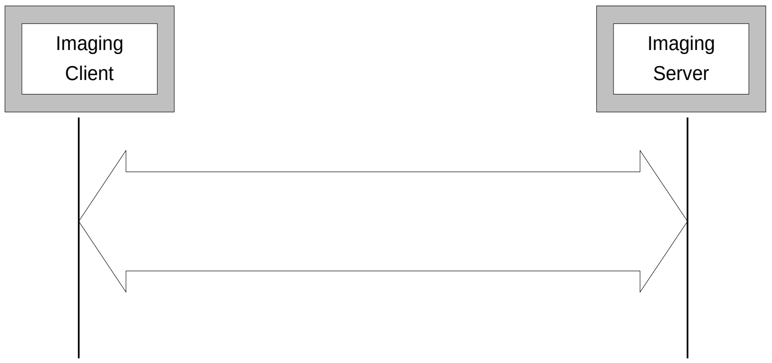


Figure 4.2: Configuration, in cases where the test case could make sense either in a primary or a secondary connection depending on the Basic Imaging Profile feature

For a test case to pass in a primary connection, the above notations translate into:

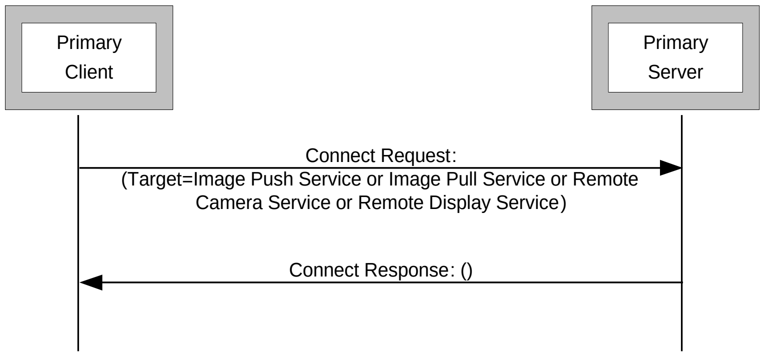

Primary Client
Primary Server
Connect Request:
(Target=Image Push Service or Image Pull Service or Remote Camera Service or Remote Display Service)
Connect Response: ()

Figure 4.3: Configuration, for a test case to pass in a primary connection

(The Imaging Client of the test case is the Primary Client. The Imaging Server is the Primary Server.)

For a test case to pass in a secondary connection, the above notations translate into:

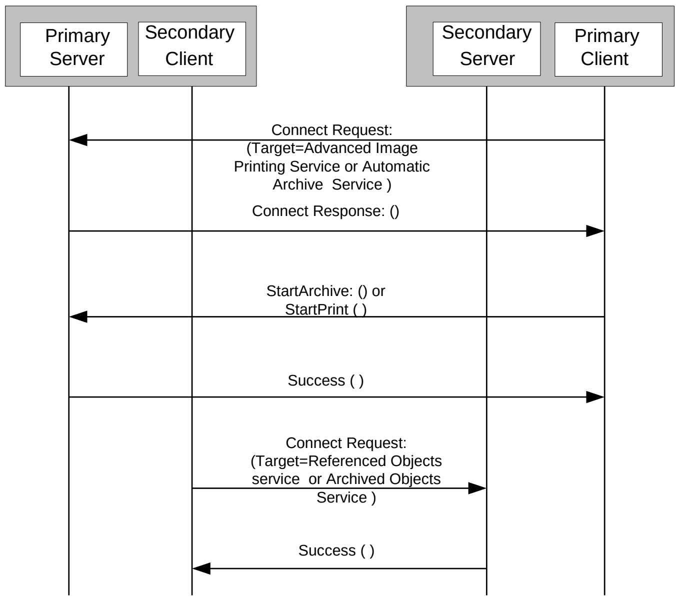

Primary Server
Secondary Client
Secondary Server
Primary Client
Connect Request: (Target=Advanced Image Printing Service or Automatic Archive Service)
Connect Response: ()
StartArchive: () or StartPrint ( )
Success ( )
Connect Request: (Target=Referenced Objects service or Archived Objects Service)
Success ( )

Figure 4.4: Configuration, for a test case to pass in a secondary connection

(The Imaging Client of the test case is the Secondary Client and the Imaging Server is the Secondary Server.)

4.1.4 Pass/Fail verdict conventions

Each test case has an Expected Outcome section. The IUT is granted the Pass verdict when all the detailed pass criteria conditions within the Expected Outcome section are met.

The convention in this Test Suite is that, unless there is a specific set of fail conditions outlined in the test case, the IUT fails the test case as soon as one of the pass criteria conditions cannot be met. If this occurs, then the outcome of the test is a Fail verdict.

4.2 Generic SDP Integrated Tests

4.2.1 Server Generic SDP Integrated Tests

4.2.1.1 Imaging

Execute the Generic SDP Integrated Tests defined in Section 6.3, Server test procedures (SGSIT), in [10] using Table 4.2 below as input:

TCID	Reference	Attribute ID name	Attribute ID definition source (Universal, Profile)	Value/secondary value	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/SR/SGSIT/SERR/BV-01-C [Service record GSIT – BIP Image Responder]	[2] 6.1.1	ServiceClassIDList	Universal	“Imaging Responder” (UUID)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder
BIP/SR/SGSIT/ATTR/BV-01-C [Attribute GSIT – Protocol Descriptor List]	[2] 6.1.1	ProtocolDescriptorList	Universal	“L2CAP” (UUID), “RFCOMM” (UUID): Channel number – skip (Uint8), “OBEX” (UUID)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder
BIP/SR/SGSIT/ATTR/BV-02-C [Attribute GSIT – Bluetooth Profile Descriptor List, BIP 1.2]	[2] 6.1.1	BluetoothProfileDescriptorList	Universal	“Imaging” (UUID): Version – “0x0102” (Uint16)	TCMT defined
BIP/SR/SGSIT/ATTR/BV-03-C [Attribute GSIT – Supported Capabilities]	[2] 6.1.1	Supported Capabilities	Profile	skip (Uint8)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder
BIP/SR/SGSIT/ATTR/BV-04-C [Attribute GSIT – Supported Features]	[2] 6.1.1	Supported Features	Profile	skip (Uint16)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder
BIP/SR/SGSIT/ATTR/BV-05-C [Attribute GSIT – Supported Functions]	[2] 6.1.1	Supported Functions	Profile	skip (Uint32)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder

TCID	Reference	Attribute ID name	Attribute ID definition source (Universal, Profile)	Value/secondary value	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/SR/SGSIT/ATTR/BV-06-C [Attribute GSIT – Total imaging data capacity]	[2] 6.1.1	Total imaging data capacity	Profile	skip (Uint64)	Present for Image Push Responder, Image Pull Responder, Advanced Image Printing Responder, Automatic Archive Responder, Remote Camera Responder, Remote Display Responder
BIP/SR/SGSIT/ATTR/BV-07-C [Attribute GSIT – GOEP L2CAP PSM]	[2] 6.1.1	GoepL2capPsm	Profile	skip (Uint16)	TCMT defined

Table 4.2: Input for the Imaging Service Record SGSIT SDP test procedure

4.2.1.2 Referenced Objects

Execute the Generic SDP Integrated Tests defined in Section 6.3, Server test procedures (SGSIT), in [10] using Table 4.3 below as input:

TCID	Reference	Attribute ID name	Attribute ID definition source (Universal, Profile)	Value/secondary value	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/AIPI/SGSIT/SERR/BV-02-C [Service record GSIT – BIP Advanced Image Printing Initiator]	[2] 6.1.2	ServiceClassIDList	Universal	“Imaging Referenced Objects” (UUID)	Present for Advanced Image Printing Initiator
BIP/AIPI/SGSIT/ATTR/BV-08-C [Attribute GSIT – Protocol Descriptor List]	[2] 6.1.2	ProtocolDescriptorList	Universal	“L2CAP” (UUID), “RFCOMM” (UUID): Channel number – skip (Uint8), “OBEX” (UUID)	Present for Advanced Image Printing Initiator
BIP/AIPI/SGSIT/ATTR/BV-09-C [Attribute GSIT – Service ID]	[2] 6.1.2	ServiceID	Universal	skip (UUID)	Present for Advanced Image Printing Initiator
BIP/AIPI/SGSIT/ATTR/BV-10-C [Attribute GSIT – Bluetooth Profile Descriptor List, BIP 1.2]	[2] 6.1.2	BluetoothProfileDescriptorList	Universal	“Imaging” (UUID): Version – “0x0102” (Uint16)	TCMT defined
BIP/AIPI/SGSIT/ATTR/BV-11-C [Attribute GSIT – Supported Functions]	[2] 6.1.2	Supported Functions	Profile	skip (Uint32)	Present for Advanced Image Printing Initiator

TCID	Reference	Attribute ID name	Attribute ID definition source (Universal, Profile)	Value/secondary value	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/AIPI/SGSIT/ATTR/BV-12-C [Attribute GSIT – GOEP L2CAP PSM]	[2] 6.1.2	GoepL2capPsm	Profile	skip (Uint16)	TCMT defined

Table 4.3: Input for the Referenced Objects Service Record SGSIT SDP test procedure

4.2.1.3 Archived Objects

Execute the Generic SDP Integrated Tests defined in Section 6.3, Server test procedures (SGSIT), in [10] using Table 4.4 below as input:

TCID	Reference	Attribute ID name	Attribute ID definition source (Universal, Profile)	Value/secondary value	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/AAI/SGSIT/SERR/BV-03-C [Service record GSIT – BIP Automatic Archive Initiator]	[2] 6.1.3	ServiceClassIDList	Universal	“Imaging Automatic Archive” (UUID)	Present for Automatic Archive Initiator
BIP/AAI/SGSIT/ATTR/BV-13-C [Attribute GSIT – Protocol Descriptor List]	[2] 6.1.3	ProtocolDescriptorList	Universal	“L2CAP” (UUID), “RFCOMM” (UUID): Channel number – skip (Uint8), “OBEX” (UUID)	Present for Automatic Archive Initiator
BIP/AAI/SGSIT/ATTR/BV-14-C [Attribute GSIT – Service ID]	[2] 6.1.3	ServiceID	Universal	skip (UUID)	Present for Automatic Archive Initiator
BIP/AAI/SGSIT/ATTR/BV-15-C [Attribute GSIT – Bluetooth Profile Descriptor List, BIP 1.2]	[2] 6.1.3	BluetoothProfileDescriptorList	Universal	“Imaging” (UUID): Version – “0x0102” (Uint16)	TCMT defined
BIP/AAI/SGSIT/ATTR/BV-16-C [Attribute GSIT – Supported Functions]	[2] 6.1.3	Supported Functions	Profile	skip (Uint32)	Present for Automatic Archive Initiator
BIP/AAI/SGSIT/ATTR/BV-17-C [Attribute GSIT – GOEP L2CAP PSM]	[2] 6.1.3	GoepL2capPsm	Profile	skip (Uint16)	TCMT defined

Table 4.4: Input for the Archived Object SGSIT SDP test procedure

4.2.1.4 Basic Imaging Profile – Attribute ID Offset String tests

Execute the Generic SDP Integrated Tests defined in Section 6.3, Server test procedures (SGSIT), in [10] using Table 4.5 below as input:

TCID	Reference	ServiceSearchPattern	Attribute ID name	Attribute ID Offset	Attribute presence (Present/Present for [role], Optionally present, TCMT defined)
BIP/SR/SGSIT/OFFS/BV-01-C [Attribute ID Offset String GSIT – Service Name]	[2] 6.1.1	Imaging Responder	ServiceName	0x0000	Optionally present
BIP/AIPI/SGSIT/OFFS/BV-02-C [Attribute ID Offset String GSIT – Service Name]	[2] 6.1.2	Imaging Referenced Objects	ServiceName	0x0000	Optionally present
BIP/AAI/SGSIT/OFFS/BV-03-C [Attribute ID Offset String GSIT – Service Name]	[2] 6.1.3	Imaging Automatic Archive	ServiceName	0x0000	Optionally present

Table 4.5: Input for the Basic Imaging Profile SGSIT Attribute ID Offset String tests

4.2.2 Client Generic SDP Integrated Tests

Execute the Generic SDP Future Compatibility Tests defined in Section 6.4, Client test procedures (CGSIT), in [10] using Table 4.6 below as input:

TCID	Reference	Service Record Service Class UUID description	Lower Tester SDP record initial conditions
BIP/IPSI/CGSIT/SFC/BV-01-C [SDP Future Compatibility – IUT is BIP Image Push Initiator]	[2] 6.2	Imaging Responder	The Lower Tester exposes a BIP Imaging Responder SDP record. The version in the Bluetooth Profile Descriptor List is greater than the most recently adopted version. All bits are set in the supported capabilities attribute, including reserved bits. All bits are set in the supported features attribute, including reserved bits. All bits are set in the supported functions attribute, including reserved bits.

Table 4.6: Input for the Client CGSIT SDP future compatibility tests

4.3 Fundamental Functional Components

Verify Fundamental Functional Components that are functions that are used in several features and reflect the most basic capabilities of a Bluetooth imaging device.

4.3.1 Fundamental Functional Components

Verify that the retrieval of the imaging-capabilities object is carried out correctly.

4.3.1.1 Get Imaging Capabilities – Client Side

- Test Purpose

  Verify that the IUT, after connection to the OBEX Image Push, Image Pull, Advanced Image Printing, Archived Objects, or Remote Display service of the Lower Tester, can request and receive the imaging-capabilities object from the Lower Tester.

- Reference

  [2] 4.4.6.3, 4.5.1

- Initial Condition

  - There is an active OBEX imaging session (Image Push, Image Pull, Advanced Image Printing, Archived Objects, or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPSI/FFC/BV-01-C [Get Imaging Capabilities – Client Side]
BIP/IPLI/FFC/BV-01-C [Get Imaging Capabilities – Client Side]
BIP/AIPI/FFC/BV-01-C [Get Imaging Capabilities – Client Side]
BIP/AAR/FFC/BV-01-C [Get Imaging Capabilities – Client Side]
BIP/RDI/FFC/BV-01-C [Get Imaging Capabilities – Client Side]

Table 4.7: Get Imaging Capabilities – Client Side test cases

- Test Procedure

  The IUT sends a GetCapabilities request to the Lower Tester.

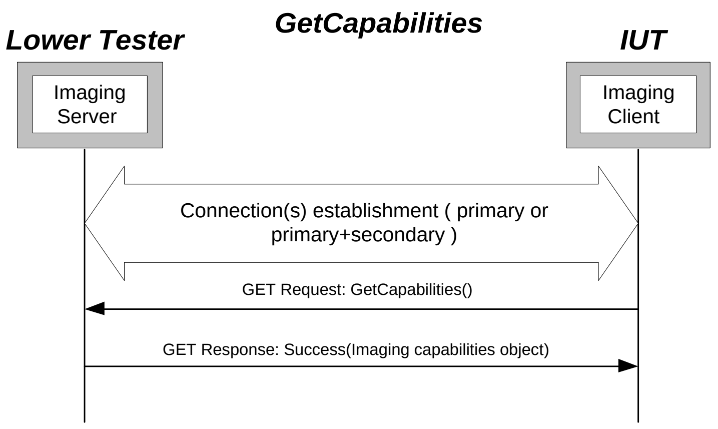

GetCapabilities
Lower Tester
IUT
Imaging Server
Imaging Client
Connection(s) establishment (primary or primary+secondary)
GET Request: GetCapabilities()
GET Response: Success(Imaging capabilities object)

Figure 4.5: Get Imaging Capabilities – Client Side

- Expected Outcome

  Pass verdict

  The GetCapabilities request is correctly understood by the Lower Tester. The Lower Tester will verify the existence and validity of the ConnectionID and Type headers. The ConnectionID is the one assigned by the Lower Tester upon opening the Imaging service session. The Type headeris set to “x-bt/img-capabilities”.

  The IUT can correctly receive the response issued by the Lower Tester (i.e., not report an error upon receiving the response).

4.3.1.2 Get Imaging Capabilities – Server Side

- Test Purpose

  Verify that the IUT, when connected through an OBEX imaging service (Image Push, Image Pull, Advanced Image Printing, Archived Objects, or Remote Display service), can accurately return to the Lower Tester its local imaging-capabilities object.

- Reference

  [2] 4.4.6.3, 4.5.1

- Initial Condition

  - There is an active OBEX imaging session (Image Push, Image Pull, Advanced Image Printing, Archived Objects, or Remote Display service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Case Configuration

Test Case
BIP/IPSR/FFC/BV-02-C [Get Imaging Capabilities – Server Side]
BIP/IPLR/FFC/BV-02-C [Get Imaging Capabilities – Server Side]
BIP/AIPR/FFC/BV-02-C [Get Imaging Capabilities – Server Side]
BIP/AAI/FFC/BV-02-C [Get Imaging Capabilities – Server Side]
BIP/RDR/FFC/BV-02-C [Get Imaging Capabilities – Server Side]

Table 4.8: Get Imaging Capabilities – Server Side test cases

- Test Procedure

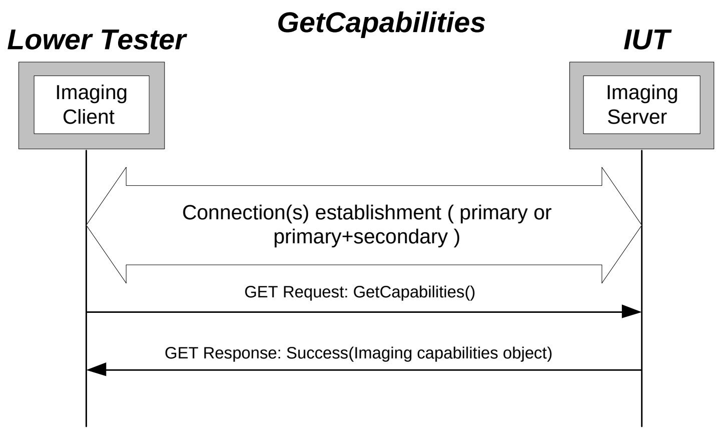

GetCapabilities
Lower Tester
IUT
Imaging Client
Imaging Server
Connection(s) establishment (primary or primary+secondary)
GET Request: GetCapabilities()
GET Response: Success(Imaging capabilities object)

Figure 4.6: Get Imaging Capabilities – Server Side

The Lower Tester issues a GetCapabilities request.

- Test Condition

  The IUT has a non-empty imaging-capabilities object.

- Expected Outcome

  Pass verdict

  A Success response containing a non-empty imaging-capabilities object is returned by the IUT.

  The GetCapabilities response is well formatted.

  - All of the elements are enumerated in the imaging-capabilities object according to the features that are supported by the IUT (based on ICS/IXIT).
  - The SDP record advertises correct features and matches what is claimed as supported by the IUT (based on ICS/IXIT).

- Notes

  When possible, try to verify that the content of each element/attribute in the imaging-capabilities object is accurate by comparing it with the ICS [5] (see the Image Push service, the Image Pull service, the Advanced Image Printing service, the Archived Objects service and the Remote Display service).

4.3.2 Get the List of Images and Related Information

Verify that the functions related with the images-listing object are correctly implemented.

4.3.2.1 Get Images List – Client Side

- Test Purpose

  Verify that the IUT, when connected to an imaging service of the Lower Tester (Image Pull, Remote Display, or Archived Objects service), can correctly request and receive the images list from the Lower Tester.

- Reference

  [2] 4.4.6.1, 4.5.6

- Initial Condition

  - There is an active OBEX imaging session (Image Push, Image Pull, or Archived Objects) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/FFC/BV-03-C [Get Images List – Client Side]
BIP/AAR/FFC/BV-03-C [Get Images List – Client Side]
BIP/RDI/FFC/BV-03-C [Get Images List – Client Side]

Table 4.9: Get Images List – Client Side test cases

- Test Procedure

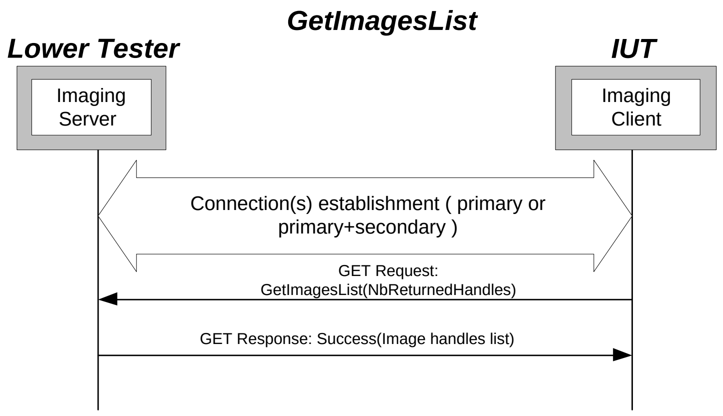

GetImagesList
Lower Tester
IUT
Imaging Server
Imaging Client
Connection(s) establishment (primary or primary+secondary)
GET Request: GetImagesList(NbReturnedHandles)
GET Response: Success(Image handles list)

Figure 4.7: Get Images List – Client Side

The IUT sends a GetImagesList request to the Lower Tester.

- Expected Outcome

<u>Pass verdict</u>

The GetImagesList request is well formatted. The Lower Tester will specifically verify the presence, validity, and order of the ConnectionID, Type, Application Parameters, and Img-Description headers. The ConnectionID is the one assigned by the Lower Tester upon opening the Imaging service session. The Type header is set to “x-bt/img-listing”. The NbReturnedHandles Application Parameters header is non-null. The ListStartOffset Application Parameters header is set to 0.

The IUT can correctly receive the response issued by the Lower Tester and in particular does not report an error upon receiving the response.

4.3.2.2 Get Images List – Server Side

- Test Purpose

Verify that the IUT, when connected to by the Lower Tester through an imaging service (Image Pull, Remote Display, or Archived Objects service), can, upon request, provide its images list to the Lower Tester.

- Reference

[2] 4.4.6.1, 4.5.6

- Initial Condition

  - There is an active OBEX imaging session (Image Pull, Remote Display, or Archived Objects service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The IUT has at least one image stored locally.

- Test Case Configuration

Test Case
BIP/IPLR/FFC/BV-04-C [Get Images List – Server Side]
BIP/AAI/FFC/BV-04-C [Get Images List – Server Side]
BIP/RDR/FFC/BV-04-C [Get Images List – Server Side]

Table 4.10: Get Images List – Server Side test cases

- Test Procedure

  The Lower Tester issues a GetImagesList request. The Img-Description header is left empty (i.e., no filtering is requested by the Lower Tester). The NbReturnedHandles is non-null. The LatestCapturedImages flag is set to 0.

- Expected Outcome

  Pass verdict

  The IUT returns a GetImagesList response that is well formatted. The Lower Tester will verify the presence and validity of the Application Parameters, Img-Description, and Body headers. The NbReturnedHandles header is accurate (i.e., correspond to the number of handles actually returned in the images-listing object). The image-handles-descriptor is empty or absent.

  The number of images returned in the images-listing object is correctly reflected in the NbReturnedHandles header value.

4.3.2.3 Get Total Number of Images – Client Side

- Test Purpose

  Verify that the IUT can connect to an imaging service of the Lower Tester (Image Pull, Remote Display, or Archived Objects service) can correctly request and receive the total number of images available on the Lower Tester.

- Reference

  [2] 4.4.6.1, 4.5.6

- Initial Condition

  - There is an active OBEX imaging session (Image Pull, Remote Display, or Archived Objects service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/FFC/BV-05-C [Get Total Number of Images – Client Side]
BIP/AAR/FFC/BV-05-C [Get Total Number of Images – Client Side]
BIP/RDI/FFC/BV-05-C [Get Total Number of Images – Client Side]

Table 4.11: Get Total Number of Images – Client Side test cases

- Test Procedure

  The IUT issues a GetImagesList request with 0 as value for the NbReturnedHandles Application Parameters header. The Img-Descriptor is left empty (i.e., no filtering is requested by the Lower Tester). The LatestCapturedImages flag is set to 0.

