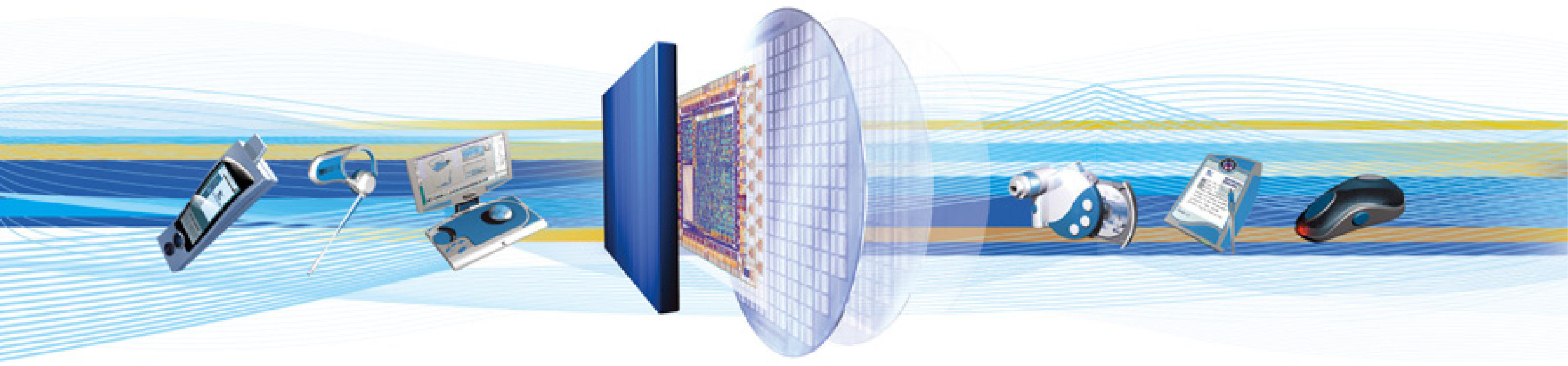
CSR Synergy®

CSR Synergy Bluetooth 18.2.2

OBEX Imaging Push Client

API Description

January 2012



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the OBEX Basic Imaging Profile (BIPC). BIPC conforms to the client side of the Image Push and Remote Camera feature ref. [BIP].

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- There is a secure and reliable transport between the profile part, i.e. BIPC and the application
- The BIPC shall only handle one request at a time
- The client can only authenticate the Imaging Responder doing a connect session

2 Description

2.1 Introduction

The scenario covered by this profile is the following:

- Usage of a Bluetooth® device e.g. a camera to send one or more images to another Bluetooth® device e.g. a mobile phone

The BIPC provides the following services to the application:

- Connection handling
- OBEX protocol handling

The application is responsible for handling the request from the BIPC and sending the response with correct data (object) as described in the IrOBEX specification. The BIPC is not checking if the data is packed correctly with white spaces in the right places, for details see ref. [BIP] and [OBEX].

2.2 Reference Model

The BIPC interfaces to the Connection Manager (CM).

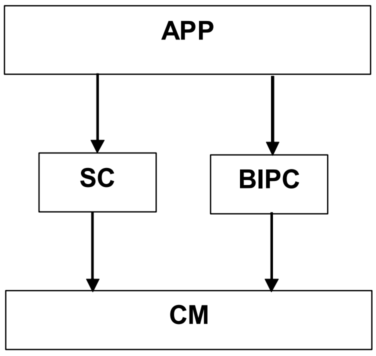


Figure 1: Reference model

2.3 Sequence Overview

When the BIPC starts up it is in IDLE state. If, in IDLE state, a connect request is received from the application, the BIPC starts to connect to the specified device, with the specified feature and the CONNECT state is entered. When the application receives a confirmation on the connect request, the application can request the relevant imaging functions. When the application disconnects the service, the BIPC re-enters IDLE state.

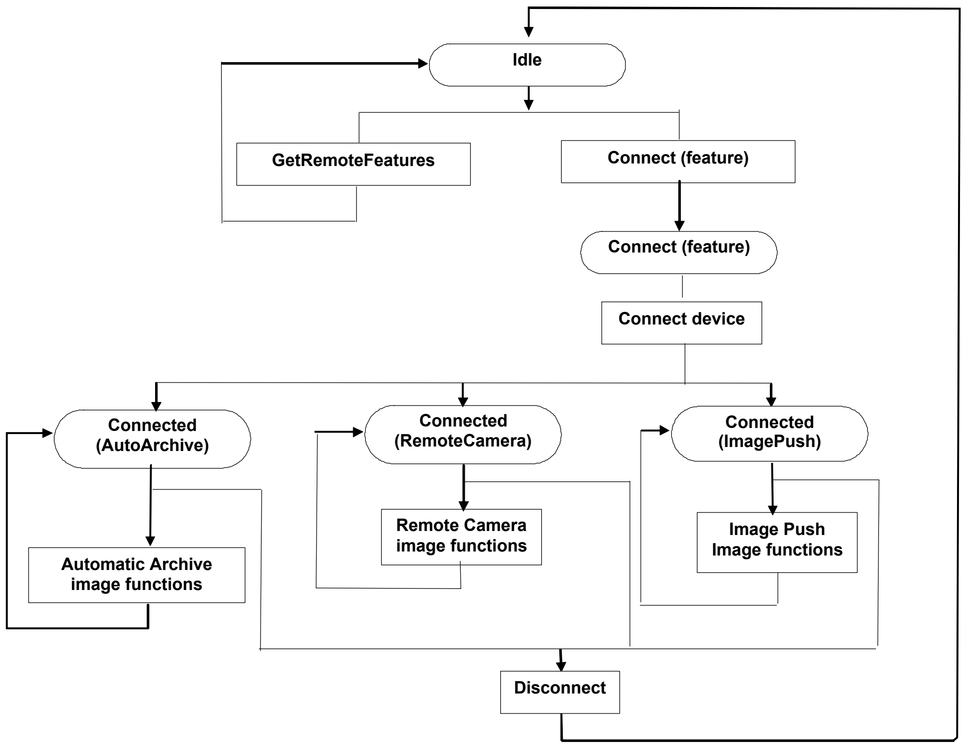


Figure 2: BIPC state diagram

The image functions availability is restricted depending on the connect state. See the list below

- Image Push available image functions
  - Get Capabilities
  - Put Image
  - Put Linked Thumbnail
  - Put Linked Attachments
- Remote Camera available image functions
  - Get Monitoring Image
  - Get Image Properties
  - Get Image
  - Get Linked Thumbnail
- Automatic Archive available image functions
  - Get Capabilities
  - Get Image List
  - Get Image Properties
  - Get Image
  - Get Linked Thumbnail
  - Get Linked Attachment
  - Delete Image

3 Interface Description

The BIPC can perform three different operations depending on which type of connection is made. Therefore, this section is divided into four parts: a common part that describes the interfaces that are the same for all three session types; Image Push specific interfaces; Remote Camera specific interfaces; and finally Automatic Archive specific interfaces.

3.1 Common Interfaces

The sequences in this section are common to the profile and not dependant on which type of connection is being made.

3.1.1 Retrieving Remote Features

Before the application connects to a BIP server it can request BIPC to query the BIP server as to which features it supports. The possible features are "Image Push", "Remote Camera" and "Automatic Archive". The signal used for requesting knowledge of the server available features is CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ. The application must supply a device address. BIPC will confirm the request with a CSR_BT_BIPC_GET_REMOTE_FEATURES_CFM containing the available features supported by both the server and client.

The application is not obliged to use this signal, and can choose to perform a 'blind' connect to the device – at the risk of the connect procedure failing due to lack of support of the feature in server.

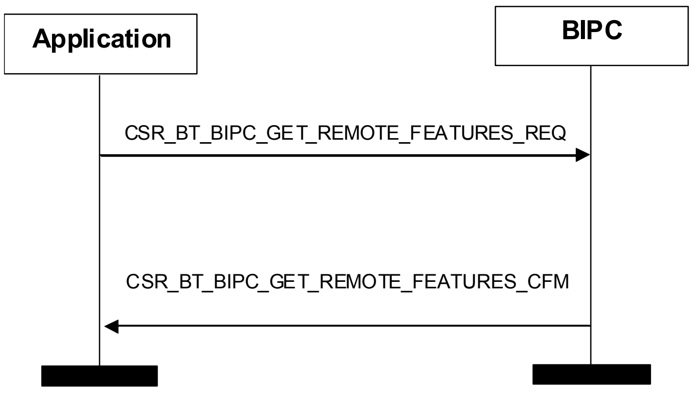

Application
BIPC
CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ
CSR_BT_BIPC_GET_REMOTE_FEATURES_CFM

Figure 3: Remote features

3.1.2 Connect

When the application wants to connect to a BIP Server it has to send a CSR_BT_BIPC_CONNECT_REQ. In this message the application has to specify which device to connect to. The parameter 'feature' must be used for specifying if the service requested if 'Image Push', 'Remote Camera' or 'Automatic Archive'. The parameter 'authorize' is controlling if the BIPC client wants to OBEX-authenticate the Imaging Responder. If the parameter is TRUE the application has to specify the password parameter. This message also has a parameter called maxPacketSize, which indicates the maximum OBEX packet size, which the application wants to receive from the server side. The value can be between 255 bytes to 64Kbytes – 1, for more information please refer to [OBEX]. If the packet size is large, it is optimising for quick file transfer, but the disadvantage will be use of big memory blocks.

The BIPC sends a CSR_BT_BIPC_CONNECT_CFM message to the application, which has the status of the connection establishment - this is the parameter result code. For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure in the connection attempt.

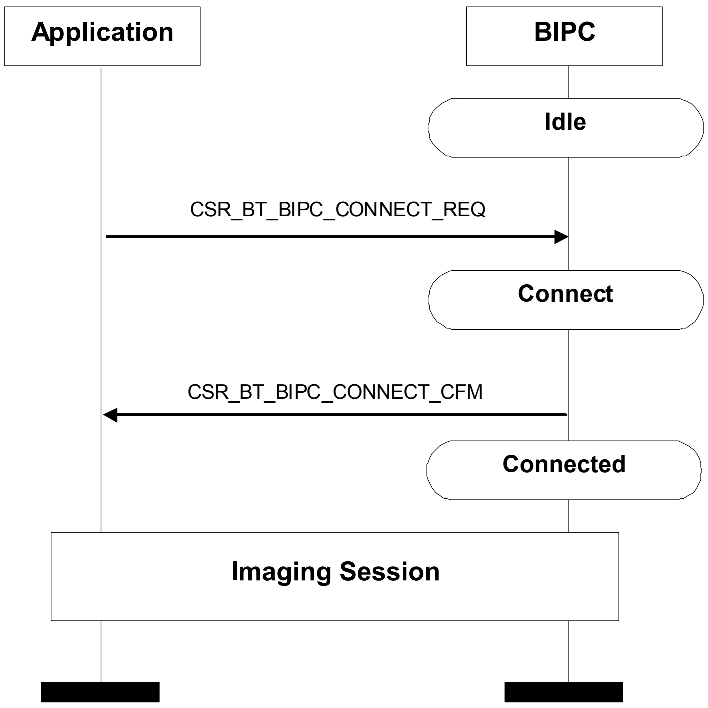
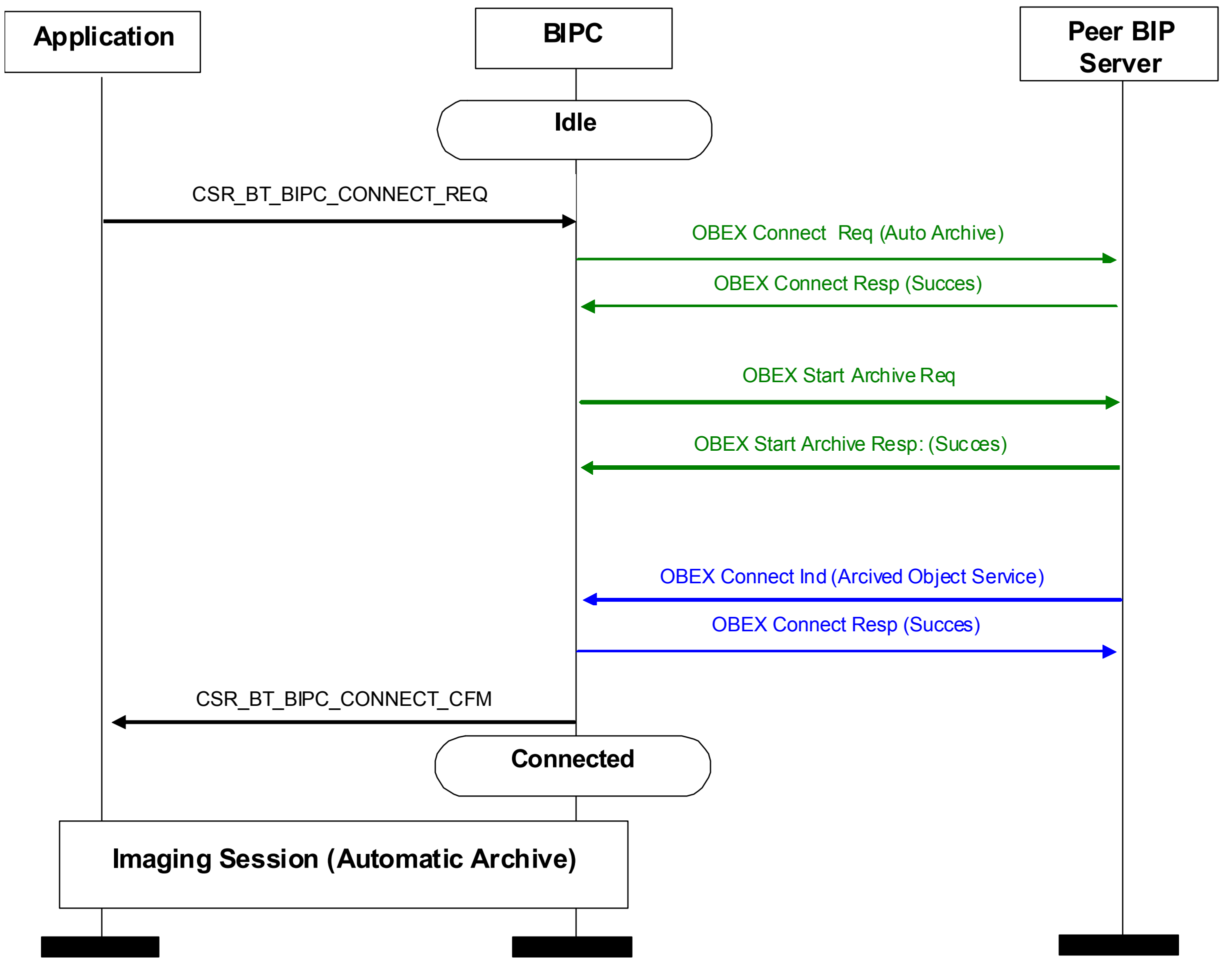


Figure 4: Connect for Image Push, Remote Camera and Automatic Archive

3.1.3 Connect with Automatic Archive

The following figure shows a system message sequence chart of the OBEX connections being established during the process of creating the Automatic Archive session. First a primary OBEX connection (shown in green) is created, and BIPC will then request the BIP peer to start archiving. This is done over the primary (green) OBEX connection. The BIP server peer should then take initiative to creating a secondary OBEX connection (shown in blue). When the secondary connection is established BIPC signals the application that the complete connection is up and the application must then act the server role, waiting for the BIP peer to take initiative - since the peer is now acting as a client.



