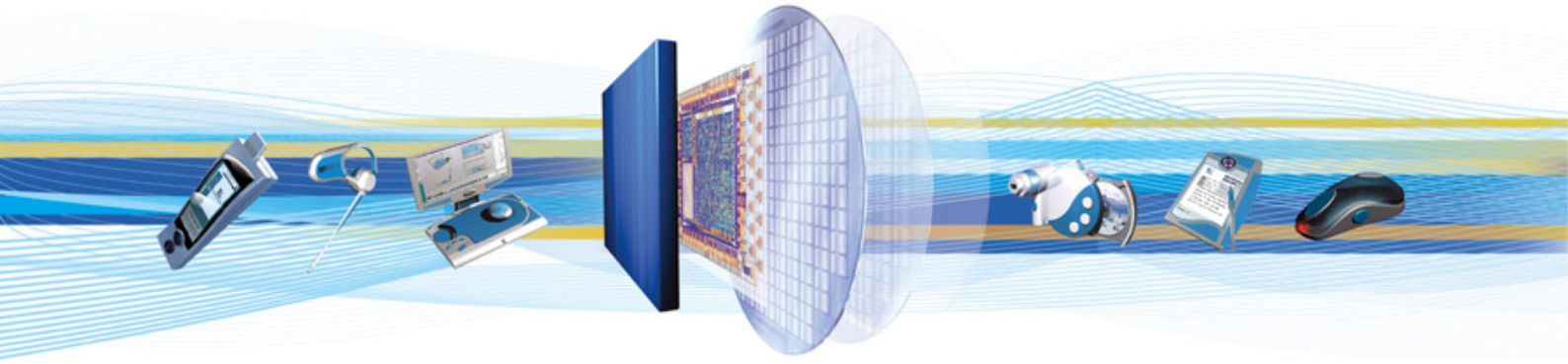
CSR Synergy Bluetooth 18.2.0

OBEX Push Client

API Description

November 2011



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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Contents

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

List of Figures

List of Tables

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the OBEX Push Client (OPC). The OPC conforms to the client side of the OBEX Push Profile, ref. [OPP].

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. OPC and the application
- The OPC shall only handle one request at the time
- Bonding (pairing) is NOT handled by the OPC

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

2 Description

2.1 Introduction

The OBEX Push Client (OPC) provides the following services to the application:

- Inquiry for devices
- Screening of these
- Service discovery parameter retrieval
- Connection handling
- OBEX protocol handling

2.2 Reference Model

The OPC interfaces to the Connection Manager (CM).

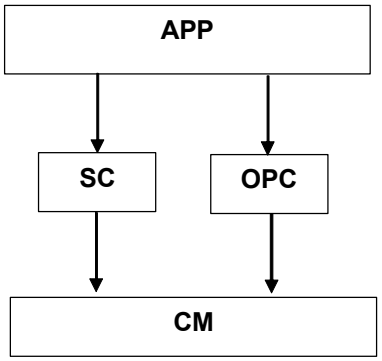


Figure 1: Reference model

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

2.3 Sequence Overview

If, in IDLE state, a connect request is received from the application, the OPC starts to connect to the specified device and the CONNECT state is entered. When the application receives a confirmation on the connect request, the application can issue a request (get or put) to start the information transfer for the object. The application can perform multiple gets/puts (one at the time) before disconnecting the connection. An exchange is performed as a put followed by a get. When the application disconnects the service, the OPC re-enters IDLE state.