- Expected Outcome

  Pass verdict

  The Lower Tester is able to recognize the request for the total number of available images. The Lower Tester will verify the presence and validity of the Application Parameters, Img-Description, and Body headers. The NbReturnedHandles header is accurate (i.e., correspond to the number of handles actually returned in the images-listing object). The image handles descriptor is empty.

The IUT correctly receives the response from the Lower Tester and in particular doesn't report any error upon receiving the response.

4.3.2.4 Get Total Number of Images – Server Side

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through an imaging service (Image Pull, Remote Display, or Archived Objects service), can, upon request, provide the total number of available images.

- Reference

  [2] 4.4.6.1, 4.5.6

- Initial Condition

  - There is an active OBEX imaging session (Image Pull, Remote Display, or Archived Objects service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The IUT has more than one image locally stored.

- Test Case Configuration

Test Case
BIP/IPLR/FFC/BV-06-C [Get Total Number of Images – Server Side]
BIP/AAI/FFC/BV-06-C [Get Total Number of Images – Server Side]
BIP/RDR/FFC/BV-06-C [Get Total Number of Images – Server Side]

Table 4.12: Get Total Number of Images – Server Side test cases

- Test Procedure

  The Lower Tester issues a GetImagesList request with 0 as the NbReturnedHandles Application Parameters header. The LatestCapturedImages flag is set to 0.

- Expected Outcome

  Pass verdict

  The IUT returns a GetImagesList response that is well formatted. The Lower Tester will specifically verify that the Body header contains an empty images-listing object.

  The NbReturnedHandles header's value corresponds to the total number of images that are locally available for transfer via Bluetooth.

4.3.2.5 Get the List of Recently Captured Images – Client Side

- Test Purpose

  Verify that the IUT can connect to an imaging service of the Lower Tester (Image Pull, Remote Display, or Archived Objects service) and correctly request and receive the list of most recently captured images on the Lower Tester.

- Reference

  [2] 4.4.6.1, 4.5.6

- Initial Condition
  - There is an active OBEX imaging session (Image Pull, Remote Display, or Archived Objects) where the IUT is the OBEX client and the Lower Tester is the OBEX server.
- Test Case Configuration

Test Case
BIP/IPLI/FFC/BV-07-C [Get the List of Recently Captured Images – Client Side]
BIP/AAR/FFC/BV-07-C [Get the List of Recently Captured Images – Client Side]
BIP/RDI/FFC/BV-07-C [Get the List of Recently Captured Images – Client Side]

Table 4.13: Get the List of Recently Captured Images – Client Side test cases

- Test Procedure

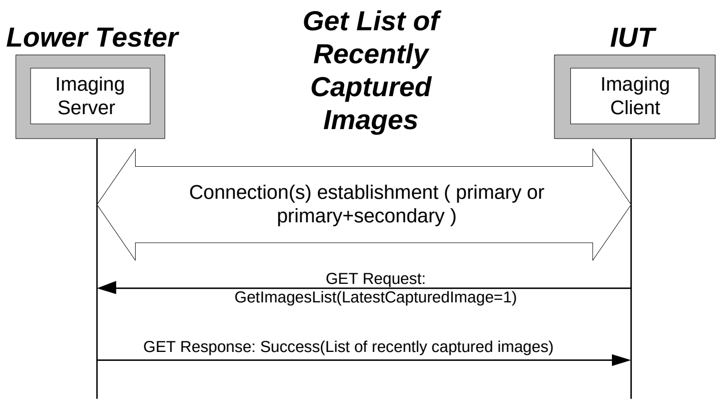

Lower Tester
Get List of Recently Captured Images
IUT
Imaging Server
Imaging Client
Connection(s) establishment (primary or primary+secondary)
GET Request: GetImagesList(LatestCapturedImage=1)
GET Response: Success(List of recently captured images)

Figure 4.8: Get the List of Recently Captured Images – Client Side

The IUT issues a GetImagesList request with the LatestCapturedImages Application Parameters header set to 1.

- Expected Outcome

<u>Pass verdict</u>

The Lower Tester is able to recognize the request for an ordered images-listing object that describes the most recently taken images.

The GetImagesList request is well formatted. The Lower Tester will especially verify that the request contains a valid ConnectionID header; a Type header of “x-bt/img-listing”, a valid Application Parameters header, and a valid image handles descriptor.

The IUT correctly receives the response from the Lower Tester and in particular doesn’t report an error upon receiving the response.

- Notes

The IUT may or may not include a non-empty image-handles-descriptor in its GetImagesList request.

4.3.2.6 Get the List of Recently Captured Images – Server Side

- Test Purpose

Verify that the IUT, when connected to an imaging service of the Lower Tester (Image Pull, Remote Display, or Archived Objects service), can correctly return the list of its most recently captured images.

- Reference

  [2] 4.4.6.1, 4.5.6

- Initial Condition

  - There is an active OBEX imaging session (Image Pull, Remote Display, or Archived Objects service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The IUT has a number of locally stored images it has captured. The dates when the images where captured are known.

- Test Case Configuration

Test Case
BIP/IPLR/FFC/BV-08-C [Get the List of Recently Captured Images – Server Side]
BIP/AAI/FFC/BV-08-C [Get the List of Recently Captured Images – Server Side]
BIP/RDR/FFC/BV-08-C [Get the List of Recently Captured Images – Server Side]

Table 4.14: Get the List of Recently Captured Images – Server Side test cases

- Test Procedure

  The Lower Tester issues a GetImagesList request with the LatestCapturedImages Application Parameters header set to 1.

- Expected Outcome

  Pass verdict

  The IUT returns a GetImagesList response that is well formatted.

  The NbReturnedHandles Application Parameters header value matches the number of handles that are actually returned in the images-listing object.

  The images-listing object contains accurate information and the order of the listed handles corresponds to the chronological order of their capture.

4.3.3 Put an image

Verify that the fundamental function for pushing images is correctly implemented.

4.3.3.1 Put an Image – Normal – Client Side

- Test Purpose

  Verify that the IUT, when connected to an Imaging service of the Lower Tester (Image Push or Remote Display service), can correctly push an image to the Lower Tester and understand the response from the Lower Tester.

- Reference

  [2] 4.4.7.2, 4.5.2

- Initial Condition

  - There is an active OBEX imaging session (Image Push or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - There is at least one image locally stored on the IUT.

- Test Case Configuration

Test Case
BIP/IPSI/FFC/BV-09-C [Put an Image – Normal – Client Side]
BIP/RDI/FFC/BV-09-C [Put an Image – Normal – Client Side]

Table 4.15: Put an Image – Normal – Client Side test cases

- Test Procedure

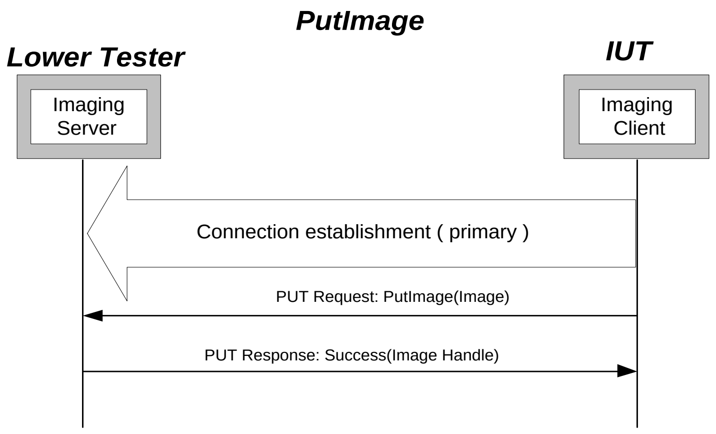

PutImage
Lower Tester
IUT
Imaging Server
Imaging Client
Connection establishment (primary)
PUT Request: PutImage(Image)
PUT Response: Success(Image Handle)

Figure 4.9: Put an Image – Normal – Client Side

A PutImage request is issued by the IUT. The Lower Tester is not allowed to issue a Partial Content response code in response to the request.

- Expected Outcome

<u>Pass verdict</u>

The Lower Tester recognizes the PutImage request. The Lower Tester verifies that the PutImage request contains a valid ConnectionID. The image descriptor is also checked and the information that it contains is accurate (the Lower Tester will open the image file to determine the properties of the image and compare them with the image-properties object).

The image descriptor that is sent in the request accurately describes the image that is sent.

The IUT properly processes the PutImage response.

4.3.3.2 Put an Image – Normal – Server Side

- Test Purpose

Verify that the IUT, when connected to by the Lower Tester through its Image Push or Remote Display service, can correctly receive an image pushed by the Lower Tester.

- Reference

[2] 4.4.7.2, 4.5.2

- Initial Condition

  - There is an active OBEX imaging session (Image Push or Remote Display service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Case Configuration

Test Case
BIP/IPSR/FFC/BV-10-C [Put an Image – Normal – Server Side]
BIP/RDR/FFC/BV-10-C [Put an Image – Normal – Server Side]

Table 4.16: Put an Image – Normal – Server Side test cases

- Test Procedure

  A PutImage request is issued by the Lower Tester.

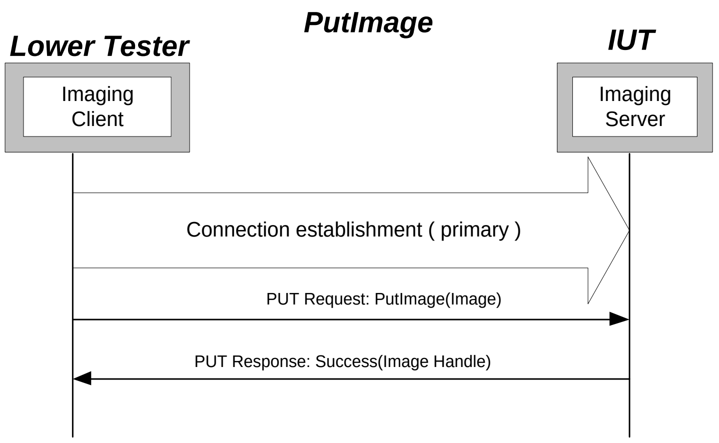

PutImage
Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (primary)
PUT Request: PutImage(Image)
PUT Response: Success(Image Handle)

Figure 4.10: Put an Image – Normal – Server Side

- Expected Outcome

  Pass verdict

  The IUT recognizes the PutImage request and accepts the image.

  The PutImage response is well formatted. The Lower Tester will specifically verify that the PutImage response contains a valid image handle.

  The IUT assigns an imaging handle to the image and returns it in the PutImage response.

- Notes

  The image sent by the Lower Tester will be in a format (encoding and size) that is known to be acceptable to the IUT. The Lower Tester chooses the format based on the IXIT.

4.3.3.3 Put an Image – Request for Thumbnail – Server Side

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through an Imaging service (Image Push or Remote Display service), can correctly receive an image pushed by the Lower Tester and can request the Lower Tester to send the thumbnail version of the image.

- Reference

  [2] 4.5.2, 4.5.9

- Initial Condition

  - There is an active OBEX imaging session (Image Push or Remote Display service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Case Configuration

Test Case
BIP/IPSR/FFC/BV-11-C [Put an Image – Request for Thumbnail – Server Side]
BIP/RDR/FFC/BV-11-C [Put an Image – Request for Thumbnail – Server Side]

Table 4.17: Put an Image – Request for Thumbnail – Server Side test cases

- Test Procedure

A PutImage request is issued by the Lower Tester.

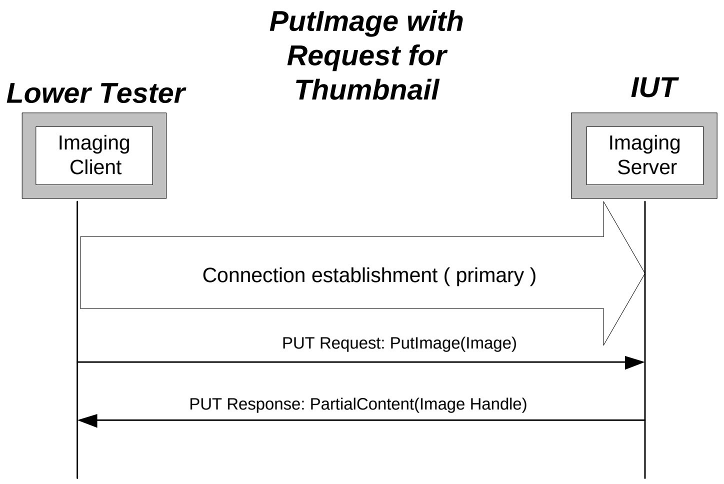

PutImage with Request for Thumbnail
Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (primary)
PUT Request: PutImage(Image)
PUT Response: PartialContent(Image Handle)

Figure 4.11: Put an Image – Request for Thumbnail – Server Side

- Expected Outcome

<u>Pass verdict</u>

The IUT recognizes the PutImage request and accepts the image.

The PutImage response issued by the IUT is well formatted. The Lower Tester will specifically verify that it contains a valid image handle.

The IUT assigns an imaging handle to the image and returns it in the PutImage response.

The IUT requests the imaging thumbnail corresponding to the received image via the Partial Content response code.

- Notes

It is not the object of this test purpose to verify support for the PutLinkedThumbnail function. Therefore, it is sufficient to verify that the response issued by the IUT contains the Partial Content response code.

The image sent by the Lower Tester will be in a format (encoding and size) that is known to be acceptable to the IUT and is different from the imaging thumbnail format. The formats supported by the IUT are declared in the IXIT [9] (see the Image Push service and the Remote Display service).

4.4 Feature Specific Functions

Verify that the functions that are specific to a given feature of the Basic Imaging Profile are correctly implemented.

4.4.1 Get a Monitoring Image

Verify that the GetMonitoring function is correctly implemented.

BIP/RCI/FSF/BV-01-C [Get a Monitoring Image – Image Not Saved – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Remote Camera service of the Lower Tester, can correctly request and receive monitoring images where the request does not trigger capturing the full size image.

- Reference

  [2] 4.4.6.5, 4.5.16

- Initial Condition

  - There is an active OBEX Remote Camera Service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

  The IUT issues a GetMonitoringImage request with StoreFlag set to 0x00.

  In response, the Lower Tester issues a GetMonitoringImage response with no Img-Handle header.

- Expected Outcome

  Pass verdict

  The Lower Tester recognizes the GetMonitoringImage requests and they are well formatted. The Lower Tester will specifically verify that the GetMonitoringImage request contains a valid ConnectionID, a Type header of “x-bt/img-monitoring”, and a StoreFlag Application Parameters header.

  The IUT correctly processes the GetMonitoringImage response and in particular does not report an error upon receiving the response.

- Notes

  The Lower Tester, if not equipped with an image capture capability, can simulate monitoring images.

BIP/RCI/FSF/BV-02-C [Get a Monitoring Image – Image Saved – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Remote Camera service of the Lower Tester, can correctly request and receive monitoring images where the request triggers capturing the full size image.

- Reference

  [2] 4.4.6.5, 4.5.16

- Initial Condition
  - There is an active OBEX Remote Camera Service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
- Test Procedure

  The IUT issues a GetMonitoringImage request with StoreFlag set to 0x01.
- Expected Outcome

  Pass verdict

  The Lower Tester recognizes the GetMonitoringImage requests. The Lower Tester will specifically verify that the GetMonitoringImage request contains a valid ConnectionID, a Type header of "x-bt/img-monitoring", and a StoreFlag Application Parameters header.

  The StoreFlag value is set to 0x01.

  The Lower Tester is returns a monitoring image. The GetMonitoringImage response contains an Img-Handle header that corresponds to the full size image available on the Lower Tester.

  The IUT correctly processes the GetMonitoringImage response and in particular does not report an error upon receiving of the response.
- Notes

  The Lower Tester, if not equipped with an image capture capability, can simulate monitoring images.

BIP/RCR/FSF/BV-03-C [Get a Monitoring Image – Image Not Saved – Server Side]

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through its Remote Camera service, can correctly return monitoring images to the Lower Tester without saving them or saving the corresponding full size images.
- Reference

  [2] 4.4.6.5, 4.5.16
- Initial Condition
  - There is an active OBEX Remote Camera service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.
- Test Procedure

  The Lower Tester issues a GetMonitoringImage request with StoreFlag set to 0x00.

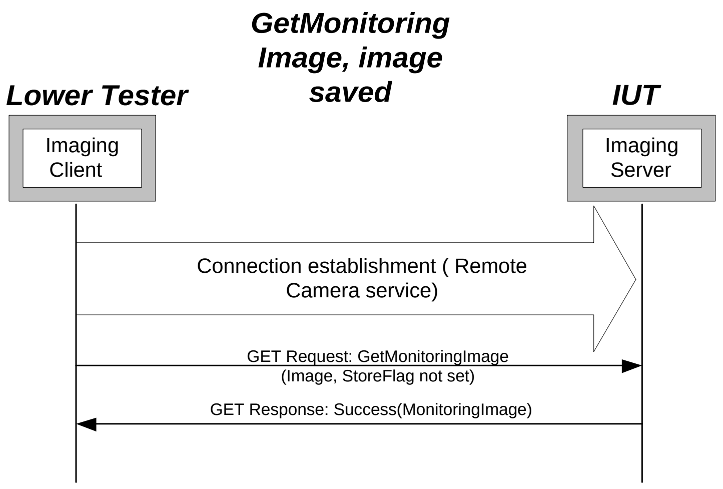

GetMonitoring Image, image saved
Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (Remote Camera service)
GET Request: GetMonitoringImage (Image, StoreFlag not set)
GET Response: Success(MonitoringImage)

Figure 4.12: BIP/RCR/FSF/BV-03-C [Get a Monitoring Image – Image Not Saved – Server Side]

- Expected Outcome

  Pass verdict

  The IUT recognizes the GetMonitoringImage requests.

  The IUT returns a GetMonitoringImage response that is well formatted. The Lower Tester will specifically verify that the response includes a monitoring image and an empty Img-Handle header.

BIP/RCR/FSF/BV-04-C [Get a Monitoring Image – Image Saved – Server Side]

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through its Remote Camera service, can correctly return monitoring images to the Lower Tester and save the corresponding full size images.

- Reference

  [2] 4.4.6.5, 4.5.16

- Initial Condition

  - There is an active OBEX Remote Camera service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Procedure

  The Lower Tester issues a GetMonitoringImage request with StoreFlag set to 0x01.

- Expected Outcome

  Pass verdict

  The IUT recognizes the GetMonitoringImage requests.

  The IUT returns a GetMonitoringImage response that is well formatted. The Lower Tester will specifically verify that the response includes a monitoring image and an Img-Handle header with a valid image handle.

  In the GetMonitoringImage response, there is an Img-Handle header corresponding to the full size image that was stored as result of the GetMonitoringImage request.

  The Lower Tester correctly processes the GetMonitoringImage response.

- Notes

  If possible, verify that the image handle that is returned by the IUT corresponds to the correct image.

4.4.2 Start Advanced Printing

Verify that the StartPrinting image is correctly implemented.

BIP/AIPI/FSF/BV-05-C [Start Advanced Printing – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Advanced Image Printing service of the Lower Tester, can initiate a print job by sending a printer control object to the Lower Tester and understand the response.

- Reference

  [2] 4.4.6.4, 4.5.12

- Initial Condition

  - There is an active OBEX Advanced Image Printing service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - There is at least one image on the IUT so the printer control object is not empty.

- Test Procedure

  The IUT builds a printer control object with at least one print job and sends it via a StartPrint request.

- Expected Outcome

  <u>Pass verdict</u>

  The Lower Tester recognizes the StartPrint request and it's well formatted. The Lower Tester will specifically verify that the StartPrint request contains a valid ConnectionID, a Type header of “x-bt/img-print”, an Application Parameter with a valid ServiceID, and a well formatted printer control object.

  The IUT can correctly process the StartPrint response and in particular does not report an error upon receiving the response.

  The Lower Tester initiates a secondary connection to the OBEX Referenced Objects service of the IUT. In this session, the Lower Tester is the OBEX client and the IUT is OBEX the server.

- Notes

  Verifying the behavior of the IUT upon receiving a Connection request for a secondary connection is out of scope for this test.

BIP/AIPR/FSF/BV-06-C [Start Advanced Printing – Server Side]

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through its Advanced Image Printing service, can correctly receive a print job and initiate the requested printing operation.

- Reference

  [2] 4.4.6.4, 4.5.12

- Initial Condition
  - There is an active OBEX Advanced Image Printing service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.
- Test Procedure

  The Lower Tester builds a printer control object with at least one print job and sends it via a StartPrint request to the IUT.
- Expected Outcome

  Pass verdict

  The IUT can recognize the StartPrint request and issues a StartPrint response.

  The Lower Tester can correctly process the StartPrint response and it's well formatted.
- Notes

  The IUT may or may not initiate a Connection request for a secondary connection – this is out of scope for this test.

4.4.3 Start Archiving

Verify that the StartArchive function is correctly implemented.

BIP/AAI/FSF/BV-07-C [Start Archiving – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Automatic Archive service of the Lower Tester, can command the Lower Tester to start archiving the images stored locally on the IUT and understand the response from the Lower Tester.
- Reference

  [2] 4.5.14
- Initial Condition
  - There is an active OBEX Automatic Archive service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
- Test Procedure

  The IUT issues a StartArchive request.
- Expected Outcome

  Pass verdict

  The Lower Tester can correctly recognize the StartArchive request and it's well formatted. The Lower Tester will verify the presence of a valid ConnectionID, a Type header of "x-bt/img-archive", and a valid ServiceID.

  The IUT can correctly process the StartArchive response and in particular does not report an error upon receiving the response.

BIP/AAR/FSF/BV-08-C [Start Archiving – Server Side]

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through its Automatic Archive service, can correctly receive a request for an Automatic Archive job.

- Reference

  [2] 4.5.14

- Initial Condition

  - There is an active OBEX Automatic Archive service session where the IUT is the OBEX server and the Lower Tester is the OBEX client. There is at least one image available on the Lower Tester.

- Test Procedure

  The Lower Tester issues a StartArchive request.

- Expected Outcome

  Pass verdict

  The IUT recognizes the StartArchive request and issues a StartArchive response.

  The Lower Tester correctly processes the StartArchive response and it's well formatted.

4.4.4 Remotely Control a Display

Verify that the RemoteDisplay function is correctly implemented.

BIP/RDI/FSF/BV-09-C [Go to Next Image – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Remote Display service of the Lower Tester, can correctly order the Lower Tester to display the next image and understand the response from the Lower Tester.

- Reference

  [2] 4.5.5

- Initial Condition

  - There is an active OBEX Remote Display Service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The Lower Tester can emulate a display. The Lower Tester has at least two locally available images and those images are ordered (the order is at the discretion of the Lower Tester).

- Test Procedure

  The IUT issues a RemoteDisplay request with NextImage as the RemoteDisplay Application Parameters header.

- Expected Outcome

  Pass verdict

  The Lower Tester recognizes the RemoteDisplay request and it's well formatted. The Lower Tester will specifically verify that the RemoteDisplay request includes a valid ConnectionID, a Type header of "x-bt/img-display", an empty Img-Handle header, and a valid Application Parameters header with NextImage as its value.

  The IUT correctly processes the RemoteDisplay response and in particular does not report an error upon receiving the response.

BIP/RDI/FSF/BV-10-C [Go to Previous Image – Client Side]

- Test Purpose

  Verify that the IUT, when connected to the Remote Display service of the Lower Tester, can correctly order the Lower Tester to display the previous image and understand the response from the Lower Tester.

- Reference

  [2] 4.5.5

- Initial Condition
  - There is an active OBEX Remote Display Service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The Lower Tester can emulate a display. The Lower Tester has at least two locally available images and those images are ordered (the order is at the discretion of the Lower Tester).