Application
BIPC
Peer BIP Server
Idle
CSR_BT_BIPC_CONNECT_REQ
OBEX Connect Req (Auto Archive)
OBEX Connect Resp (Succes)
OBEX Start Archive Req
OBEX Start Archive Resp: (Succes)
OBEX Connect Ind (Arcived Object Service)
OBEX Connect Resp (Succes)
CSR_BT_BIPC_CONNECT_CFM
Connected
Imaging Session (Automatic Archive)

Figure 5: System MSC of the OBEX connections setup needed for Automatic Archive

The BIPC profile has the full responsibility of the primary and secondary connections. The application has no need to concern it self with the OBEX connections.

3.1.4 Authentication

An Imaging Responder can authenticate BIPC on every operation individually. If the application receives a CSR_BT_BIPC_AUTHENTICATE_IND it must response with a CSR_BT_BIPC_AUTHENTICATE_RES signal using the password or pin number that the Imaging Responder requires. An example of the authenticate sequence is illustrated below.

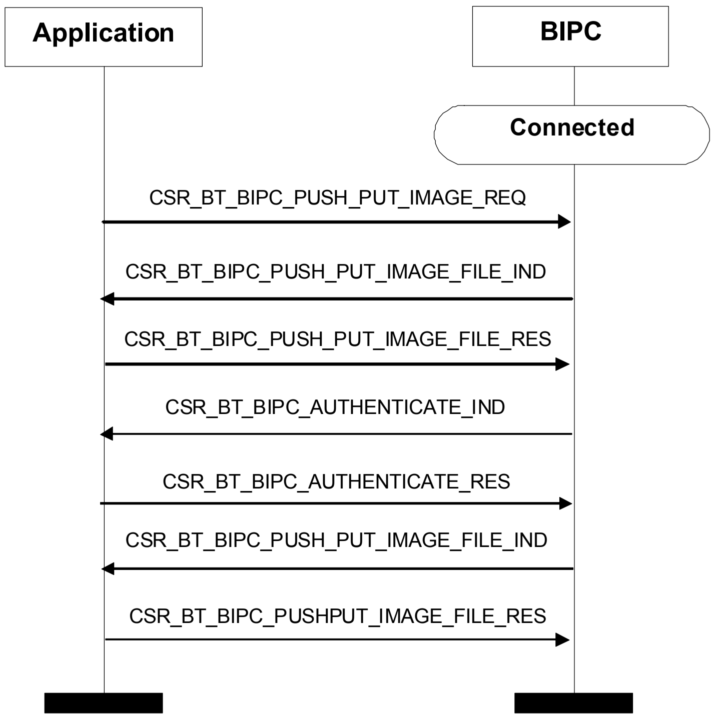

Application
BIPC
Connected
CSR_BT_BIPC_PUSH_PUT_IMAGE_REQ
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES
CSR_BT_BIPC_AUTHENTICATE_IND
CSR_BT_BIPC_AUTHENTICATE_RES
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND
CSR_BT_BIPC_PUSHPUT_IMAGE_FILE_RES

Figure 6: OBEX authentication

3.1.5 Abort Operation

The orderly sequence of request (from an OBEX client) followed by response (from an OBEX server) has one exception. An abort operation may be requested in the middle of a request/response sequence. It cancels the current operation. The application can terminate a multi-packet operation by sending an abort request (CSR_BT_BIPC_ABORT_REQ). The response (CSR_BT_BIPC_ABORT_CFM) is received indicating that the abort request is a success. It is also indicating that the abort request is received and the Imaging Responder is now resynchronized with the client. If anything else is returned the BIPC will disconnect the link and send CSR_BT_BIPC_DISCONNECT_IND to the application. An example of the abort sequence is illustrated below.

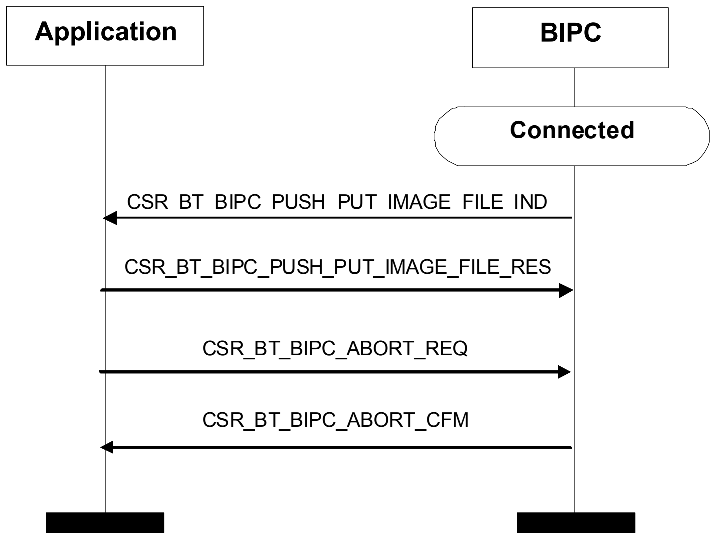

Application
BIPC
Connected
CSR BT BIPC PUSH PUT IMAGE FILE IND
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES
CSR_BT_BIPC_ABORT_REQ
CSR_BT_BIPC_ABORT_CFM

Figure 7: Abort operation

3.1.6 Disconnect

Sending a CSR_BT_BIPC_DISCONNECT_REQ disconnects the current connection (if any). The disconnect procedure may take some time and is confirmed with a CSR_BT_BIPC_DISCONNECT_IND signal. This is the normal disconnect scenario for Image Push and Remote Camera. Due to the role reversal during Automatic archive this disconnect procedure would be abnormal during an automatic archive session.

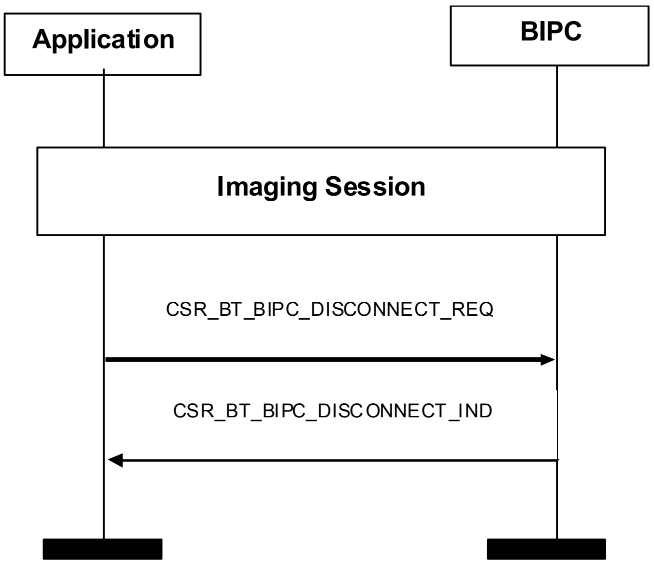


Figure 8: Normal disconnect for Image Push and Remote Camera, abnormal for Automatic Archive

For Image Push and Remote Camera sessions: In case the peer side prematurely disconnects, the BIPC sends a CSR_BT_BIPC_DISCONNECT_IND to the application and enters IDLE state. However, in the case of an automatic archive session it is expected due to the role reversal that the peer takes initiative to disconnect the session.

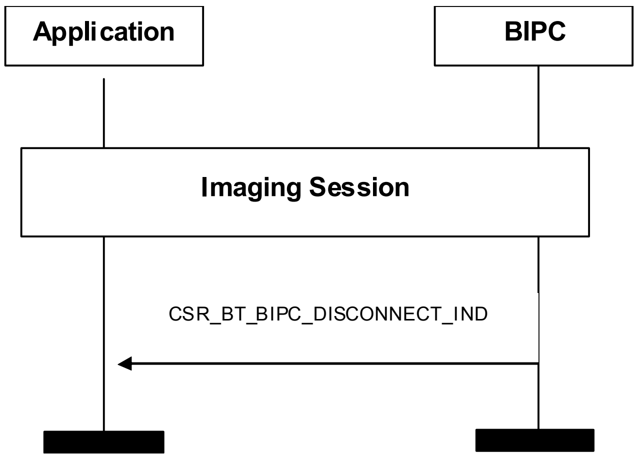


Figure 9: Abnormal disconnect for Image Push and Remote Camera, normal for Automatic Archive

3.1.7 Disconnect with Automatic Archive

The normal disconnect scenario for a server is to wait for the client to initiate disconnection. During automatic archive BIPC and the application has the role of server, and is therefore waiting for the peer BIP entity to finish the imaging session by disconnecting the Archived Objects OBEX connection (shown in blue). When BIPC receives this OBEX disconnect it reverts to its normal role as client, informs the application that the automatic archive connection is lost, and then disconnects the primary OBEX connection (shown in green).

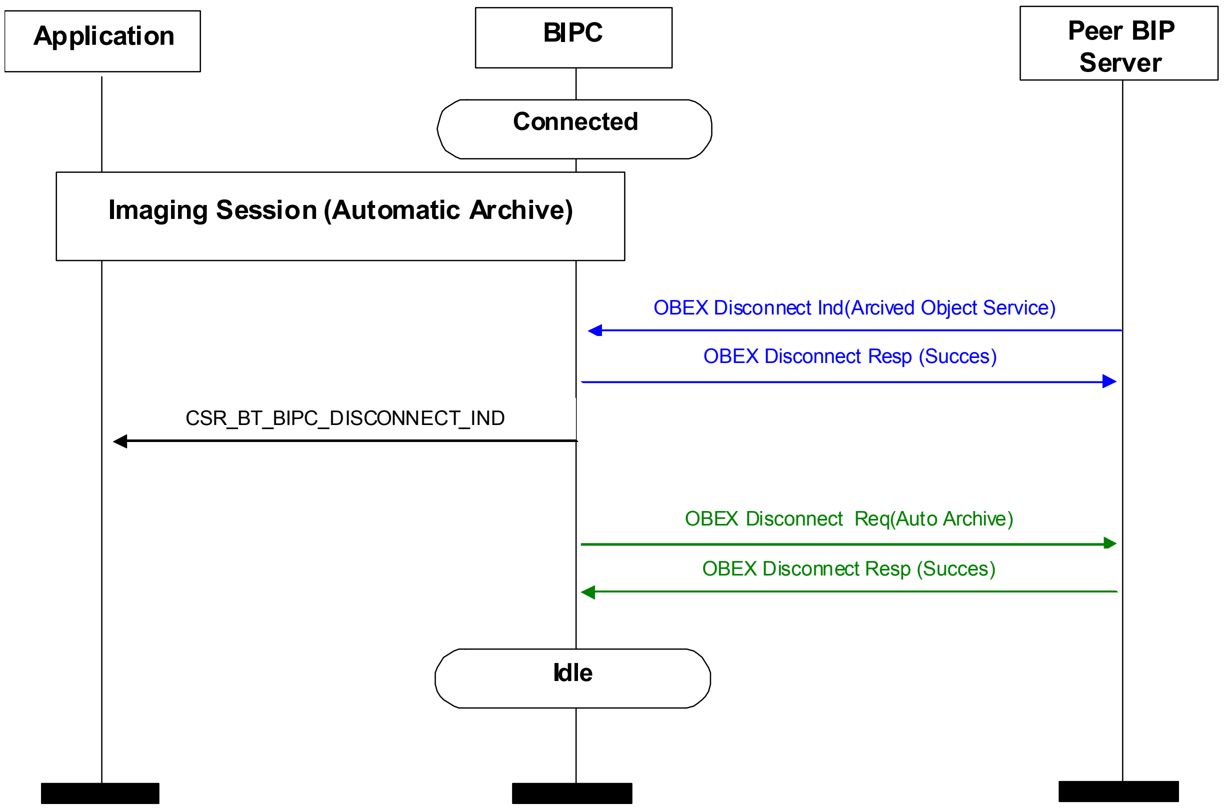

Application
BIPC
Peer BIP Server
Connected
Imaging Session (Automatic Archive)
OBEX Disconnect Ind(Arcived Object Service)
OBEX Disconnect Resp (Succes)
CSR_BT_BIPC_DISCONNECT_IND
OBEX Disconnect Req(Auto Archive)
OBEX Disconnect Resp (Succes)
Idle

Figure 10: System MSC of the normal disconnect scenario showing OBEX disconnections

In the abnormal case that the application wishes to terminate the automatic archive connection the application can request a disconnection of the image session. BIPC will upon such an request close the transport layer (shown in blue) of the secondary OBEX connection and make a OBEX disconnect of the primary connection (shown in green).

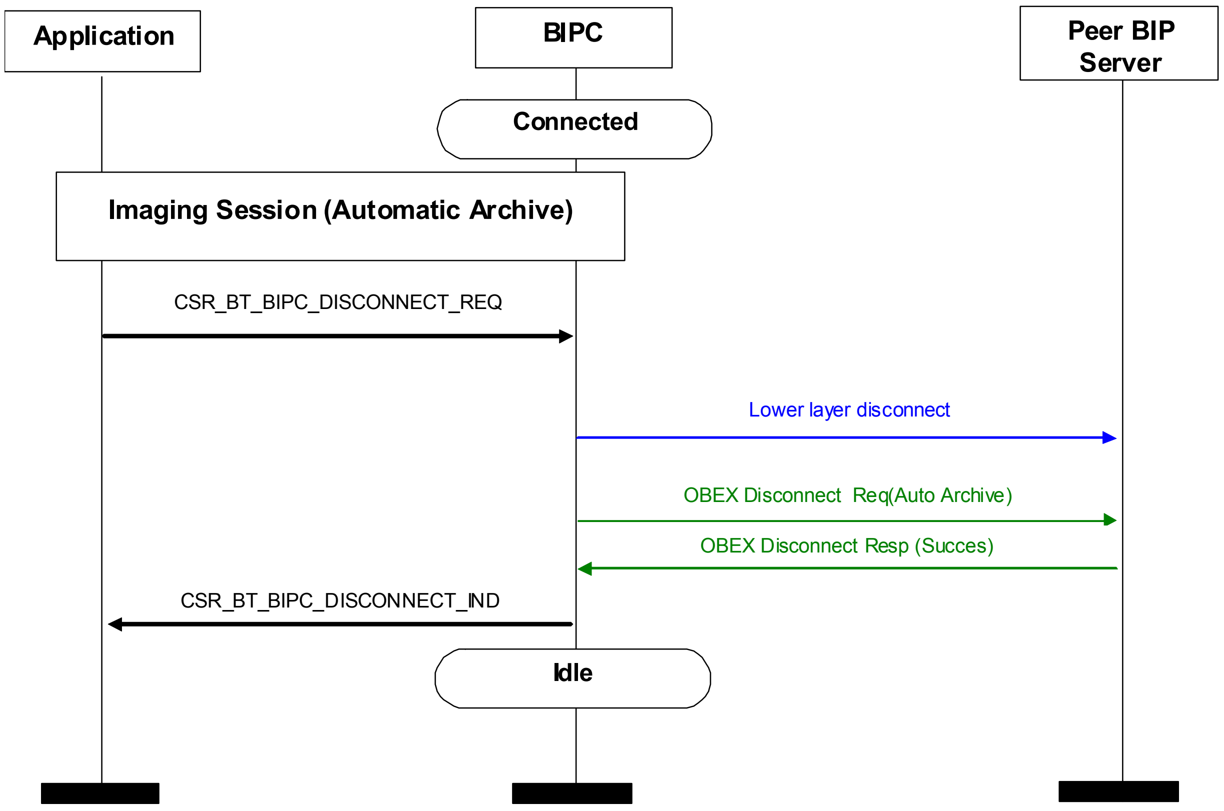

Application
BIPC
Peer BIP Server
Connected
Imaging Session (Automatic Archive)
CSR_BT_BIPC_DISCONNECT_REQ
Lower layer disconnect
OBEX Disconnect Req(Auto Archive)
OBEX Disconnect Resp (Succes)
CSR_BT_BIPC_DISCONNECT_IND
Idle

Figure 11: System MSC of the abnormal disconnect scenario showing OBEX disconnections

The BIPC profile has the full responsibility of the primary and secondary connections. When the application keeps the role reversal during an automatic archive imaging session it has no need to concern it self with the OBEX connections.