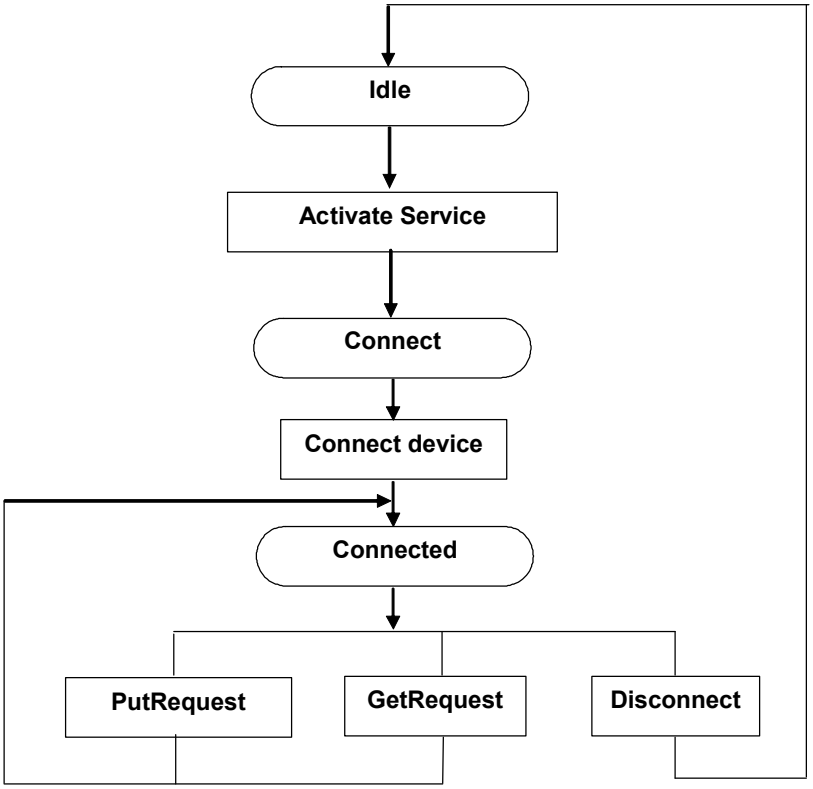


Figure 2: OPC state diagram

CSR Synergy Bluetooth 18.2.0 OBEX Push Client API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3 Interface Description

3.1 Connect

When the application wants to connect to an Object Push Server it has to send a CSR_BT_OPC_CONNECT_REQ. In this message the application has to specify which device to connect to. The OPC sends a CSR_BT_OPC_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.

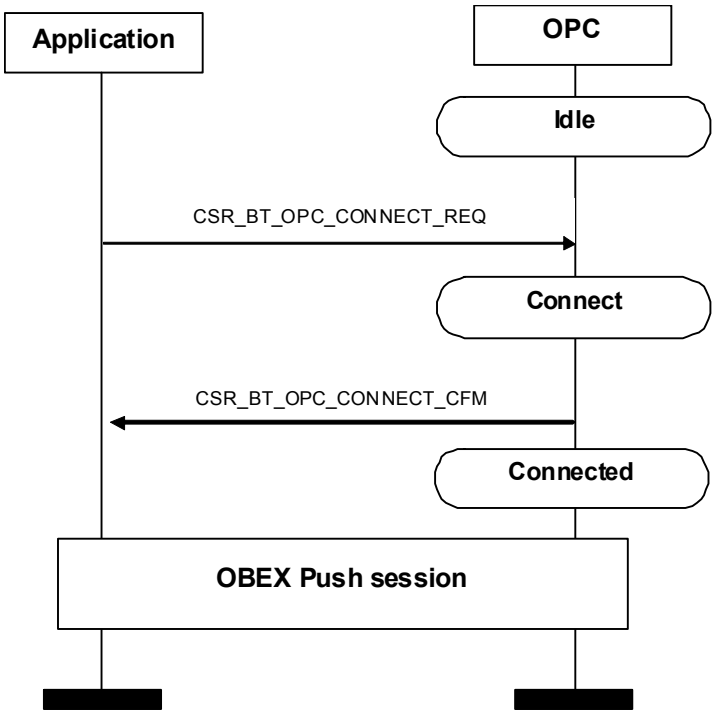


Figure 3: Connect

CSR Synergy Bluetooth 18.2.0 OBEX Push Client API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3.2 Cancel Connect

The application can cancel an outgoing connection request by sending a CSR_BT_OPC_CANCEL_CONNECT_REQ. If the outgoing connection can be cancelled the responses will be a CSR_BT_OPC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