- Test Procedure

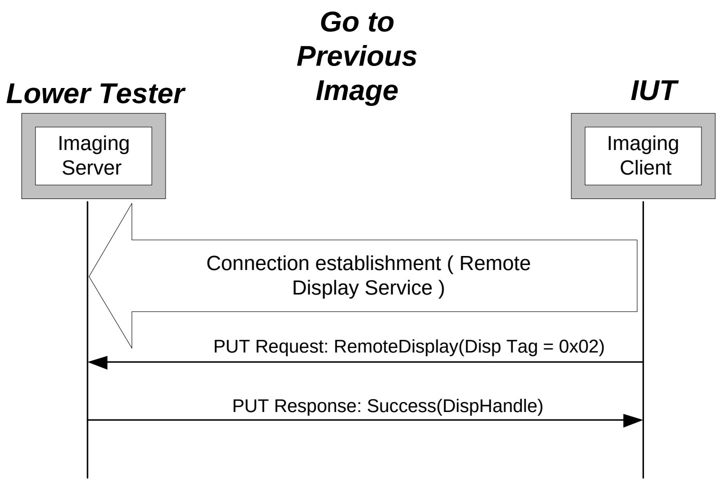

Go to Previous Image
Lower Tester
IUT
Imaging Server
Imaging Client
Connection establishment (Remote Display Service)
PUT Request: RemoteDisplay(Disp Tag = 0x02)
PUT Response: Success(DispHandle)

Figure 4.13: BIP/RDI/FSF/BV-10-C [Go to Previous Image – Client Side]

The IUT issues a RemoteDisplay request with PreviousImage as the RemoteDisplay Application Parameters header.

- Expected Outcome

  Pass verdict

  The Lower Tester recognizes the RemoteDisplay request and it's well formatted. The Lower Tester will specifically verify that the RemoteDisplay request includes a valid ConnectionID, a Type header of "x-bt/img-display", an empty Img-Handle header, and a valid Application Parameters header with PreviousImage as its value.

  The IUT correctly processes the RemoteDisplay response and in particular does not report an error upon receiving the response.

BIP/RDR/FSF/BV-11-C [Go to Next Image – Server Side]

- Test Purpose

  Verify that the IUT, when connected through its Remote Display service by the Lower Tester, can correctly display the next image.

- Reference

  [2] 4.5.5

- Initial Condition

  - There is an active Remote Display Service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The IUT has at least two locally available images and those images are ordered (the order is at the discretion of the IUT). The IUT is displaying the first image.

- Test Procedure

  The Lower Tester issues two RemoteDisplay requests with NextImage as the RemoteDisplay Application Parameters header.

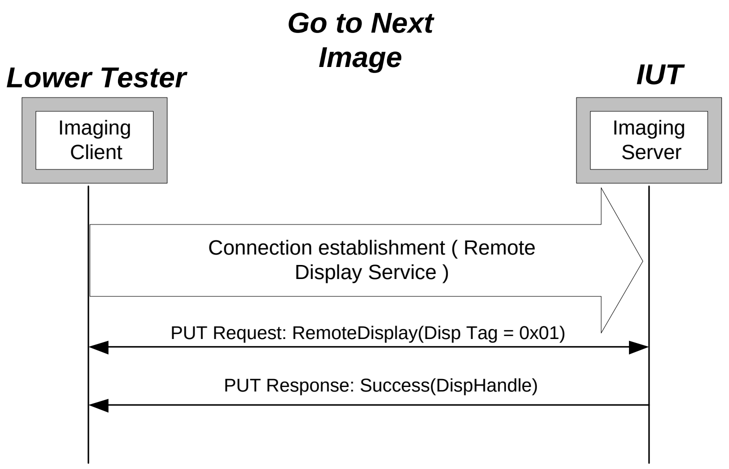

Go to Next Image
Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (Remote Display Service)
PUT Request: RemoteDisplay(Disp Tag = 0x01)
PUT Response: Success(DispHandle)

Figure 4.14: BIP/RDR/FSF/BV-11-C [Go to Next Image – Server Side]

- Expected Outcome

  Pass verdict

  The IUT recognizes the RemoteDisplay request.

Upon receiving the first RemoteDisplay request, the IUT displays an image. Upon receiving the second RemoteDisplay request, the IUT displays another image.

The Lower Tester correctly processes the RemoteDisplay responses and they're well formatted. The Lower Tester will specifically verify that the responses contain a valid image handle.

BIP/RDR/FSF/BV-12-C [Go to Previous Image – Server Side]

- Test Purpose

  Verify that the IUT, when connected to by the Lower Tester through its Remote Display service, can correctly display the previous image.

- Reference

  [2] 4.5.5

- Initial Condition

  - There is an active Remote Display Service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The IUT has at least two locally available images and those images are ordered (the order is at the discretion of the IUT). The IUT is displaying the last image.

- Test Procedure

  The Lower Tester issues two RemoteDisplay requests with PreviousImage as the RemoteDisplay Application Parameters header.

- Expected Outcome

  Pass verdict

  The IUT recognizes the RemoteDisplay request.

  Upon receiving the first RemoteDisplay request, the IUT displays an image. Upon receiving the second RemoteDisplay request, the IUT displays another image.

  The Lower Tester correctly processes the RemoteDisplay responses and they're well formatted. The Lower Tester will specifically verify that the responses contain a valid image handle.

4.5 Multi Functions Sequences

Submit the IUT to a number of typical function sequences. As is the case for the functional tests, these tests are not meant to be absolutely exhaustive, but to exercise a number of typical situations that the IUT ought to handle properly.

4.5.1 Push Images

Verify that the IUT is capable of pushing images or receiving images as defined in the Image Push feature and the Remote Display feature in, respectively, Sections 4.3.1 and 4.3.6 of [2].

BIP/IPSI/MFS/BV-01-C [Use the Capabilities Object to Put Images – Client Side]

- Test Purpose

  Verify that the linkage between the GetCapabilities function and the PutImage function is correct (i.e., that the IUT sends images according to the image-formats element in the Lower Tester's imaging-capabilities object).

- Reference

  [2] 4.4.6.3, 4.5.1, 4.5.2

- Initial Condition

  - There is an active OBEX session (Image Push or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The IUT has at least one image stored locally and available to be pushed. If the IUT supports encodings or sizes other than imaging thumbnail, the image is not a thumbnail.

- Test Procedure

  The above sequence is repeated three times with the following variations on the imaging-capabilities object; this may include disconnection of the OBEX channel for each iteration.

  The imaging-capabilities object indicates that JPEG160*120 is the sole image encoding/size supported by the Lower Tester.

  The imaging-capabilities object indicates support for one and only one of the encodings supported by the IUT and all possible sizes (“1*1-65535*65535”). Supported encodings are indicated in the IXIT [9], for the Image Push service and the Remote Display service.

  The imaging-capabilities object indicates support for all the encodings supported by the IUT and one size chosen among those supported by the IUT. Supported encodings and sizes are indicated in the IXIT [9], for the Image Push service and the Remote Display service.

- Expected Outcome

  Pass verdict

  Upon receiving the first GetCapabilities response, the IUT issues a PutImage request that contains an imaging thumbnail as the body and an accurate image descriptor.

  Upon receiving the second GetCapabilities response, the IUT issues a PutImage request that contains an image using an encoding indicated as supported by the Lower Tester. The image can be of any size.

  Upon receiving the third GetCapabilities response, the IUT issues a PutImage request that contains an image using a size indicated as supported by the Lower Tester. The encoding is chosen by the IUT.

BIP/IPSR/MFS/BV-02-C [Accurately Provide the Capability Object – Server Side]

- Test Purpose

  Verify that the linkage between the GetCapabilities function and the PutImage function is correct (i.e., that the IUT does support the image formats it advertises in its imaging-capabilities object).

- Reference

  [2] 4.4.6.3, 4.5.1, 4.5.2

- Initial Condition

  - There is an active OBEX session (Image Push or Remote Display service) where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Procedure

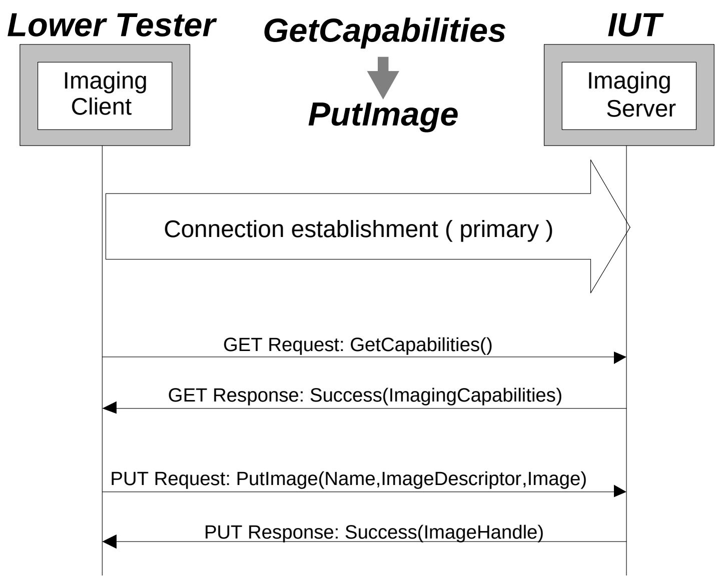

Lower Tester
GetCapabilities
PutImage
IUT
Imaging Client
Imaging Server
Connection establishment (primary)
GET Request: GetCapabilities()
GET Response: Success(ImagingCapabilities)
PUT Request: PutImage(Name,ImageDescriptor,Image)
PUT Response: Success(ImageHandle)

Figure 4.15: BIP/IPSR/MFS/BV-02-C [Accurately Provide the Capability Object – Server Side]

The Lower Tester, upon receiving the imaging-capabilities object from the IUT, will pick at random one of the encodings/sizes described as supported by the IUT and send an image to the IUT in that format.

- Expected Outcome

  Pass verdict

  The image sent by the Lower Tester is correctly received by the IUT.

  If the IUT is in a Remote Display session, the displayed image does not change as the result of pushing a new image.

BIP/IPSI/MFS/BV-03-C [Accurately Push a Thumbnail – Client Side]

- Test Purpose

  Verify the linkage between the PutImage function and the PutLinkedThumbnail function (i.e., that IUT can send the thumbnail corresponding to the image previously sent to the Lower Tester).

- Reference

  [2] 4.5.2, 4.5.3

- Initial Condition
  - There is an active OBEX session (Image Push or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The IUT has at least one image locally stored and available for pushing to the Lower Tester in a format different from the imaging thumbnail format.

- Test Procedure

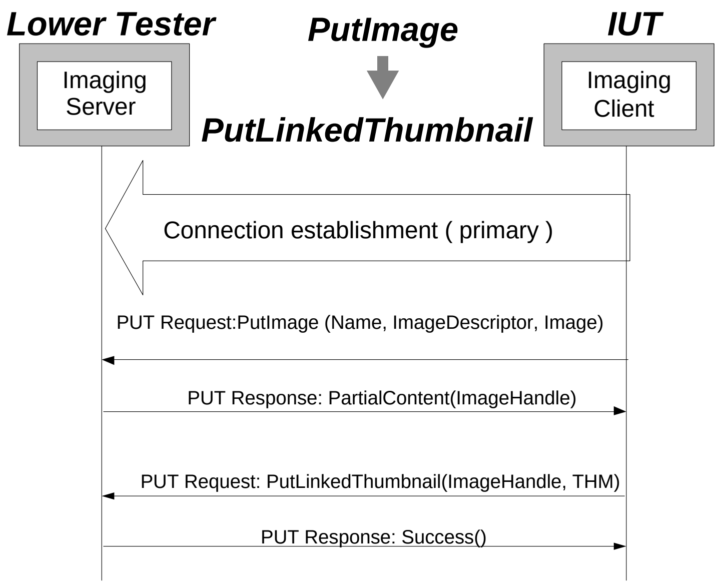

Lower Tester
PutImage
PutLinkedThumbnail
IUT
Imaging Server
Imaging Client
Connection establishment (primary)
PUT Request:PutImage (Name, ImageDescriptor, Image)
PUT Response: PartialContent(ImageHandle)
PUT Request: PutLinkedThumbnail(ImageHandle, THM)
PUT Response: Success()

Figure 4.16: BIP/IPSI/MFS/BV-03-C [Accurately Push a Thumbnail – Client Side]

The following sequence is executed:

The IUT issues a PutImage request containing an image to send to the Lower Tester. The image is not an imaging thumbnail.

The Lower Tester responds to the PutImage request with a PutImage response containing the image handle assigned to the received image and a Partial Content response code.

IUT issues a PutLinkedThumbnail request containing the thumbnail that corresponds to the previously transmitted image with its image handle.

The Lower Tester responds with Success response code to the PutLinkedThumbnail request.

- Expected Outcome

Pass verdict

The image sent by the IUT in the PutImage Request is not an imaging thumbnail.

The IUT issues a PutLinkedThumbnail request containing the thumbnail corresponding to the previously transmitted image. The Lower Tester will specifically verify that the PutLinkedThumbnail request contains a valid ConnectionID and a Type header of “x-bt/img-thm”.

The PutLinkedThumbnail request is issued immediately upon receiving the PutImage request.

The image handle contained in the PutLinkedThumbnail request from the IUT is the same as the one in the PutImage response from the Lower Tester.

- Notes

The image handle for the transmitted image is assigned randomly by the Lower Tester.

BIP/IPSR/MFS/BV-04-C [Request a Thumbnail – Server Side]

- Test Purpose

  Verify the linkage between the PutImage function and the PutLinkedThumbnail function (i.e., that the IUT can request a thumbnail and correctly receive and process it).

- Reference

  [2] 4.5.2, 4.5.3

- Initial Condition

  - There is an active OBEX session (Image Push or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The image to be transmitted from the Lower Tester is a non-thumbnail image supported by IUT and for which the IUT needs a subsequent thumbnail request.

- Test Procedure

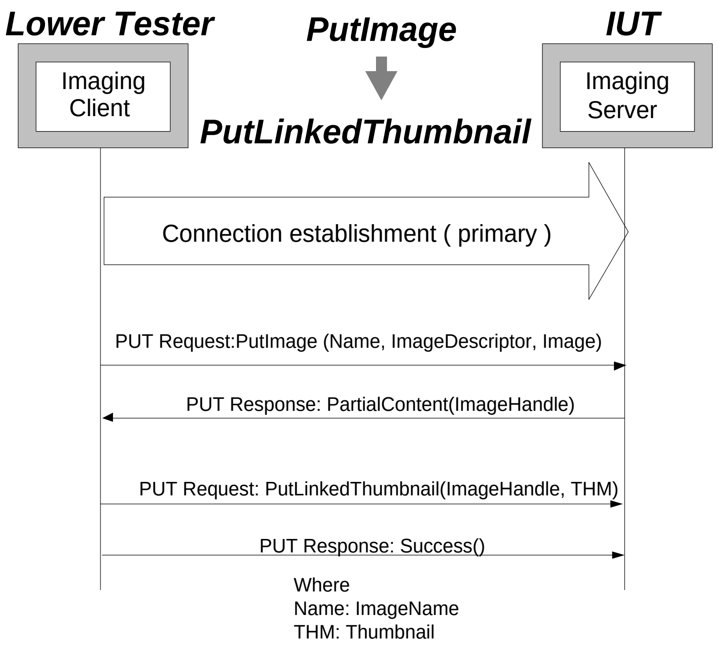

Lower Tester
PutImage
IUT
Imaging Client
PutLinkedThumbnail
Imaging Server
Connection establishment (primary)
PUT Request:PutImage (Name, ImageDescriptor, Image)
PUT Response: PartialContent(ImageHandle)
PUT Request: PutLinkedThumbnail(ImageHandle, THM)
PUT Response: Success()
Where
Name: ImageName
THM: Thumbnail

Figure 4.17: BIP/IPSR/MFS/BV-04-C [Request a Thumbnail – Server Side]

The Lower Tester issues a PutImage request to the IUT containing an image that is not an imaging thumbnail.

The IUT responds to the PutImage request with a PutImage response containing the image handle assigned to the received image and a Partial Content response code.

The Lower Tester issues a PutLinkedThumbnail request containing the thumbnail that corresponds to the previously transmitted image with its image handle.

The IUT receives the thumbnail and responds with a Success response code to the PutLinkedThumbnail request.

- Expected Outcome

  Pass verdict

  The IUT responds to PutImage request with a Partial Content response code.

The IUT receives a PutLinkedThumbnail request containing the thumbnail and responds with a Success response code.

- Notes

  Whenever possible, check on the IUT that the received thumbnail has been linked to the correct image.

BIP/IPSI/MFS/BV-05-C [Push an Attachment – Client Side]

- Test Purpose

  Verify that an IUT that pushes an image, then tries to send an associated attachment does so using the proper image handle (i.e., verify the linkage between a PutImage response and a PutLinkedAttachment request).

- Reference

  [2] 4.4.6.3, 4.5.2, 4.5.4

- Initial Condition

  - There is an active OBEX Image Push service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
  - The IUT has at least one image with an associated attachment that's available for transmission to the Lower Tester (supported attachment formats are declared in the IXIT [9]).

- Test Procedure

  The following sequence is executed:

  1. The IUT issues a PutImage request containing an image to the Lower Tester.
  2. The Lower Tester responds to the PutImage request with a PutImage response containing the image handle that has been assigned to the received image.
  3. The IUT issues a PutLinkedAttachment request containing an attachment file associated with the previously transmitted image along with its image handle.
  4. The Lower Tester responds with a Success response code.

- Expected Outcome

  Pass verdict

  The PutLinkedAttachment request is well formatted. The Lower Tester will specifically verify that the PutLinkedAttachment request contains a valid ConnectionID, a Type header of "x-bt/img-attachment", a valid image handle, a valid attachment descriptor, and the attachment file.

  The attachment descriptor correctly describes the attachment (the Lower Tester will open the attachment file following the indications passed on in the attachment descriptor and confirm that the information given in the attachment descriptor is accurate).

  The image handle used by the IUT in the PutLinkedAttachment request is the one that the Lower Tester assigned to the associated image.

BIP/IPSR/MFS/BV-06-C [Accurately Receive an Attachment – Server Side]

- Test Purpose

  Verify that an IUT correctly processes the attachment files sent by the Lower Tester (i.e., verify the linkage between PutImage responses and subsequent PutLinkedAttachment operations).

- Reference

  [2] 4.4.6.3, 4.5.2, 4.5.4

- Initial Condition

  - There is an active OBEX Image Push service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.
  - The Lower Tester has knowledge of the attachment formats that the IUT supports and use only attachment files that are known to be supported (supported attachment formats are declared in the IXIT [9]).

- Test Procedure

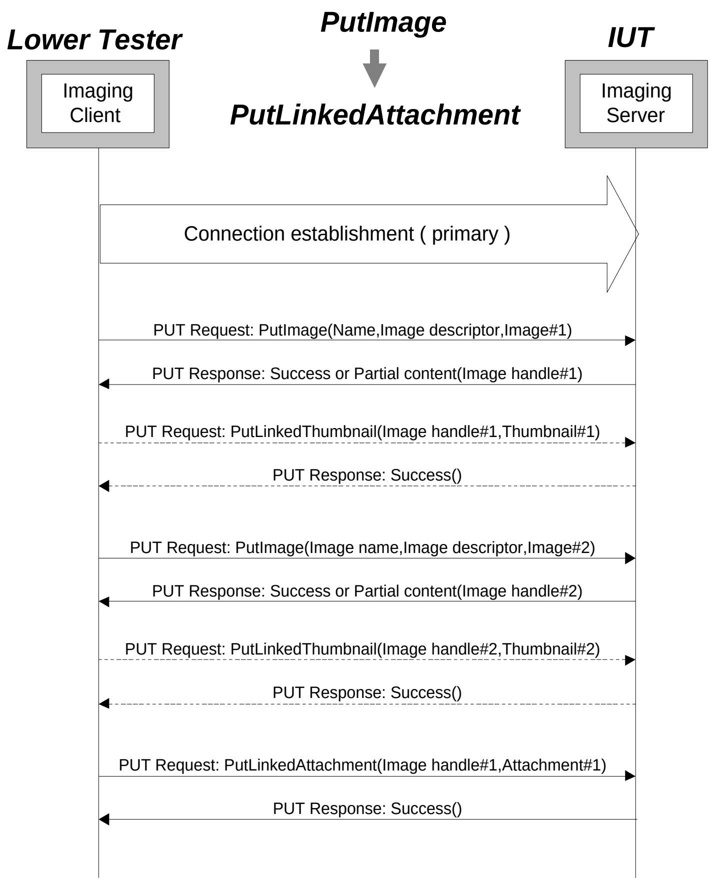

Lower Tester
PutImage
PutLinkedAttachment
IUT
Imaging Client
Imaging Server
Connection establishment (primary)
PUT Request: PutImage(Name,Image descriptor,Image#1)
PUT Response: Success or Partial content(Image handle#1)
PUT Request: PutLinkedThumbnail(Image handle#1,Thumbnail#1)
PUT Response: Success()
PUT Request: PutImage(Image name,Image descriptor,Image#2)
PUT Response: Success or Partial content(Image handle#2)
PUT Request: PutLinkedThumbnail(Image handle#2,Thumbnail#2)
PUT Response: Success()
PUT Request: PutLinkedAttachment(Image handle#1,Attachment#1)
PUT Response: Success()

Figure 4.18: BIP/IPSR/MFS/BV-06-C [Accurately Receive an Attachment – Server Side]

1. The Lower Tester issues a PutImage request containing image #1.
2. The IUT responds to the PutImage request with a PutImage response containing the image handle assigned to image #1. Note that, depending on the implementation, the IUT might request the associated thumbnail. If this is the case, the Lower Tester will send the imaging thumbnail with a PutLinkedThumbnail operation before moving to step 3.
3. The Lower Tester issues a second PutImage request containing image #2.
4. The IUT responds to the PutImage request with a PutImage response containing the image handle assigned to image #2. Note that, depending on the implementation, the IUT might request the associated thumbnail. If this is the case, the Lower Tester will send the imaging thumbnail with a PutLinkedThumbnail operation before moving to step 5.

5. The Lower Tester issues a PutLinkedAttachment request containing the attachment file associated with image #1 together with image handle #1.
6. The IUT receives the attachment file and responds with a Success response code to the PutLinkedAttachment request.

- Expected Outcome

  Pass verdict

  The PutLinkedAttachment response is well formatted.

  The IUT correctly associates the received attachment with the proper image.

4.5.2 Pull Images

Verify that the IUT is capable of correctly retrieving or delivering images as defined in the Image Pull, Automatic Archive, and Remote Camera features of [2].

4.5.2.1 Use the Capability Object to Pull Images Lists – Client Side

- Test Purpose

  Verify that the IUT can retrieve the imaging-capabilities object, use it to learn which filtering parameters supported by the Lower Tester, and properly apply the filtering parameters to a request for an images-listing object.

- Reference

  [2] 4.4.6.3, 4.5.1, 4.5.6

- Initial Condition

  - There is an active OBEX Imaging service session (Image Pull service, Archived Objects service or Remote Display service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-07-C [Use the Capability Object to Pull Images Lists – Client Side]
BIP/AAR/MFS/BV-07-C [Use the Capability Object to Pull Images Lists – Client Side]
BIP/RDI/MFS/BV-07-C [Use the Capability Object to Pull Images Lists – Client Side]

Table 4.18: Use the Capability Object to Pull Images Lists – Client Side test cases

- Test Procedure

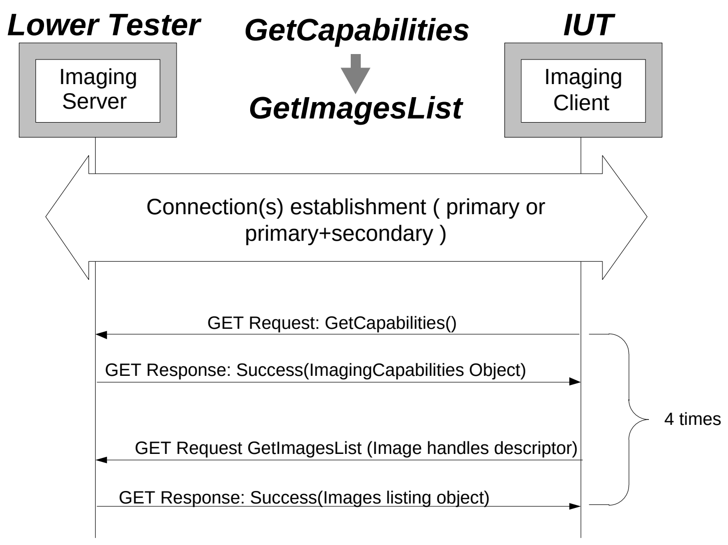

Lower Tester
GetCapabilities
GetImagesList
IUT
Imaging Server
Imaging Client
Connection(s) establishment (primary or primary+secondary)
GET Request: GetCapabilities()
GET Response: Success(ImagingCapabilities Object)
GET Request GetImagesList (Image handles descriptor)
GET Response: Success(Images listing object)
4 times

Figure 4.19: Use the Capability Object to Pull Images Lists – Client Side

The above sequence is repeated four times with four different imaging-capabilities objects being returned by the Lower Tester.