3.1.8 Get Instance Queue Identifier

As mentioned earlier most signals require a queue handle, as BIPC is capable of running multiple instances. The application gets this queue handle from the profile by issuing a CSR_BT_BIPC_GET_INSTANCES_QID_REQ. This is responded with a CSR_BT_BIPC_GET_INSTANCES_QID_CFM containing the queue handles. The application is responsible for remembering this handles and providing it in all signals.

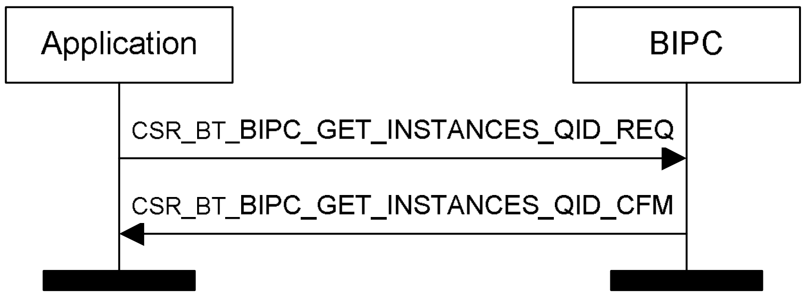

Application
BIPC
CSR_BT_BIPC_GET_INSTANCES_QID_REQ
CSR_BT_BIPC_GET_INSTANCES_QID_CFM

Figure 12: Get Instances QID handling

3.2 Image Push Interfaces

The sequences in this section can only occur when the BIPC is connected in an Image Push session. In the Image Push session images are pushed from the client to the server.

3.2.1 Get Capabilities

The application can retrieve the imaging-capabilities object from the Imaging Push Server by sending a CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ.

In case the image being pushed is large enough to require several OBEX packets, BIPC sends a CSR_BT_BIPC_PUSH_GET_CAPABILITIES_IND to the application, which the application must respond with a CSR_BT_BIPC_PUSH_GET_CAPABILITIES_RES message. Please notice that the CSR_BT_BIPC_PUSH_GET_CAPABILITIES_IND/ CSR_BT_BIPC_PUSH_GET_CAPABILITIES_RES sequence may be repeated.

When the Get Capabilities procedure is finished, BIPC sends a CSR_BT_BIPC_PUSH_GET_CAPABILITIES_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the imaging-capabilities object has been retrieved with success. Any other response code indicates that the imaging-capabilities object could not be retrieved from the server.

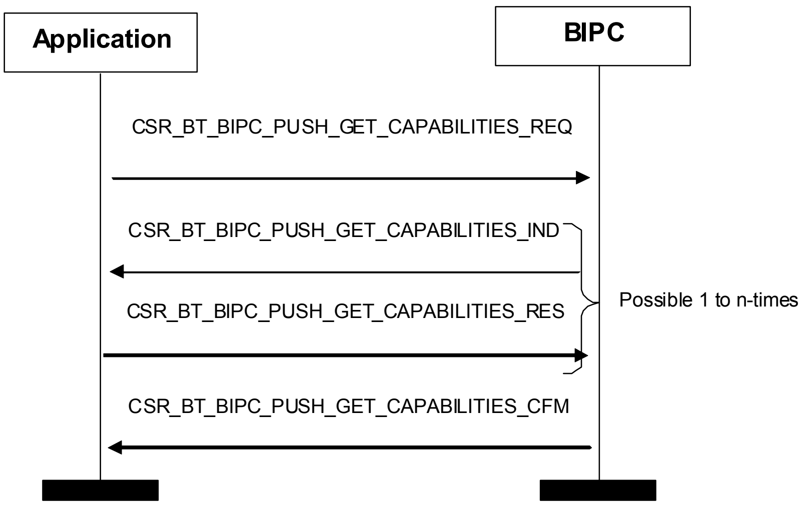

Application
BIPC
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_IND
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_RES
Possible 1 to n-times
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_CFM

Figure 13: Get capabilities

3.2.2 Put Image

The application can push an image to the Imaging Push Server by sending a CSR_BT_BIPC_PUSH_PUT_IMAGE_REQ. BIPC then sends a CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND to the application, which the application must respond with a CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES message. In case the image being pushed is large enough to require several OBEX packets the CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND/ CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES message sequence is repeated.

The Imaging Push Server may use the response code to request that BIPC sends the thumbnail version of the image it just received. In this case BIPC sends a CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_IND message to the application. When receiving this message the application must respond with a CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_RES message. In case the thumbnail version of the images is large enough to require several OBEX packets the CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_IND/ CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_RES message sequence is repeated.

When the Put Image procedure is finished, BIPC sends a CSR_BT_BIPC_PUSH_PUT_IMAGE_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the image, and if requested the thumbnail version of it, is pushed to the server with success. Any other response code indicates a failure in the Put Image procedure.

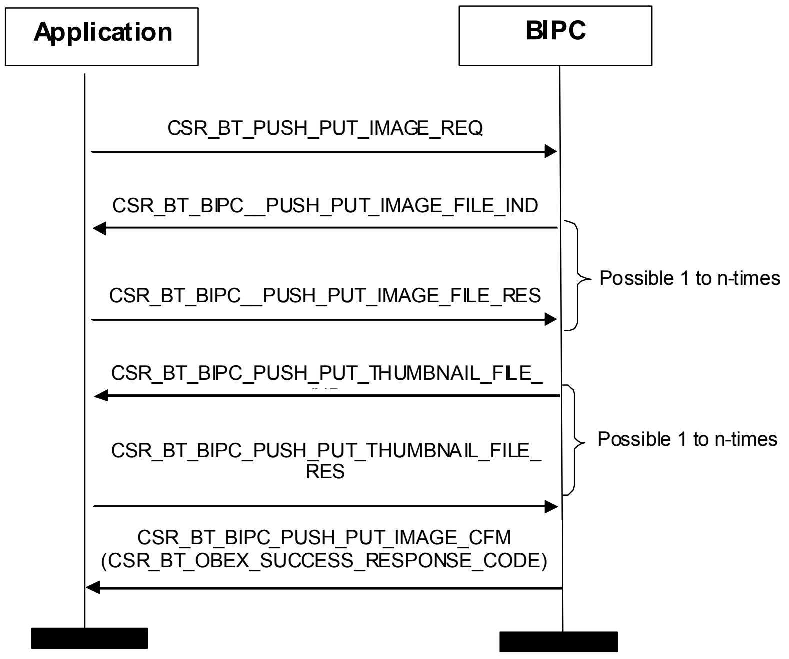

Application
BIPC
CSR_BT_PUSH_PUT_IMAGE_REQ
CSR_BT_BIPC__PUSH_PUT_IMAGE_FILE_IND
CSR_BT_BIPC__PUSH_PUT_IMAGE_FILE_RES
Possible 1 to n-times
CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_IND
CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE_RES
Possible 1 to n-times
CSR_BT_BIPC_PUSH_PUT_IMAGE_CFM (CSR_BT_OBEX_SUCCESS_RESPONSE_CODE)

Figure 14: Put image

3.2.3 Put Attachment

Please notice, that the application must only request the Put Attachment procedure if this feature is supported by the Imaging Push Server. (The parameter “supportedFunctions” in the CSR_BT_BIPC_CONNECT_CFM message is describing which features the Imaging Push Server supports).

The application can send attachments associated with an image to the Imaging Push Server after the image has been sent to the server within the OBEX context of an OBEX session. This means that the Put Attachment procedure must be used within the OBEX session during which the image was sent to the Imaging Push Server.

To send an attachment associated with an image the application must send a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_REQ message. BIPC then sends a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND to the application, which the application must respond with a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES message. In case the image being pushed is large enough to require several OBEX packets the CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND/ CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES message sequence is repeated.

When the Put Attachment procedure is finished, BIPC sends a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure.

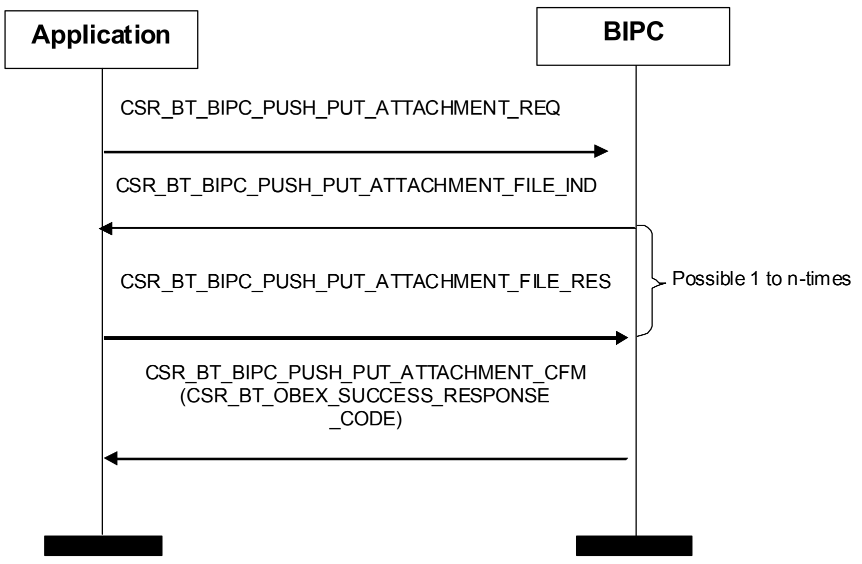

Application
BIPC
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_REQ
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES
Possible 1 to n-times
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_CFM (CSR_BT_OBEX_SUCCESS_RESPONSE _CODE)

Figure 15: Put attachment

3.3 Remote Camera Interfaces

The sequences in this section can only occur when the BIPC is connected in a Remote Camera session. In the Remote Camera session the client is most often connected to a server which can record images. The client can pull monitoring images and recorded images from the server.

3.3.1 Get Monitoring Image

When an OBEX connection for Remote Camera has been made the CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ is used for retrieving a monitoring image from the Image Responder. The application chooses whether the Imaging Responder must release its shutter and store the full image.

When the image handle has been retrieved from the server it is send to the application in the signal CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_IND. The application must respond with CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_RES when it is ready to receive the monitoring image. Following this, the image, or part of it is send to the application using CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_IND, the application is required to respond with CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_RES when it is ready to receive more data. The procedure is finished upon reception by the application of the signal CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_CFM.

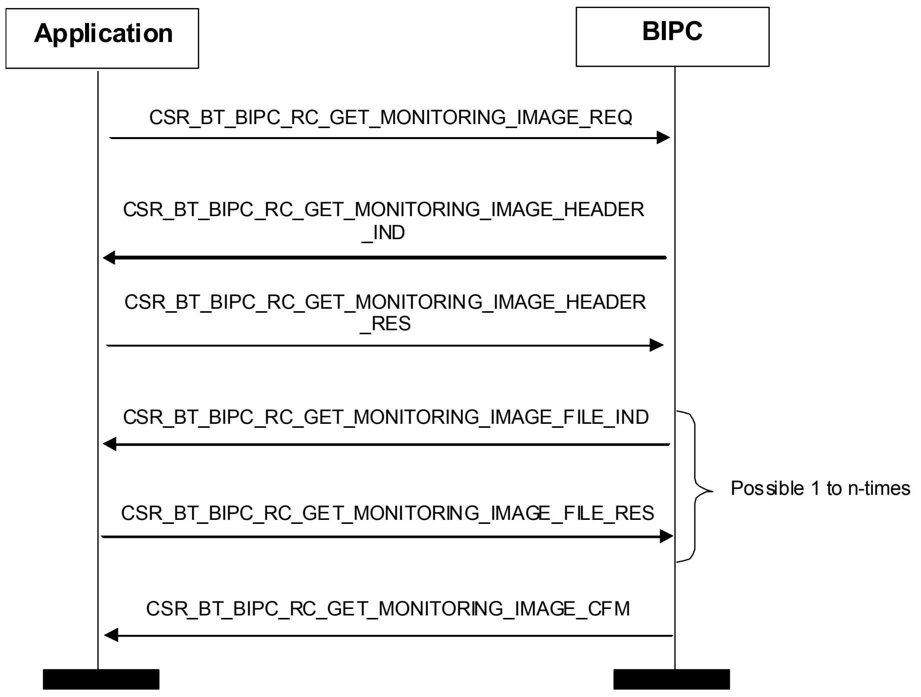

Application
BIPC
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_IND
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_RES
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_IND
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_RES
Possible 1 to n-times
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_CFM

Figure 16: Get monitoring image

3.3.2 Get Image Properties

When an OBEX connection for Remote Camera has been made signal CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ can be used for retrieving information about a specific image from the Imaging Responder. The image properties are a detailed description of the image's characteristics.

The image properties or parts there of, are presented to the application in the signal CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_IND and CFM. The indications must be responded with a CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_RES

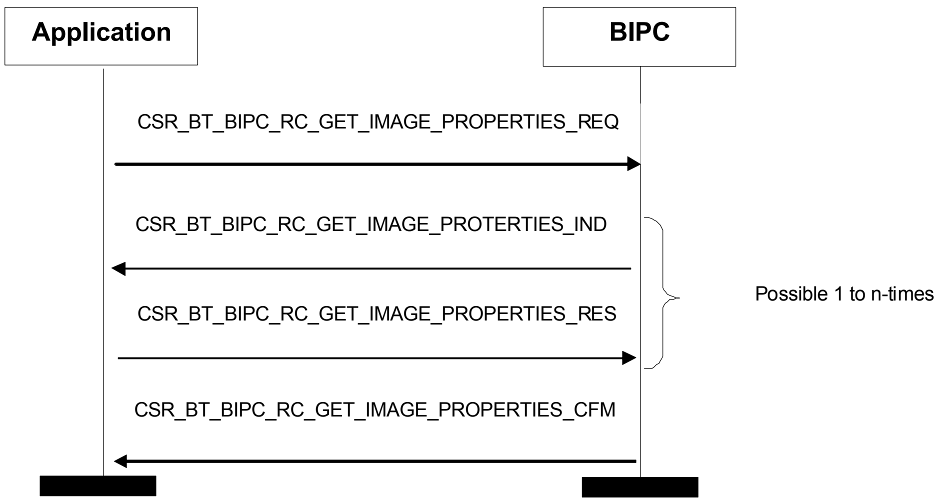

Application
BIPC
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ
CSR_BT_BIPC_RC_GET_IMAGE_PROTERTIES_IND
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_RES
Possible 1 to n-times
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_CFM

Figure 17: Get image properties

3.3.3 Get Image

When an OBEX connection for Remote Camera has been made the signal CSR_BT_BIPC_RC_GET_IMAGE_REQ can be used for retrieving a specific image from the Imaging Responder. An image handle is supplied to identify the image, and an image description is supplied to inform the Imaging Responder of which format to use.

