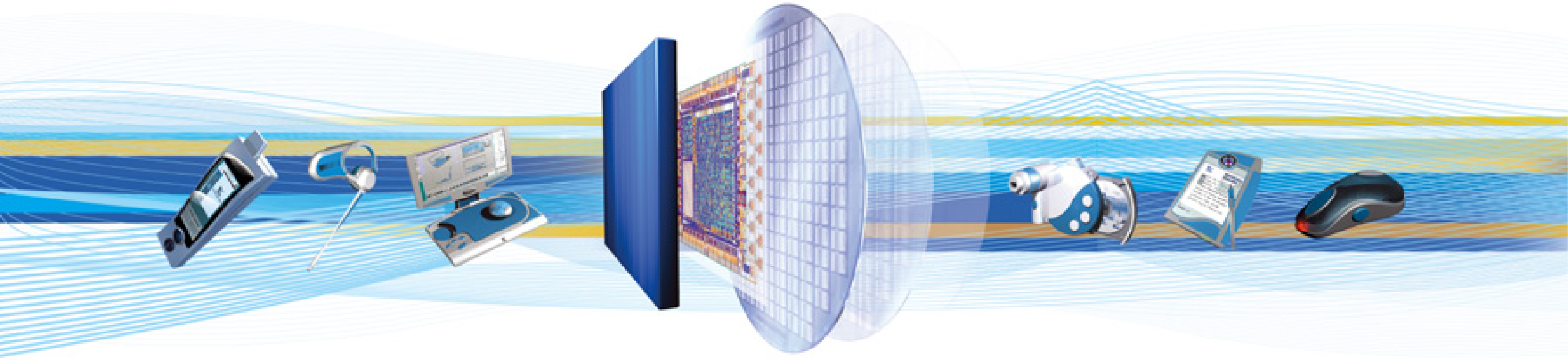
CSR Synergy®

CSR Synergy Bluetooth 18.2.0

DI – Device Identification

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 functionality and message interface specified by the Device identification profile and is implemented as part of the Service Discovery (SD) module in CSR Synergy Bluetooth.

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- The DI shall only handle one registration/unregistration request per application at the time
- The DI shall only handle one read service record request per application at the time

2 Description

2.1 Introduction

The DI functionality provides a set of functions, especially designed to register a Device Identification service record and reading of DI service records from remote Bluetooth® devices.

The services offered by the DI functionality to an application are:

- Search for DI service records on remote devices
- Registration of DI service records on the local device

2.2 Reference Model

To use the DI functionality, the application must communicate through the SD, which will take care of all necessary communication down towards the CM afterwards.

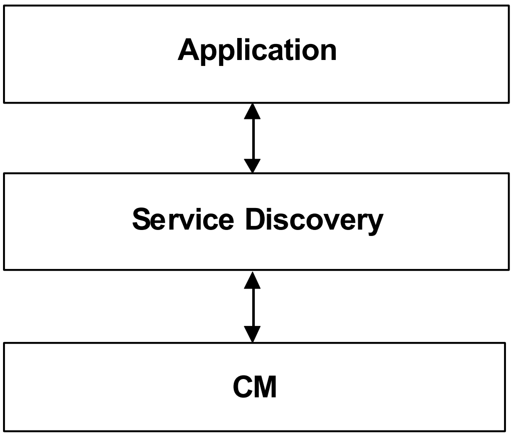


Figure 1: Reference model

3 Interface Description

3.1 The Device Identification Service Record

A DI service record contains several parameters, described in the ‘Device Identification Profile Specification’. Among the parameters are:

- Vendor ID
- Product ID
- Product version
- Primary Record information

Special care must be taken with the usage of the ‘PrimaryRecord’ parameter in the application, again please see the ‘Device Identification Profile Specification’ for extended explanations as to how this parameter should be treated in the device.

3.2 Registration of a DI Service Record

The registration of a DI service e record is done through the SD by using the CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ primitive. When a registration request has been sent from the application, the SD will return a result confirmation, using the CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM primitive.

Please note that an application can only register one DI service record at the time, but several DI Service Records can be registered one after the other.

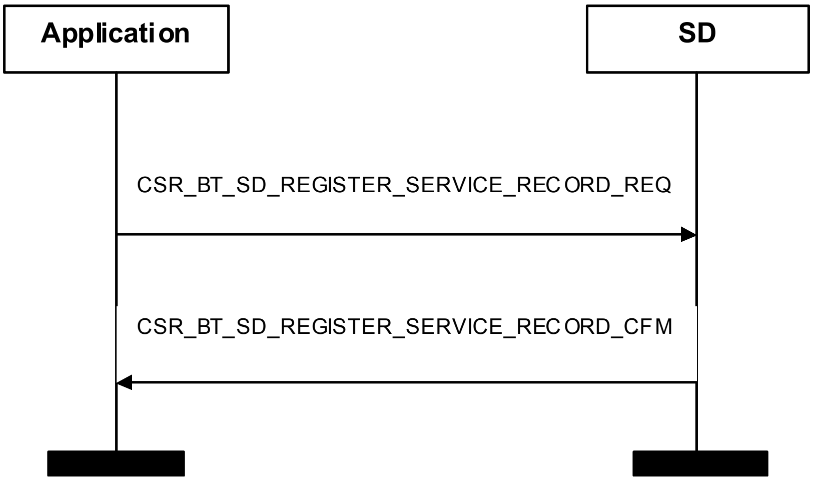

Application
SD
CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ
CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM

Figure 2: DI service record registration sequence

In the confirmation (CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM), the result of the registration can be extracted from the *result* field, as well as the *serviceHandle* (used for unregistering the DI service record), explained in the following.

3.2.1 DI Service Record Format

The *data* field in the CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ primitive is a byte stream in a proprietary format. An short overview of this is shown in figure Figure 3.