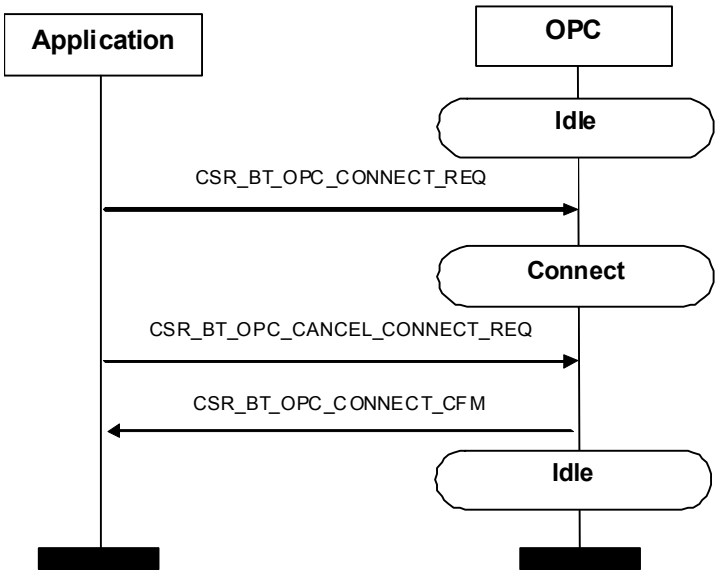


Figure 4: Cancel Connect I

Please note that if the application requests a CSR_BT_OPC_CANCEL_CONNECT_REQ while the OPC is sending a CSR_BT_OPC_CONNECT_CFM with the response code CSR_BT_OBEX_SUCCESS_RESPONSE_CODE to the application, then will OPC consider the CSR_BT_OPC_CANCEL_CONNECT_REQ as a CSR_BT_OPC_DISCONNECT_REQ and the application will receive a CSR_BT_OPC_DISCONNECT_IND.

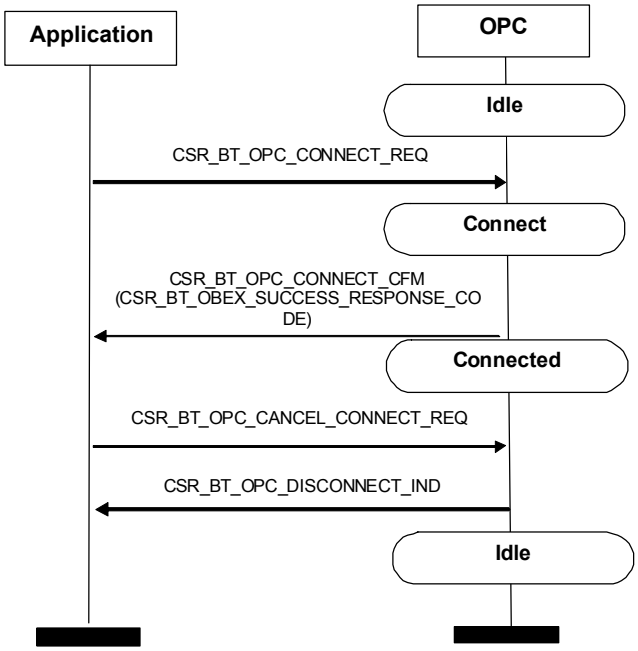


Figure 5: Cancel Connect II

CSR Synergy Bluetooth 18.2.0 OBEX Push Client API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3.3 Object Transfer

Objects are transmitted to the server by issuing a CSR_BT_OPC_PUT_REQ. OPC then send a CSR_BT_OPC_PUT_OBJECT_IND to the application, which the application must respond with a CSR_BT_OPC_PUT_OBJECT_RES message. In case the object being pushed is large enough to require several OBEX packets the CSR_BT_OPC_PUT_OBJECT_IND/CSR_BT_OPC_PUT_OBJECT_RES message sequence is repeated.

When the Put Object procedure is finished, OPC sends a CSR_BT_OPC_PUT_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the Object is pushed to the server with success. Any other response code indicates a failure in the Put Object procedure.