The image or parts there of, are presented to the application in the signal CSR_BT_BIPC_RC_GET_IMAGE_IND and CFM. The indications must be responded with a CSR_BT_BIPC_RC_GET_IMAGE_RES

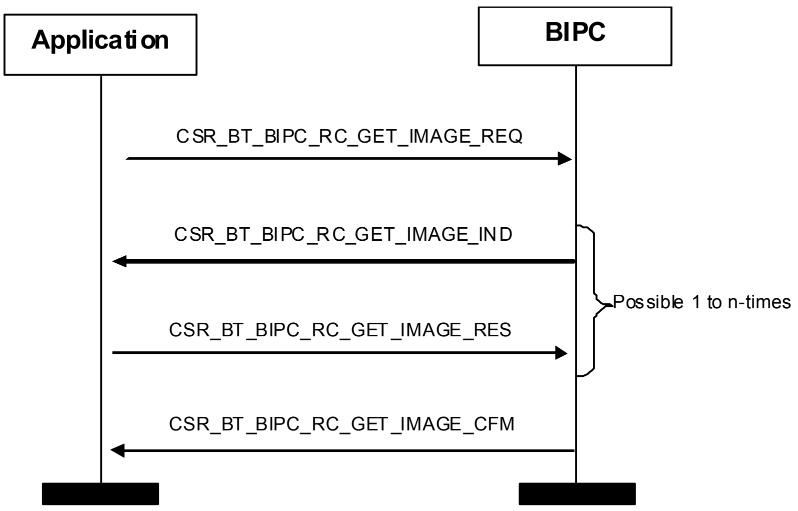

Application
BIPC
CSR_BT_BIPC_RC_GET_IMAGE_REQ
CSR_BT_BIPC_RC_GET_IMAGE_IND
Possible 1 to n-times
CSR_BT_BIPC_RC_GET_IMAGE_RES
CSR_BT_BIPC_RC_GET_IMAGE_CFM

Figure 18: Get image

3.3.4 Get Linked Thumbnail

When an OBEX connection for Remote Camera has been made signal CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ can be used for retrieving the thumbnail of a specific image from the Imaging Responder.

The thumbnail or parts there of, are presented to the application in the signal CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_IND and CFM. The indications must be responded with a CSR_BT_BIPC_RC_GET_LINKED_THUMBNIAL_RES.

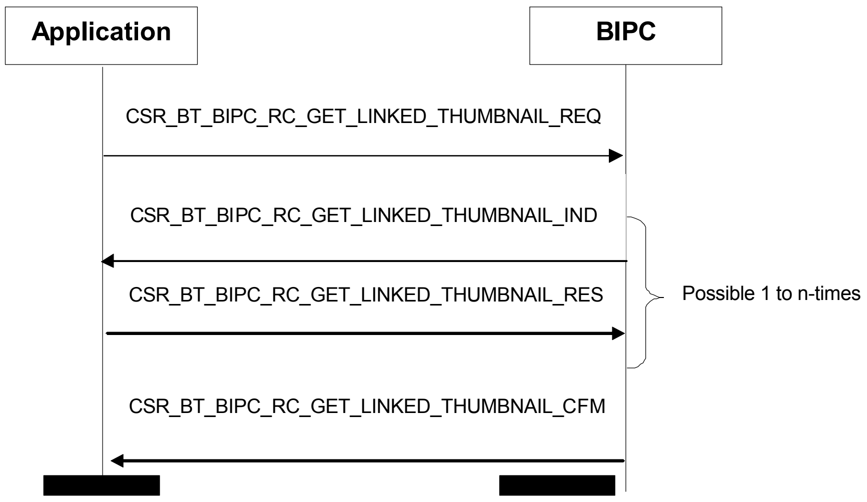

Application
BIPC
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_IND
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_RES
Possible 1 to n-times
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_CFM

Figure 19: Get linked thumbnail

3.4 Automatic Archive Interfaces

The sequences in this section can only occur when the BIPC is connected in an Automatic Archive session. In the Automatic Archive session the client asks the server to perform Automatic Archive and the server then retrieves and stores images from the client. In fact the server will start acting as a client and the client will start acting as a server, until the Automatic Archive session is completed.

3.4.1 Get Image List

When an OBEX connection for Automatic Archive has been made the get image list function can be used by the peer for retrieving a list of images available from the client. The application must provide information to BIPC about the list of images according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get image list function is CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER _RES. The transfer of the image list object or part of it is requested by the signal CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND, the response CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_RES contains the image list object or a fragment of it. See also the figure below.

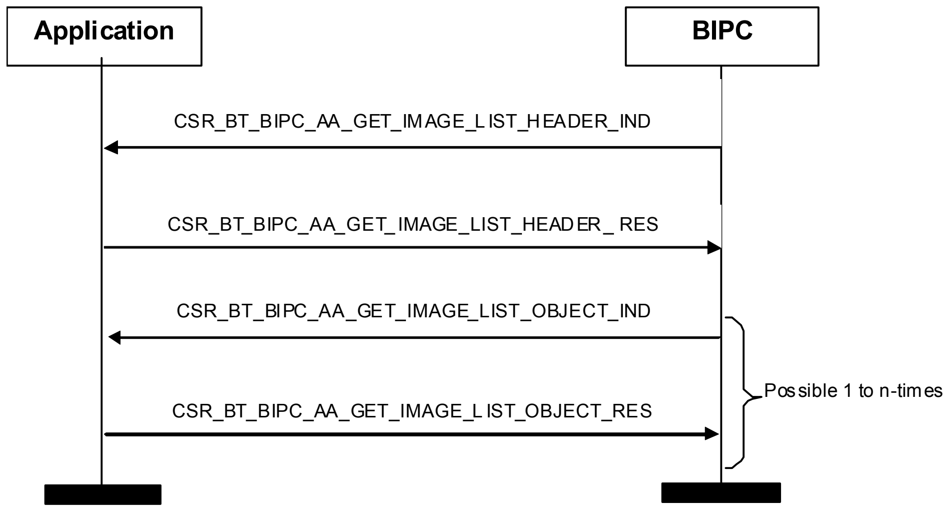

Application
BIPC
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_IND
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_ RES
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_RES
Possible 1 to n-times

Figure 20: Get image list

3.4.2 Get Capabilities

When an OBEX connection for Automatic Archive has been made the get capabilities function can be used by the peer for retrieving the imaging capabilities of the application. This information must be provided by the application to BIPC according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get capabilities function is CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER _RES. The transfer of the capabilities object or part of it is requested by the signal CSR_BT_BIPC_AA_GET_CAPABILITIES_OBJECT_IND, the response CSR_BT_BIPC_AA_GET_CAPABILITIES_OBJECT_RES contains the capabilities object or a fragment of it. See also the figure below.

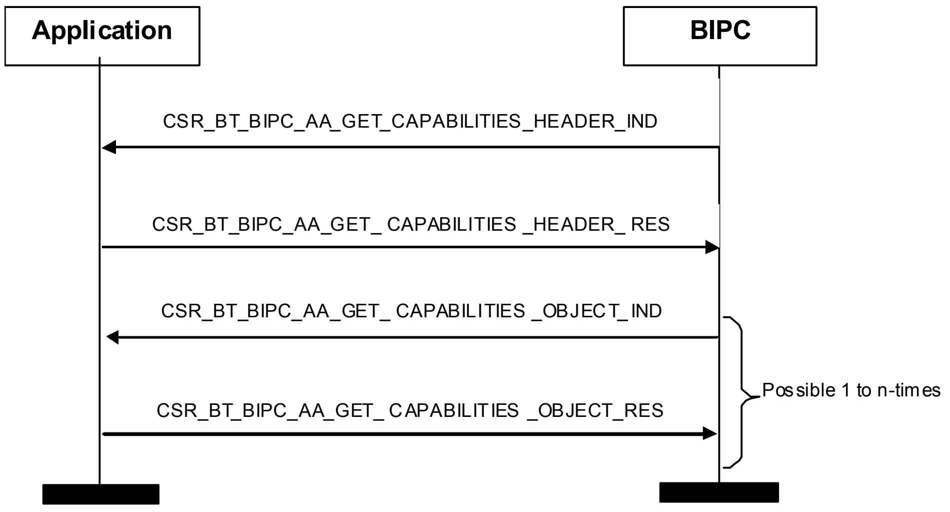

Application
BIPC
CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_IND
CSR_BT_BIPC_AA_GET_ CAPABILITIES _HEADER_ RES
CSR_BT_BIPC_AA_GET_ CAPABILITIES _OBJECT_IND
CSR_BT_BIPC_AA_GET_ CAPABILITIES _OBJECT_RES
Possible 1 to n-times

Figure 21: Get capabilities

3.4.3 Get Image Properties

When an OBEX connection for Automatic Archive has been made the get image properties function can be used by the peer for retrieving the image properties from the application of an image handle previously disclosed in an image list. The image properties of the referenced image must be provided by the application to BIPC according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get image properties function is CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER _RES. The transfer of the image properties object or part of it is requested by the signal CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES:OBJECT_IND, the response CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_OBJECT_RES contains the image properties object or a fragment of it. See also the figure below.

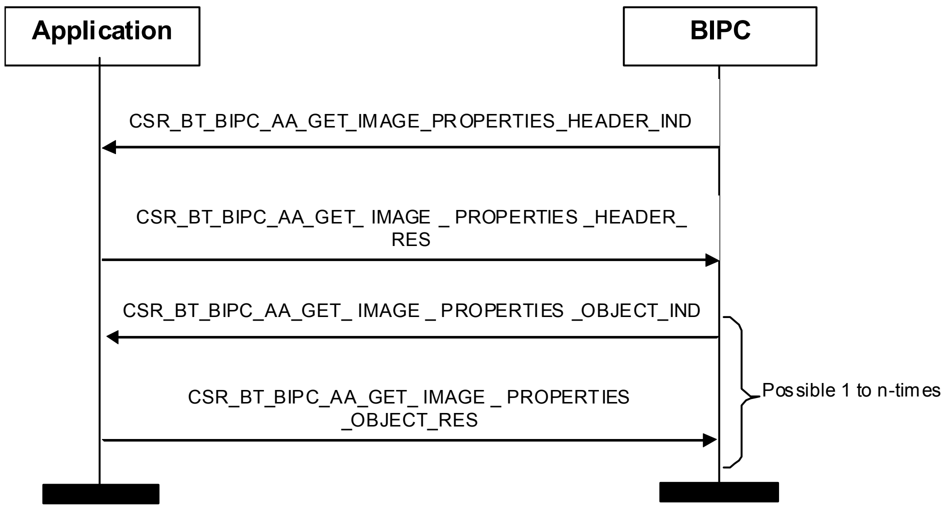

Application
BIPC
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER_IND
CSR_BT_BIPC_AA_GET_ IMAGE _ PROPERTIES _HEADER_ RES
CSR_BT_BIPC_AA_GET_ IMAGE _ PROPERTIES _OBJECT_IND
CSR_BT_BIPC_AA_GET_ IMAGE _ PROPERTIES _OBJECT_RES
Possible 1 to n-times

Figure 22: Get image properties

3.4.4 Get Image

When an OBEX connection for Automatic Archive has been made the get image function can be used by the peer for retrieving an image from the application referenced by an image handle previously disclosed in an image list. The image of the referenced image must be provided by the application to BIPC according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get image function is CSR_BT_BIPC_AA_GET_IMAGE_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_IMAGE_HEADER _RES. The transfer of the image or part of it is requested by the signal CSR_BT_BIPC_AA_GET_IMAGE_OBJECT_IND, the response CSR_BT_BIPC_AA_GET_IMAGE_OBJECT_RES contains the image or a fragment of it. See also the figure below.

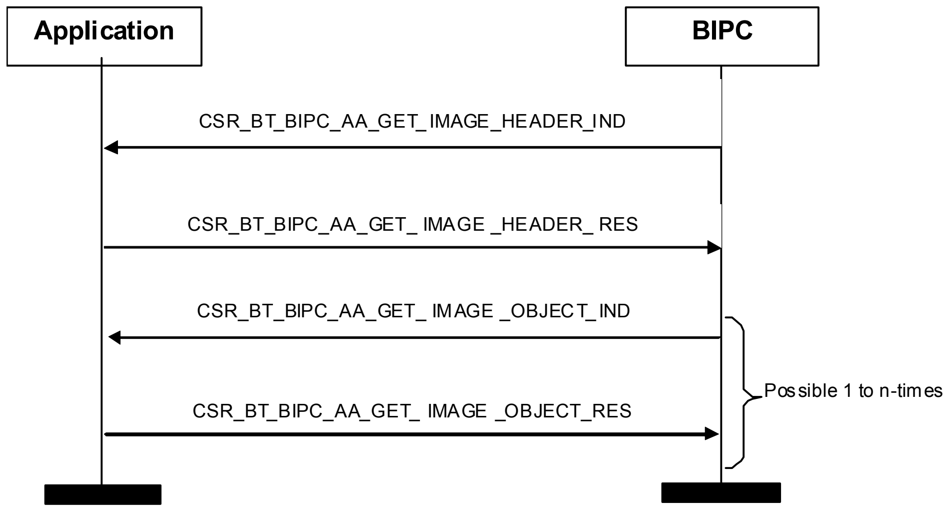

Application
BIPC
CSR_BT_BIPC_AA_GET_IMAGE_HEADER_IND
CSR_BT_BIPC_AA_GET_ IMAGE _HEADER_ RES
CSR_BT_BIPC_AA_GET_ IMAGE _OBJECT_IND
Possible 1 to n-times
CSR_BT_BIPC_AA_GET_ IMAGE _OBJECT_RES

Figure 23: Get image

3.4.5 Get Linked Attachment

When an OBEX connection for Automatic Archive has been made the get linked attachment function can be used by the peer for retrieving from the application an attachment associated to an image handle previously disclosed in an image list. The attachment of the referenced image must be provided by the application to BIPC according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get linked attachment function is CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_LINKED_ ATTACHMENT _HEADER _RES. The transfer of the attachment object or part of it is requested by the signal CSR_BT_BIPC_AA_GET_LINKED_ ATTACHMENT_OBJECT_IND, the response CSR_BT_BIPC_AA_GET_LINKED_ ATTACHMENT_OBJECT_RES contains the attachment or a fragment of it. See also the figure below.