Imaging-capabilities object 1: Indicates support for “creation” as the sole filtering parameter.

Imaging-capabilities object 2: Indicates support for “modified” as the sole filtering parameter.

Imaging-capabilities object 3: Indicates support for “pixel-size” as the sole filtering parameter.

Imaging-capabilities object 4: Indicates support for “encoding” as the sole filtering parameter.

For each iteration of the sequence, the GetImagesList request contains an image handles descriptor that uses only the filtering parameter that has been indicated as supported by the Lower Tester. If the IUT never uses the corresponding filtering parameter (see the ICS [5], for the Image Pull service, for the Archived Objects service, and for the Remote Display service), the GetImagesList request can contain an empty Img-Description header or can carry an image handles descriptor with no image-handle-filtering-parameters element or attributes.

- Expected Outcome

Pass verdict

For each iteration of the GetImagesList request, the image handles descriptor contains only the filtering parameter that has been indicated as supported by the Lower Tester.

- Notes

If a filtering parameter is never used by the IUT in normal operation, it is allowable to send a GetImagesList request with an empty image handles descriptor or an empty Img-Description header. It is mentioned in the ICS that this filtering parameter is never used.

4.5.2.2 Use the Image Property Object – Client Side

- Test Purpose

Verify that the IUT can correctly read the images-listing object, use the image-properties object to individually check the images for versions other than the native format and the imaging thumbnail format, and request an image in a format other than the native one or the mandatory imaging thumbnail.

- Reference

  [2] 4.4.6.1, 4.4.6.2, 4.5.6, 4.5.7, 4.5.8

- Initial Condition

  - There is an active OBEX Imaging service session (Image Pull service or Archived Objects service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-08-C [Use the Image Property Object – Client Side]
BIP/AAR/MFS/BV-08-C [Use the Image Property Object – Client Side]

Table 4.19: Use the Image Property Object – Client Side test cases

- Test Procedure

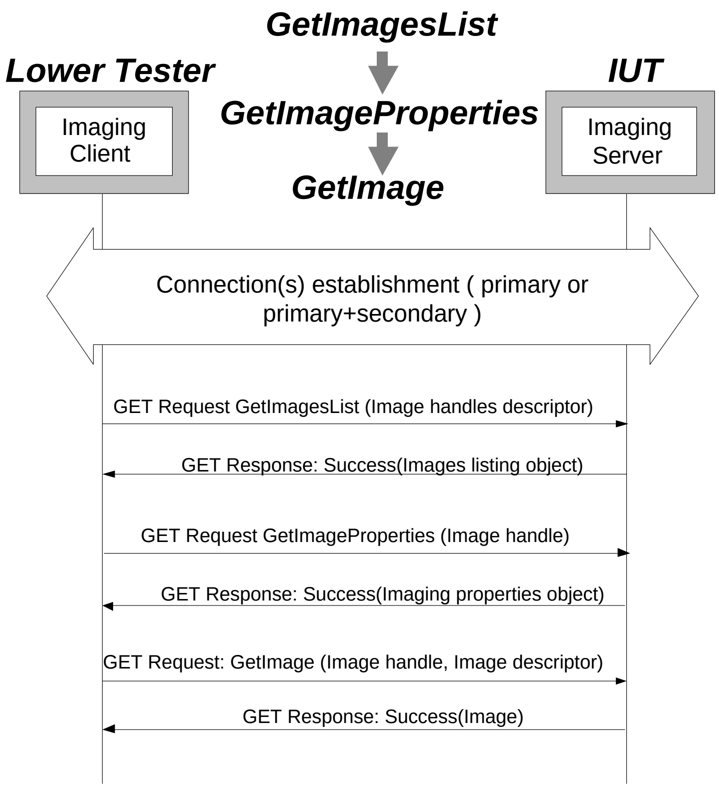

GetImagesList
Lower Tester
IUT
GetImageProperties
Imaging Client
Imaging Server
GetImage
Connection(s) establishment (primary or primary+secondary)
GET Request GetImagesList (Image handles descriptor)
GET Response: Success(Images listing object)
GET Request GetImageProperties (Image handle)
GET Response: Success(Imaging properties object)
GET Request: GetImage (Image handle, Image descriptor)
GET Response: Success(Image)

Figure 4.20: Use the Image Property Object – Client Side

In the above sequence chart, N is the maximum number of handles in an images-listing object supported by the IUT (as indicated in the IXIT [9], for the Image Pull service and the Archived Objects service). n is an integer value that is greater than N/4.

The IUT issues a GetImagesList request with an empty image handles descriptor or an empty Img-Description header.

Upon receiving the images-listing object from the Lower Tester, the IUT retrieves at least one quarter of the images that are available on the Lower Tester and ask for the imaging thumbnail variant. This is done by using a GetImage request with an image descriptor matching the imaging thumbnail format.

- Expected Outcome

  Pass verdict

  The GetImageProperties and GetImage requests are well formatted. The Lower Tester will specifically verify that the GetImageProperties request includes a valid ConnectionID, a Type header of “x-bt/img-properties”, and a valid image handle. The GetImage request contains a valid ConnectionID, a valid image handle, and a well formatted image descriptor.

  The image handle used in the GetImage request is that of an image that has a non-thumbnail variant (the associated image descriptor describes the format of this non-thumbnail variant).

- Notes

  The Lower Tester will make sure that the variant is offered in a format that is supported by the IUT as declared in the IXIT [9], for the Image Pull service and the Archived Objects service. For IUTs that support only the mandatory imaging thumbnail format, this test does not apply.

4.5.2.3 Pull an Image as Thumbnail – Client Side

- Test Purpose

  Verify that the IUT can correctly use the images-listing object to retrieve the thumbnail format of the available images (i.e., test the linkage between the GetImagesList function and the GetImage function with imaging thumbnail as the image descriptor).

- Reference

  [2] 4.4.6.1, 4.5.6, 4.5.8

- Initial Condition

  - There is an active OBEX Imaging service session (Image Pull service or Archived Objects service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-09-C [Pull an Image as Thumbnail – Client Side]
BIP/AAR/MFS/BV-09-C [Pull an Image as Thumbnail – Client Side]

Table 4.20: Pull an Image as Thumbnail – Client Side test cases

- Test Procedure

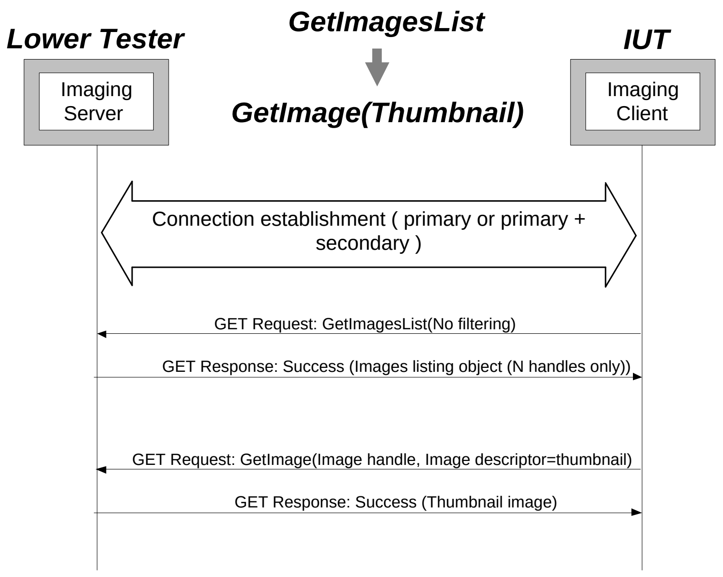

Lower Tester
GetImagesList
GetImage(Thumbnail)
IUT
Imaging Server
Imaging Client
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles only))
GET Request: GetImage(Image handle, Image descriptor=thumbnail)
GET Response: Success (Thumbnail image)

Figure 4.21: Pull an Image as Thumbnail – Client Side

In the above sequence chart, N is the maximum number of handles in an images-listing object supported by the IUT (as indicated in the IXIT [9], for the Image Pull service and the Archived Objects service). n is an integer value that is greater than N/4.

The IUT issues a GetImagesList request with an empty image handles descriptor or an empty Img-Description header.

Upon receiving the images-listing object from the Lower Tester, the IUT retrieves at least one quarter of the images that are available on the Lower Tester and ask for the imaging thumbnail variant. This is done by using a GetImage request with an image descriptor matching the imaging thumbnail format.

- Expected Outcome

<u>Pass verdict</u>

The GetImage function is well formatted. The Lower Tester will specifically verify that the GetImage request contains valid ConnectionID, image handle, and image descriptor.

The image handle is chosen among the image handles returned by the Lower Tester in the images-listing object. The image descriptor describes an imaging thumbnail.

4.5.2.4 Pull a Thumbnail – Client Side

- Test Purpose

Verify that the IUT can correctly use the images-listing object to retrieve thumbnails (i.e., verify the linkage between the GetImagesList function and the GetLinkedThumbnail function).

- Reference

[2] 4.4.6.1, 4.4.7.2, 4.5.6, 4.5.8

- Initial Condition

  - There is an active OBEX session (Image Pull or Archived Objects service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-10-C [Pull a Thumbnail – Client Side]
BIP/AAR/MFS/BV-10-C [Pull a Thumbnail – Client Side]

Table 4.21: Pull a Thumbnail – Client Side test cases

- Test Procedure

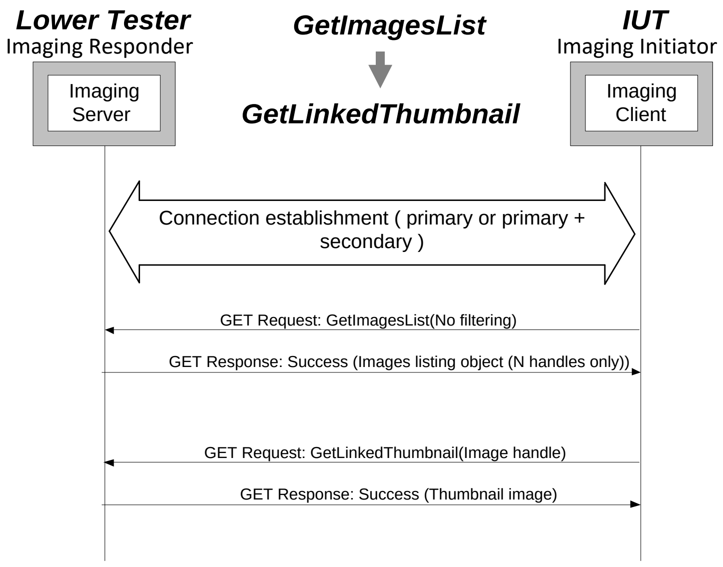

Lower Tester
Imaging Responder
Imaging Server
GetImagesList
GetLinkedThumbnail
IUT
Imaging Initiator
Imaging Client
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles only))
GET Request: GetLinkedThumbnail(Image handle)
GET Response: Success (Thumbnail image)

Figure 4.22: Pull a Thumbnail – Client Side

In the sequence chart, N is the maximum number of handles in an images-listing object supported by the IUT, as indicated in the IXIT [9] for the Image Pull service and the Archived Objects service. n is an integer value and is greater than N/4.

The IUT issues a GetImagesList request with an empty image handles descriptor or an empty Img-Description header.

Upon receiving the images-listing object from the Lower Tester, the IUT retrieves at least one quarter of the images that are available on the Lower Tester and ask for the imaging thumbnail variant. This is done by using the GetLinkedThumbnail function.

- Expected Outcome

<u>Pass verdict</u>

The GetLinkedThumbnail requests is well formatted. The Lower Tester will specifically verify that the GetLinkedThumbnail request includes a valid ConnectionID, a Type header of “x-bt/img-thm”, and a valid image handle.

The image handle that is chosen in the GetLinkedThumbnail request is one of the handles advertised by the Lower Tester in the images-listing object.

- Notes

Verifying that the GetImage response was well received is out of scope for this test.

4.5.2.5 Pull a Native Image – Client Side

- Test Purpose

  Verify that the IUT can correctly use the images-listing object to retrieve the available images in native format.

- Reference

  [2] 4.4.6.1, 4.4.6.2, 4.4.7.2, 4.5.6, 4.5.8

- Initial Condition

  - There is an active OBEX Imaging service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-11-C [Pull a Native Image – Client Side]
BIP/AAR/MFS/BV-11-C [Pull a Native Image – Client Side]

Table 4.22: Pull a Native Image – Client Side test cases

- Test Procedure

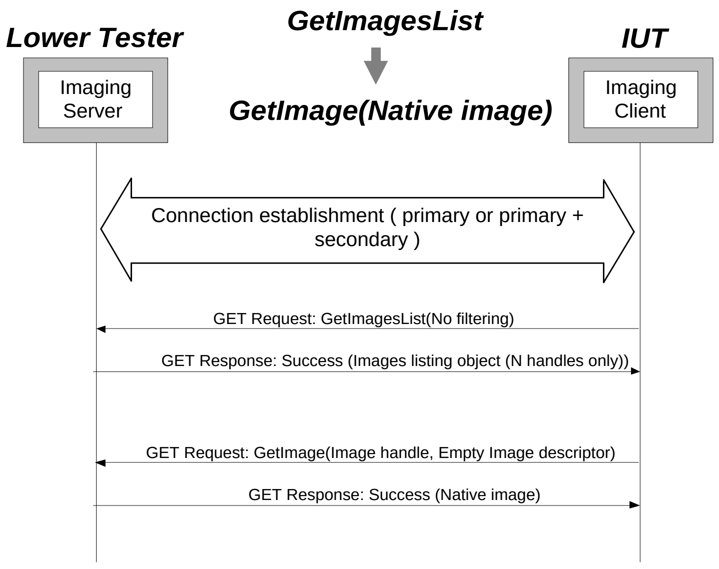

Lower Tester
GetImagesList
GetImage(Native image)
IUT
Imaging Server
Imaging Client
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles only))
GET Request: GetImage(Image handle, Empty Image descriptor)
GET Response: Success (Native image)

Figure 4.23: Pull a Native Image – Client Side

In the above sequence chart, N is the maximum number of handles in an images-listing object supported by the IUT, as specified in the IXIT [9], for the Image Pull service and the Archived Objects service. n is an integer that is superior to N/4.

The IUT issues a GetImagesList request with an empty image handles descriptor or an empty Img-Description header.

Upon receiving the images-listing object from the Lower Tester, the IUT retrieves at least one quarter of the images that are available on the Lower Tester and ask for the native format. This is done by using the GetImage request with an empty Img-Description header.

- Expected Outcome

  Pass verdict

  The GetImagesList and GetImage requests are well formatted.

  The GetImage request contains an empty Img-Description header.

  The GetImage function is well formatted. The Lower Tester will specifically verify that the GetImage request contains valid ConnectionID, image handle, and an empty Img-Description header.

  The image handle is chosen among the image handles returned by the Lower Tester in the images-listing object.

4.5.2.6 Pull Images – Server Side

- Test Purpose

  Verify that the IUT can correctly act as an Image Pull responder or Automatic Archive initiator (i.e., confirm the correctness of the implementations of all the mandatory functions applicable to an Image Pull responder or an Automatic Archive initiator).

- Reference

  [2] 4.4.2, 4.4.6.1, 4.4.6.2, 4.4.6.3, 4.5.1, 4.5.6, 4.5.7, 4.5.8, 4.5.9

- Initial Condition

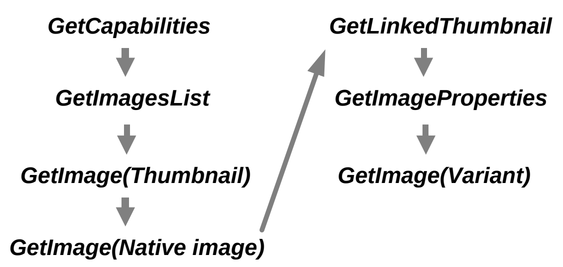
  - There is an active OBEX session (Image Pull or Archived Objects service) where the Lower Tester is the OBEX client and the IUT is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLR/MFS/BV-12-C [Pull Images – Server Side]
BIP/AAI/MFS/BV-12-C [Pull Images – Server Side]

Table 4.23: Pull Images – Server Side test cases

- Test Procedure



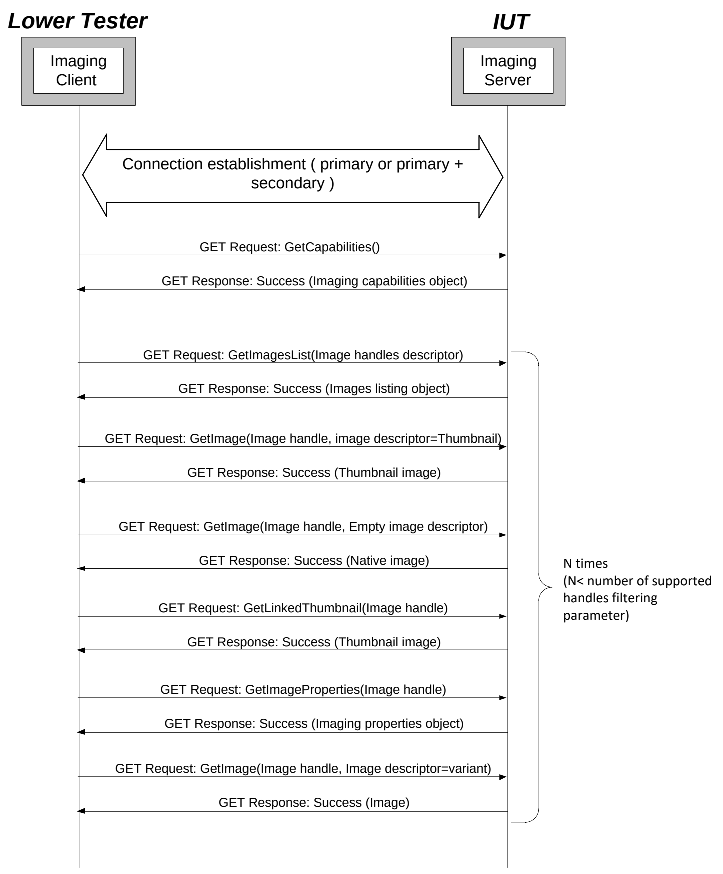

Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (primary or primary + secondary)
GET Request: GetCapabilities()
GET Response: Success (Imaging capabilities object)
GET Request: GetImagesList(Image handles descriptor)
GET Response: Success (Images listing object)
GET Request: GetImage(Image handle, image descriptor=Thumbnail)
GET Response: Success (Thumbnail image)
GET Request: GetImage(Image handle, Empty image descriptor)
GET Response: Success (Native image)
GET Request: GetLinkedThumbnail(Image handle)
GET Response: Success (Thumbnail image)
GET Request: GetImageProperties(Image handle)
GET Response: Success (Imaging properties object)
GET Request: GetImage(Image handle, Image descriptor=variant)
GET Response: Success (Image)
N times (N< number of supported handles filtering parameter)

Figure 4.24: Pull Images – Server Side

1. The Lower Tester first requests the imaging-capabilities object of the IUT.
2. Upon receiving the imaging-capabilities object and reviewing the filtering parameters that are supported by the IUT, the Lower Tester requests the list of images using "encoding" as the filtering parameters that are declared as supported by the IUT.
3. From the images-listing object it receives, the Lower Tester will pick one image at random and successively request its thumbnail (using a GetImage request with the image descriptor of an Imaging thumbnail), its native version (using a GetImage request with no imaging descriptor), and its thumbnail again (using the GetLinkedThumbnail function).
4. The Lower Tester then retrieves the image-properties object corresponding to the image it picked in step 2.
5. If the image has one or more non-thumbnail variants, the Lower Tester will request one of those variants chosen at random. If no non-thumbnail variant is available, the Lower Tester will request either the native version or its associated imaging thumbnail using a GetImage request with the corresponding image descriptor.

6. The test is then repeated three times from step 2, the Lower Tester successively using “modified”, “pixel” and “encoding” as the filtering parameter.

- Expected Outcome

  Pass verdict

  The GetImage, GetLinkedThumbnail, and GetImageProperties responses is well formatted.

  When receiving a GetImagesList with a filtering parameter that is indicated as not supported in its imaging-capabilities object, the IUT returns a GetImagesList response with a non-filtered images-listing object and an imaging handles descriptor with no filtering element.

  When receiving a GetImagesList with a filtering parameter that is indicated as supported in its imaging-capabilities object, the IUT returns a GetImagesList response with a filtered images-listing object and an image handles descriptor identical to the one received to indicate that the filtering parameter sent by the Lower Tester was indeed applied to the images-listing object.

  The various GetImage responses include the relevant image variants (native, thumbnail, or non-thumbnail variant).

  The GetLinkedThumbnail responses include an imaging thumbnail.

4.5.2.7 Delete an Image – Client Side

- Test Purpose

  Verify that the IUT can delete images on the Lower Tester.

- Reference

  [2] 4.4.6.1, 4.5.6, 4.5.11

- Initial Condition

  - There is an OBEX Imaging service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-13-C [Delete an Image – Client Side]
BIP/AAR/MFS/BV-13-C [Delete an Image – Client Side]

Table 4.24: Delete an Image – Client Side test cases

- Test Procedure

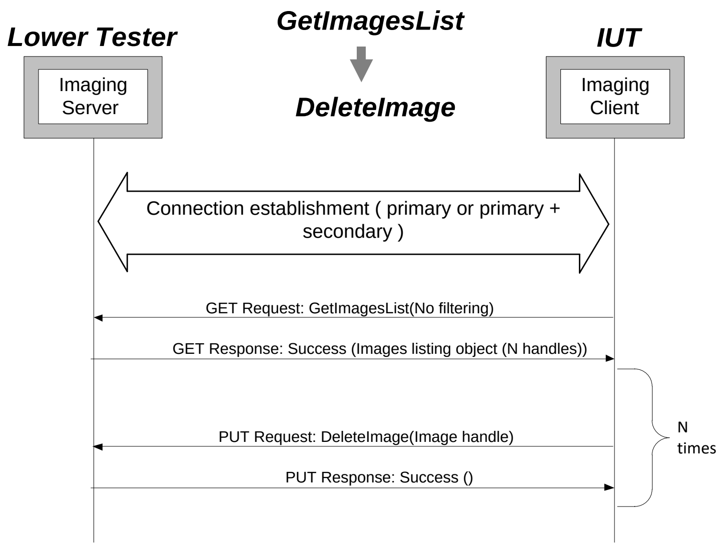

Lower Tester
GetImagesList
DeleteImage
IUT
Imaging Server
Imaging Client
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles))
PUT Request: DeleteImage(Image handle)
PUT Response: Success ()
N times

Figure 4.25: Delete an Image – Client Side

In the above sequence chart, N is the number of handles returned by the Lower Tester in the images-listing object.

The IUT retrieves the images-listing object by issuing a GetImagesList request with no filtering parameters (the Img-Description header is empty or the image handles descriptor is empty).

Upon receiving the images-listing object, the IUT deletes one image after another in the reverse order as the one used in the images-listing object.

- Expected Outcome

<u>Pass verdict</u>

The DeleteImage request is well formatted. The Lower Tester will specifically verify that the DeleteImage request includes a valid ConnectionID and image handle.

The image handle used in the DeleteImage request belongs to the list of image handles returned by the Lower Tester in the images-listing object.

The images are deleted in the reverse order of their appearance in the images-listing object.

4.5.2.8 Delete an Image – Server Side

- Test Purpose

Verify that it is possible to delete pictures on the IUT and validate the linkage between the images-listing object and the image deletion operation.

