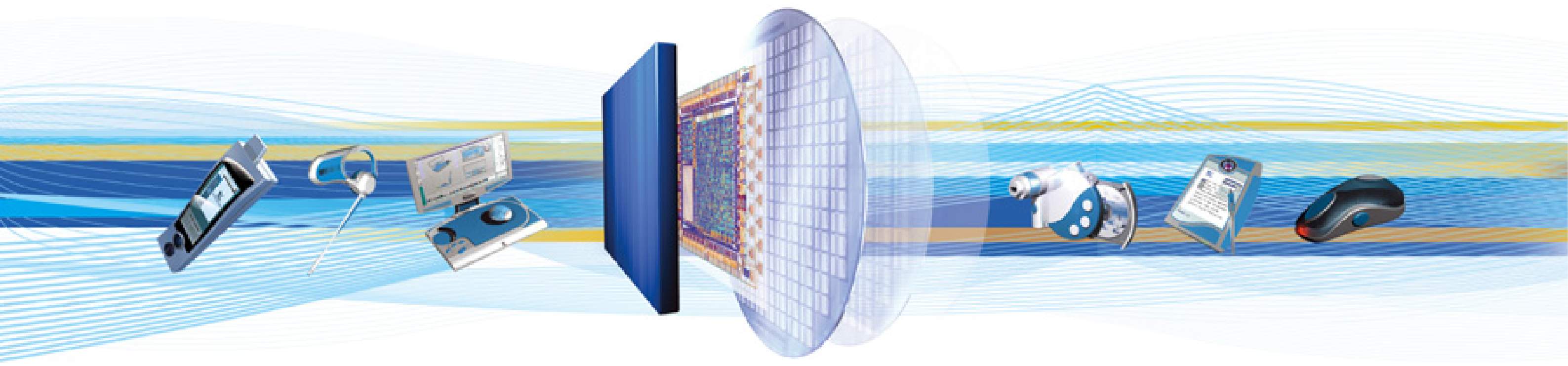
CSR Synergy ®

CSR Synergy Bluetooth 18.2.0

Obex Push Server

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

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the OBEX Push Server (OPS). The OPS conforms to the server 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. OPS and the application
- The OPS shall only handle one request at the time
- Bonding (pairing) is NOT handled by the OPS

2 Description

2.1 Introduction

The OBEX Push Server (OPS) must be activated by the application. When it is activated it is able to provide the application with incoming objects and provide the application the ability to send objects.

The OPS provides Service Discovery handling. It sets the Service Discovery Record according to the activation parameters.

2.2 Reference Model

The OPS interfaces to the Connection Manager (CM).

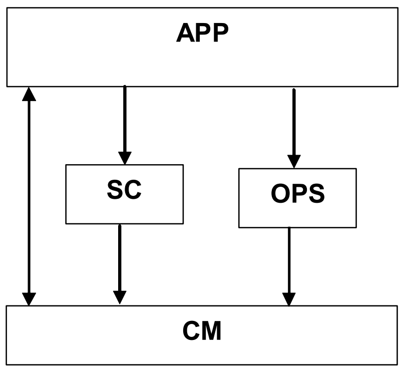


Figure 1: Reference model

2.3 Sequence Overview

The OPS starts up being in IDLE state. When the application activates OPS, the server enters ACTIVE state and is ready to handle incoming requests. The server remains in this state until deactivated by the application. When deactivated it re-enters IDLE state.