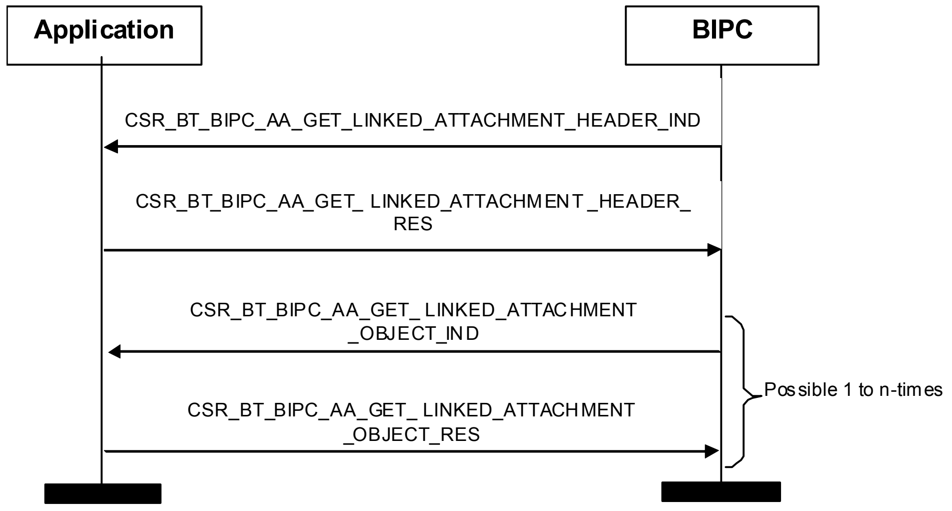

Application
BIPC
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_IND
CSR_BT_BIPC_AA_GET_ LINKED_ATTACHMENT _HEADER_ RES
CSR_BT_BIPC_AA_GET_ LINKED_ATTACHMENT _OBJECT_IND
Possible 1 to n-times
CSR_BT_BIPC_AA_GET_ LINKED_ATTACHMENT _OBJECT_RES

Figure 24: Get linked attachment

3.4.6 Get Linked Thumbnail

When an OBEX connection for Automatic Archive has been made the get linked thumbnail function can be used by the peer for retrieving from the application a thumbnail of the image associated to an image handle previously disclosed in an image list. The thumbnail of the referenced image must be provided by the application to BIPC according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the get linked thumbnail function is CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_IND. The header response is send using CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_RES. The transfer of the thumbnail object or part of it is requested by the signal CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT_IND, the response CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT_RES contains the thumbnail or a fragment of it. See also the figure below.

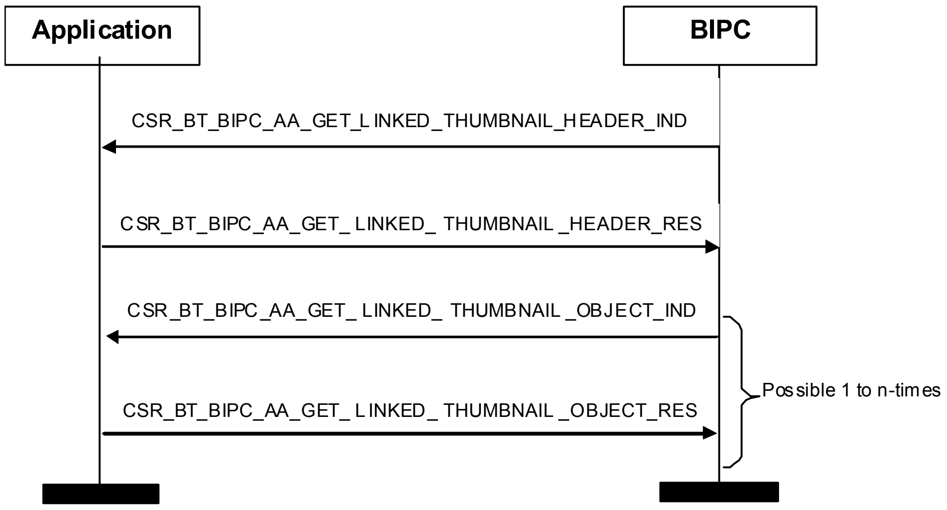

Application
BIPC
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_IND
CSR_BT_BIPC_AA_GET_ LINKED_ THUMBNAIL _HEADER_RES
CSR_BT_BIPC_AA_GET_ LINKED_ THUMBNAIL _OBJECT_IND
CSR_BT_BIPC_AA_GET_ LINKED_ THUMBNAIL _OBJECT_RES
Possible 1 to n-times

Figure 25: Get linked thumbnail

3.4.7 Delete Image

When an OBEX connection for Automatic Archive has been made the delete image function can be used by the peer for the application to delete the image associated to an image handle previously disclosed in an image list. The image in question must be deleted by the application according to the indications received. The application can choose to reject the request by transmitting an appropriate response code.

The signal used for initiating the delete image function is CSR_BT_BIPC_AA_DELETE_IMAGE_IND. The response is send using CSR_BT_BIPC_AA_DELETE_IMAGE_RES. See also the figure below.

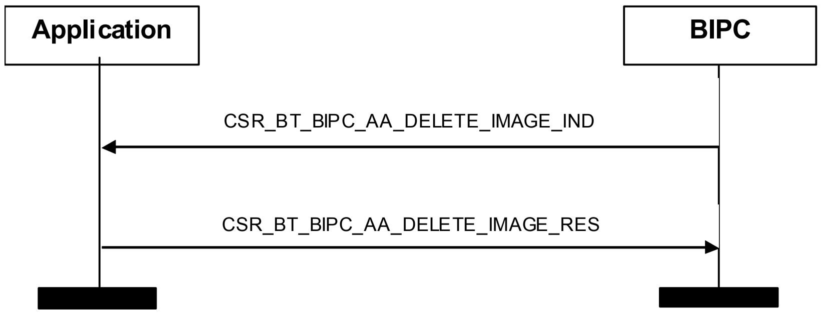

Application
BIPC
CSR_BT_BIPC_AA_DELETE_IMAGE_IND
CSR_BT_BIPC_AA_DELETE_IMAGE_RES

Figure 26: Delete image

3.4.8 Abort

When an OBEX connection for Automatic Archive has been made the abort function can be used by the peer for aborting the current OBEX procedure. The application must stop transmitting responses to previous indications, and be able to handle the discontinuation of the current procedure.

The signal used for indicating the abort function is CSR_BT_BIPC_AA_ABORT_IND. See also the figure below.

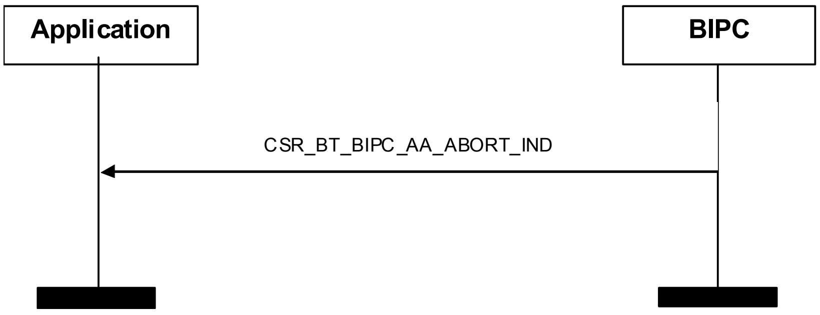


Figure 27: Abort indication

3.5 Payload Encapsulated Data

3.5.1 Using Offsets

As many OBEX messages contain multiple parameters with variable length, some of the parameters are based on *offsets* instead of standard pointers to the data. Signals with offset-based data can easily be recognized as they have both a *payload* and a *payloadLength* parameter. The *payload* contains the actual data, on which the offset is based. For example, a typical signal may contain the following:

```
    CsrBtBipPrim   type;
    CsrUint8           result;
    CsrUint16      ucs2nameOffset;
    CsrUint16      bodyOffset;
    CsrUint16      bodyLength;
    CsrUint16      payloadLength;
    CsrUint8           *payload;
```

In this example, two offset parameters can be found, namely *ucs2nameOffset* and *bodyOffset*. To obtain the actual data, the offset value is added to the *payload* pointer, which yields a pointer to the data, i.e.:

```
      CsrUint8  *ucs2name;
      ucs2name = (CsrUint8*)(primitive->payload + primitive->ucs2nameOffset);
```

As can be seen, the offset contains the number of bytes within the *payload* where the information begins. Similarly, the body data can be retrieved using the following:

```
    CsrUint8 *body;
    body = (CsrUint8*)(primitive->payload + primitive->bodyOffset);
```

And to illustrate the usage of the *length* parameter, which is also a common parameter, to copy the body one would typically use:

```
    CsrMemCpy( copyOfBody, body, primitive->bodyLength );
```

Offset parameters will always have an “Offset” suffix on the name, and offsets are *always* relative to the “payload” parameter.

If the `bodyOffset` or the `bodyLength` is 0 (zero) this means that the signal does not contain any body. The same holds when the `payloadLength` is 0 (zero), which means that there is not payload.

3.5.2 Payload Memory

When the application receives a signal which has a *payload* parameter, the application must always free the payload pointer to avoid memory leaks, for example

```
    CsrPfree(primitive->payload);
    CsrPfree(primitive);
```

will free both the payload data and the message itself. Note that when the payload has been freed, offsets can not be used anymore, as the actual data is contained within the payload.

Signals that do not use the *payload* parameter must still have each of their pointer-based parameters freed.

Likewise, the profile will free any pointers received as parameters in API signals or functions

4 OBEX Basic Imaging Profile Client Primitives

This section gives an overview of the primitives and parameters in the interface. Detailed information can be found in the corresponding csr_bt_bipc_prim.h file.

4.1 List of All Primitives

Primitives:	**Reference:**
CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ	See section 4.2.1
CSR_BT_BIPC_GET_REMOTE_FEATURES_CFM	See section 4.2.1
CSR_BT_BIPC_CONNECT_REQ	See section 4.2.2
CSR_BT_BIPC_CONNECT_CFM	See section 4.2.2
CSR_BT_BIPC_AUTHENTICATE_IND	See section 4.2.3
CSR_BT_BIPC_AUTHENTICATE_RES	See section 4.2.3
CSR_BT_BIPC_ABORT_REQ	See section 4.2.4
CSR_BT_BIPC_ABORT_CFM	See section 4.2.4
CSR_BT_BIPC_DISCONNECT_REQ	See section 4.2.5
CSR_BT_BIPC_DISCONNECT_IND	See section 4.2.5
CSR_BT_BIPC_SECURITY_OUT_REQ	See section 4.2.6
CSR_BT_BIPC_SECURITY_OUT_CFM	See section 4.2.6
CSR_BT_BIPC_REGISTER_QID_REQ	See section 4.2.7
CSR_BT_BIPC_GET_INSTANCES_QID_REQ	See section 4.2.8
CSR_BT_BIPC_GET_INSTANCES_QID_CFM	See section 4.2.8
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ	See section 4.3.1
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_RES	See section 4.3.1
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_IND	See section 4.3.1
CSR_BT_BIPC_ PUSH_GET_CAPABILITIES_CFM	See section 4.3.1
CSR_BT_BIPC_ PUSH_PUT_IMAGE_REQ	See section 4.3.2
CSR_BT_BIPC_ PUSH_PUT_IMAGE_CFM	See section 4.3.2
CSR_BT_BIPC_ PUSH_PUT_IMAGE_FILE_IND	See section 4.3.2
CSR_BT_BIPC_ PUSH_PUT_IMAGE_FILE_RES	See section 4.3.2
CSR_BT_BIPC_ PUSH_PUT_THUMBNAIL_FILE_IND	See section 4.3.3
CSR_BT_BIPC_ PUSH_PUT_THUMBNAIL_FILE_RES	See section 4.3.3
CSR_BT_BIPC_ PUSH_PUT_ATTACHMENT_REQ	See section 4.3.4
CSR_BT_BIPC_ PUSH_PUT_ATTACHMENT_CFM	See section 4.3.4
CSR_BT_BIPC_ PUSH_PUT_ATTACHMENT_FILE_IND	See section 4.3.4
CSR_BT_BIPC_ PUSH_PUT_ATTACHMENT_FILE_RES	See section 4.3.4
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ	See section 4.4.1
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_IND	See section 4.4.1
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_RES	See section 4.4.1
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_CFM	See section 4.4.1
CSR_BT_BIPC_RC_GET_IMAGE_REQ	See section 4.4.2
CSR_BT_BIPC_RC_GET_IMAGE_CFM	See section 4.4.2
CSR_BT_BIPC_RC_GET_IMAGE_RES	See section 4.4.2

Primitives:	Reference:
CSR_BT_BIPC_RC_GET_IMAGE_IND	See section 4.4.2
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ	See section 4.4.3
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_CFM	See section 4.4.3
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_RES	See section 4.4.3
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_IND	See section 4.4.3
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ	See section 4.4.4
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_CFM	See section 4.4.4
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_RES	See section 4.4.4
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_IND	See section 4.4.4
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_RES	See section 4.4.4
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_IND	See section 4.4.4
CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_IND	See section 4.5.1
CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_RES	See section 4.5.1
CSR_BT_BIPC_AA_GET_CAPABILITIES_OBJECT_IND	See section 4.5.1
CSR_BT_BIPC_AA_GET_CAPABILITIES_OBJECT_RES	See section 4.5.1
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER_IND	See section 4.5.2
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER_RES	See section 4.5.2
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_OBJECT_IND	See section 4.5.2
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_OBJECT_RES	See section 4.5.2
CSR_BT_BIPC_AA_GET_IMAGE_HEADER_IND	See section 4.5.3
CSR_BT_BIPC_AA_GET_IMAGE_HEADER_RES	See section 4.5.3
CSR_BT_BIPC_AA_GET_IMAGE_OBJECT_IND	See section 4.5.3
CSR_BT_BIPC_AA_GET_IMAGE_OBJECT_RES	See section 4.5.3
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_IND	See section 4.5.4
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_RES	See section 4.5.4
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT_IND	See section 4.5.4
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT_RES	See section 4.5.4
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_IND	See section 4.5.5
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_RES	See section 4.5.5
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND	See section 4.5.6
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_RES	See section 4.5.6
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_IND	See section 4.5.7
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_RES	See section 4.5.7
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_IND	See section 4.5.8
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_RES	See section 4.5.8
CSR_BT_BIPC_AA_DELETE_IMAGE_IND	See section 0
CSR_BT_BIPC_AA_DELETE_IMAGE_RES	See section 0
CSR_BT_BIPC_AA_ABORT_IND	See section 4.5.10

Table 1: List of all primitives

4.2 Common Primitives

4.2.1 CSR_BT_BIPC_GET_REMOTE_FEATURES

Parameters / Primitives	type	appHandle	deviceAddr	pHandleInst	supportedFeatures	srmpOn
CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ	✓	✓	✓			✓
CSR_BT_BIPC_GET_REMOTE_FEATURES_CFM	✓			✓	✓	

Table 2: CSR_BT_BIPC_GET_REMOTE_FEATURES Primitives

Description

Before the application connects to a BIP server it can request BIPC to query the BIP server as to which features it supports. The possible features are “Image Push”, “Remote Camera” and “Automatic Archive”. The signal used for requesting knowledge of the server available features is CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ. The application must supply a device address. BIPC will confirm the request with a CSR_BT_BIPC_GET_REMOTE_FEATURES_CFM containing the available features supported by both the server and client.

Parameters

Type	Signal identity, CSR_BT_BIPC_GET_REMOTE_FEATURES_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® address of the device to connect to.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
supportedFeatures	An Imaging feature flags that Indicates the features the Imaging Server supports. For a detailed description of the Imaging feature flags, please refer to [BIP].
srmpOn	Reserved for future use. Set to FALSE.