- Reference

[2] 4.4.6.1, 4.5.6, 4.5.11

- Initial Condition

  - There is an OBEX session (Image Pull or Archived Objects service) active where the Lower Tester is the OBEX client and the IUT is the OBEX server.
  - The IUT has at least one image locally stored and available for deletion.

- Test Case Configuration

Test Case
BIP/IPLR/MFS/BV-14-C [Delete an Image – Server Side]
BIP/AAI/MFS/BV-14-C [Delete an Image – Server Side]

Table 4.25: Delete an Image – Server Side test cases

- Test Procedure

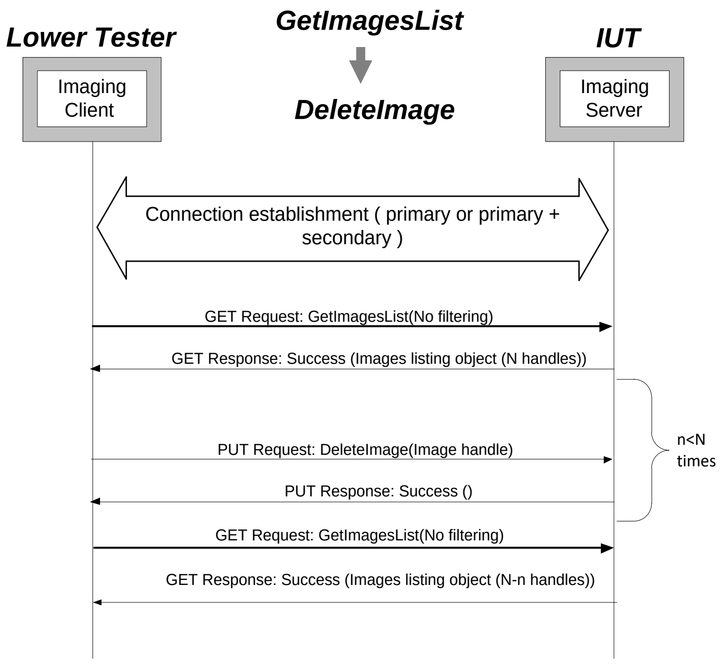

Lower Tester
GetImagesList
DeleteImage
IUT
Imaging Client
Imaging Server
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles))
PUT Request: DeleteImage(Image handle)
PUT Response: Success ()
n<N times
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N-n handles))

Figure 4.26: Delete an Image – Server Side

In the above sequence chart, N is the number of handles returned by the IUT in the first images-listing object.

The Lower Tester will first issue a request for the list of images without filtering parameters.

Upon receiving the images-listing object from the IUT, the Lower Tester will pick n images at random (where n<N) to delete among the ones advertised in the list and order their deletion with a series of DeleteImage requests.

The Lower Tester will then issue a second request for the images-listing object and compare the images that are left with the original images-listing object.

- Expected Outcome

<u>Pass verdict</u>

The DeleteImage function is well formatted. The Lower Tester will specifically verify that the request contains a valid ConnectionID and image handle and that there is no Body or EndOfBody header.

All the images whose deletion was requested by the Lower Tester has disappeared from the final images-listing object.

- Notes

  In some IUT implementations, it is conceivable that the deletion of one image automatically results in the deletion of a series of images. For instance, on cameras that can take several images in a raw (or burst) mode, the deletion of the first raw image of the raw implies deleting all the other images that belong to the same series. This behavior is legal and might result in a final images-listing object that contains fewer images than N-n.

4.5.2.9 Get an Attachment – Client Side

- Test Purpose

  Verify that the IUT can properly discover the properties of an attachment and retrieve it (i.e., check the linkage between the interpretation of the images-listing object, the GetImageProperties request, and the GetLinkedAttachment request).

- Reference

  [2] 4.4.4, 4.4.6.1, 4.4.6.3, 4.5.6, 4.5.7, 4.5.10

- Initial Condition

  - There is an active OBEX Imaging session (Image Pull or Archived Objects service) where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLI/MFS/BV-15-C [Get an Attachment – Client Side]
BIP/AAR/MFS/BV-15-C [Get an Attachment – Client Side]

Table 4.26: Get an Attachment – Client Side test cases

- Test Procedure

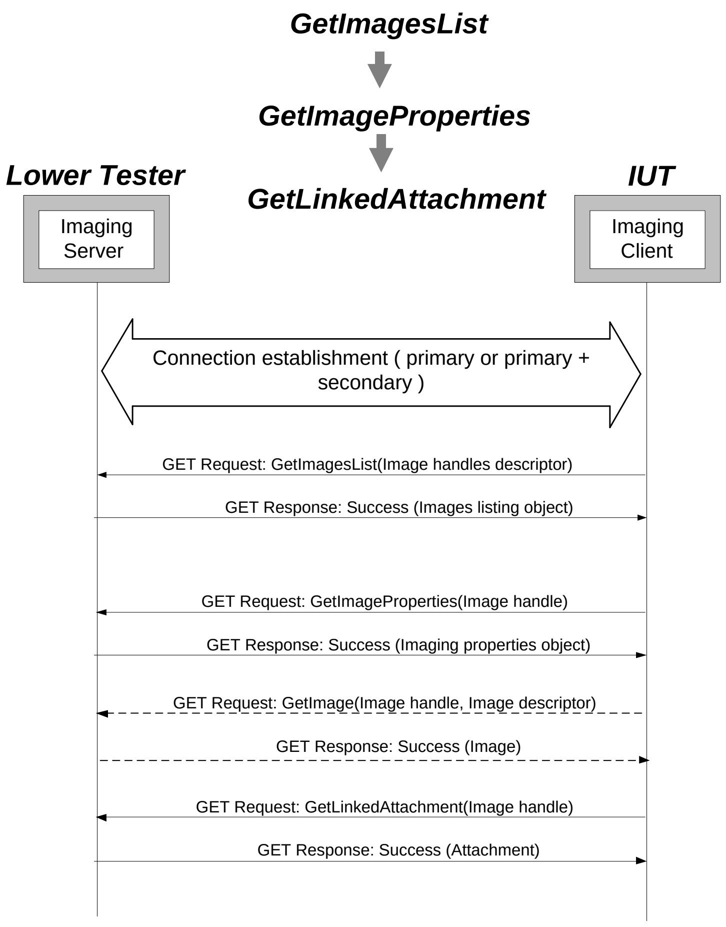

GetImagesList
GetImageProperties
GetLinkedAttachment
Lower Tester
Imaging Server
IUT
Imaging Client
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(Image handles descriptor)
GET Response: Success (Images listing object)
GET Request: GetImageProperties(Image handle)
GET Response: Success (Imaging properties object)
GET Request: GetImage(Image handle, Image descriptor)
GET Response: Success (Image)
GET Request: GetLinkedAttachment(Image handle)
GET Response: Success (Attachment)

Figure 4.27: Get an Attachment – Client Side

The IUT retrieves the list of images available on the Lower Tester by issuing a GetImagesList request. The IUT may or may not include a non-empty image handles descriptor in the request.

Upon receiving the list of images, the IUT chooses one image and retrieves its image-properties object via a GetImageProperties request.

The IUT may or may not request the image from the Lower Tester, as in some implementations the IUT may be unable to issue a GetLinkedAttachment request on an image that has not been retrieved.

The IUT retrieves one attachment associated with the image chosen by issuing a GetLinkedAttachment request.

- Expected Outcome

<u>Pass verdict</u>

The GetImageProperties and GetLinkedAttachment requests are well formatted. The Lower Tester will verify that the GetImageProperties request contains a valid ConnectionID, a Type header of “x-bt/img-properties”, and a valid image handle in the Img-Handle header. The GetLinkedAttachment request includes a valid ConnectionID, a Type header of “x-bt/img-attachment”, a valid image handle, and a valid file name.

The image handle passed by the IUT in the GetImageProperties and GetLinkedAttachment requests belongs to one of the handles listed by the Lower Tester in the images-listing object.

- Notes

  Associated with each image listed by the Lower Tester will be a number of attachments. Based on the information provided in the IXIT [9] regarding support of attachments by the IUT, the Lower Tester will make sure that for each image, at least one of the attachments is in a format supported by the IUT.

4.5.2.10 Provide an Attachment – Server Side

- Test Purpose

  Verify that the IUT, when connected through an Imaging service (Image Pull or Archived Objects service), can properly advertise the existence of and deliver attachment files associated with image files.

- Reference

  [2] 4.4.4, 4.4.6.1, 4.4.6.3, 4.5.6, 4.5.7, 4.5.10

- Initial Condition

  - There is an active OBEX Image Pull or Archived Objects service session where the Lower Tester is the OBEX client and the IUT is the OBEX server.

- Test Case Configuration

Test Case
BIP/IPLR/MFS/BV-16-C [Provide an Attachment – Server Side]
BIP/AAI/MFS/BV-16-C [Provide an Attachment – Server Side]

Table 4.27: Provide an Attachment – Server Side test cases

- Test Procedure

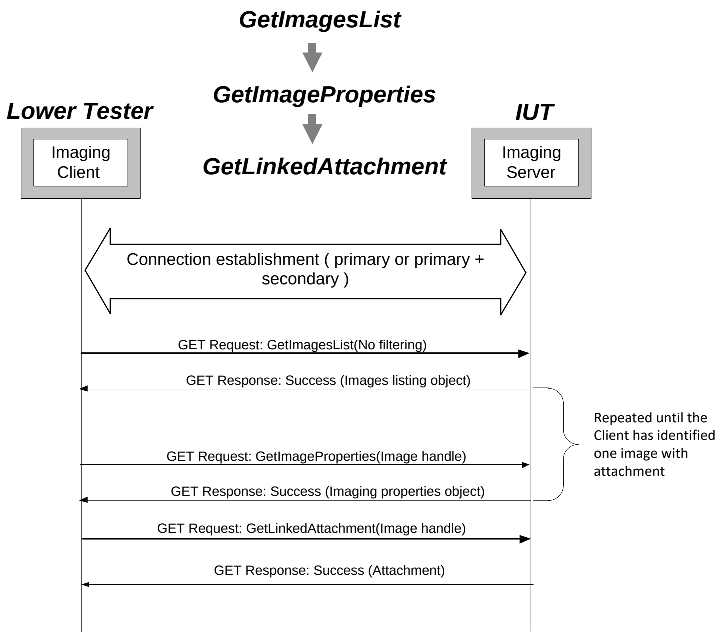

GetImagesList
GetImageProperties
GetLinkedAttachment
Lower Tester
Imaging Client
IUT
Imaging Server
Connection establishment (primary or primary + secondary)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object)
GET Request: GetImageProperties(Image handle)
GET Response: Success (Imaging properties object)
Repeated until the Client has identified one image with attachment
GET Request: GetLinkedAttachment(Image handle)
GET Response: Success (Attachment)

Figure 4.28: Provide an Attachment – Server Side

The Lower Tester retrieves the list of images available on the Lower Tester by issuing a GetImagesList request without using any of the filtering parameters.

Upon receiving the list of images, the Lower Tester will look for an image with associated attachments by retrieving the image-properties object for each image in the images-listing object.

The Lower Tester retrieves the attachments associated with the image.

- Expected Outcome

  Pass verdict

  The GetImageProperties and GetLinkedAttachment responses are well formatted. The Lower Tester will specifically check that the GetImageProperties response contains a well formatted image-properties object.

  The GetLinkedAttachment requests issued by the Lower Tester is properly processed and result in a response with the Success response code.

  The attachment delivered by the IUT in the GetLinkedAttachment has the same characteristics as those described in the image-properties object (the Lower Tester will open the attachment file and check that the declaration in the mage-properties object was correct).

4.5.3 Remotely Order an Advanced Printing Job

Verify that the IUT can correctly order or perform a print job, as specified in the Advanced Image Printing feature of [2].

BIP/AIPI/MFS/BV-17-C [Use the Imaging-Capabilities Object – Client Side]

- Test Purpose

  Verify the linkage between the GetCapabilities request in the primary connection and the GetPartialImage response in the secondary connection (i.e., that the IUT is capable of correctly interpreting the imaging-capabilities object of the Lower Tester and provides image chunks in a format the Lower Tester supports).

- Reference

  [2] 4.4.6.3, 4.4.6.4, 4.5.1, 4.5.13, 4.5.14, 5.5

- Initial Condition

  - There is an active OBEX Advanced Image Printing session where the IUT is the OBEX client (for the primary connection) and the Lower Tester is OBEX server (also for the primary connection).
  - The IUT has at least two and up to ten images to be transmitted and described in the printer control object.
  - The image-formats attribute in the imaging-capabilities object are set randomly by the Lower Tester but will contain at least one image format that is known to be supported by the IUT (based on the IXIT [9]), so the IUT can always find a format that will suit the Lower Tester.

- Test Procedure

  The Lower Tester sends its imaging-capabilities object in response to the GetCapabilities request.

  The IUT issues a StartPrint request containing a printing control object.

  The Lower Tester issues a StartArchive response followed by a Connect request to the Referenced Objects service of the IUT.

  The IUT accepts the Connect request and the secondary session begins.

The Lower Tester retrieves images according to previously received printing control object and the IUT transmits the images in a format that is known to be supported by the Lower Tester (as declared in the Lower Tester’s imaging-capabilities object). The first image is retrieved in chunks of 1Kbyte long. This function is then repeated to complete the print jobs specified in the printer control object, but the images are retrieved in one block (PartialFileStartOffset always set to 0 and PartialFileLength always set to the 0xFFFFFFFF in the Application Parameters headers of the GetPartialImage requests).

The Lower Tester issues a Disconnect request.

IUT accepts the Disconnect request.

- Expected Outcome

  Pass verdict

  The formats of transmitted images are actually available as described in the imaging-capabilities object.

BIP/AIPR/MFS/BV-18-C [Use the Imaging-Capabilities Object – Server Side]

- Test Purpose

  Verify that the image format and DPOF options that are described in the imaging-capabilities object are actually supported by the IUT (i.e., test the linkage between the imaging-capabilities object and the GetPartialImage function).

- Reference

  [2] 4.4.6.3, 4.4.6.4, 4.5.1, 4.5.13, 4.5.14, 5.5

- Initial Condition

  - There is an active OBEX Advanced Image Printing session where the IUT is the OBEX server (for the primary connection) and the Lower Tester is OBEX client (for the secondary connection).

- Test Procedure

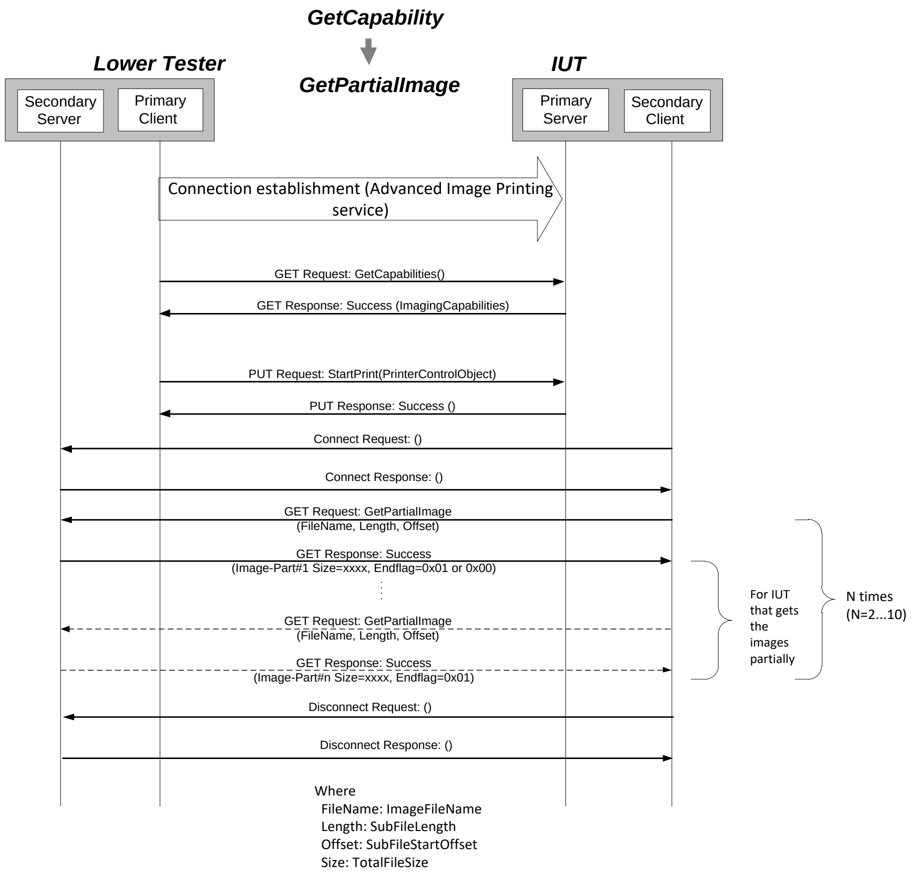

GetCapability
GetPartialImage
Lower Tester
IUT
Secondary Server
Primary Client
Primary Server
Secondary Client
Connection establishment (Advanced Image Printing service)
GET Request: GetCapabilities()
GET Response: Success (ImagingCapabilities)
PUT Request: StartPrint(PrinterControlObject)
PUT Response: Success ()
Connect Request: ()
Connect Response: ()
GET Request: GetPartialImage (FileName, Length, Offset)
GET Response: Success (Image-Part#1 Size=xxxx, Endflag=0x01 or 0x00)
GET Request: GetPartialImage (FileName, Length, Offset)
GET Response: Success (Image-Part#n Size=xxxx, Endflag=0x01)
For IUT that gets the images partially
N times (N=2...10)
Disconnect Request: ()
Disconnect Response: ()
Where
FileName: ImageFileName
Length: SubFileLength
Offset: SubFileStartOffset
Size: TotalFileSize

Figure 4.29: BIP/AIPR/MFS/BV-18-C [Use the Imaging-Capabilities Object – Server Side]

The Lower Tester issues a GetCapabilities request to the IUT.

The IUT sends its imaging-capabilities object in response to the GetCapabilities request.

The Lower Tester issues a StartPrint request containing a printing control object.

Upon receiving the StartPrint request, the IUT issues a StartPrint response followed by a Connect request to the Referenced Objects service of the Lower Tester.

The Lower Tester accepts the Connect request and the secondary session is initiated.

The IUT retrieves the images according to the previously received printer control object and the Lower Tester transmits the images in a format described as supported in the imaging-capabilities object of the IUT.

The IUT, after having correctly received all the images necessary to perform the print job, issues a disconnect request in the secondary session.

The Lower Tester accepts the Disconnect request.

IUT accepts the Disconnect request.

- Expected Outcome

<u>Pass verdict</u>

The print job described in the printing control object is output.

The images are correctly received and printed by the IUT.

4.5.4 Remotely Control a Camera

Verify that the IUT can correctly perform the Remote Control feature as defined in [2].

BIP/RCI/MFS/BV-19-C [Take and Retrieve an Image – Client Side]

- Test Purpose

  Verify that the IUT can instruct the Imaging Server to store an image and subsequently retrieve it in a valid format and using the proper handle.

- Reference

  [2] 4.4.6.2, 4.4.6.5, 4.5.7, 4.5.8, 4.5.16

- Initial Condition

  - There is an OBEX Remote Camera service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

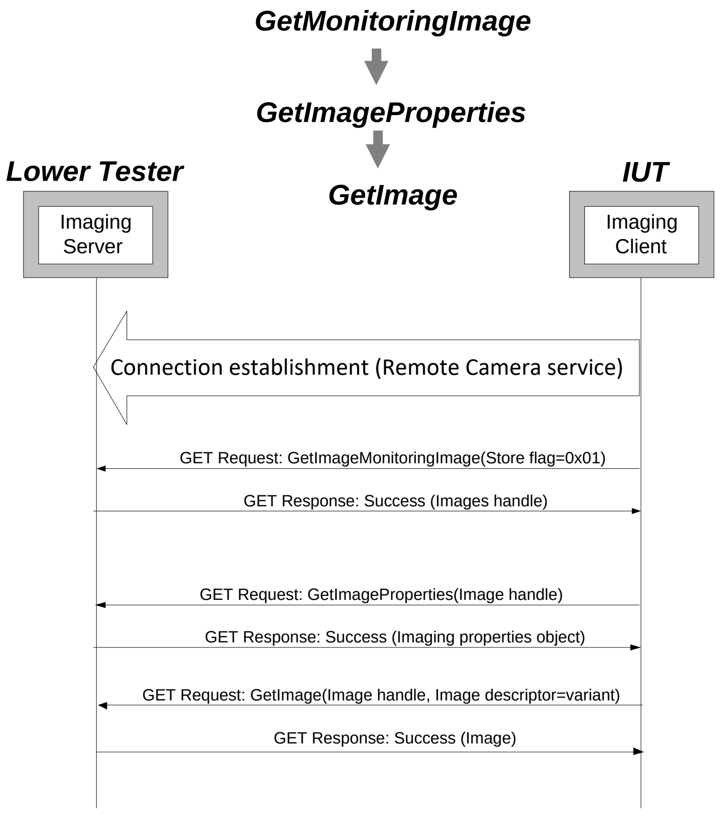

GetMonitoringImage
GetImageProperties
GetImage
Lower Tester
IUT
Imaging Server
Imaging Client
Connection establishment (Remote Camera service)
GET Request: GetImageMonitoringImage(Store flag=0x01)
GET Response: Success (Images handle)
GET Request: GetImageProperties(Image handle)
GET Response: Success (Imaging properties object)
GET Request: GetImage(Image handle, Image descriptor=variant)
GET Response: Success (Image)

Figure 4.30: BIP/RCI/MFS/BV-19-C [Take and Retrieve an Image – Client Side]

The IUT issues a request for a monitoring image (the StoreFlag is set to 0x01, indicating to the Lower Tester that the full image corresponding to the monitoring image should be stored and assigned an image handle).

The Lower Tester responds with a GetMonitoringImage response that includes the monitoring image object and its associated image handle.

The IUT requests the image-properties object corresponding to the stored image.

The Lower Tester returns the image-properties object of the stored image.

The IUT retrieves the full Image using a GetImage request and an image descriptor that describes one of the formats that are indicated as available in the image-properties object of the stored image.

- Expected Outcome

  Pass verdict

  The GetImageProperties and GetImage requests are well formatted. The Lower Tester will especially verify that the GetImageProperties request contains a valid ConnectionID, a Type header of “x-bt/img-properties”, and a valid Image handle. The GetImage request contains a valid ConnectionID, a Type header of “x-bt/img-thm”, and a valid image handle.

  The GetMonitoringImage contains a StoreFlag set to 0x01.

  The handle passed by the IUT in the GetImage request is the handle that was returned by the Lower Tester in the GetMonitoringImage response.

  The image format chosen by the IUT in its GetImage request is described as available by the Lower Tester in the image’s image-properties object.

- Notes

  The GetMonitoringImage function is tested in BIP/RCI/FSF/BV-01-C [Get a Monitoring Image – Image Not Saved – Client Side] and BIP/RCI/FSF/BV-02-C [Get a Monitoring Image – Image Saved – Client Side].

BIP/RCI/MFS/BV-20-C [Take and Retrieve an Image as Imaging Thumbnail – Client Side]

- Test Purpose

  Verify that the IUT can order an image to be stored and subsequently retrieve the imaging thumbnail version of the image using the proper handle.