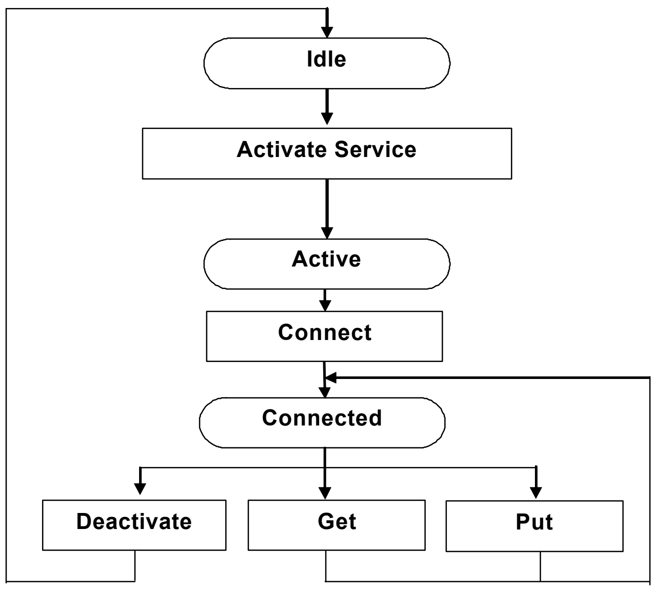


Figure 2: OPS state diagram

3 Interface Description

3.1 OPS API

3.1.1 Activation

Sending a CSR_BT_OPS_ACTIVATE_REQ to the OPS activates the OPS. The OPS then registers a Service Record, which contains the supported formats list, in the Service Discovery Server. The OPS is now ready to handle incoming requests.

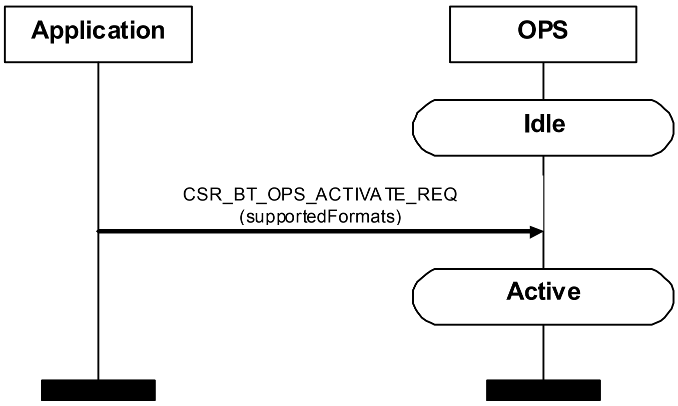


Figure 3: OPS activation

Please note that whether or not the Bluetooth device will be discoverable, i.e. can be found by other Bluetooth devices, it must be controlled by the application. For more information, please refer to [CM]. After initialization of CSR Synergy Bluetooth the Bluetooth® device is set up to be discoverable.

3.1.2 Incoming Objects

When objects (vCards, vNotes, etc.) are received by the OPS, it passes them on to the application in a CSR_BT_OPS_PUT_IND message. The application responds with a CSR_BT_OPS_PUT_RES, which contains the result of the "put". If the client side sends the body part fragmented the OPS sends additional indications (CSR_BT_OPS_PUT_NEXT_INDs) until the finalFlag parameter is set. This indicates end of body to the application.