4.2.2 CSR_BT_BIPC_CONNECT

Primitives \ Parameters	type	appHandle	maxPacketSize	deviceAddr	authorize	feature	pHandleInst	resultCode	resultSupplier	supportedFunctions	obexPeerMaxPacketSize	totalImagingDataCapacity	realmLength	*realm	passwordLength	*password	*userId	length	count	btConnId	windowSize	srmEnable
CSR_BT_BIPC_CONNECT_REQ	✓	✓	✓	✓	✓	✓							✓	✓	✓	✓	✓	✓	✓		✓	✓
CSR_BT_BIPC_CONNECT_CFM	✓						✓	✓	✓	✓	✓	✓								✓		

Table 3: CSR_BT_BIPC_CONNECT Primitives

Description

To start an OBEX Imaging Push session against an Imaging Push server, the application must send a CSR_BT_BIPC_CONNECT_REQ. BIPC will then respond with a CSR_BT_BIPC_CONNECT_CFM. In case the response code in the confirmation message is CSR_BT_OBEX_SUCCESS_RESULT_CODE an OBEX connection is established. Any other value indicates a failure in the attempt to initiate an OBEX connection.

The connect messages between the OBEX Imaging Push client and Server is guarded by a timer, thus if for some reason the server do not reply to the OBEX connect request within a fixed time interval the Bluetooth connection is disconnected direct. The timeout functionality is per default set to five seconds, or twenty seconds if the client request OBEX authentication. The timeout value can be disable, or change by changing CSR_BT_OBEX_CONNECT_TIMEOUT or CSR_BT_OBEX_CONNECT_WITH_AUTH_TIMEOUT, which is define in csr_bt_user_config.default.h. Note if the value of CSR_BT_OBEX_CONNECT_TIMEOUT or CSR_BT_OBEX_CONNECT_WITH_AUTH_TIMEOUT is change, it will influence all OBEX profiles.

The function:

```
CsrBtBipcConnectReqSend (CsrSchedQid       theAppHandle,
                         CsrUint16         theMaxPacketSize,
                         CsrBtDeviceAddr   theDestination,
                         CsrUint8          feature,
                         CsrBool           theAuthorize,
                         CsrUint16         realmLength,
                         CsrUint8          *realm,
                         CsrUint16         passwordLength,
                         CsrUint8          *password,
                         CsrCharString     *userId,
                         CsrUint32         length,
                         CsrUint32         count
                         CsrUint16         windowSize,
                         CsrBool           srmEnable )
```

defined in csr_bt_bipc_lib.h, builds and sends the CSR_BT_BIPC_CONNECT_REQ primitive to the BIPC profile.

Parameters

type	Signal identity, CSR_BT_BIPC_CONNECT_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
maxPacketSize	The maximum OBEX packet size allowed sending to the client application.

deviceAddr	The Bluetooth® address of the device to connect to.
authorize	If TRUE BIPC will initiate OBEX authentication against the server during the connection procedure.
feature	Must always be set to CSR_BT_IMAGE_PUSH_FEATURE, CSR_BT_REMOTE_CAMERA_FEATURE, or CSR_BT_AUTO_ARCHIVE_FEATURE. These are defined in csr_bt_bip_common.h.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
supportedFunctions	An Imaging functions flags that Indicates the function the Imaging Server supports. For a detailed description of the Imaging functions flags, please refer to [BIP]. Please notice that it is mandatory for an Imaging Push server to support the GetCapabilities and PutImage functions, while the PutLinkedThumbnail and PutAttachment functions are optional.
obexPeerMaxPacketSize	Indicates the maximum size OBEX packet that is allowed to send to the server.
totalImagingDataCapacity[8]	Indicates the maximum memory available for image storage on the server.
realmLength	Number of bytes in realm of type CsrUint16
*realm	A displayable string indicating for the user which userid and/or password to use. The first byte of the string is the character set of the string. The table below shows the different values for character set. Note that this pointer must be pfree by the application, and that this pointer can be NULL because the realm field is optional to set by the peer device.

Char set Code	Meaning
0	ASCII
1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4
5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8

9	ISO-8859-9
0xFF = 255	UNICODE

passwordLength	The length of the response password.
*password	Containing the challenge password of the OBEX authentication. This is a pointer which shall be allocated by the application.
*userId	Pointer to a zero terminated string (ASCII) containing the userId for the authentication. Note that the userId is ignored right now.
length	Length is use to express the approximate total length of the bodies of all the objects in the transaction. If set to 0 this header will not be include.
count	Count is use to indicate the number of objects that will be sent during this connection. If set to 0 this header will not be include
btConnId	Identifier which shall be used when using AMPM, for more information please refer to [AMPM].
windowSize	Controls how many packets the OBEX profile, and lower protocol layers, are allowed to cache on the data receive side. A value of zero (0) will cause the system to auto-detect this value.
srmEnable	TRUE enables local support for Single Response Mode (SRM). If SRM is enabled BIPC allows that PUT and GET commands, multiple OBEX request packets (PUT) or OBEX response packet (GET), can be send immediately, without waiting for the remote device. Please note, SRM can only be enabled if both sides support it. For more information about SRM, please refer to [GOEP2.0].

4.2.3 CSR_BT_BIPC_AUTHENTICATE

Primitives \ Parameters	type	pHandleInst	options	realmLength	* realm	deviceAddr	authPasswordLength	*authPassword	*authUserId	chalRealmLength	*chalRealm	chalPasswordLength	*chalPassword	*chalUserId
CSR_BT_BIPC_AUTHENTICATE_IND	✓	✓	✓	✓	✓	✓								
CSR_BT_BIPC_AUTHENTICATE_RES	✓						✓	✓	✓	✓	✓	✓	✓	✓

Table 4: CSR_BT_BIPC_AUTHENTICATE Primitives

Description

The indication and response signal is used when the Imaging Push server wants to OBEX authenticate the application. The application has to response with the password or pin number in the responsePassword and responseUserId for the server to identify the proper password using the `CsrBtBipcAuthenticateResSend`-function. If the client wants to challenge the server back, this is done using the `CsrBtBipcAuthenticateWithChalResSend`-function. **NOTE: It is *not* recommended to make the client challenge back the server.** Most commercially available servers will not handle that scenario correctly and some will disconnect the client.

Parameters

type	Signal identity, CSR_BT_BIPC_AUTHENTICATE_IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
options	Challenge information of type CsrUint8. Bit 0 controls the responding of a valid user Id. If bit 0 is set it means that the application must response with a user Id in a CSR_BT_BIPC_AUTHENTICATE_RES message. If bit 0 is not set the application can just zero terminated the first character of the user Id string. Bit 1 indicates the access mode being offered by the sender. If bit 1 is set the access mode is read only. If bit 1 is not set the sender gives full access, e.g. both read and write. Bit 2 - 7 is reserved.
realmLength	Number of bytes in realm of type CsrUint16 **Note** in this release version the 'realmLength' parameter is always set to 0x0000
* realm	A displayable string indicating for the user which userid and/or password to use. The first byte of the string is the character set of the string. The table below shows the different values for character set. Note that this pointer must be CsrPfree by the application, and that this pointer can be NULL because the realm field is optional to set by the peer device. **Note** in this release version the 'realm' pointer is always set to NULL

Char set Code	Meaning
0	ASCII

1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4
5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8
9	ISO-8859-9
0xFF = 255	UNICODE

deviceAddr	The Bluetooth address of the device that has initiated the OBEX authentication procedure
*authPassword	Contains the response password of the OBEX authentication. This is a pointer which shall be allocated by the application.
authPasswordLength	The length of the response password.
*authUserId	Pointer to a zero terminated string (ASCII) containing the userId for the authentication. This is a pointer which shall be allocated by the application.
chalRealmLength	Number of bytes in chalRealm of type CsrUint16
* chalRealm	A displayable string indicating to the server which userid and/or password to use in the authentication response. See *realm for details
*chalPassword	Contains the password of the OBEX authentication challenge. This is a pointer which shall be allocated by the application.
chalPasswordLength	The length of the challenge password.
*chalUserId	Pointer to a zero terminated string (ASCII) containing the userId for the authentication challenge. This is a pointer which shall be allocated by the application.

4.2.4 CSR_BT_BIPC_ABORT

Parameters / Primitives	type	pHandleInst
CSR_BT_BIPC_ABORT_REQ	✓	
CSR_BT_BIPC_ABORT_CFM	✓	✓

Table 5: CSR_BT_BIPC_ABORT Primitive

Description

The CSR_BT_BIPC_ABORT_REQ is used when the apps decides to terminate a multi-packet operation (such as GET/PUT) before it normally ends. The CSR_BT_BIPC_ABORT_CFM indicates that the server has received the abort response and the server is now resynchronized with the client. If the server does not respond the Abort Request or it response with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, the profile will disconnect the Bluetooth connection and send a CSR_BT_DISCONNECT_IND to the application.

Parameters

type	Signal identity, CSR_BT_BIPC_ABORT_REQ/CFM.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.

4.2.5 CSR_BT_BIPC_DISCONNECT

Primitives \ Parameters	type	normalDisconnect	pHandleInst	reasonCode	reasonSupplier
CSR_BT_BIPC_DISCONNECT_REQ	✓	✓			
CSR_BT_BIPC_DISCONNECT_IND	✓		✓	✓	✓

Table 6: CSR_BT_BIPC_DISCONNECT Primitive

Description

To disconnect a connection to an Imaging Push server (if any), the application must send a CSR_BT_BIPC_DISCONNECT_REQ to BIPC. When disconnected, the BIPC will respond with a CSR_BT_BIPC_DISCONNECT_IND. If the link is dropped in the middle of a session the application will receive a CSR_BT_BIPC_DISCONNECT_IND indicating that the OBEX Imaging push session is finished, and BIPC is ready to start a new session.

The disconnect messages between the OBEX Imaging Push client and Server is guarded by a timer, thus if for some reason the server do not reply to the OBEX disconnect request within a fixed time interval the Bluetooth connection is disconnected direct. The timeout functionality is per default set to five seconds. The timeout value can be disable, or change by changing CSR_BT_OBEX_DISCONNECT_TIMEOUT, which is define in csr_bt_user_config.default.h. Note if the value of CSR_BT_OBEX_DISCONNECT_TIMEOUT is change, it will influence all OBEX profiles.

Parameters

type	Signal identity, CSR_BT_BIPC_DISCONNECT_REQ/IND.
normalDisconnect	FALSE defines an Abnormal disconnect sequence where the Bluetooth connection is release direct. TRUE defines a normal disconnect sequence where the OBEX connection is release before the Bluetooth connection.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
reasonCode	The reason code of the operation. Possible values depend on the value of reasonSupplier. If e.g. the reasonSupplier == CSR_BT_SUPPLIER_CM then the possible reason codes can be found in csr_bt_cm_prim.h. If the reasonSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
reasonSupplier	This parameter specifies the supplier of the reason given in reasonCode. Possible values can be found in csr_bt_result.h

4.2.6 CSR_BT_BIPC_SECURITY_OUT

Primitives \ Parameters	type	appHandle	secLevel	pHandleInst	resultCode	resultSupplier
CSR_BT_BIPC_SECURITY_OUT_REQ	✓	✓	✓			
CSR_BT_BIPC_SECURITY_OUT_CFM	✓			✓	✓	✓

Table 7: CSR_BT_BIPC_SECURITY_OUT Primitives

Description

Applications that wish to change the enforcement to a specific profile security level, i.e. authentication, encryption and/or authorisation, can use this API to set up the security level for *new* connections. Note that this API is for the local device only and can be used from within any state.

The *CSR_BT_SECURITY_OUT_REQ* signal sets up the security level for new outgoing connections. Already established and pending connections are not altered. Note that *authorisation* should not be used for outgoing connections as that may be confusing for the user – there is really no point in requesting an outgoing connection and afterwards having to authorise as they are both locally-only decided procedures.

Note, that any attempts to set security to a less secure level than the mandatory security level will be rejected. See csr_bt_profiles.h for mandatory security settings. The default settings used by CSR Synergy Bluetooth are set to require authentication and encryption.

Note that if MITM protection is requested and the remote device does not have the required IO capabilities, pairing/bonding will fail and connections to the remote device *cannot* be made. See [SC] for further details.

Parameters

type	Signal identity CSR_BT_BIPC_SECURITY_OUT_REQ/CFM.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
appHandle	Application handle to which the confirm message is sent.
secLevel	The application must specify one of the following values:

- CSR_BT_SEC_DEFAULT : Use default security settings
- CSR_BT_SEC_MANDATORY : Use mandatory security settings
- CSR_BT_SEC_SPECIFY : Specify new security settings

If CSR_BT_SEC_SPECIFY is set the following values can be OR'ed additionally:

- CSR_BT_SEC_AUTHORISATION: Require authorisation
- CSR_BT_SEC_AUTHENTICATION: Require authentication
- CSR_BT_SEC_ SEC_ENCRYPTION: Require encryption (implies
- authentication)
- CSR_BT_SEC_MITM: Require MITM protection (implies encryption)

resultCode The result code of the operation. Possible values depend on the value of

	resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.2.7 CSR_BT_BIPC_REGISTER_QID_REQ

Primitives \ Parameters	type	qId
CSR_BT_BIPC_REGISTER_QID_REQ	✓	✓

Table 8: CSR_BT_BIPC_REGISTER_QID_REQ Primitives

Description

Used by the application to register a `CsrSchedQid` in the profile. This `CsrSchedQid` is used for identifying which “application” the signaling is to be send to.

Parameters

type	Signal identity CSR_BT_BIPC_REGISTER_QID_REQ.
qId	This value is used for identifying which BIPC instance the signal is coming from.

4.2.8 CSR_BT_BIPC_GET_INSTANCES_QID

Primitives \ Parameters	type	qId	phandlesListSize	*phandlesList
CSR_BT_BIPC_GET_INSTANCSE_QID_REQ	✓	✓		
CSR_BT_BIPC_GET_INSTANCES_QID_CFM	✓		✓	✓

Table 9: CSR_BT_BIPC_GET_INSTANCES_QID Primitives

Description

Retrieves list of registered BIPC-instances from the BIPC-manager.

Parameters

type	Signal identity, CSR_BT_BIPC_GET_INSTANCES_QID_REQ/CFM,
qId	This value is used for identifying which BIPC instance the signal is to be sent to. Must always be CSR_BT_BIPC_IFACEQUEUE.
phandlesListSize	Number of phandles that is returned in the CSR_BT_BIPC_GET_INSTANCES_QID_CFM.
*phandlesList	Pointer to array of phandles for registered BIPC instances. These values must be used for distinguishing the registered BIPC instances when sending signals to, or receiving signals from them. They are carried in the pHandleInst or qId parameters of the signals and library functions.

Library Function

A library function is provided by CSR_BT_BIPC_lib.h, and should be used for building and sending the CSR_BT_BIPC_GET_INSTANCES_QID_REQ signal:

```
void CsrBtBipcGetInstancesQIDReqSend(CsrSchedQid appHandle);
```

The parameters are as described in the table above.