- Reference

  [2] 4.4.6.2, 4.4.6.5, 4.5.7, 4.5.8, 4.5.16

- Initial Condition

  - There is an OBEX Remote Camera service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

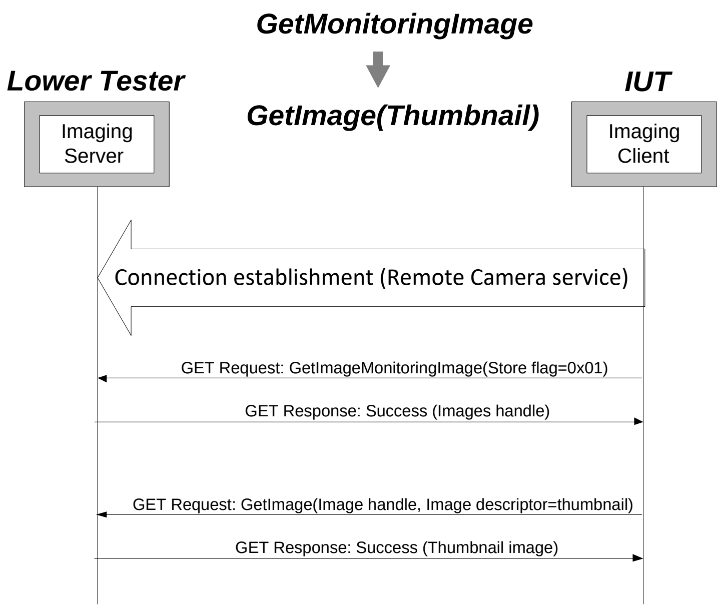

GetMonitoringImage
GetImage(Thumbnail)
Lower Tester
Imaging Server
IUT
Imaging Client
Connection establishment (Remote Camera service)
GET Request: GetImageMonitoringImage(Store flag=0x01)
GET Response: Success (Images handle)
GET Request: GetImage(Image handle, Image descriptor=thumbnail)
GET Response: Success (Thumbnail image)

Figure 4.31: BIP/RCI/MFS/BV-20-C [Take and Retrieve an Image as Imaging Thumbnail – Client Side]

The IUT issues a request for the monitoring image (the StoreFlag is set to 0x01, indicating to the Lower Tester that the full image corresponding to the monitoring image should be stored and assigned an image handle).

The Lower Tester responds with a GetMonitoringImage response that includes the monitoring image object and its associated image handle.

The IUT retrieves the imaging thumbnail of the image using a GetImage request with the image descriptor of an imaging thumbnail and the image handle of the stored.

- Expected Outcome

Pass verdict

The GetImage request is well formatted. The Lower Tester will especially verify that the GetImage request contains a valid ConnectionID, a Type header of “x-bt/img-img”, and a valid image handle.

The GetMonitoringImage contains a StoreFlag set to 0x01.

The handle passed by the IUT in the GetImage request is the handle that was returned by the Lower Tester in the GetMonitoringImage response.

The image descriptor used in the GetImage request is that of an imaging thumbnail.

- Notes

The GetMonitoringImage function is tested in BIP/RCI/FSF/BV-01-C [Get a Monitoring Image – Image Not Saved – Client Side] and BIP/RCI/FSF/BV-02-C [Get a Monitoring Image – Image Saved – Client Side].

BIP/RCI/MFS/BV-21-C [Take an Image and Retrieve its Native Version – Client Side]

- Test Purpose

  Verify that the IUT can order an image to be stored and subsequently retrieve the native image of the image using the proper handle.

- Reference

  [2] 4.4.6.5, 4.5.8, 4.5.16

- Initial Condition

  - There is an OBEX Remote Camera service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

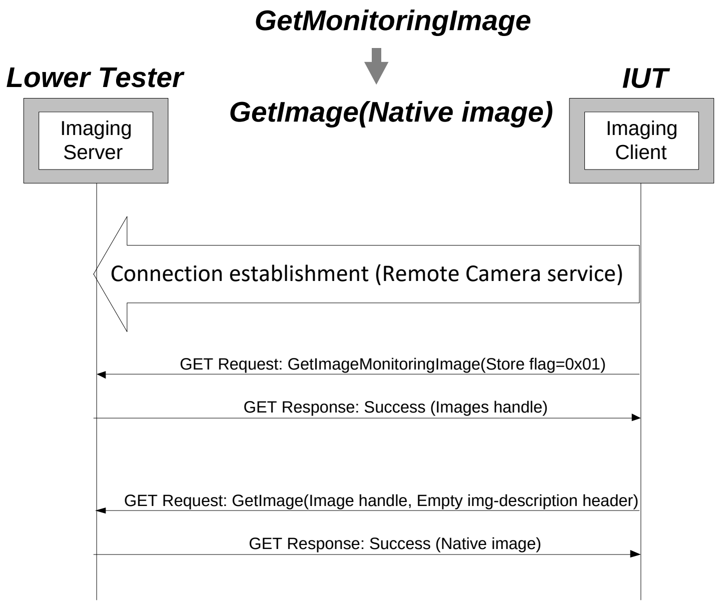

GetMonitoringImage
GetImage(Native image)
Lower Tester
Imaging Server
IUT
Imaging Client
Connection establishment (Remote Camera service)
GET Request: GetImageMonitoringImage(Store flag=0x01)
GET Response: Success (Images handle)
GET Request: GetImage(Image handle, Empty img-description header)
GET Response: Success (Native image)

Figure 4.32: BIP/RCI/MFS/BV-21-C [Take an Image and Retrieve its Native Version – Client Side]

The IUT issues a request for the monitoring image (the StoreFlag is set to 0x01, indicating to the Lower Tester that the full image corresponding to the monitoring image should be stored and assigned an image handle).

The Lower Tester responds with a GetMonitoringImage response that includes the monitoring image object and its associated image handle.

The IUT retrieves the native image of the monitoring image using a GetImage request with the image handle of the stored image and an empty Img-Description header.

- Expected Outcome

  <u>Pass verdict</u>

  The GetImage request is well formatted. The Lower Tester will especially verify that the GetImage request contains a valid ConnectionID, an empty Img-Description header, and a valid image handle.

  The GetMonitoringImage contains a StoreFlag set to 0x01.

  The handle passed by the IUT in the GetImage request is the handle that was returned by the Lower Tester in the GetMonitoringImage response.

- Notes

  The GetMonitoringImage function is tested in BIP/RCI/FSF/BV-01-C [Get a Monitoring Image – Image Not Saved – Client Side] and BIP/RCI/FSF/BV-02-C [Get a Monitoring Image – Image Saved – Client Side].

BIP/RCI/MFS/BV-22-C [Take and Retrieve an Image as Imaging Thumbnail using the GetLinkedThumbnail Function – Client Side]

- Test Purpose

  Verify that the IUT can order an image to be stored and subsequently retrieve the imaging thumbnail associated with this image using the proper handle and the GetLinkedThumbnail function.

- Reference

  [2] 4.4.6.5, 4.5.9, 4.5.16

- Initial Condition

  - There is an OBEX Remote Camera service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

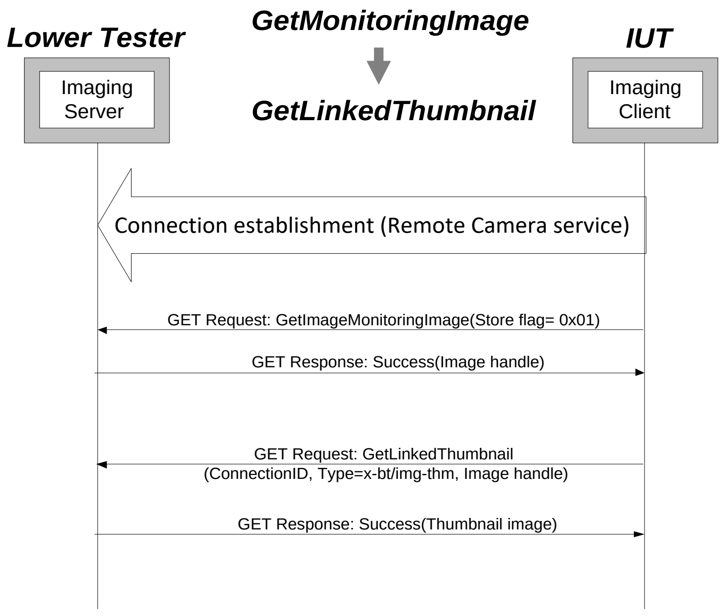

Lower Tester
GetMonitoringImage
GetLinkedThumbnail
IUT
Imaging Server
Imaging Client
Connection establishment (Remote Camera service)
GET Request: GetImageMonitoringImage(Store flag= 0x01)
GET Response: Success(Image handle)
GET Request: GetLinkedThumbnail (ConnectionID, Type=x-bt/img-thm, Image handle)
GET Response: Success(Thumbnail image)

Figure 4.33: BIP/RCI/MFS/BV-22-C [Take and Retrieve an Image as Imaging Thumbnail using the GetLinkedThumbnail Function – Client Side]

The IUT issues a request for the monitoring image (the StoreFlag is set to 0x01, indicating to the Lower Tester that the full image corresponding to the monitoring image should be stored and assigned an image handle).

The Lower Tester responds with a GetMonitoringImage response that includes the monitoring image object and its associated image handle.

The IUT then retrieves the imaging thumbnail of the stored image using a GetLinkedThumbnail request with the image handle of the stored image.

- Expected Outcome

  Pass verdict

  The GetLinkedThumbnail request is well formatted. The Lower Tester will especially verify that the GetLinkedThumbnail request contains a valid ConnectionID, a Type header of “x-bt/img-thm”, and a valid image handle.

  The GetMonitoringImage contains a StoreFlag set to 0x01.

  The handle passed by the IUT in the GetLinkedThumbnail request is the handle that was returned by the Lower Tester in the GetMonitoringImage response.

- Notes

  The GetMonitoringImage function is tested in BIP/RCI/FSF/BV-01-C [Get a Monitoring Image – Image Not Saved – Client Side] and BIP/RCI/FSF/BV-02-C [Get a Monitoring Image – Image Saved – Client Side].

BIP/RCR/MFS/BV-23-C [Act as Remote Camera Imaging Responder – Server Side]

- Test Purpose

  Verify that the IUT can correctly act as a Remote Camera server (i.e., properly implements all the mandatory functions defined for the server side of the Remote Camera feature).

- Reference

  [2] 4.3.5, 4.4.6.2, 4.4.6.5, 4.5.7, 4.5.8, 4.5.9, 4.5.16

- Initial Condition

  - There is an OBEX Remote Camera service session active where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Procedure

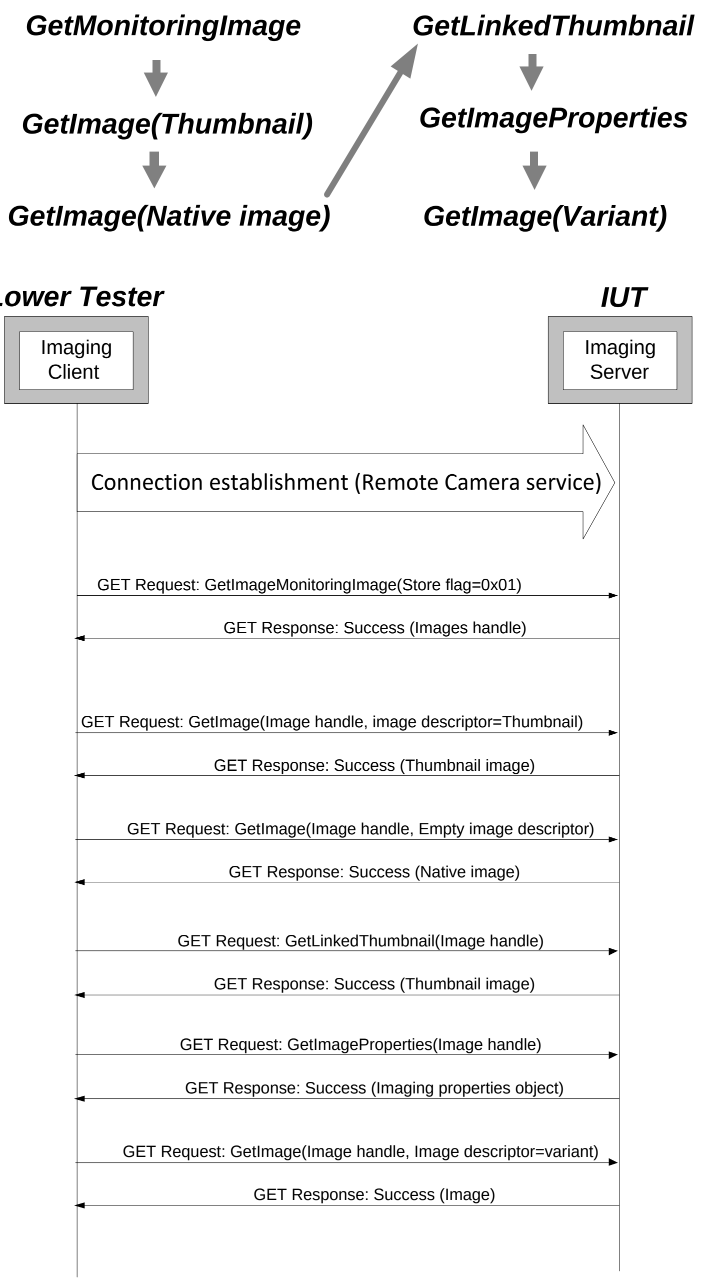

GetMonitoringImage
GetImage(Thumbnail)
GetImage(Native image)
GetLinkedThumbnail
GetImageProperties
GetImage(Variant)
Lower Tester
IUT
Imaging Client
Imaging Server
Connection establishment (Remote Camera service)
GET Request: GetImageMonitoringImage(Store flag=0x01)
GET Response: Success (Images handle)
GET Request: GetImage(Image handle, image descriptor=Thumbnail)
GET Response: Success (Thumbnail image)
GET Request: GetImage(Image handle, Empty image descriptor)
GET Response: Success (Native image)
GET Request: GetLinkedThumbnail(Image handle)
GET Response: Success (Thumbnail image)
GET Request: GetImageProperties(Image handle)
GET Response: Success (Imaging properties object)
GET Request: GetImage(Image handle, Image descriptor=variant)
GET Response: Success (Image)

Figure 4.34: BIP/RCR/MFS/BV-23-C [Act as Remote Camera Imaging Responder – Server Side]

The Lower Tester first requests a monitoring image to be returned by the IUT and the corresponding image to be stored.

The Lower Tester retrieves the imaging thumbnail of the stored image using a GetImage request with the image handle that was returned by the IUT in the GetMonitoringImage response and the image descriptor for an imaging thumbnail.

The Lower Tester retrieves the native image of the stored image using a GetImage request with the image handle that was returned by the IUT in the GetMonitoringImage response and an empty Img-Description header.

The Lower Tester retrieves the imaging thumbnail of the stored image using a GetLinkedThumbnail response and the image handle that was returned by the IUT in the GetMonitoringImage response.

The Lower Tester issues a request for the image-properties object of the stored image.

After reading the image-properties object and discovering that the image has one or more non-thumbnail variants available, the Lower Tester will request one of those variants chosen at random. If no non-thumbnail variant is available, the Lower Tester will request either the native version or its associated imaging thumbnail using a GetImage request with the corresponding image descriptor.

- Expected Outcome

  Pass verdict

  The GetImage, GetLinkedThumbnail, and GetImageProperties responses are well formatted.

  The various GetImage responses include the relevant image variants (native, thumbnail, or non-thumbnail variant).

  The GetLinkedThumbnail responses always include an imaging thumbnail.

- Notes

  The GetMonitoringImage function is tested in BIP/RCR/FSF/BV-03-C [Get a Monitoring Image – Image Not Saved – Server Side] and BIP/RCR/FSF/BV-04-C [Get a Monitoring Image – Image Saved – Server Side].

4.5.5 Remotely Pilot a Display Device

Verify that the IUT can correctly perform the Remote Display feature as defined in [2].

BIP/RDI/MFS/BV-24-C [Select an Image to Display After Having Pushed It – Client Side]

- Test Purpose

  Verify that the IUT push an image to the Lower Tester and request that it be displayed.

- Reference

  [2] 4.4.7.2, 4.5.2, 4.5.5

- Initial Condition

  - There is an OBEX Remote Display service session active where the IUT is the OBEX client and the Lower Tester is the OBEX server.

- Test Procedure

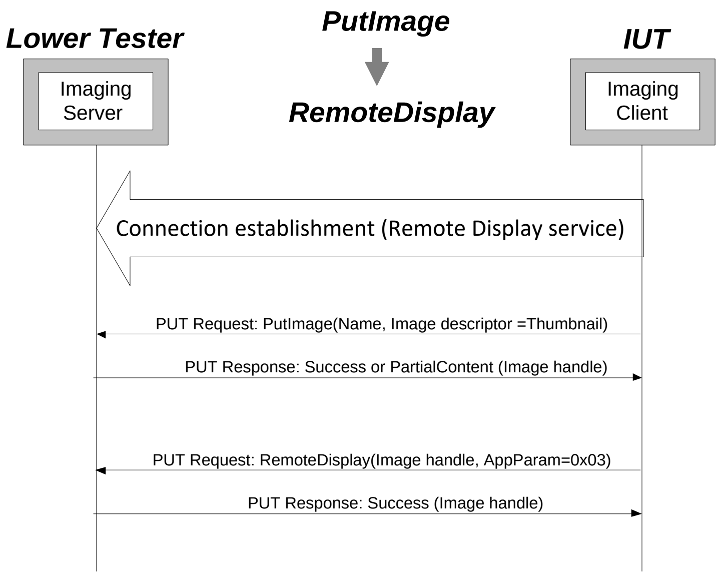

Lower Tester
Imaging Server
PutImage
RemoteDisplay
IUT
Imaging Client
Connection establishment (Remote Display service)
PUT Request: PutImage(Name, Image descriptor =Thumbnail)
PUT Response: Success or PartialContent (Image handle)
PUT Request: RemoteDisplay(Image handle, AppParam=0x03)
PUT Response: Success (Image handle)

Figure 4.35: BIP/RDI/MFS/BV-24-C [Select an Image to Display After Having Pushed It – Client Side]

The IUT pushes an image to the Lower Tester using the PutImage function.

The Lower Tester returns a Success response code accompanied with the image handle assigned to the image.

Upon receiving the image handle, the IUT uses it to request that the Lower Tester's displaying it by issuing a RemoteDisplay request with the SelectImage command.

- Expected Outcome

<u>Pass verdict</u>

The RemoteDisplay request is well formatted. The Lower Tester will specifically verify that the RemoteDisplay request contains a valid ConnectionID, a Type header of "x-bt/img-display", the Application Parameters header set to the SelectImage command, and a valid image handle.

The image handle passed by the IUT in the RemoteDisplay request is the identical to the one received from the Lower Tester in the PutImage response.

- Notes

The Remote Display function is tested in BIP/RDI/FSF/BV-09-C [Go to Next Image – Client Side] and BIP/RDI/FSF/BV-10-C [Go to Previous Image – Client Side] (Application Parameters values NextImage and PreviousImage only).

BIP/RDI/MFS/BV-25-C [Select an Image to Display from the Images-Listing Object – Client Side]

- Test Purpose

Verify that the IUT can select and trigger the display of images from the images-listing object.

- Reference

[2] 4.5.6, 4.5.5

- Initial Condition
  - There is an active OBEX Imaging service session where the IUT is the OBEX client and the Lower Tester is the OBEX server.
- Test Procedure

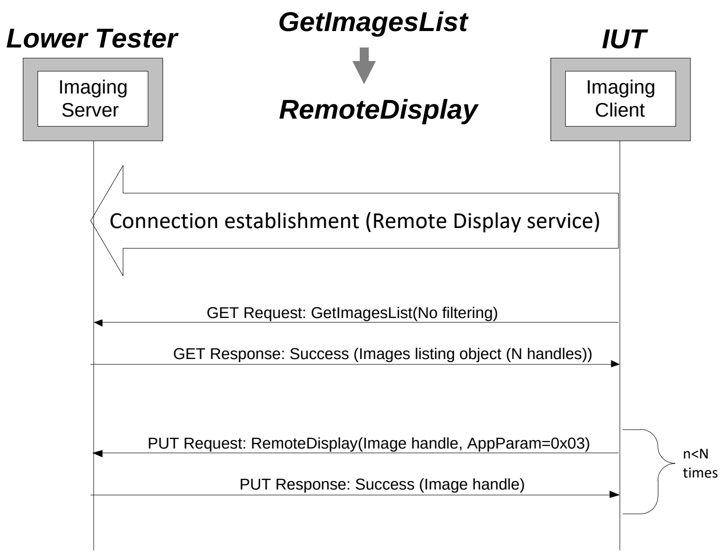

Lower Tester
GetImagesList
RemoteDisplay
IUT
Imaging Server
Imaging Client
Connection establishment (Remote Display service)
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object (N handles))
PUT Request: RemoteDisplay(Image handle, AppParam=0x03)
PUT Response: Success (Image handle)
n<N times

Figure 4.36: BIP/RDI/MFS/BV-25-C [Select an Image to Display from the Images-Listing Object – Client Side]

In the above sequence chart, N is the maximum number of image handles in an images-listing object supported by the IUT (as declared in the IXIT [9]).

The IUT retrieves the list of Images available for display on the Lower Tester with a GetImageList request.

The Lower Tester returns an images-listing object that contains N handles.

The IUT chooses an image to display by issuing a RemoteDisplay request with SelectImage as the Application Parameters header and the image handle of the image of interest.

The Lower Tester returns a Success response code after displaying the selected image.

- Expected Outcome

  Pass verdict

  The RemoteDisplay request is well formatted. The Lower Tester will specifically verify that the RemoteDisplay request includes a valid ConnectionID, a Type header of “x-bt/img-display”, a valid image handle, and SelectImage as the Application Parameters header.

  The image handle that is passed by the IUT in the RemoteDisplay request belongs to the list of handles returned by the Lower Tester in the images-listing object.

- Notes

  The Remote Display function is tested in BIP/RDI/FSF/BV-09-C [Go to Next Image – Client Side] and BIP/RDI/FSF/BV-10-C [Go to Previous Image – Client Side] (Application Parameters values NextImage and PreviousImage only).

BIP/RDR/MFS/BV-26-C [Act as Remote Display Feature Responder – Server Side]

- Test Purpose

  Verify that the IUT correctly supports all the mandatory functions of a Remote Display server.

- Reference

  [2] 4.3.6, 4.4.6.3, 4.5.1, 4.5.2, 4.5.3, 4.5.5, 4.5.6

- Initial Condition

  - There is an active OBEX Remote Display Service session where the IUT is the OBEX server and the Lower Tester is the OBEX client.