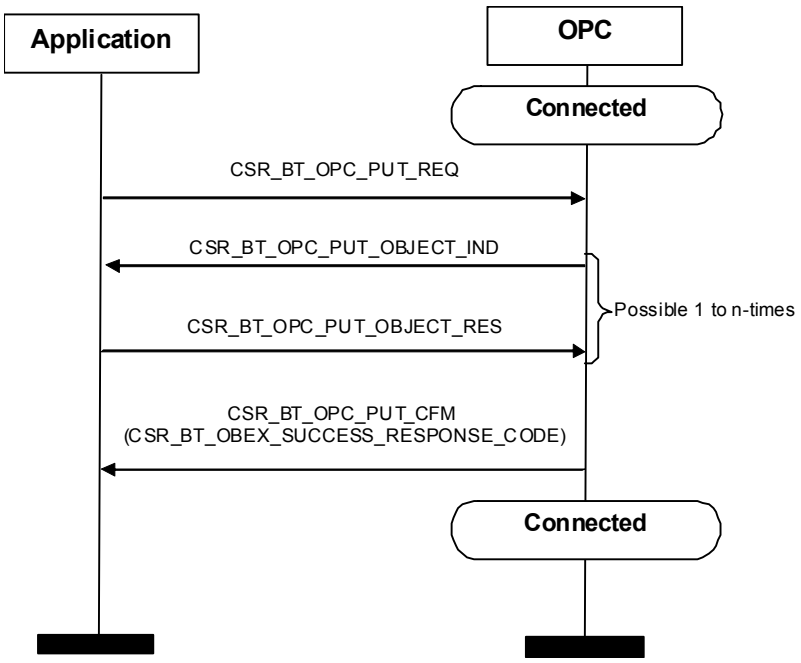


Figure 6: Put object

Default objects of a specific type can be pulled from the server by issuing a CSR_BT_OPC_GET_HEADER_REQ.

The server responds with the result of the operation in a CSR_BT_OPC_GET_HEADER_CFM signal. Where the result code:

- CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the object has been retrieved with success
- CSR_BT_OBEX_CONTINUE_RESPONSE_CODE indicates that the server will respond with multiple fragments
- Any other response code indicates that the object could not be retrieved from the server

If the result in the CSR_BT_OPC_GET_HEADER_CFM signal is CSR_BT_OBEX_CONTINUE_RESPONSE_CODE the server will respond with multiple fragments. In this case the application must send a CSR_BT_OPC_GET_OBJECT_REQ signal in order to get the next fragment. The server will then respond with the result of the operation in a CSR_BT_OPC_GET_OBJECT_CFM signal.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Please notice that the CSR_BT_OPC_GET_OBJECT_REQ/CSR_BT_OPC_GET_OBJECT_CFM sequence must be repeated as long as the result code in the CSR_BT_OPC_GET_OBJECT_CFM signal is CSR_BT_OBEX_CONTINUE_RESPONSE_CODE.

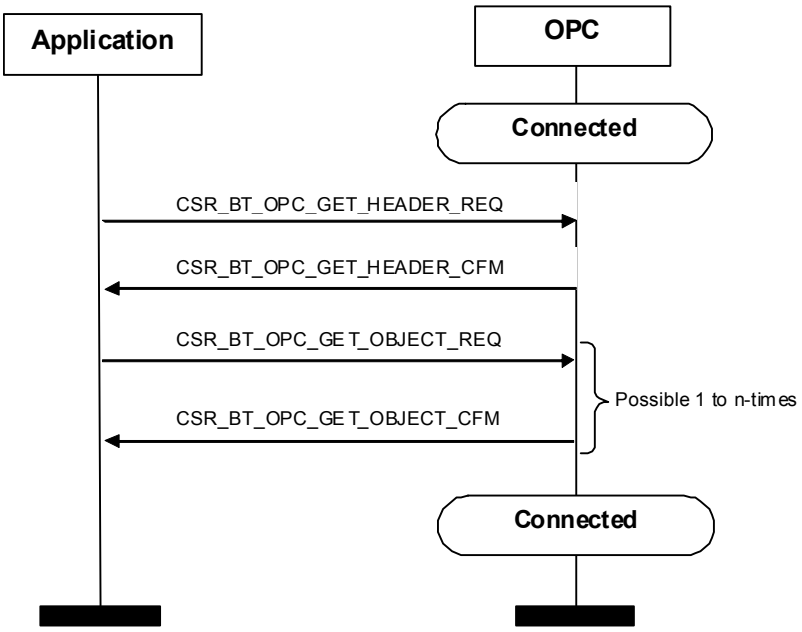


Figure 7: Get default object

3.4 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_OPC_ABORT_REQ). The response (CSR_BT_OPC_ABORT_CFM) is received indicating that the abort request is a success. It is also indicating that the abort request is received and the Object Push Server is now re-synchronized with the client. If anything else is returned the OPC will disconnect the link and send CSR_BT_OPC_DISCONNECT_IND to the application.