4.3 Image Push Primitives

This section contains description of primitives that are specific to Image Push

4.3.1 CSR_BT_BIPC_PUSH_GET_CAPABILITIES

Parameters / Primitives	type	pHandleInst	responseCode	capabilitiesObjectLength	capabilitiesObjectOffset	payloadLength	*payload	srmpOn
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ	✓							✓
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_RES	✓							✓
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_IND	✓	✓		✓	✓	✓	✓	
CSR_BT_BIPC_PUSH_GET_CAPABILITIES_CFM	✓	✓	✓	✓	✓	✓	✓	

Table 10: CSR_BT_BIPC_PUSH_GET_CAPABILITIES Primitives

Description

The CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ is used for retrieving the imaging-capabilities object from the Imaging Push server.

Parameters

type	Signal identity, CSR_BT_BIPC_PUSH_GET_CAPABILITIES_REQ/RES/IND/CFM.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
responseCode	A successful response is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
capabilitiesObjectLength	The length of the Get Capabilities object,
capabilitiesObjectOffset	Offset (relative to payload) of the imaging-capabilities object or part of it (i.e. in case of a multi packet operation).
payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, the BIP Client can instruct the BIP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait;

however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.3.2 CSR_BT_BIPC_PUSH_PUT_IMAGE & CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE

Primitives \ Parameters	type	*ucs2imageName	*imageDescriptor	imageDescriptorLength	pHandleInst	responseCode	imageHandle	imageFileLength	finalFlag	*imageFile
CSR_BT_BIPC_PUSH_PUT_IMAGE_REQ	✓	✓	✓	✓						
CSR_BT_BIPC_PUSH_PUT_IMAGE_CFM	✓				✓	✓	✓			
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND	✓				✓			✓		
CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES	✓							✓	✓	✓

Table 11: CSR_BT_BIPC_PUSH_PUT_IMAGE & CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE Primitives

Description

The CSR_BT_BIPC_PUSH_PUT_IMAGE_REQ is used for pushing an image to the Imaging Push server. Although not mandatory requirement, it is highly recommended that a CSR_BT_BIPC_PUSH_PUT_IMAGE procedure attempts to be preceded by retrieving from the imaging-capabilities object, using the CSR_BT_BIPC_PUSH_GET_CAPABILITIES procedure of the Imaging Push server, so that images are sent to the server in a format the server supports.

Parameters

type	Signal identity, CSR_BT_BIPC_PUSH_PUT_IMAGE_REQ/CFM & CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND/RES.
*ucs2imageName	The image name describes the name of the image being sent. The imageName must be a null terminated 16 bit Unicode text string (UCS2). The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string
*imageDescriptor	The image descriptor describes the properties of the image being pushed. For a description of the definition for the image descriptor, together with the elements, and the attributes used in the image descriptor, please refer to [BIP]. The function “buildImgDescriptorHeader” under csr_bt_obex_string.c can be used for building the imageDescriptor.
imageDescriptorLength	The length of the imageDescriptor. The function “returnImgDescriptionLength“ under csr_bt_obex_string.c can be used for returning the imageDescriptorLength.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the image (and if requested by the server, the thumbnail version of it) is pushed to the server with success. Any other response code indicates a failure in the put image procedure.

	The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
imageHandle	The handle of the images, which the attachment is associated with.
imageFileLength	The length of the image, or the length of a part of it. In the CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_IND the value indicates the maximum size that the image (or a part of it) can be in the CSR_BT_BIPC_PUSH_PUT_IMAGE_FILE_RES message
finalFlag	The finalFlag must be set to TRUE if the image fits in one packet, or if it is the last packet of a multi packet operation.
*imageFile	The image or a part of it (i.e., in case of a multi packet operation).

4.3.3 CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE

Primitives \ Parameters	type	pHandleInst	thumbnailFileLength	*thumbnailFile	finalFlag
CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _IND	✓	✓	✓		
CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _RES	✓		✓	✓	✓

Table 12: CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE Primitives

Description

The CSR_BT_BIPC_PUSH_PUT_THUMBNAIL procedure is a scaled-down version of the CSR_BT_BIPC_PUSH_PUT_IMAGE procedure that does not use the image-descriptor header. It is only possible to push the thumbnail version of an image; it is not possible to push any other format using this procedure. BIPC only sends the CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _IND message to the application if the Imaging Push server requests a thumbnail version of an image.

Parameters

type	Signal identity, CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
thumbnailFileLength	The length of the thumbnail file object or the length of a part of it. In the CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _IND the value indicates the maximum size that the thumbnail file (or a part of it) can be in the CSR_BT_BIPC_PUSH_PUT_THUMBNAIL_FILE _RES message
*thumbnailFile	The thumbnail file object or a part of it (i.e., in case of a multi packet operation).
finalFlag	The finalFlag must be set to TRUE if the thumbnail file object fits in one packet, or if it is the last packet of a multi packet operation.

4.3.4 CSR_BT_BIPC_PUSH_PUT_ATTACHMENT & CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE

Primitives \ Parameters	type	*attachmentDescriptor	attachmentDescriptorLength	imageHandle	pHandleInst	responseCode	attachmentFileLength	finalFlag	*attachmentFile
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_REQ	✓	✓	✓	✓					
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_CFM	✓				✓	✓			
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND	✓				✓		✓		
CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES	✓						✓	✓	✓

Table 13: CSR_BT_BIPC_PUSH_PUT_ATTACHMENT & CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE Primitives

Description

To send an attachment associated with an image the application must send a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_REQ message. BIPC then sends a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND to the application, which the application must respond with a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES message. In case the image being pushed is large enough to require several OBEX packets the CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_IND / CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_FILE_RES message sequence is repeated.

When the Put Attachment procedure is finished BIPC sends a CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Image procedure.

Parameters

type	Signal identity, CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_REQ/CFM & CSR_BT_BIPC_PUSH_PUT_ATTACHMENT_IND/RES.
*attachmentDescriptor	The attachmentDescriptor describes the properties of the attachment being push. For a description of the definition for the attachment descriptor, together with the elements, and the attributes used in the attachment descriptor, please refer to [BIP]. The function “buildImgAttachmentDescriptorHeader” under csr_bt_obex_string.c can be used for building the attachmentDescriptor.
attachmentDescriptorLength	The length of the attachmentDescriptor. The function “returnImgAttachDescriptionLength“ under csr_bt_obex_string.c can be used for returning the attachmentDescriptorLength.
imageHandle	The handle of the images, which the attachment is associated with.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.

responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
attachmentFileLength	The length of the attachment object, or the length of a part of it. In the CSR_BT_BIPC_PUT_ATTACHMENT_IND the value indicates the maximum size that the attachment object (or part of it) can be in the CSR_BT_BIPC_PUT_ATTACHMENT_RES message.
finalFlag	The finalFlag must be set to TRUE if the attachment object fits in one packet, or if it is the last packet of a multi packet operation.
*attachmentFile	The attachment object or a part of it (i.e., in case of a multi packet operation).

4.4 Remote Camera Primitives

This section contains description of primitives that are specific to Remote Camera.

4.4.1 CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES

Parameters / Primitives	type	imageHandle	pHandleInst	propertiesObjectOffset	propertiesObjectLength	payloadLength	*payload	responseCode	srmpOn
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ	✓	✓							✓
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_IND	✓		✓	✓	✓	✓	✓		
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_RES	✓								✓
CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_CFM	✓		✓	✓	✓	✓	✓	✓	

Table 14: CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES Primitives

Description

The CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ is used for retrieving the image-properties object from the Imaging Responder.

Parameters

type	Signal identity, CSR_BT_BIPC_RC_GET_IMAGE_PROPERTIES_REQ/IND/RES/CFM
imageHandle	The imagehandle is used for identifying the relevant image on the server.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
propertiesObjectOffset	The length of the properties object,
propertiesObjectLength	Offset (relative to payload) of the properties object or part of it (i.e., in case of a multi packet operation).
payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, the BIP Client can instruct the BIP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE.

If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.4.2 CSR_BT_BIPC_RC_GET_IMAGE

Primitives \ Parameters	type	imageHandle	imageDestiptorLength	*imageDescriptor	pHandleInst	imageObjectOffset	imageObjectLength	payloadLength	*payload	responseCode	srmpOn
CSR_BT_BIPC_RC_GET_IMAGE_REQ	✓	✓	✓	✓							✓
CSR_BT_BIPC_RC_GET_IMAGE_IND	✓				✓	✓	✓	✓	✓		
CSR_BT_BIPC_RC_GET_IMAGE_RES	✓										✓
CSR_BT_BIPC_RC_GET_IMAGE_CFM	✓				✓	✓	✓	✓	✓	✓	

Table 15: CSR_BT_BIPC_RC_GET_IMAGE Primitives

Description

The CSR_BT_BIPC_RC_GET_IMAGE_REQ is used for retrieving the image from the Imaging Responder. The format of the requested image is supplied in the image-descriptor object of the request.

Parameters

type	Signal identity, CSR_BT_BIPC_RC_GET_IMAGE_REQ/IND/RES/CFM
imageHandle	The imagehandle is used for identifying the relevant image on the server.
*imageDescriptor	The image descriptor describes the properties of the requested image. For a description of the definition for the image descriptor, together with the elements, and the attributes used in the image descriptor, please refer to [BIP]. The function "buildImgDescriptorHeader" under csr_bt_obex_string.c can be used for building the imageDescriptor.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
imageDescriptorLength	The length of the imageDescriptor. The function "returnImgDescriptionLength" under csr_bt_obex_string.c can be used for returning the imageDescriptorLength.
imageObjectOffset	The length of the image object,
imageObjectLength	Offset (relative to payload) of the image object or part of it (i.e., in case of a multi packet operation).
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure.

	The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, the BIP Client can instruct the BIP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.4.3 CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL

Primitives \ Parameters	type	imageHandle	pHandleInst	thumbnailObjectOffset	thumbnailObjectLength	payloadLength	*payload	responseCode	srmpOn
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ	✓	✓							✓
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_IND	✓		✓	✓	✓	✓	✓		
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_RES	✓								✓
CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_CFM	✓		✓	✓	✓	✓	✓	✓	

Table 16: CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL Primitives

Description

The CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ is used for retrieving the thumbnail of an image from the Imaging Responder.

Parameters

type	Signal identity, CSR_BT_BIPC_RC_GET_LINKED_THUMBNAIL_REQ/IND/RES/CFM
imageHandle	The imagehandle is used for identifying the relevant image on the server.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
thumbnailObjectOffset	The length of the image object,
thumbnailObjectLength	Offset (relative to payload) of the image object or part of it (i.e., in case of a multi packet operation).
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, the BIP Client can instruct the BIP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.4.4 CSR_BT_BIPC_RC_GET_MONITORING_IMAGE /_HEADER/ _FILE

Primitives \ Parameters	type	storeFlag	pHandleInst	imageHandle	imageFileOffset	imageFileLength	payloadLength	*payload	responseCode	srmpOn
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ	✓	✓								✓
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_IND	✓		✓	✓						
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_RES	✓									✓
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_IND	✓		✓		✓	✓	✓	✓		
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_RES	✓									✓
CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_CFM	✓		✓						✓	

Table 17: CSR_BT_BIPC_RC_GET_MONITORING_IMAGE /_HEADER /_FILE Primitives

Description

The CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ is used for retrieving a monitoring image from the Imaging Responder. As part of the request the storeFlag can be set, this implies that the Imaging Responder's shutter is released and the full image is stored on the server.

Parameters

type	Signal identity, CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_REQ/CFM, CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_HEADER_IND/RES, or CSR_BT_BIPC_RC_GET_MONITORING_IMAGE_FILE_IND/RES
storeFlag	The application can indicate whether the monitoring image procedure should be accompanied by releasing the shutter and storing the corresponding full size image on the server.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
imageHandle	If the storeFlag is set the imageHandle is returned. The imagehandle can be used later for retrieving the full size image.
ImageFileOffset	The length of the monitoring-image object,
imageFileLength	Offset (relative to payload) of the monitoring-image object or part of it (i.e., in case of a multi packet operation).
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the attachment is pushed to the server with success. Any other response code indicates a failure in the Put Attachment procedure.

	The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, the BIP Client can instruct the BIP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5 Automatic Archive Primitives

This section contains description of primitives that are specific to Automatic Archive. It is worth mentioning that the primitives in this section deviate from the primitives in the previous sections. The reason for this is that the BIP client when connected using the Automatic Archive feature is working as a secondary server. The secondary server is the one to react to the actions initiated from the secondary client.

4.5.1 CSR_BT_BIPC_AA_GET_CAPABILITIES /_HEADER /_OBJECT

Primitives \ Parameters	type	pHandleInst	connectionId	responseCode	capabilitiesObjectLength	*capabilitiesObject	srmpOn
CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_IND	✓	✓	✓				
CSR_BT_BIPS_AA_GET_CAPABILITIES_HEADER_RES	✓			✓			✓
CSR_BT_BIPS_AA_GET_CAPABILITIES_OBJECT_IND	✓	✓	✓				
CSR_BT_BIPS_AA_GET_CAPABILITIES_OBJECT_RES	✓			✓	✓	✓	✓

Table 18: CSR_BT_BIPC_AA_GET_CAPABILITIES /_HEADER /_OBJECT Primitives

Description

This signal is part of an operation where the secondary client has requested to retrieve the imaging-capabilities object from the secondary BIP server. The imaging-capabilities object is a mandatory object that describes in detail the various options, formats and attributes that are supported by the secondary BIP server.

An example of an Imaging-capabilities object for the BIP Image Push client is illustrated below:

```
<imaging-capabilities version=”1.0”>
<preferred-format encoding=”JPEG” pixel=”1280*960” />
<image-formats encoding=”JPEG” pixel=”160*120” maxsize=”5000” />
<image-formats encoding=”JPEG” pixel=”320*240” />
<image-formats encoding=”JPEG” pixel=”640*480” />
<image-formats encoding=”JPEG” pixel=”1280*960” />
<attachment-formats content-type=”audio/basic” />
<filtering-parameters created=”1” modified=”1” />
</imaging-capabilities>
```

For a description of the definition for the imaging-capabilities object, together with the elements, and the attributes used in the imaging-capabilities object, please refer to [BIP].

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_CAPABILITIES_HEADER_IND/CSR_BT_RES, CSR_BT_BIPS_AA_GET_CAPABILITIES_OBJECT_IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.

responseCode	For accepting the GetCapabilities request the code is: CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. However, if the response is large enough to require multiple CSR_BT_BIPC_AA_GET_CAPABILITIES_OBJECT packets, only the last response must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, and the other must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetCapabilities request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
capabilitiesObjectLength	The length of the capabilities object part being sent.
*capabilitiesObject	The imaging-capabilities object or part of it being sent.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.2 CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES /_HEADER /_OBJECT

Primitives \ Parameters	type	pHandleInst	connectionId	imageHandleOffset	payloadLength	*payload	responseCode	allowedImageLength	propertiesObjectLength	*propertiesObject	srmpOn
CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES _HEADER_IND	✓	✓	✓	✓	✓	✓					
CSR_BT_BIPS_AA_GET_IMAGE_PROPERTIES _HEADER_RES	✓						✓				✓
CSR_BT_BIPS_AA_GET_IMAGE_PROPERTIES _OBJECT _IND	✓	✓	✓					✓			
CSR_BT_BIPS_AA_GET_IMAGE_PROPERTIES _OBJECT _RES	✓						✓		✓	✓	✓

Table 19: CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES /_HEADER /_OBJECT Primitives

Description

The header indication and response signal is the first part of an operation where the client requests image properties of an image on the secondary BIP server. If the secondary BIP server accepts the request from the secondary client the following messages will be CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_OBJECT indications/responses.