- Test Procedure

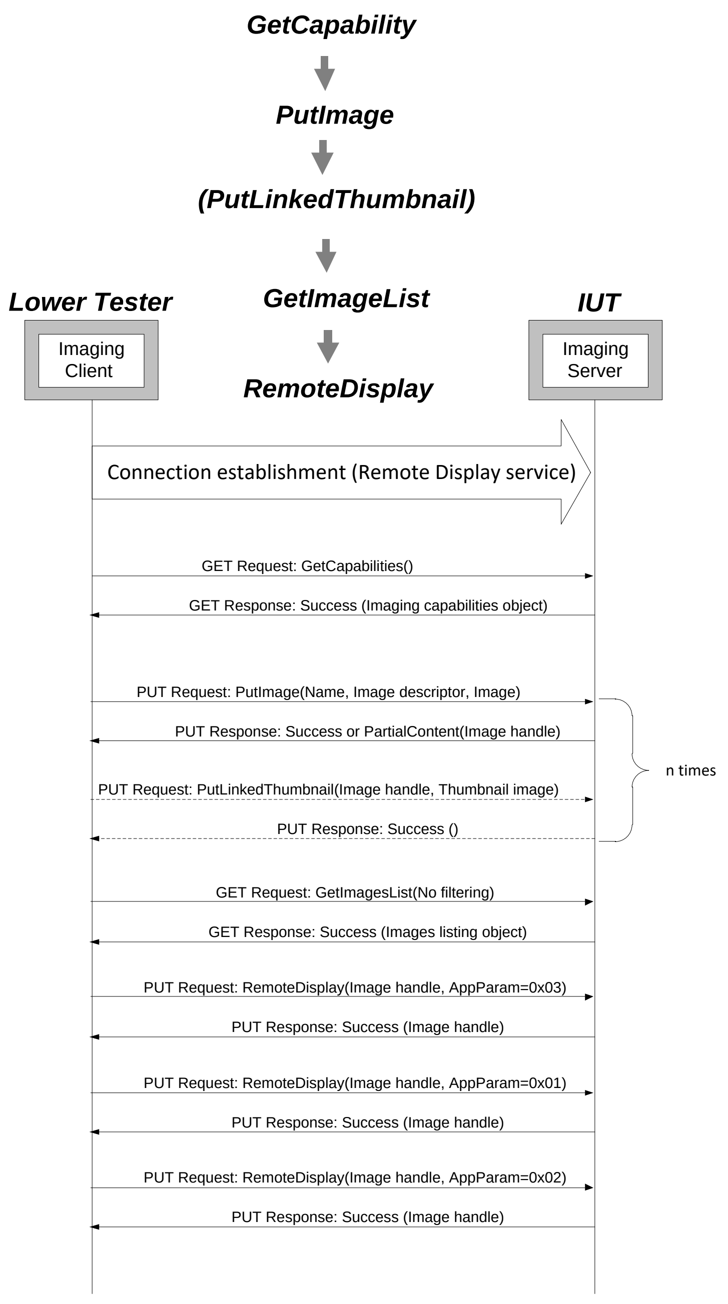

GetCapability
PutImage
(PutLinkedThumbnail)
GetImageList
RemoteDisplay
Lower Tester
Imaging Client
IUT
Imaging Server
Connection establishment (Remote Display service)
GET Request: GetCapabilities()
GET Response: Success (Imaging capabilities object)
PUT Request: PutImage(Name, Image descriptor, Image)
PUT Response: Success or PartialContent(Image handle)
n times
PUT Request: PutLinkedThumbnail(Image handle, Thumbnail image)
PUT Response: Success ()
GET Request: GetImagesList(No filtering)
GET Response: Success (Images listing object)
PUT Request: RemoteDisplay(Image handle, AppParam=0x03)
PUT Response: Success (Image handle)
PUT Request: RemoteDisplay(Image handle, AppParam=0x01)
PUT Response: Success (Image handle)
PUT Request: RemoteDisplay(Image handle, AppParam=0x02)
PUT Response: Success (Image handle)

Figure 4.37: BIP/RDR/MFS/BV-26-C [Act as Remote Display Feature Responder – Server Side]

1. In the above sequence chart, n is at least equal to three and will be chosen at random by the Lower Tester.
2. The Lower Tester retrieves the imaging-capabilities object of the IUT and checks the formats that the IUT supports.
3. The Lower Tester pushes a series of images to the IUT. At least three images are pushed. The format in which each image is sent is chosen at random by the Lower Tester, selecting from the formats that the IUT has declared as supported in its imaging-capabilities object.
4. In response to each PutImage request, the IUT may or may not request the associated imaging thumbnail from the Lower Tester.
5. The Lower Tester retrieves the list of images available for display on the IUT. Note that the list of images is equal to or larger than three.
6. The Lower Tester chooses one image to display by issuing a RemoteDisplay request with the Application Parameters header set to SelectImage.
7. The IUT displays the corresponding image and returns a Success code.
8. The Lower Tester orders the next image to be displayed by issuing a RemoteDisplay request with NextImage as the Application Parameters header.
9. The IUT displays the next image (as recommended in the profile, it is most desirable that the next-previous order corresponds to the order of the image handles in the images-listing object that was returned by the IUT in step 2).
10. The Lower Tester orders the previous image to be displayed by issuing a RemoteDisplay request with PreviousImage as the Application Parameters header.

- Expected Outcome

  Pass verdict

  The RemoteDisplay responses are well formatted. The Lower Tester will specifically verify that the RemoteDisplay responses contain a valid image handle.

  The image handles returned by the IUT in the RemoteDisplay responses are the correct ones (i.e., the ones corresponding to the image that is being displayed).

  The IUT accepts the images that are being pushed by the Lower Tester regardless of their format (given that the Lower Tester will only send formats that have been advertised as supported by the IUT in its imaging-capabilities object).

  After having performed “display next” and “display previous” operations, the IUT again is displaying the first image that it first displayed.

- Notes

  The Remote Display function is tested in BIP/RDR/FSF/BV-11-C [Go to Next Image – Server Side] and BIP/RDR/FSF/BV-12-C [Go to Previous Image – Server Side] (Application Parameters values NextImage and PreviousImage only).

4.6 Push Images

4.6.1 Push Images

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator has the ability to push images to the Responder in a format supported by the Responder, after receiving the Responder’s imaging-capabilities object following a GetCapabilities request, together with their attachments (if needed) and their thumbnail (if requested by the Responder).

Imaging Responder: Verify that the Imaging Responder can correctly respond to a GetCapabilities request from the Imaging Initiator and properly receive the images and possibly attachments sent by the Initiator after receiving a PutImage request.

- Reference

  [2] 4.3.1

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has started the Image Push feature, is connected to the Imaging Responder, and has successfully performed all authentication procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication procedures, if any.

- Test Case Configuration

Test Case
BIP/IPSI/PSH/BV-01-C [Push Images]
BIP/IPSR/PSH/BV-01-C [Push Images]

Table 4.28: Push Images test cases

- Test Procedure

  Imaging Initiator:

  The Initiator sends a GetCapabilities request to the Imaging Responder to retrieve information about the capabilities of the Imaging Responder.

  An image to push is selected on the Initiator and is pushed to the Responder, possibly together with associated attachments.

  Imaging Responder:

  The Imaging Responder accepts the images and possibly the attachments sent by the Initiator, and it may inform the user of the Imaging Responder about the new imaging files.

- Expected Outcome

  Pass verdict

  Imaging Initiator:

  No error is reported. The user may be informed of the success of the push operations.

  If the Imaging Responder requests a PutLinkedThumbnail, the Initiator is able to respond and provide the correct matching thumbnail.

  Imaging Responder:

  The Responder correctly accepts the images and is able to use them (indicating that the format under which the images were sent was indeed chosen by the Initiator in accordance with the capabilities of the Responder).

4.6.2 Push Images – GetCapabilities not supported by Initiator

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator has the ability to push images to the Responder, when GetCapabilities is not supported, and their thumbnail (if requested by the Responder).

  Imaging Responder: Verify that the Imaging Responder can correctly respond to a PutImage request from the Imaging Initiator when GetCapabilities is not supported.

- Reference

  [2] 4.3.1

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has started the Image Push feature, is connected to the Imaging Responder, and has successfully performed all authentication procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication procedures, if any.

- Test Case Configuration

Test Case
BIP/IPSI/PSH/BV-02-C [Push Images – GetCapabilities not supported by Initiator]
BIP/IPSR/PSH/BV-02-C [Push Images – GetCapabilities not supported by Initiator]

Table 4.29: Push Images – GetCapabilities not supported by Initiator test cases

- Test Procedure

  Imaging Initiator: An image to push is selected on the Initiator and is pushed to the Responder.

  Imaging Responder: The Imaging Responder accepts the images if the images are in a supported format, and it may inform the user of the Imaging Responder about the new imaging files.

- Expected Outcome

  Pass verdict

  Imaging Initiator:

  No error is reported. The user may be informed of the success of the push operations.

  If the Imaging Responder requests a PutLinkedThumbnail, the Initiator is able to respond and provide the correct matching thumbnail.

  Imaging Responder:

  The Responder correctly accepts the images if the images are in a supported format and is able to use them.

BIP/IPSR/PSH/BI-01-C [Push Images – Unsupported PutImage Request by Initiator]

- Test Purpose

  Imaging Initiator: Verify that the IUT (Imaging Responder) can appropriately respond to a PutImage request from the Lower Tester (Imaging Initiator) when GetCapabilities is not supported and an unsupported image format is chosen. The unsupported format can be derived from the BIP IXIT [9].

Imaging Responder: Verify that the Imaging Responder can correctly respond to a PutImage request from the Imaging Initiator when GetCapabilities is not supported.

- Reference

  [2] 4.3.1, 5.3

- Initial Condition

  - The IUT has entered the Bluetooth Imaging Mode, is connected to the Lower Tester, and has successfully performed all authentication procedures, if any.
  - Image format not listed in the IXIT as supported [9] is selected by the Lower Tester.

- Test Procedure

  An image to push is selected on the Lower Tester with a format that is not supported by the IUT and is pushed to the IUT. The IUT responds with an OBEX error.

- Expected Outcome

  Pass verdict

  The IUT responds appropriately with an OBEX error.

- Notes

  In the case that all possible formats are supported by the IUT this test case is waived.

4.7 Pull Images

4.7.1 Pull Images

- Test Purpose

  Imaging Initiator: Verify the Imaging Initiator has the ability to browse through the images available on the Responder and retrieve them in the format of its choice, together with their imaging thumbnail (if needed) and their attachments (if desired).

  Imaging Responder: Verify that the Imaging Responder can correctly advertise its locally stored images and deliver them to the Initiator upon request, possibly followed by their thumbnail versions and associated attachments.

- Reference

  [2] 4.3.2

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has started the Image Pull feature, is connected to the Imaging Responder, and has successfully performed all authentication related procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication procedures, if any. The Imaging Responder has at least one image stored locally.

- Test Case Configuration

Test Case
BIP/IPLI/PLL/BV-01-C [Pull Images]
BIP/IPLR/PLL/BV-01-C [Pull Images]

Table 4.30: Pull Images test cases

- Test Procedure

  Imaging Initiator:

  1. Select the image to retrieve from the list of images available on the Imaging Responder.
  2. The user may be given the opportunity to filter the list of images available on the Imaging Responder, by encoding, pixel-size, creation or modification dates. If so, filtering is used.
  3. Retrieve the selected image, together with its associated attachments (if any) and in a format supported by the Imaging Initiator.
  4. Repeat step 3 as many times as necessary to confirm that all the images in the list can be transferred correctly.

  Imaging Responder:

  5. The user of the Imaging Responder might be prompted for approval of the Image Pull requests coming for the Imaging Initiator. If so, the user approves the transfer of the imaging data.

- Expected Outcome

  <u>Pass verdict</u>

  Imaging Initiator:

  - The Imaging Initiator correctly selects available filtering parameters from the Responder’s imaging-capabilities object.
  - The attributes of each image enumerated in the images-listing object match the filtering parameters supplied by the Initiator.
  - All the images in the images-listing object can be retrieved in all their native and variant forms.
  - The Imaging Initiator selects and specifies the variant from the image-properties object.
  - Attachments to the image are correctly transferred.
  - The properties of each retrieved image (pixel size, encoding, etc.) conform to the image descriptor supplied with the GetImage request.

  Imaging Responder:

  - No error is reported. The user may be informed of the completion of the pull operations.
  - The list of images displayed at Imaging Initiator has been filtered correctly in the Imaging Responder.
  - All the variant forms enumerated in the image-properties object are available from the Imaging Responder.
  - All the enumerated filtering parameters in the imaging-capabilities object are effective in the Imaging Responder.

4.8 Remotely Order Advanced Print Jobs

4.8.1 Advanced Printing – Normal

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator has the ability to correctly specify print jobs.

  Imaging Responder: Verify that the Imaging Responder can accurately print the images in accordance with the print jobs specified by the Imaging Initiator.

- Reference

  [2] 4.3.3, 4.4.5.1

- Initial Condition
  - Imaging Initiator: The Imaging Initiator has started the Advanced Image Printing feature, is connected to the Imaging Responder, and has successfully performed all authentication procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication procedures, if any.

- Test Case Configuration

Test Case
BIP/AIPI/ADP/BV-01-C [Advanced Printing – Normal]
BIP/AIPR/ADP/BV-01-C [Advanced Printing – Normal]

Table 4.31: Advanced Printing – Normal test cases

- Test Procedure

  Imaging Initiator:

  The user may be informed of the capabilities of the Imaging Responder.

  The user of the Imaging Initiator selects one or a number of images to be printed, so that the full set of Advanced Image Printing capabilities supported by the Imaging Responder can be utilized.

  The user initiates the print job in accordance with the user’s choice of images and printing options.

  Imaging Responder:

  The Imaging Responder prints images that are stored locally on the Imaging Initiator device.

- Expected Outcome

  Pass verdict

  Imaging Initiator:

  - The user may be informed of the success of the print jobs.

  Imaging Responder:

  - The responder successfully prints one or more images.

- The images that are printed are the images that had been selected on the Imaging Initiator.
- The printing output corresponds to the options and settings that were specified by the Imaging Initiator.

4.9 Automatically Archive Images

4.9.1 Automatically Archive Images

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator can command the Imaging Responder to archive its locally stored images and properly deliver those images to the Responder. The process starts either automatically when the Initiator comes within Bluetooth range of the responder, or after a user interaction on the Initiator.

  Imaging Responder: Verify that the Imaging Responder has the ability to perform the image archiving operation commanded by the Initiator.

- Reference

  [2] 4.3.4

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has entered the Automatic Archive feature, is connected to the Imaging Responder, and has successfully performed all authentication related, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication related, if any. The Imaging Initiator has at least one image stored locally.

- Test Case Configuration

Test Case
BIP/AAI/ACH/BV-01-C [Automatically Archive Images]
BIP/AAR/ACH/BV-01-C [Automatically Archive Images]

Table 4.32: Automatically Archive Images test cases

- Test Procedure

  Imaging Initiator:

  (Depending on the implementation, it might be enough to enter the Bluetooth Automatic Archive feature to trigger the archiving operation.)

  The user of the Imaging Initiator triggers the archiving operation.

  Imaging Responder:

  The Imaging Responder downloads the imaging files from the Imaging Initiator according to its algorithm.

- Expected Outcome

  Pass verdict

  Imaging Initiator: No error is reported.

Imaging Responder: The Imaging Responder was able to download the images the Imaging Initiator according to its algorithm and settings (for instance, whether the Responder is to download all the images stored on the Initiator or only selected ones).

4.10 Remotely Capture Images

4.10.1 Receive Monitoring Images and Have Full Size Images Stored

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator has the ability to correctly receive monitoring images and remotely trigger the capture of full size images to be stored on the Imaging Responder.

  Imaging Responder: Verify that the Imaging Responder can accurately deliver monitoring images to the Initiator while storing the corresponding full size images when requested by the Initiator.

- Reference

  [2] 4.3.5, 4.4.5.5

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has started the Remote Camera feature, is connected to the Imaging Responder, and has successfully performed all authentication procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication related procedures, if any.

- Test Case Configuration

Test Case
BIP/RCI/RMC/BV-02-C [Receive Monitoring Images and Have Full Size Images Stored]
BIP/RCR/RMC/BV-02-C [Receive Monitoring Images and Have Full Size Images Stored]

Table 4.33: Receive Monitoring Images and Have Full Size Images Stored test cases

- Test Procedure

  Imaging Initiator: The Imaging Initiator requests monitoring images from the Imaging Responder and receives them. Depending on the implementation of the Remote Camera feature, the Imaging Initiator might also have the ability to trigger the Imaging Responder to actually capture and save an image. The Imaging Initiator may or may not retrieve the stored image.

  Imaging Responder: The Imaging Responder produces monitoring images and sends them back to the Imaging Initiator. If requested to do by the Imaging Initiator, the Imaging Responder will capture and store an image. If requested by the Initiator, the Responder sends the locally stored images back to the Initiator.

- Expected Outcome

  <u>Pass verdict</u>

  Imaging Initiator:

  The Imaging Initiator was able to trigger the Imaging Responder to capture an image and the monitoring images received from the Imaging Responder correspond to the expectations of the Imaging Initiator.

If the Imaging Initiator retrieved other imaging data from the Responder (i.e., images in a format different from the monitoring image format), those images are useable by the Initiator (i.e., in a format that is supported by the Initiator).

Imaging Responder:

The Imaging Responder was able to transmit monitoring images to the Initiator. Upon request by the Initiator, the Responder was capture images, store the corresponding image files, and, if necessary, transmit the image file to the Initiator in an appropriate format.

4.11 Remotely Control a Display

4.11.1 Control the Display of a Remote Device

- Test Purpose

  Imaging Initiator: Verify that the Imaging Initiator has the ability to send images to the Responder and control their display sequence.

  Imaging Responder: Verify that the Imaging Responder has the ability to receive images from the Initiator and display them following the instructions of the Initiator.

- Reference

  [2] 4.3.6

- Initial Condition

  - Imaging Initiator: The Imaging Initiator has entered the Remote Display feature, is connected to the Imaging Responder, and has successfully performed all authentication procedures, if any.
  - Imaging Responder: The Imaging Responder has entered the Bluetooth Imaging Mode, is connected to an Imaging Initiator, and has successfully performed all authentication procedures, if any. The Imaging Responder is not displaying an image.

- Test Case Configuration

Test Case
BIP/RDI/RMD/BV-01-C [Control the Display of a Remote Device]
BIP/RDR/RMD/BV-01-C [Control the Display of a Remote Device]

Table 4.34: Control the Display of a Remote Device test cases

- Test Procedure

  Imaging Initiator:

  The Imaging Initiator may start by sending a number of images to the Imaging Responder.

  The Imaging Initiator may inform the user of the images that are available for display on the Imaging Responder via a list of images displayed to the user.

  The user or an automatic process selects the images to be displayed or simply orders the Responder to proceed to the next or previous image (next and previous refer to the display order as determined by the Imaging Responder).

  Imaging Responder:

  The Imaging Responder displays images in accordance to the instructions from the Imaging Initiator.

- Expected Outcome

  <u>Pass verdict</u>

  Imaging Initiator: No error is reported.

  Imaging Responder: The Responder displays the image following the Initiator’s instructions.

5 Test case mapping

The Test Case Mapping Table (TCMT) maps test cases to specific requirements in the ICS. The IUT is tested in all roles for which support is declared in the ICS document.

The columns for the TCMT are defined as follows:

Item: Contains a logical expression based on specific entries from the associated ICS document. Contains a logical expression (using the operators AND, OR, NOT as needed) based on specific entries from the applicable ICS document(s). The entries are in the form of y/x references, where y corresponds to the table number and x corresponds to the feature number as defined in the ICS document for Basic Imaging Profile (BIP) [5].

Feature: A brief, informal description of the feature being tested.

Test Case(s): The applicable test case identifiers are required for Bluetooth Qualification if the corresponding y/x references defined in the Item column are supported. Further details about the function of the TCMT are elaborated in [7].

For the purpose and structure of the ICS/IXIT, refer to [7].

Item	Feature	Test Case(s)
Service Records		
BIP 1/2 OR BIP 1/4 OR BIP 1/6 OR BIP 1/8 OR BIP 1/10 OR BIP 1/12	Imaging Responder SDP service	BIP/SR/SGSIT/ATTR/BV-01-C BIP/SR/SGSIT/ATTR/BV-03-C BIP/SR/SGSIT/ATTR/BV-04-C BIP/SR/SGSIT/ATTR/BV-05-C BIP/SR/SGSIT/ATTR/BV-06-C BIP/SR/SGSIT/OFFS/BV-01-C BIP/SR/SGSIT/SERR/BV-01-C
BIP 0/3 AND (BIP 1/2 OR BIP 1/4 OR BIP 1/6 OR BIP 1/8 OR BIP 1/10 OR BIP 1/12)	Imaging Responder SDP attribute: BluetoothProfileDescriptorList – BIP v1.2	BIP/SR/SGSIT/ATTR/BV-02-C
BIP 39/3 AND (BIP 1/2 OR BIP 1/4 OR BIP 1/6 OR BIP 1/8 OR BIP 1/10 OR BIP 1/12)	Imaging Responder SDP attribute: GoepL2capPsm	BIP/SR/SGSIT/ATTR/BV-07-C
BIP 1/5	Referenced Objects service record	BIP/AIPI/SGSIT/SERR/BV-02-C BIP/AIPI/SGSIT/ATTR/BV-08-C BIP/AIPI/SGSIT/ATTR/BV-09-C BIP/AIPI/SGSIT/ATTR/BV-11-C BIP/AIPI/SGSIT/OFFS/BV-02-C
BIP 0/3 AND BIP 1/5	Referenced Objects SDP attribute: BluetoothProfileDescriptorList – BIP v1.2	BIP/AIPI/SGSIT/ATTR/BV-10-C
BIP 39/3 AND BIP 1/5	Referenced Objects SDP attribute: GoepL2capPsm	BIP/AIPI/SGSIT/ATTR/BV-12-C

Item	**Feature**	**Test Case(s)**
BIP 1/7	Automatic Archive service record	BIP/AAI/SGSIT/ATTR/BV-13-C BIP/AAI/SGSIT/ATTR/BV-14-C BIP/AAI/SGSIT/ATTR/BV-16-C BIP/AAI/SGSIT/OFFS/BV-03-C BIP/AAI/SGSIT/SERR/BV-03-C
BIP 0/3 AND BIP 1/7	Automatic Archive SDP attribute: BluetoothProfileDescriptorList – BIP v1.2	BIP/AAI/SGSIT/ATTR/BV-15-C
BIP 39/3 AND BIP 1/7	Automatic Archive SDP attribute: GoepL2capPsm	BIP/AAI/SGSIT/ATTR/BV-17-C
BIP 1/1	Successful Connection with future SDP Record value – BIP Image Push Initiator	BIP/IPSI/CGSIT/SFC/BV-01-C
Image Push Feature		
BIP 2/1	Get Imaging Capabilities – Client	BIP/IPSI/FFC/BV-01-C
BIP 2/2	Put an Image – Normal – Client	BIP/IPSI/FFC/BV-09-C
BIP 2/1 AND BIP 3/2 AND BIP 2/2	Use the Capabilities Object to Put Images – Client	BIP/IPSI/MFS/BV-01-C
BIP 2/2 AND BIP 2/3	Push a Thumbnail – Client	BIP/IPSI/MFS/BV-03-C
BIP 2/2 AND BIP 2/4	Push an Attachment – Client	BIP/IPSI/MFS/BV-05-C
BIP 4/1	Get Imaging Capabilities – Server	BIP/IPSR/FFC/BV-02-C
BIP 4/2	Put an Image – Normal – Server	BIP/IPSR/FFC/BV-10-C
BIP 4/2 AND BIP 6/1	Put an Image – Request for Thumbnail – Server	BIP/IPSR/FFC/BV-11-C
BIP 4/1 AND BIP 4/2	Provide the Capability Object – Server	BIP/IPSR/MFS/BV-02-C
BIP 6/1 AND BIP 4/2 AND BIP 4/3	Request a Thumbnail – Server	BIP/IPSR/MFS/BV-04-C
BIP 4/2 AND BIP 4/4	Receive an Attachment – Server	BIP/IPSR/MFS/BV-06-C
Image Pull Feature		
BIP 7/1	Get Imaging Capabilities – Client	BIP/IPLI/FFC/BV-01-C
BIP 7/2	Get Images List – Client	BIP/IPLI/FFC/BV-03-C
BIP 7/2 AND BIP 8/2	Get Total Number of Images – Client	BIP/IPLI/FFC/BV-05-C
BIP 7/2 AND BIP 8/3	Get the List of Recently Captured Images – Client	BIP/IPLI/FFC/BV-07-C
BIP 7/2 AND BIP 7/4 AND BIP 8/1	Use the Capability Object to Pull Images Lists – Client	BIP/IPLI/MFS/BV-07-C
BIP 7/2 AND BIP 7/3 AND BIP 7/4 AND BIP 9/2	Use the Image Property Object – Client	BIP/IPLI/MFS/BV-08-C
BIP 7/2 AND BIP 7/4 AND BIP 9/1	Pull an Image as Thumbnail – Client	BIP/IPLI/MFS/BV-09-C
BIP 7/3 AND BIP 7/5 AND BIP 9/1	Pull a Thumbnail – Client	BIP/IPLI/MFS/BV-10-C
BIP 7/2 AND BIP 7/4 AND BIP 9/3	Pull a Native Image – Client	BIP/IPLI/MFS/BV-11-C