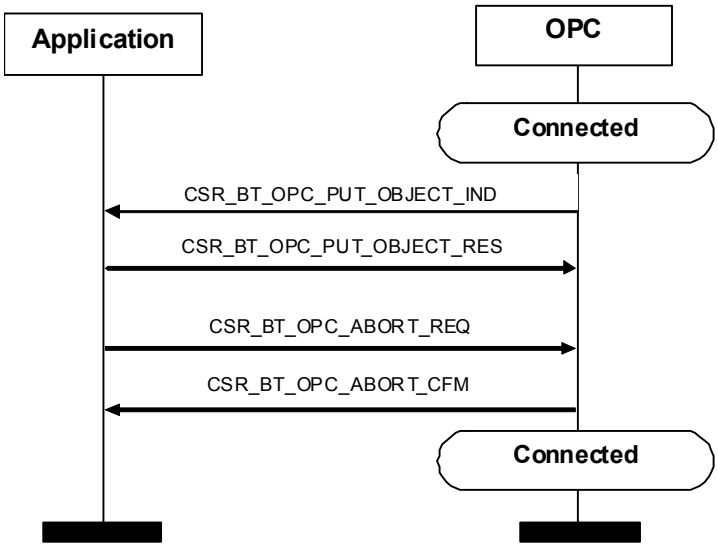


Figure 8: Abort operation

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3.5 Disconnect

Sending a CSR_BT_OPC_DISCONNECT_REQ disconnects the current connection (if any). The disconnect may take some time and is confirmed with a CSR_BT_OPC_DISCONNECT_IND signal. After the connection is disconnected OPC enters IDLE state.

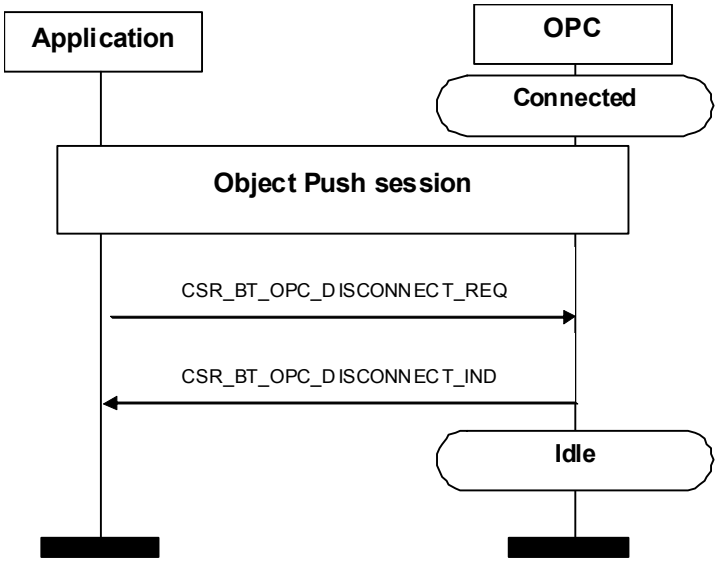


Figure 9: Normal disconnect

In case the peer side prematurely disconnects, the OPC sends a CSR_BT_OPC_DISCONNECT_IND to the application and enters IDLE state.

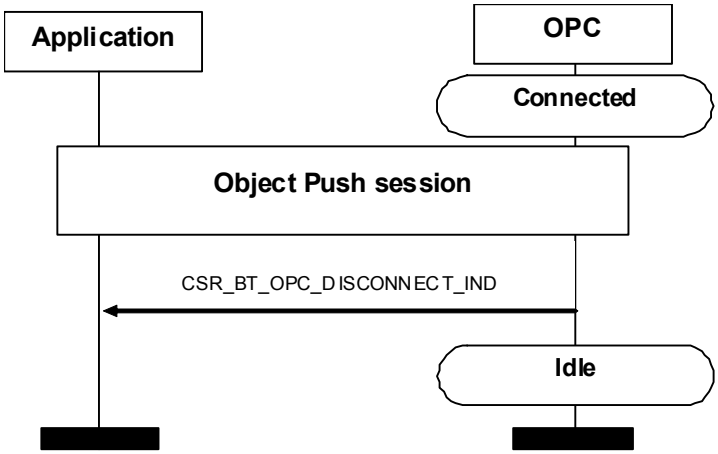


Figure 10: Abnormal disconnect

3.6 Payload Encapsulated Data

3.6.1 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:

CSR Synergy Bluetooth 18.2.0 OBEX Push Client API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

```
CsrBtCommonPrim    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), it 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.

3.6.2 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.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4 OBEX Push 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_opc_prim.h file.

4.1 List of All Primitives

Primitives	Reference
CSR_BT_OPC_CONNECT_REQ	See section 4.2
CSR_BT_OPC_CONNECT_CFM	See section 4.2
CSR_BT_OPC_ABORT_REQ	See section 4.3
CSR_BT_OPC_ABORT_CFM	See section 4.3
CSR_BT_OPC_DISCONNECT_REQ	See section 4.4
CSR_BT_OPC_DISCONNECT_IND	See section 4.4
CSR_BT_OPC_PUT_REQ	See section 4.5
CSR_BT_OPC_PUT_CFM	See section 4.5
CSR_BT_OPC_PUT_OBJECT_IND	See section 4.5
CSR_BT_OPC_PUT_OBJECT_RES	See section 4.5
CSR_BT_OPC_GET_HEADER_REQ	See section 4.6
CSR_BT_OPC_GET_HEADER_CFM	See section 4.6
CSR_BT_OPC_GET_OBJECT_REQ	See section 4.6
CSR_BT_OPC_GET_OBJECT_CFM	See section 4.6
CSR_BT_OPC_CANCEL_CONNECT_REQ	See section 4.7
CSR_BT_OPC_SECURITY_OUT_REQ	See section 4.8
CSR_BT_OPC_SECURITY_OUT_CFM	See section 4.8

Table 1: List of all primitives

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.2 CSR_BT_OPC_CONNECT

Primitives \ Parameters	type	appHandle	maxPacketSize	deviceAddr	resultCode	resultSupplier	maxPeerPacketSize	supportedFeatures	length	count	btConnId	windowSize	smEnable
CSR_BT_OPC_CONNECT_REQ	✓	✓	✓	✓					✓	✓		✓	✓
CSR_BT_OPC_CONNECT_CFM	✓				✓	✓	✓	✓			✓		

Table 2: CSR_BT_OPC_CONNECT Primitives

Description

To start a session for exchange of object against the server, the application sends a CSR_BT_OPC_CONNECT_REQ, with the max packet size allowed to send from the server. The server responds with a CSR_BT_OPC_CONNECT_CFM. In case the result in the confirmation is CSR_BT_OBEX_SUCCESS_RESULT_CODE the remaining parameters are valid, otherwise these parameters are invalid.

The connect messages between the OBEX 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. The timeout value can be disabled, or changed by changing CSR_BT_OBEX_CONNECT_TIMEOUT, which is defined in csr_bt_user_config.default.h. Note if the value of CSR_BT_OBEX_CONNECT_TIMEOUT is change, it will influence all OBEX profiles.

The function:

CsrBtOpcConnectReqSend (CsrSchedQid theAppHandle, CsrUint16 theMaxPacketSize ,
CsrBtDeviceAddr theDestination, CsrUint32 length, CsrUint32 count,
CsrUint16 windowSize, CsrBool srmEnable)

defined in csr_bt_opc_lib.h, builds and sends the CSR_BT_OPC_CONNECT_REQ primitive to the OPC profile.

Parameters

type	Signal identity, CSR_BT_OPC_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 server device.
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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

	values can be found in csr_bt_result.h
maxPeerPacketSize	The maximum OBEX packet size being allowed to send to the server application.
supportedFeatures	The formats supported by the server. This applies both to incoming and outgoing objects. The following values are possible (defined in csr_bt_obex.h): • VCARD_2_1_SUPPORT • VCARD_3_0_SUPPORT • VCAL_1_0_SUPPORT • ICAL_2_0_SUPPORT • VNOTE_SUPPORT • VMESSAGE_SUPPORT • OTHER_TYPE_SUPPORT (other that the above specified) Multiple formats may be combined. In this case the values are binary OR'ed.
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 OPC allows that PUT and GET commands, multiple OBEX request packets (PUT) or OBEX response packet (GET), can be sent 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].

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.3 CSR_BT_OPC_ABORT

Primitives \ Parameters	type
CSR_BT_OPC_ABORT_REQ	✓
CSR_BT_OPC_ABORT_CFM	✓

Table 3: CSR_BT_OPC_ABORT Primitives

Description