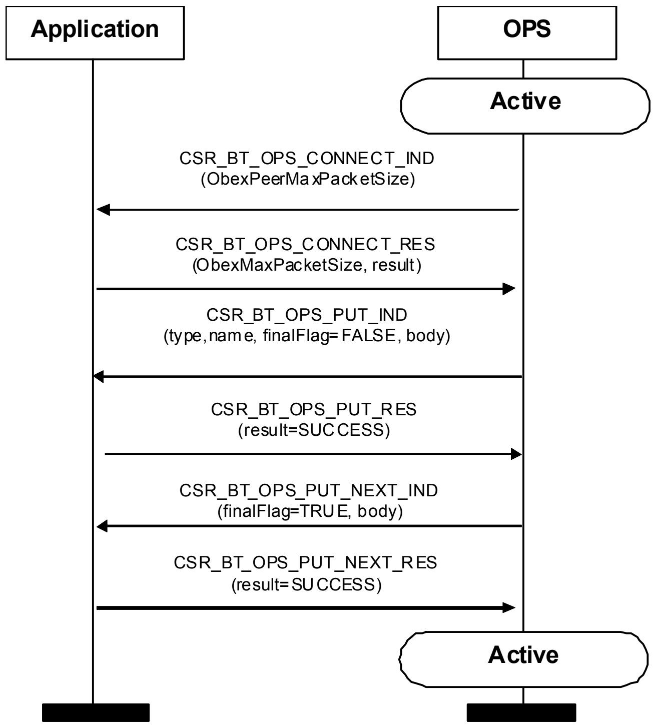


Figure 4: Incoming message handling

3.1.3 Outgoing Objects

When the OPS receives a request to send an object to the client side, it sends a CSR_BT_OPS_GET_IND message to the application with the bodyType parameter set to the requested type of object. The application responds with a CSR_BT_OPS_GET_RES with the appropriate result code. If the application wants to fragment the "body" information due to memory considerations it can set the finalFlag to FALSE and will hence receive CSR_BT_OPS_GET_NEXT_INDs until the finalFlag is set to TRUE in the following CSR_BT_OPS_GET_NEXT_RES.

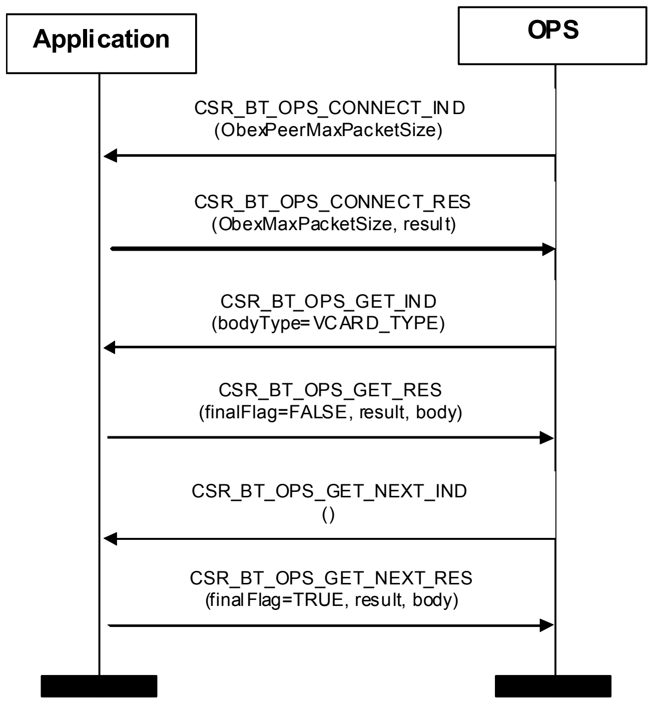


Figure 5: Outgoing message handling

3.1.4 Deactivation

Sending a CSR_BT_OPS_DEACTIVATE_REQ to the OPS deactivates the OPS. This procedure can take some time depending on the current OPS activity. When deactivated, the OPS confirms the deactivation with a CSR_BT_OPS_DEACTIVATE_CFM message.

Any transaction in progress will be terminated immediately when this message is received by the OPS.