Item	Feature	Test Case(s)
BIP 7/7 AND BIP 7/2	Delete an Image – Client	BIP/IPLI/MFS/BV-13-C
BIP 7/2 AND BIP 7/3 AND BIP 7/4 AND BIP 7/6	Get an Attachment – Client	BIP/IPLI/MFS/BV-15-C
BIP 10/1	Get Imaging Capabilities – Server	BIP/IPLR/FFC/BV-02-C
BIP 10/2	Get Images List – Server	BIP/IPLR/FFC/BV-04-C
BIP 10/2	Get Total Number of Images – Server	BIP/IPLR/FFC/BV-06-C
BIP 10/2 AND BIP 13/1	Get the List of Recently Captured Images – Server	BIP/IPLR/FFC/BV-08-C
BIP 10/1 AND BIP 10/2 AND BIP 10/3 AND BIP 10/4 AND BIP 10/5	Pull Images – Server	BIP/IPLR/MFS/BV-12-C
BIP 10/2 AND BIP 10/7	Delete an Image – Server	BIP/IPLR/MFS/BV-14-C
BIP 10/2 AND BIP 10/3 AND BIP 10/6	Provide an Attachment – Server	BIP/IPLR/MFS/BV-16-C
Advanced Printing Feature		
BIP 14/1	Get Imaging Capabilities – Client	BIP/AIPI/FFC/BV-01-C
BIP 14/2	Start Advanced Printing – Client	BIP/AIPI/FSF/BV-05-C
BIP 14/1 AND BIP 14/2 AND BIP 14/3	Use the Imaging-Capabilities Object – Client	BIP/AIPI/MFS/BV-17-C
BIP 15/1	Get Imaging Capabilities – Server	BIP/AIPR/FFC/BV-02-C
BIP 15/2	Start Advanced Printing – Server	BIP/AIPR/FSF/BV-06-C
BIP 15/1 AND BIP 15/2 AND BIP 15/3 AND BIP 15/4	Use the Imaging-Capabilities Object – Server	BIP/AIPR/MFS/BV-18-C
Automatic Archive Feature		
BIP 17/8	Start Archiving – Client	BIP/AAI/FSF/BV-07-C
BIP 17/1	Get Imaging Capabilities – Server	BIP/AAI/FFC/BV-02-C
BIP 17/2	Get Images List – Server	BIP/AAI/FFC/BV-04-C
BIP 17/2	Get Total Number of Images – Server	BIP/AAI/FFC/BV-06-C
BIP 17/2 AND BIP 18/1	Get the List of Recently Captured Images – Server	BIP/AAI/FFC/BV-08-C
BIP 17/1 AND BIP 17/2 AND BIP 17/3 AND BIP 17/4 AND BIP 17/5	Pull Images – Server	BIP/AAI/MFS/BV-12-C
BIP 17/2 AND BIP 17/7	Delete an Image – Server	BIP/AAI/MFS/BV-14-C
BIP 17/2 AND BIP 17/3 AND BIP 17/6	Provide an Attachment – Server	BIP/AAI/MFS/BV-16-C
BIP 19/8	Start Archiving – Server	BIP/AAR/FSF/BV-08-C
BIP 19/1	Get Imaging Capabilities – Client	BIP/AAR/FFC/BV-01-C
BIP 19/2	Get Images List – Client	BIP/AAR/FFC/BV-03-C
BIP 19/2 AND BIP 20/2	Get Total Number of Images – Client	BIP/AAR/FFC/BV-05-C
BIP 19/2 AND BIP 20/3	Get the List of Recently Captured Images – Client	BIP/AAR/FFC/BV-07-C

Item	Feature	Test Case(s)
BIP 19/1 AND BIP 20/1	Use the Capability Object to Pull Images Lists – Client	BIP/AAR/MFS/BV-07-C
BIP 19/2 AND BIP 19/3 AND BIP 19/4 AND BIP 21/2	Use the Image Property Object – Client	BIP/AAR/MFS/BV-08-C
BIP 19/2 AND BIP 19/4 AND BIP 21/1	Pull an Image as Thumbnail – Client	BIP/AAR/MFS/BV-09-C
BIP 19/2 AND BIP 19/5	Pull a Thumbnail – Client	BIP/AAR/MFS/BV-10-C
BIP 19/2 AND BIP 19/4 AND BIP 21/3	Pull a Native Image – Client	BIP/AAR/MFS/BV-11-C
BIP 19/2 AND BIP 19/7	Delete an Image – Client	BIP/AAR/MFS/BV-13-C
BIP 9/2 AND BIP 19/3 AND BIP 19/6	Get an Attachment – Client	BIP/AAR/MFS/BV-15-C
Remote Camera Feature		
BIP 22/1	Get a Monitoring Image – Client	BIP/RCI/FSF/BV-01-C BIP/RCI/FSF/BV-02-C
BIP 22/1 AND BIP 22/2 AND BIP 22/3 AND BIP 23/2	Take and Retrieve an Image – Client	BIP/RCI/MFS/BV-19-C
BIP 22/1 AND BIP 22/3 AND BIP 23/1	Take and Retrieve an Image as Imaging Thumbnail – Client	BIP/RCI/MFS/BV-20-C
BIP 22/1 AND BIP 22/3 AND BIP 23/3	Take an Image and Retrieve its Native Version – Client	BIP/RCI/MFS/BV-21-C
BIP 22/1 AND BIP 22/4 AND BIP 23/1	Take and Retrieve an Image as Imaging Thumbnail using the GetLinkedThumbnail Function – Client	BIP/RCI/MFS/BV-22-C
BIP 24/1	Get a Monitoring Image – Server	BIP/RCR/FSF/BV-03-C BIP/RCR/FSF/BV-04-C
BIP 24/1 AND BIP 24/2 AND BIP 24/3 AND BIP 24/4	Act as Remote Camera Imaging Responder – Server	BIP/RCR/MFS/BV-23-C
Remote Display Feature		
BIP 25/1	Get Imaging Capabilities – Client	BIP/RDI/FFC/BV-01-C
BIP 25/2	Put an Image – Normal – Client	BIP/RDI/FFC/BV-09-C
BIP 25/5 AND BIP 26/1	Remotely Control a Display - Client	BIP/RDI/FSF/BV-09-C BIP/RDI/FSF/BV-10-C
BIP 25/4	Get Images List – Client	BIP/RDI/FFC/BV-03-C BIP/RDI/FFC/BV-05-C BIP/RDI/FFC/BV-07-C
BIP 25/2 AND BIP 25/5	Select an Image to Display After Having Pushed It – Client	BIP/RDI/MFS/BV-24-C
BIP 25/4 AND BIP 25/5	Select an Image to Display from the Images-Listing Object – Client	BIP/RDI/MFS/BV-25-C
BIP 29/1	Get Imaging Capabilities – Server	BIP/RDR/FFC/BV-02-C
BIP 29/2	Put an Image – Normal – Server	BIP/RDR/FFC/BV-10-C
BIP 29/2 AND BIP 32/1	Put an Image – Request for Thumbnail – Server	BIP/RDR/FFC/BV-11-C

Item	Feature	Test Case(s)
BIP 29/5	Remotely Control a Display - Server	BIP/RDR/FSF/BV-11-C BIP/RDR/FSF/BV-12-C
BIP 29/4	Get Images List – Server	BIP/RDR/FFC/BV-04-C BIP/RDR/FFC/BV-06-C BIP/RDR/FFC/BV-08-C
BIP 29/1 AND BIP 29/2 AND BIP 29/4 AND BIP 29/5	Act as Remote Display Feature Responder – Server	BIP/RDR/MFS/BV-26-C
BIP 25/1 AND BIP 25/4	Use the Capability Object to Pull Images Lists – Client	BIP/RDI/MFS/BV-07-C
Image Push		
BIP 1/1 AND BIP 2/1	Push Images - Image Push Initiator	BIP/IPSI/PSH/BV-01-C
BIP 1/2	Push Images - Image Push Responder	BIP/IPSR/PSH/BV-01-C
BIP 1/1 AND NOT BIP 2/1	Push Images – GetCapabilities not supported by Initiator – IUT is Initiator	BIP/IPSI/PSH/BV-02-C
BIP 1/2	Push Images – GetCapabilities not supported by Initiator – IUT is Responder	BIP/IPSR/PSH/BV-02-C
BIP 1/2	Push Images – Unsupported PutImage Request by Initiator – IUT is Responder	BIP/IPSR/PSH/BI-01-C
Image Pull		
BIP 1/3	Pull Images - Image Pull Initiator	BIP/IPLI/PLL/BV-01-C
BIP 1/4	Pull Images - Image Pull Responder	BIP/IPLR/PLL/BV-01-C
Advanced Printing		
BIP 1/5	Advanced Printing - Advanced Image Printing Initiator	BIP/AIPI/ADP/BV-01-C
BIP 1/6	Advanced Printing - Advanced Image Printing Responder	BIP/AIPR/ADP/BV-01-C
Automatic Archive		
BIP 1/7	Automatically Archive Images - Automatic Archive Initiator	BIP/AAI/ACH/BV-01-C
BIP 1/8	Automatically Archive Images - Automatic Archive Responder	BIP/AAR/ACH/BV-01-C
Remote Camera		
BIP 1/9	Remotely Capture Images - Remote Camera Initiator	BIP/RCI/RMC/BV-02-C
BIP 1/10	Remotely Capture Images - Remote Camera Responder	BIP/RCR/RMC/BV-02-C
Remote Display		
BIP 1/11	Control the Display of a Remote Device - Remote Display Initiator	BIP/RDI/RMD/BV-01-C

Item	Feature	Test Case(s)
BIP 1/12	Control the Display of a Remote Device - Remote Display Responder	BIP/RDR/RMD/BV-01-C
GOEP 2.0 or later		
BIP 0/3 AND BIP 39/2 AND (BIP 1/2 OR BIP 1/6 OR BIP 1/8 OR BIP 1/12)	Server: GOEP v2.0 or later Features backward compatibility, PUT over RFCOMM	BIP/SR/GOEP/BC/BV-01-C
BIP 0/3 AND BIP 39/2 AND (BIP 1/1 OR BIP 1/5 OR BIP 1/7 OR BIP 1/11)	Client: GOEP v2.0 or later Features backward compatibility, PUT over RFCOMM	BIP/CL/GOEP/BC/BV-02-C
BIP 0/3 AND BIP 39/2 AND (BIP 1/4 OR BIP 1/5 OR BIP 1/7 OR BIP 1/10 OR BIP 1/12)	Server: GOEP v2.0 or later Features backward compatibility, GET over RFCOMM	BIP/SR/GOEP/BC/BV-03-C
BIP 0/3 AND BIP 39/2 AND (BIP 1/3 OR BIP 1/6 OR BIP 1/8 OR BIP 1/9 OR BIP 1/11)	Client: GOEP v2.0 or later Features backward compatibility, GET over RFCOMM	BIP/CL/GOEP/BC/BV-04-C
BIP 0/3 AND BIP 39/1 AND BIP 39/3	Client: GOEP v20 or later, OBEX over L2CAP	BIP/CL/GOEP/CON/BV-01-C
BIP 0/3 AND BIP 39/1 AND BIP 39/3	Server: GOEP v20 or later, OBEX over L2CAP	BIP/SR/GOEP/SRM/BI-03-C
BIP 0/3 AND BIP 37/14 AND (BIP 1/1 OR BIP 1/5 OR BIP 1/7 OR BIP 1/11)	Client: OBEX SRM, Image Push, Advanced Image Printing, Automatic Archive, Remote Display	BIP/CL/GOEP/SRM/BV-01-C BIP/CL/GOEP/SRM/BV-03-C
BIP 0/3 AND BIP 38/14 AND (BIP 1/2 OR BIP 1/6 OR BIP 1/8 OR BIP 1/12)	Server: OBEX SRM, Image Push, Advanced Image Printing, Automatic Archive, Remote Display	BIP/SR/GOEP/SRM/BV-04-C BIP/SR/GOEP/SRM/BI-02-C
BIP 0/3 AND BIP 37/14 AND (BIP 1/3 OR BIP 1/6 OR BIP 1/8 OR BIP 1/9 OR BIP 1/11)	Client: OBEX SRM, Image Pull, Advanced Image Printing, Automatic Archive, Remote Camera, Remote Display	BIP/CL/GOEP/SRM/BV-05-C BIP/CL/GOEP/SRM/BV-07-C
BIP 0/3 AND BIP 37/14 AND BIP 37/15 AND BIP 37/16 AND (BIP 1/3 OR BIP 1/6 OR BIP 1/8 OR BIP 1/9 OR BIP 1/11)	Client: Send/Receive OBEX SRMP header	BIP/CL/GOEP/SRMP/BV-05-C
BIP 0/3 AND BIP 38/14 AND (BIP 1/4 OR BIP 1/5 OR BIP 1/7 OR BIP 1/10 OR BIP 1/12)	Server: OBEX SRM	BIP/SR/GOEP/SRM/BV-08-C BIP/SR/GOEP/SRM/BI-05-C
BIP 0/3 AND BIP 37/14 AND BIP 37/15 AND (BIP 1/1 OR BIP 1/5 OR BIP 1/7 OR BIP 1/11)	Client: OBEX SRM, Receive OBEX SRMP header	BIP/CL/GOEP/SRMP/BV-01-C
BIP 0/3 AND BIP 38/14 AND BIP 38/16 AND (BIP 1/2 OR BIP 1/6 OR BIP 1/8 OR BIP 1/12)	Server: Send OBEX SRMP header	BIP/SR/GOEP/SRMP/BV-03-C

Item	Feature	Test Case(s)
BIP 0/3 AND BIP 37/14 AND BIP 37/16 AND (BIP 1/3 OR BIP 1/6 OR BIP 1/8 OR BIP 1/9 OR BIP 1/11)	Client: OBEX SRM, Send OBEX SRMP header	BIP/CL/GOEP/SRMP/BV-04-C
BIP 0/3 AND BIP 37/14 AND BIP 37/15 AND (BIP 1/3 OR BIP 1/6 OR BIP 1/8 OR BIP 1/9 OR BIP 1/11)	Client: Receive OBEX SRMP header	BIP/CL/GOEP/SRMP/BI-01-C BIP/CL/GOEP/SRMP/BV-06-C
BIP 0/3 AND BIP 38/14 AND BIP 38/15 AND (BIP 1/4 OR BIP 1/5 OR BIP 1/7 OR BIP 1/10 OR BIP 1/12)	Server: OBEX SRM	BIP/SR/GOEP/SRMP/BV-02-C BIP/SR/GOEP/SRMP/BI-02-C
BIP 0/3 AND (BIP 1/2 OR BIP 1/4 OR BIP 1/5 OR BIP 1/6 OR BIP 1/7 OR BIP 1/8 OR BIP 1/10 OR BIP 1/12)	Server: BIP v1.1 or later	BIP/SR/GOEP/ROB/BV-01-C
BIP 0/3 AND (BIP 1/2 OR BIP 1/4 OR BIP 1/5 OR BIP 1/6 OR BIP 1/7 OR BIP 1/8 OR BIP 1/10 OR BIP 1/12)	Server: BIP v1.1 or later	BIP/SR/GOEP/ROB/BV-02-C

Table 5.1: Test case mapping

6 Revision history and acknowledgments

Revision History

Publication Number	Revision Number	Date	Comments
	0.9	2001-09-20	Reorganization of the Test Suite Structure
	0.96	2002-05-15	BTI and BQRB comments included and approved
		2002-09-18	Correction of a remaining typo
0	1.0 Final	2002-11-15	Final
1	1.1	2004-08-18	Incorporate TSE 586 (Correction for TP/MFS/BV-20-C), TSE 587 (Correction for TP/MFS/BV-20-C), and TSE 588 (Correction for TP/MFS/BV-22-C) and made editorial changes
	1.1.1r1	2005-02-15	Editorial and Format changes. Change document number. Incorporate TSE 693 for TP/MFS/BV-01-C, TP/MFS/BV-07-C, and TP/MFS/BV-17-C. Incorporate TSE 706 for TCMT (TP/RMC/BV-02-I).
2	1.1.1	2005-03-04	Incorporate TSE 744 for TCMT (TP/SIR/BV-01-I, TP/SIR/BV-02-I, and TP/SIR/BV-03-I). Prepare for publication.
3	1.1.2	2005-03-07	Remove unneeded references to TP/RMC/BV-02-I from TCMT.
	1.1.3r0	2005-08-22	TSE 724: map TP/FSF/BV-11| BV-12 to Responder/Server role.
	1.1.3r1	2005-09-21	TSE 724: map TP/FSF/BV-11| BV-12 to Responder/Server role. In TCMT
4	1.1.3	2005-10-03	Prepare for publication.
	1.1.4r0-1	2008-09-01	TSE 2567: TP/MFS/BV-01-C TSE 2455: TP/FFC/BV-02-C Incorporate reviewer's comments—re-insert missing graphics
5	1.1.4	2008-12-02	Prepare for publication.
	1.1.5r0	2009-04-28	TSE 2867 TP/FFC/BV-10-C, TP/FFC/BV-09-C, TP/FFC/BV-11-C, update graphics
6	1.1.5	2009-07-28	Prepare for publication.
	1.1.6r0	2010-07-26	Update for OBEX 1.2
	1.1.6r1	2010-08-09	Updated conformance section.
7	1.1.6r2	2010-08-17	Updated TCMT to match updated ICS table numbers, prepare for publication.

Publication Number	Revision Number	Date	Comments
	1.1.7r0-1	2011-10-28	TSE 4214: TP/SIR/BV-03-I: TCMT change TSE 4491: TP/PSH/BV-01-I: Remove extra pass/fail verdict criteria TSE 3827: TP/PSH/BV-01-I: TP, Test proc, Expected Outcome TSE 4493: TP/MFS/BV-01/02/03/04/05/06/07/08/09/11/12/13/14/18/19/21/22/23/24/25/26-C: Fix test case IDs in Notes section. Additional corrections by AC.
8	1.1.7	2012-03-30	Prepare for publication.
	1.2.0r0	2012-06-13	Version update to accommodate new specification version 1.2 TSE 4492: New test cases TP/PSH/BV-02-I and TP/PSH/BI-01-I
9	1.2.0	2012-07-24	Prepare for publication.
	1.2.1r1	2012-10-05	TSE 4494: Rename of test case TP/SIR/BV-03-I from “PIN” to “Security,” added reference, editorial correction to the initial condition, revised the test procedure and edited the pass/fail verdicts to expand the scope of the tests to devices with SM4 support. Added SM4 GAP ICS to the TCMT for test case TP/SIR/BV-03-I.
	1.2.1r2	2012-10-22	Updated TOC
	1.2.1r3	2012-11-15	TSE 5014: Add BIP 0/3 to the TCMT as an OR statement where BIP 0/2 appears so the test cases also map to v1.2,
10	1.2.1	2012-11-19	Prepare for Publication
	1.2.2r01	2013-09-26	TSE 5283: Updated Test Condition and added a Note section to TP/PSH/BI-01-I.
11	1.2.2	2013-12-03	Prepare for Publication
	1.2.3r00	2017-03-07	TSE 8754: Updated Test Spec Template and miscellaneous editorials.
	1.2.3r01	2017-05-09	Converted to new Test Case ID conventions as defined in TSTO v4.1.
12	1.2.3	2017-07-03	Approved by BTI. Prepared for TCRL 2017-1 publication.
	1.2.1.0	2018-11-09	Updated version number to 1.2.1.0 to align with adoption of the specification 1.2.1
13	1.2.1.0	2018-11-21	Approved by BTI. Prepared for TCRL 2018-2 publication.

Publication Number	Revision Number	Date	Comments
	p14r00–r03	2020-10-20 – 2020-12-09	TSE 15265 (rating 1): Updated TCMT and TCRL to synchronize TCIDs and descriptions. Consistency Checker fixes; integration review feedback to fix TCID order of GOEP test strings (in the TCMT and TCRL). Additional fixes to align w/ ICS (problem found during Launch Studio configuration). Backed out TCMT deletions from Consistency Checker; TCRL and TCMT will need to be updated via TSE to account for TCs that no longer exist in the test suite. Template-related editorials, including: moving revision history and contributors to the end of the doc; replacing Conformance and Pass/Fail Verdict Conventions text; updating TCID heading styles; adding publication number, assigning last published v1.2.1.0 as p13, and updating document number; updating copyright page and footer text (and logo) to align with most recent DMR.
14	p14	2020-12-22	Approved by BTI on 2020-12-02. Prepared for TCRL 2020-1 publication.
	p15r00	2021-03-30	TSE 16313 (rating 4): Updates to address consistency issues and missing test cases. Added existing TCs (showing only in TCMT/TCRL) to test cases in body of doc (added TCs: BIP/RDI/FFC/BV-03-C; BIP/AAR/FFC/BV-03-C, -05-C, and -07-C; BIP/AAR/MFS/BV-07-C – -11-C, -13-C, and -15-C; BIP/AAI/FFC/BV-06-C and -08-C; and BIP/AAI/MFS/BV-12-C, -14-C, and -16-C; affected existing TCs: BIP/IPLI/FFC/BV-03-C, -05-C, and -07-C; BIP/IPLI/MFS/BV-07-C – -11-C, -13-C, -15-C; BIP/IPLR/FFC/BV-06-C and -08-C; and BIP/IPLR/MFS/BV-12-C, -14-C, and -16-C) and added new TCs BIP/RDI/FFC/BV-05-C and -07-C; BIP/RDR/FFC/BV-06-C and -08-C; and BIP/RDI/MFS/BV-07-C. Updated TCMT and TCRL accordingly.
15	p15	2021-07-13	Approved by BTI on 2021-06-03. Prepared for TCRL 2021-1 publication.
	p16r00	2021-09-20	TSE 17475 (rating 2): Updated TCMT entries to better outline role support. Performed template-related formatting fixes. Updated the introduction text before the TCMT to align with the template. Updated copyright page to align with v2 of the DNMD. Added captions to figures.
16	p16	2022-01-25	Approved by BTI on 2021-12-19. Prepared for TCRL 2021-2 publication.

Publication Number	**Revision Number**	**Date**	**Comments**
	p17r00–r07	2023-10-04 – 2024-05-07	TSE 23937 (rating 1): Converted -I tests to -C tests. Deleted TCIDs BIP/AAI/SIR/BV-01-I – -03-I, BIP/AAR/SIR/BV-01-I – -03-I, BIP/AIPI/SIR/BV-01-I – -03-I, BIP/AIPR/SIR/BV-01-I – -03-I, BIP/IPLI/SIR/BV-01-I – -03-I, BIP/IPLR/SIR/BV-01-I – -03-I, BIP/IPSI/SIR/BV-01-I – -03-I, BIP/IPSR/SIR/BV-01-I – -03-I, BIP/RCI/SIR/BV-01-I – -03-I, BIP/RCR/SIR/BV-01-I – -03-I, BIP/RDI/SIR/BV-01-I – -03-I, and BIP/RDR/SIR/BV-01-I – -03-I. Updated the TCMT and TCRL accordingly. TSE 23938 (rating 2): Updated TCMT entries to support resolution of GAP ILDs in the BIP ICS. TSE 24521 (rating 4): Added new GSIT section with new TCs BIP/AAI/SGSIT/ATTR/BV-13-C – -17-C, BIP/AAI/SGSIT/OFFS/BV-03-C, BIP/AAI/SGSIT/SERR/BV-03-C, BIP/AIPI/SGSIT/ATTR/BV-08-C – -12-C, BIP/AIPI/SGSIT/OFFS/BV-02-C, BIP/AIPI/SGSIT/SERR/BV-02-C, BIP/SR/SGSIT/ATTR/BV-01-C – -07-C, BIP/SR/SGSIT/OFFS/BV-01-C, BIP/SR/SGSIT/SERR/BV-01-C, and BIP/IPSI/CGSIT/SFC/BV-01-C. Removed BIP/SR/GOEP/CON/BV-02-C from the TCMT and TCRL. Updated the TCMT accordingly. Added SDP TS and GOEP TS to the references list, updated the TCID Conventions, and revised the TSS overview. Updated document to align with the latest TS template, including setting up TCIDs in TC Config tables.
17	p17	2024-07-01	Approved by BTI on 2024-05-22. Prepared for TCRL 2024-1 publication.

Acknowledgments

Name	**Company**
Alicia Courtney	Broadcom
Tatsuo Arai	Casio Computer
Ryohei Yamamoto	Casio Computer
Erwin Weinans	Ericsson
Hiroshi Tanaka	Fujifilm
Mikio Watanabe	Fujifilm
Paolo Fontani	HP
Holt Mebane	HP
Franc Camara	Microsoft
Stephane Bouet	Nokia Mobile Phones
Martin Roter	Nokia Mobile Phones
Takayasu Fujiya	Toshiba
Kiyofumi Sanda	Toshiba

Name	**Company**
Yosuke Tajika	Toshiba