The CSR_BT_OPC_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_OPC_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_OPC_ABORT_REQ/CFM.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.4 CSR_BT_OPC_DISCONNECT

Parameters / Primitives	type	normalDisconnect	reasonCode	reasonSupplier
CSR_BT_OPC_DISCONNECT_REQ	✓	✓		
CSR_BT_OPC_DISCONNECT_IND	✓		✓	✓

Table 4: CSR_BT_OPC_DISCONNECT Primitives

Description

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

The disconnect messages between the OBEX 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 disabled, or changed by changing CSR_BT_OBEX_DISCONNECT_TIMEOUT, which is defined 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_OPC_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.
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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.5 CSR_BT_OPC_PUT & CSR_BT_OPC_PUT_OBJECT

Parameters / Primitives	type	finalFlag	responseCode	lengthOfObject	*ucs2name	objectLength	*object	bodyTypeLength	*bodyType
CSR_BT_OPC_PUT_REQ	✓			✓	✓			✓	✓
CSR_BT_OPC_PUT_CFM	✓		✓						
CSR_BT_OPC_PUT_OBJECT_IND	✓					✓			
CSR_BT_OPC_PUT_OBJECT_RES	✓	✓				✓	✓		

Table 5: CSR_BT_OPC_PUT & CSR_BT_OPC_PUT_OBJECT Primitives

Description

To push an object to the Push server, the application must send a CSR_BT_OPC_PUT_REQ to the OPC. The OPC then sends a CSR_BT_OPC_PUT_OBJECT_IND to the application, which the application must respond with a CSR_BT_OPC_PUT_OBJECT_RES message. In case the object being pushed is large enough to require several OBEX packets the CSR_BT_OPC_PUT_OBJECT_IND/CSR_BT_OPC_PUT_OBJECT_RES message sequence is repeated.

When the Put object procedure is finish OPC sends a CSR_BT_OPC_PUT_CFM to the application. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that the object is pushed to the server with success. Any other response code indicates a failure in the Put object procedure.

Parameters

type	Signal identity, CSR_BT_OPC_PUT_REQ/CFM & CSR_BT_OPC_PUT_OBJECT_IND/RES.
finalFlag	The finalFlag must be set to TRUE if the object fits in one packet, or if it is the last packet of a multi packet operation.
responseCode	The valid result codes are defined in csr_bt_obex.h. The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that an OBEX object has been push with success, CSR_BT_OBEX_CONTINUE_RESPONSE_CODE indicates that the object is large enough to require several OBEX packets, and any other response code indicates a failure in object push attempt. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
lengthOfObject	The total length of the object being push
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
objectLength	The length of the object, or the length of a part of it. In the CSR_BT_OPC_PUT_OBJECT_IND the value indicates the maximum size that the object (or part of it) can be in the CSR_BT_OPC_PUT_OBJECT_RES message.
*object	The object or a part of it (i.e. in case of a multi packet operation).

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

bodyTypeLength	Length of the bodyType parameter, including the null-char termination.
*bodyType	A null terminated 8 bit ASCII text string describing the MIME type of the object, such as vCard, vCal, etc. The usual supported types are: • “text/x-vcard” • “text/x-vcalendar” • “text/x-vnote” • “text/x-vmessage” More types can be found on the following URL: http://www.iana.org/assignments/media-types/index.html

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.6 CSR_BT_OPC_GET_HEADER & CSR_BT_OPC_GET_OBJECT

Parameters / Primitives	type	responseCode	rtotalObjectSize	bodyType	objectBodyOffset	objectBodyLength	ucs2nameOffset	payloadLength	*payload	srmpOn
CSR_BT_OPC_GET_HEADER_REQ	✓			✓						✓
CSR_BT_OPC_GET_HEADER_CFM	✓	✓	✓		✓	✓	✓	✓	✓	
CSR_BT_OPC_GET_OBJECT_REQ	✓									✓
CSR_BT_OPC_GET_OBJECT_CFM	✓	✓			✓	✓		✓	✓	

Table 6: CSR_BT_OPC_GET_HEADER & CSR_BT_OPC_GET_OBJECT Primitives

Description