This object signal is the other part of the operation where the secondary client has requested image properties on an image from the secondary BIP server. Part of or the entire image properties object is returned.

An example of an image properties object is illustrated below:

```
<image-properties version=”1.0” handle=”1000001”>
<native encoding=”JPEG” pixel=”1280*1024” size=”1048476” />
<variant encoding=”JPEG” pixel=”640*480” />
<variant encoding=”JPEG” pixel=”160*120” />
<variant encoding=”GIF” pixel=”80*60-640*480” />
<attachment content-type=”text/plain” name=”DSCF0001.txt”
size=”5120” />
<attachment content-type=”audio/basic” name=”DSCF0001.wav”
size=”102400” />
</image-properties>
```

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_HEADER_IND/RES, CSR_BT_BIPS_AA_GET_IMAGE_PROPERTIES_OBJECT_IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
imageHandleOffset	The handle of the image the secondary client is requesting properties on. The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at

	payload[imageHandleOffset]. The image handle is always 7 bytes in length.
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	For accepting the GetImageProperties request, the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. However, if the response is large enough to require multiple CSR_BT_BIPC_AA_GET_IMAGE_PROPERTIES_OBJECT packets, only the last response must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, and the other must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetImageProperties request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
allowedImageLength	Indicates how much data is maximal allowed to return in the response. This must be obeyed by the application. Please be aware that this number may change from indication to indication.
propertiesObjectLength	The length of the image descriptor.
*propertiesObject	The secondary client can supply an image descriptor in the indication. This descriptor specifies which format the secondary client wants to receive the image in. The application is responsible for making sure that the image obeys these specifications prior to being transmitted. An empty descriptor is allowed in which case the image is transmitted in its native format. The descriptor is in XML and the specification of the format is found in [BIP]. Furthermore, see example above.
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.3 CSR_BT_BIPC_AA_GET_IMAGE /_HEADER /_OBJECT

Parameters / Primitives	type	pHandleInst	connectionId	imageHandleOffset	descriptorLength	descriptorOffset	payloadLentgth	*payload	imageTotalLength	responseCode	allowedObjectLength	imageObjectLength	*imageObject	srmpOn
CSR_BT_BIPC_AA_GET_IMAGE_HEADER_IND	✓	✓	✓	✓	✓	✓	✓	✓						
CSR_BT_BIPS_AA_GET_IMAGE_HEADER_RES	✓								✓	✓				✓
CSR_BT_BIPS_AA_GET_IMAGE_OBJECT_IND	✓	✓	✓								✓			
CSR_BT_BIPS_AA_GET_IMAGE_OBJECT_RES	✓									✓		✓	✓	✓

Table 20: CSR_BT_BIPC_AA_GET_IMAGE /_HEADER /_OBJECT Primitives

Description

The header indication and response signal is the first part of an operation where the secondary client requests an image file from the secondary BIP server. If the secondary BIP server accepts the request from the secondary client the following messages will be CSR_BT_BIPS_AA_GET_IMAGE_OBJECT indications/responses.

An example of the image descriptor object that can be received from the secondary BIP automatic archive client is illustrated below:

```
<image-descriptor version="1.0">
<image encoding="JPEG" pixel="1280*960" size="5000000" />
</image-descriptor>
```

This object signal is the other part of the operation where the secondary client has requested an image from the secondary BIP server, part of or the entire image is returned.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_IMAGE_HEADER_IND/RES, CSR_BT_BIPS_AA_GET_IMAGE_OBJECT_IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
imageHandleOffset	The handle of the image the client is requesting properties on. The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at payload[imageHandleOffset]. The image handle is always 7 bytes in length.
descriptorlength	The length of the image descriptor.
descriptorOffset	The secondary client can supply an image descriptor in the indication. This descriptor specifies which format the secondary client wants to receive the image in. The application is responsible for making sure that the image obeys these specifications prior to being transmitted.

	An empty descriptor is allowed in which case the image is transmitted in its native format. The descriptor is in XML and the specification of the format is found in [BIP]. Furthermore, see example above. The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at payload[descriptorOffset].
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
imageTotalLength	Holds the total length of the image being sent. This parameter is Mandatory but can be set to 0, in case the byte size of the transferred image is not known (this could happen, for instance, if the image is being produced on the fly)
responseCode	For accepting the GetImage request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. However, if the response is large enough to require multiple CSR_BT_BIPC_AA_GET_IMAGE_OBJECT packets, only the last response must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, and the other must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetImage request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
allowedObjectLength	Indicates how much data is maximal allowed to return in the response. This must be obeyed by the application. Please be aware that this number may change from indication to indication.
imageObjectLength	The length of the image object,
*imageObject	Offset (relative to payload) of the image object or part of it (i.e., in case of a multi packet operation).
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.4 CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL /_HEADER /_OBJECT

Primitives \ Parameters	type	pHandleInst	connectionId	imageHandleOffset	payloadLentgth	*payload	responseCode	allowedObjectLength	thumbnailObjectLength	*thumbnailObject	srmpOn
CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_IND	✓	✓	✓	✓	✓	✓					
CSR_BT_BIPS_AA_GET_LINKED_THUMBNAIL_HEADER_RES	✓						✓				✓
CSR_BT_BIPS_AA_GET_LINKED_THUMBNAIL_OBJECT_IND	✓	✓	✓					✓			
CSR_BT_BIPS_AA_GET_LINKED_THUMBNAIL_OBJECT_RES	✓						✓		✓	✓	✓

Table 21: CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL /_HEADER /_OBJECT Primitives

Description

The header indication and response signal is the first part of an operation where the secondary client requests an image file from the secondary BIP server. If the secondary BIP server accepts the request from the secondary client the following messages will be CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT indications/responses.

This object signal is the following part of the operation where the secondary client has requested a linked thumbnail of an image from the secondary BIP server. Part of or the entire thumbnail is returned.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_HEADER_IND/RES, CSR_BT_BIPS_AA_GET_ LINKED_THUMBNAIL_OBJECT_IND/RES.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
imageHandleOffset	The handle of the image the secondary client is requesting properties on. The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at payload[imageHandleOffset]. The image handle is always 7 bytes in length.
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	For accepting the GetLinkedThumbnail request, the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. However, if the response is large enough to require multiple CSR_BT_BIPC_AA_GET_LINKED_THUMBNAIL_OBJECT packets, only the last response must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, and the other must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetLinkedThumbnail request:

	CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
allowedObjectLength	Indicates how much data is maximal allowed to return in the response. This must be obeyed by the application. Please be aware that this number may change from indication to indication.
thumbnailObjectLength	The length of the thumbnail object,
*thumbnailObject	Offset (relative to payload) of the thumbnail object or part of it (i.e., in case of a multi packet operation).
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.5 CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER

Primitives \ Parameters	type	connectionId	pHandleInst	nbReturnedHandles	listStartOffset	latestCapturedImages	allowedDescriptorLength	imageDescriptorOffset	imageDescriptorLength	payloadLength	*payload	responseCode	*imageDescriptor	srmpOn
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_RES	✓			✓					✓			✓	✓	✓

Table 22: CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER Primitives

Description

A typical first operation when performing Automatic Archive is to learn what images are on the BIP secondary server. This is initiated with the CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER. The operation is initiated by the secondary BIP client.

An example of the image handles descriptor object is illustrated below:

```
<image-handles-descriptor version="1.0">
<filtering-parameters created="20070101T000000Z-20070101T235959Z" />
</image-handles-descriptor>
```

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_IMAGE_LIST_HEADER_IND/RES,
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
nbReturnedHandles	In the header indication this value is used for specifying how many items could be returned in the image list. In the header response this value is used for specifying the number of items actually returned. More information on the nbReturnedHandles can be found in [BIP].
listStartOffset	This value is used for specifying what offset in the image list on the secondary server to start the image list to be returned to the secondary client. E.g. a previous operation could have retrieved the first 10 image list elements. In the second operation this parameter would then be set to 11.
latestCapturedImages	This boolean is used for indicating to the secondary server if the image list should be sorted descending in respect to capture date.
allowedDescriptorLength	This value identifies the maximum allowed length of the returned image descriptor.
imageDescriptorOffset	In the indication this descriptor represents the criteria the server wishes the client applies to the image list before sending it. The value in this parameter indicates the offset in the payload data where the

	image descriptor data starts. I.e. the data is located at payload[imageDescriptorOffset]. An example of an image handles descriptor is seen above. Detailed information is found in [BIP].
imageDescriptorLength	This value is used for identifying the length of the image descriptor.
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	For accepting the GetImageList request, the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. The following response codes reject the GetImageList request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
*imageDescriptor	A pointer to an image handles descriptor. The descriptor transmitted in the response is the criteria actually applied by the secondary server. Ideally the two descriptors are identical. Please be aware that the length of this descriptor must not exceed the maximally allowed OBEX packet size. An example of an image handles descriptor is seen above. Detailed information is found in [BIP].
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.6 CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT

Primitives \ Parameters	type	connectionId	pHandleInst	allowedDescriptorLength	responseCode	imageListObjectLength	*imageListObject	srmpOn
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND	✓	✓	✓	✓				
CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_RES	✓				✓	✓	✓	✓

Table 23: CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT Primitives

Description

The signal CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND is used for indicating the reception of (part of) an image list. This signal will have been preceded by the GetImageListHeader signal exchange.

An example of the image listing object is illustrated below:

```
<images-listing version=”1.0”>
<image handle=”1000001” created=”20070801T060000Z” />
<image handle=”1000003” created=”20070801T060115Z”
modified=”20070808T101500Z” />
<image handle=”1000004” created=”20070801T080137Z” />
</images-listing>
```

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_IMAGE_LIST_OBJECT_IND/RES,
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
allowedObjectLength	This value indicates the maximum allowed length of the image list
responseCode	If the image list fits in one response or the final part is being transmitted the responseCode must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. All intermediate responses must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetImageList request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.

	The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
imageListObjectLength	This value indicates the length of the image list
*imageListObject	Pointer to the image list
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.7 CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER

Primitives \ Parameters	type	pHandleInst	connectionId	imageHandleOffset	attachmentNameOffset	attachmentNameLength	payloadLength	*payload	responseCode	srmpOn
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT _ HEADER _IND	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT _ HEADER_RES	✓								✓	✓

Table 24: CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER Primitives

Description

The CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_IND is used for retrieving a linked attachment associated with an image from the BIP Automatic Archive secondary server.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_HEADER_IND/RES,
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
imageHandleOffset	The imageHandle is used for identifying the relevant image on the client. The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at payload[imageHandleOffset]. The image handle is always 7 bytes in length.
attachmentNameOffset	The name of the attachment file The value in this parameter indicates the offset in the payload data where the image handle file data starts. I.e. the data is located at payload[imageHandleOffset].
attachmentNameLength	The length of the attachment name
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	For accepting the GetLinkedAttachment request, the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. The following response codes reject the GetLinkedAttachment request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE

	If the connectionId is invalid it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.8 CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT

Primitives \ Parameters	type	pHandleInst	connectionId	allowedObjectLength	responseCode	attachmentObjectLength	*attachmentObject	srmpOn
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_IND	✓	✓	✓	✓				
CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_RES	✓				✓	✓	✓	✓

Table 25: CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT Primitives

Description

The CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_IND is used for retrieving a linked attachment associated with an image from the BIP Automatic Archive secondary server.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_GET_LINKED_ATTACHMENT_OBJECT_IND/RES,
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.
allowedObjectLength	The maximum allowed length of the object.
responseCode	If the attachment fits in one response or the final part is being transmitted the responseCode must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. All intermediate responses must be CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. The following response codes reject the GetLinkedAttachment request: CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE CSR_BT_OBEX_BAD_REQUEST_RESPONSE_CODE CSR_BT_OBEX_NOT_ACCEPTABLE_RESPONSE_CODE CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE If the connectionId is invalid, it is recommended that the operation is being rejected with the response code: CSR_BT_OBEX_SERVICE_UNAVAILABLE_RESPONSE_CODE. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].
attachmentObjectLength	The length of the attachment object
*attachmentObject	Pointer to the attachment object
srmpOn	If Single Response Mode is enabled, see section 4.2.2, BIPC secondary server

can instruct BIPS secondary client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE.

If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause BIPS secondary client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5.9 CSR_BT_BIPC_AA_DELETE_IMAGE

Primitives \ Parameters	type	pHandleInst	connectionId	imageHandleOffset	payloadLength	*payload	responseCode
CSR_BT_BIPC_AA_DELETE_IMAGE_IND	✓	✓	✓	✓	✓	✓	
CSR_BT_BIPC_AA_DELETE_IMAGE_RES	✓						✓

Table 26: CSR_BT_BIPC_AA_DELETE_IMAGE Primitives

Description

The CSR_BT_BIPS_AA_DELETE_IMAGE_IND is used for deleting an image on the BIP Automatic Archive secondary server.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_DELETE_IMAGE_IND/RES,
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request. .
imageHandleOffset	This value is used for identifying which BIPS instance the signal is coming from
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
responseCode	A successful response is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. The meaning of the response codes for the Basic Imaging Profile is described in [BIP].

4.5.10 CSR_BT_BIPC_AA_ABORT

Primitives \ Parameters	type	pHandleInst	connectionId
CSR_BT_BIPC_AA_ABORT_IND	✓	✓	✓

Table 27: CSR_BT_BIPC_AA_ABORT Primitives

Description

The CSR_BT_BIPS_AA_ABORT_IND is used for aborting an ongoing operation with the BIP Automatic Archive secondary server.

Parameters

type	Signal identity, CSR_BT_BIPC_AA_ABORT_IND,
pHandleInst	This value is used for identifying which BIPC instance the signal is coming from.
connectionId	The connection Id for this secondary session, the BIP secondary client must use this Id in the request.

5 Document References

Document	Reference
Basic Imaging Profile Revision V11r00 28 August 2010	[BIP]
GENERIC OBJECT EXCHANGE PROFILE Revision V20r00 26 August 2010	[GOEP2.0]
IrDA Object Exchange Protocol - IrOBEX Version 1.2 or Version 1.5	[OBEX]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc	[SC]
CSR Synergy Bluetooth, AMPM – Alternate MAC and PHY Manager API Description, api-0148-ampm.pdf	[AMPM]
CSR Synergy Bluetooth. CM – Connection Manager API Description, doc. no. api-0101-cm	[CM]

Terms and Definitions

BlueCore®	Group term for CSR’s range of Bluetooth wireless technology chips
Bluetooth®	A set of technologies providing audio and data transfer over short-range radio connections
BIPC	OBEX Basic Imaging Profile (Client)
CSR	Cambridge Silicon Radio
SDC	Service Discovery Client
SIG	Special Interest Group
UniFi™	Group term for CSR’s range of chips designed to meet IEEE 802.11 standards
SRM	Single Response Mode
SRMP	Single Response Mode Parameters
GOEP	Generic Object Exchange Profile
AMPM	Alternate MAC and PHY Manager

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0
2	23 JAN 12	Ready for release 18.2.2

TradeMarks, Patents and Licences

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.