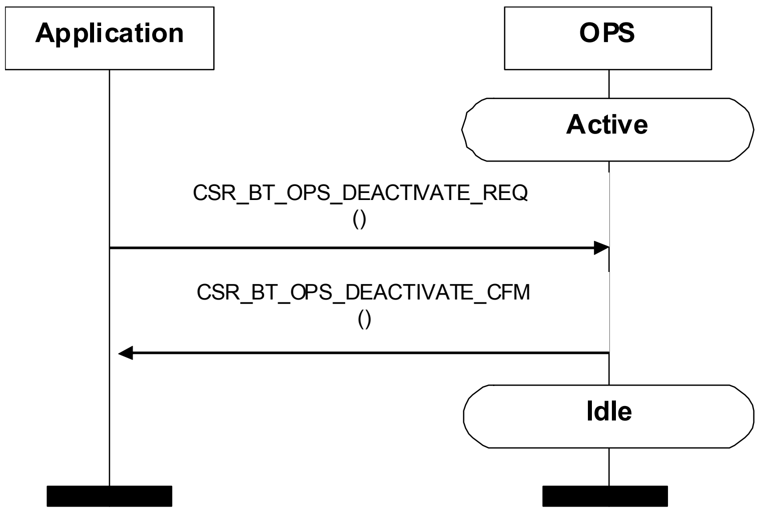


Figure 6: OPS deactivation

3.2 Payload Encapsulated Data

3.2.1 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:

```
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.

3.2.2 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 in an API signal.

4 OBEX Push Server Primitives

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

4.1 List of All Primitives

Primitives	Reference
CSR_BT_OPS_ACTIVATE_REQ	See section 4.2
CSR_BT_OPS_DEACTIVATE_REQ	See section 4.3
CSR_BT_OPS_DEACTIVATE_CFM	See section 4.3
CSR_BT_OPS_PUT_IND	See section 4.4
CSR_BT_OPS_PUT_RES	See section 4.4
CSR_BT_OPS_PUT_NEXT_IND	See section 4.4
CSR_BT_OPS_PUT_NEXT_RES	See section 4.4
CSR_BT_OPS_GET_IND	See section 4.5
CSR_BT_OPS_GET_RES	See section 4.5
CSR_BT_OPS_GET_NEXT_IND	See section 4.5
CSR_BT_OPS_GET_NEXT_RES	See section 4.5
CSR_BT_OPS_ABORT_IND	See section 4.6
CSR_BT_OPS_CONNECT_IND	See section 4.7
CSR_BT_OPS_CONNECT_RES	See section 4.7
CSR_BT_OPS_DISCONNECT_IND	See section 4.8
CSR_BT_OPS_SECURITY_IN_REQ	See section 4.9
CSR_BT_OPS_SECURITY_IN_CFM	See section 4.9

Table 1: List of all primitives

4.2 CSR_BT_OPS_ACTIVATE

Parameters / Primitives	type	appHandle	supportedFormats	obexMaxPacketSize	windowSize	srmEnable
CSR_BT_OPS_ACTIVATE_REQ	✓	✓	✓	✓	✓	✓

Table 2: CSR_BT_OPS_ACTIVATE Primitives

Description

This signal is used for activating the OPS and making it connectable. The process includes:

1. Registering the OBEX Push service in the service discovery database
2. Enabling page scan

The OPS will remain activated until a CSR_BT_OPS_DEACTIVATE_REQ is received.

Parameters

type — Signal identity, CSR_BT_OPS_ACTIVATE_REQ.

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.

supportedFormats — The formats being supported in this application. This applies both to incoming and outgoing objects.

The following values are possible (defined in csr_bt_obex.h):

- VCARD_2_1_SUPPORT (corresponds to “vCard 2.1” (0x01) in the OPP specification)
- VCARD_3_0_SUPPORT (corresponds to “vCard 3.0” (0x02) in the OPP specification)
- VCAL_1_0_SUPPORT (corresponds to “vCal 1.0” (0x03) in the OPP specification)
- ICAL_2_0_SUPPORT (corresponds to “iCal 2.0” (0x04) in the OPP specification)
- VNOTE_SUPPORT (corresponds to “vNote” (0x05) in the OPP specification)
- VMESSAGE_SUPPORT (corresponds to “vMessage” (0x06) in the OPP specification)
- OTHER_TYPE_SUPPORT (corresponds to “any type of object” (0xFF) in the OPP specification)
- ANY_TYPE_SUPPORT (corresponds to OR’ing all of the above formats)

Multiple formats can be combined by binary OR’ing the values. If ANY_TYPE_SUPPORT is set, support of all format types are claimed in the Service Record.

NOTICE: It is not necessary to have a default object for each supported format. The application shall respond with a “NOT FOUND” response code in this case.

obexMaxPacketSize — To control the maximum allowed OBEX packet size the application can receive. There is a define CSR_BT_MAX_OBEX_SIGNAL_LENGTH (in csr_bt_usr_config.h) to be used for this value, the max allowed value is 64K bytes – 1.

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 OPS 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].

4.3 CSR_BT_OPS_DEACTIVATE

Primitives \ Parameters	type
CSR_BT_OPS_DEACTIVATE_REQ	✓
CSR_BT_OPS_DEACTIVATE_CFM	✓

Table 3: CSR_BT_OPS_DEACTIVATE Primitives

Description

This signal deactivates the OPS. The service cannot be re-activated until after the application has received a CSR_BT_OPS_DEACTIVATE_CFM.

The service will no longer be connectable.

The signal will stop any ongoing transaction.

Parameters

type	Signal identity, CSR_BT_OPS_DEACTIVATE_REQ/CFM.

4.4 CSR_BT_OPS_PUT & CSR_BT_OPS_PUT_NEXT

Primitives \ Parameters	type	responseCode	finalFlag	totalObjectSize	bodyTypeOffset	bodyTypeLength	ucs2nameOffset	bodyOffset	bodyLength	payloadLength	*payload	srmpOn
CSR_BT_OPS_PUT_IND	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
CSR_BT_OPS_PUT_RES	✓	✓										✓
CSR_BT_OPS_PUT_NEXT_IND	✓		✓		✓	✓	✓	✓	✓	✓	✓	