To pull an object from the object Push server, the application must send a CSR_BT_OPC_GET_HEADER_REQ to the OPC. The OPC then sends a CSR_BT_OPC_GET_HEADER_CFM to the application. If the responseCode in the CSR_BT_OPC_GET_HEADER_CFM message is CSR_BT_OBEX_CONTINUE_RESPONSE_CODE the server will respond with multiple fragments. In this case the application must send a CSR_BT_OPC_GET_OBJECT_REG message to OPC in order to get the next fragment. OPC will send the result of this operation in the CSR_BT_OPC_GET_OBJECT_CFM message. Please notice that the CSR_BT_OPC_GET_OBJECT_REG/CSR_BT_OPC_GET_OBJECT_CFM sequence must be repeated as long as the result code in the CSR_BT_OPC_GET_OBJECT_CFM signal is CSR_BT_OBEX_CONTINUE_RESPONSE_CODE.

Parameters

type — Signal identity, CSR_BT_OPC_GET_HEADER_REQ/CFM & CSR_BT_OPC_GET_OBJECT_REQ/CFM.

responseCode — The valid result codes are defined in csr_bt_obex.h.
The CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates that an OBEX object has been pulled with success, CSR_BT_OBEX_CONTINUE_RESPONSE_CODE indicates that the server will respond with multiple fragments, and any other response code indicates a failure in pulling an object.

The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.

totalObjectSize — The total length of the object to receive

bodyType — The following body types are valid: (defined in csr_bt_obex.h)

- VCARD_TYPE
- VCAL_TYPE (can also be an iCal object)
- VNOTE_TYPE
- VMESSAGE_TYPE

objectBodyOffset — The payload-relative offset of the object or part of it (i.e., in case of a multi packet operation).

objectBodyLength — The length of the object, or the length of a part of it.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

ucs2nameOffset	Offset of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
payloadLength	Number of bytes in the payload parameter.
*payload	Pointer to the OBEX data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabledd, see section 4.2, the OPP Client can instruct the OPP 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.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.7 CSR_BT_OPC_CANCEL_CONNECT

Primitives \ Parameters	type
CSR_BT_OPC_CANCEL_CONNECT_REQ	✓

Table 7: CSR_BT_OPC_CANCEL_CONNECT Primitives

Description

The CSR_BT_OPC_CANCEL_CONNECT_REQ can be used the cancel an ongoing connect procedure. If the OPC succeed to cancel the ongoing connection attempt the application will receive a CSR_BT_OPC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

Parameters

type Signal identity, CSR_BT_OPC_CANCEL_CONNECT_REQ

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.8 CSR_BT_OPC_SECURITY_OUT

Primitives \ Parameters	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_OPC_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_OPC_SECURITY_OUT_CFM	✓			✓	✓

Table 8: CSR_BT_OPC_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 both are 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_OPC_SECURITY_OUT_REQ/CFM.

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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

	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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

5 Document References

Document	Reference
OBJECT PUSH PROFILE Revision V12r00 26 August 2010	[OPP]
IrDA Object Exchange Protocol - IrOBEX Version 1.2 or Version 1.5	[OBEX]
GENERIC OBJECT EXCHANGE PROFILE Revision V20r00 26 August 2010	[GOEP2.0]
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]

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
OPS	OBEX Push Server
OPC	OBEX Push Client
SIG	Special Interest Group
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
AMPM	Alternate MAC and PHY Manager
SRM	Single Response Mode
SRMP	Single Response Mode Parameters
GOEP	Generic Object Exchange Profile

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.