CSR_BT_OPS_PUT_NEXT_RES	✓	✓										✓

Table 4: CSR_BT_OPS_PUT & CSR_BT_OPS_PUT_NEXT Primitives

Description

The OPS passes incoming objects on to the application with the CSR_BT_OPS_PUT_IND signal. The application can then store the objects in the "Inbox". The result of the store operation is given to the OPS with the CSR_BT_OPS_PUT_RES signal. The result can contain error codes corresponding to the reason for failure.

Parameters

type	Signal identity, CSR_BT_OPS_PUT_IND/RES and CSR_BT_OPS_PUT_NEXT_IND/RES. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_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.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
totalObjectSize	The total length of the object to send.
bodyTypeOffset	Offset into the payload where 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 NOTICE: The body type is received from the peer side. It is allowed NOT to send the body type, hence if no body type is received, the body type is set to NULL. In this case, the type must be determined from the name (extension) or body content.

bodyTypeLength	Length of the body information.
ucs2nameOffset	Payload-relative offset for a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyOffset	The offset within payload where the body data starts.
bodyLength	The number of bytes in the body (contained within the payload).
payloadLength	Length of payload.
*payload	Pointer to the payload, which contains all data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabled, see section 4.2, the OPP server can instruct the OPP Client to wait for the next response packet during a PUT Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first PUT response and may be used in consecutive PUT response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a PUT response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5 CSR_BT_OPS_GET & CSR_BT_OPS_GET_NEXT

Primitives \ Parameters	type	finalFlag	responseCode	bodyType	totalObjectSize	*ucs2name	*body	bodyLength	srmpOn
CSR_BT_OPS_GET_IND	✓			✓					
CSR_BT_OPS_GET_RES	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_BT_OPS_GET_NEXT_IND	✓			✓					
CSR_BT_OPS_GET_NEXT_RES	✓	✓	✓				✓	✓	✓

Table 5: CSR_BT_OPS_GET & CSR_BT_OPS_GET_NEXT Primitives

Description

The OPS passes outgoing requests on to the application with the CSR_BT_OPS_GET_IND signal. The application can then send the object to the client side with a CSR_BT_OPS_GET_RES in case of success or use an appropriate result code to respond to the request. The body can be fragmented by setting the finalFlag to FALSE until the last fragment, which contains finalFlag TRUE. If fragmentation is used, the first indication/response will be CSR_BT_OPS_GET_IND/CSR_BT_OPS_GET_RES. The following messages will be CSR_BT_OPS_GET_NEXT_IND/ CSR_BT_OPS_GET_NEXT_RES indications/responses.

Please Note that in a CSR_BT_OPS_GET_RES the sum of the length of the ucs2name + bodyLength must not exceed the OBEX packet size which the server is allowed to send to the client. See section 4.7.

Parameters

type	Signal identity, CSR_BT_OPS_GET_IND/RES and CSR_BT_OPS_GET_NEXT_IND/RES.
finalFlag	Indicate that the body (object) fits in one response packet or the last part of multiple responses.
responseCode	The valid response codes are defined in csr_bt_obex.h. For success in the request the code 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.
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
totalObjectSize	The total length of the object to send.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null

	terminated UTF8 text string into a null terminated UCS2 text string, and the function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string,
bodyLength	The length of the body (object).
*body	The object itself. "body" is a CsrUint8 pointer to the object.
srmpOn	If Single Response Mode is enabled, see section 4.2, the OPP server can instruct the OPP 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 the 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.

4.6 CSR_BT_OPS_ABORT

Parameters / Primitives	type	descriptionOffset	descriptionLength	payloadLength	*payload
CSR_BT_OPS_ABORT_IND	✓	✓	✓	✓	✓

Table 6: CSR_BT_OPS_ABORT Primitives

Description

In case the client side wants to abort a command (put or get) it can send an abort message to the server. In this case the OPS will send a CSR_BT_OPS_ABORT_IND message to the application so it can abort the current transactions. The application can then act accordingly, for example, close any open files and get ready for new requests from the client (or another client).

Parameters

type	Signal identity, CSR_BT_OPS_ABORT_IND.
descriptionOffset	Offset into the payload for a null terminated 16 bit Unicode text string (UCS2) containing the reason for the abort. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string
descriptionLength	Length in bytes of the description.
payloadLength	Length of the payload.
*payload	OBEX payload data chunk, on which the offset is based.

4.7 CSR_BT_OPS_CONNECT

Primitives \ Parameters	type	obexPeerMaxPacketSize	deviceAddr	responseCode	length	count	cid	btConnId
CSR_BT_OPS_CONNECT_IND	✓	✓	✓		✓	✓	✓	✓
CSR_BT_OPS_CONNECT_RES	✓			✓				

Table 7: CSR_BT_OPS_CONNECT Primitives

Description

This signal is indicating that a Push client is starting a session. The application can then accept or deny with the result.

Parameters

type	Signal identity, CSR_BT_OPS_CONNECT_IND/RES.
obexPeerMaxPacketSize	The maximum obex packet size allowed sending to the client application.
deviceAddr	The Bluetooth address which is connected to the device.
responseCode	The valid response codes are defined (in csr_bt_obex_h). For accepting a connection the code 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.
length	The length parameter contains the length in bytes of the bodies of all the objects that the sender plans to send. Note this length cannot be guarantee correct, so while the value may be useful for status indicators and resource reservations, OPS application should not die if the length is not correct. If 0 this parameter were not included in the received OBEX Connect Request packet.
count	Count is use to indicate the number of objects that will be sent by the sender during this connection. If 0 this parameter were not included in the received OBEX connect Request packet.
btConnId	Identifier which shall be used when using AMPM, for more information please refer to [AMPM].

4.8 CSR_BT_OPS_DISCONNECT

Parameters / Primitives	type	deviceAddr	reasonCode	reasonSupplier
CSR_BT_OPS_DISCONNECT_IND	✓	✓	✓	✓

Table 8: CSR_BT_OPS_DISCONNECT Primitives

Description

This signal is indicating that the OBEX Push session is finished, and is ready for a new one.

Parameters

type	Signal identity, CSR_BT_OPS_DISCONNECT_IND.
deviceAddr	The Bluetooth address which is connected to the device
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. All values which are currently not specified are the respective prim.h files or csr_bt_obex.h is 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

4.9 CSR_BT_OPS_SECURITY_IN

Primitives \ Parameters	type	appHandle	secLevel	resultCode	resultsSupplier
CSR_BT_OPS_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_OPS_SECURITY_IN_CFM	✓			✓	✓

Table 9: CSR_BT_OPS_SECURITY_IN 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_IN_REQ* signal sets up the security level for new incoming connections. Already established or pending connections are not altered.

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_OPS_SECURITY_IN_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_ CSR_BT_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

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. CM – Connection Manager API Description, doc. no. api-0101-cm	[CM]
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®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
OPC	OBEX Push Client
OPS	OBEX Push Server
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

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