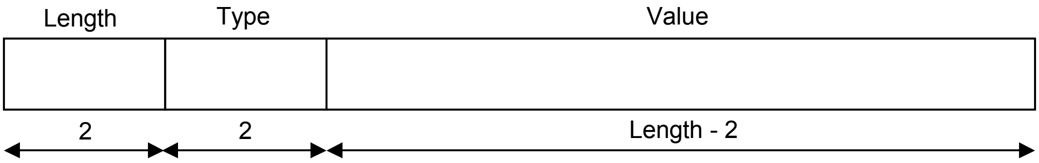


Figure 3: An overview of the data format in the data sequence. Values are measured in bytes

Data types are allocated for compatibility with the EIR format, which means that types of 0x00FF and below are reserved for identifiers allocated for EIR by the Bluetooth SIG.

The two types identified by the DI parser are 0x0100 (CSR_BT_TAG_SDR_ENTRY) and 0x0101 (CSR_BT_TAG_SDR_ATTRIBUTE_ENTRY), defined in profiles.h.

Types in the interval 0x0102 to 0xFFFF are reserved for future CSR Synergy Bluetooth extensions. In order to provide extensibility, any parser of this format must silently ignore types that it does not know, and skip to the next field.

3.2.2 Encoding of the DI Service Record Format

It is not intended for the applications to encode the data field of the CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ themselves. Instead, CSR Synergy Bluetooth includes a set of helper functions, which can be used for encoding and sending the CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ primitive. These helper functions are placed in CSR_BT_SD_lib.h.

Register a DI service Record version 1.3:

```
CsrBool CsrBtSdRegisterDiServiceRecordV13(CsrSchedQid apphandle,
                                    CsrUint16  vendorId,
                                    CsrUint16  productId,
                                    CsrUint16  version,
                                    CsrBool    primaryRecord,
                                    CsrUint16  vendorIdSource,
                                    CsrUint8  *serviceDescription,
                                    CsrUint16  serviceDescriptionLen,
                                    CsrUint8  *clientExecutableUrl,
                                    CsrUint16  clientExecutableUrlLen,
                                    CsrUint8  *documentationUrl,
                                    CsrUint16  documentationUrlLen);
```

This function will place the vendorId, productId, version, primaryRecord, VendorIdSource as well as the serviceDescription, clientExecutableUrl and the documentationUrl strings into the data field structure, and send the signal to the SD.

The three pointer parameters will be dynamically allocated inside the register helper function, and must hence be freed by the caller afterwards. The only exception to this is when no data/string is present. In this case, the pointer parameters shall be set to NULL and the length to 0 (zero). The function returns a boolean value, indicating whether the creation and allocation of the data-field succeeded. The result of the registration itself will be returned to the application in the CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM primitive, sent from the SD

3.3 Unregistration of a DI Service Record

This command unregisters a DI Service Record. Be aware that this signal should be in compliance with the specification guidelines especially for the PrimaryRecord parameter.

The CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ primitive needs to contain a valid serviceHandle previously received in a CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM. If an application sends this signal when it has no active DI service records (no valid serviceHandle), it is considered an error, and the confirmation primitive sent back to the application will not contain a CSR_BT_SUCCESS-result.

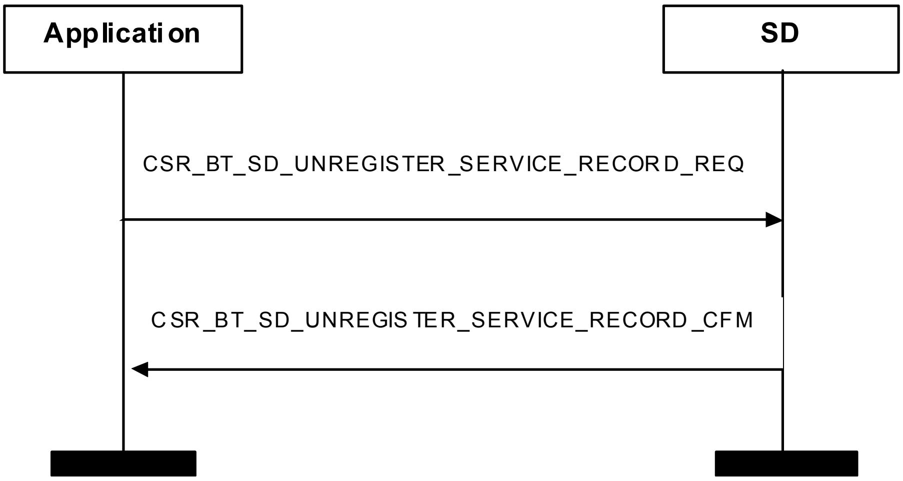

Application
SD
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_CFM

Figure 4: Unregister DI service record sequence

The helper function for sending the unregistration request of a DI service record primitive is:

```
CsrBtSdUnregisterServiceRecordReqSend(CsrSchedQid appHandle,
                                      CsrUint32  flags,
                                      CsrUint32  serviceHandle);
```

Where the *flags* field is reserved for future usage and should be set to 0 (zero) to avoid backwards compatibility issues.

3.4 Read Device Information (DI) about a Device

This command allows an application to read all the information a remote Bluetooth® device has stored in its Device Identification service records. The CSR_BT_SD_READ_SERVICE_RECORD_REQ primitive is sent to the SD, using a helper function, and the results (the number of DI service records on the remote device), are sent up to the application, one by one, with the CSR_BT_SD_READ_SERVICE_RECORD_IND primitive. When there are no more DI service records available in the remote device, the CSR_BT_SD_READ_SERVICE_RECORD_CFM primitive is sent to the application to indicate the end of the sequence.

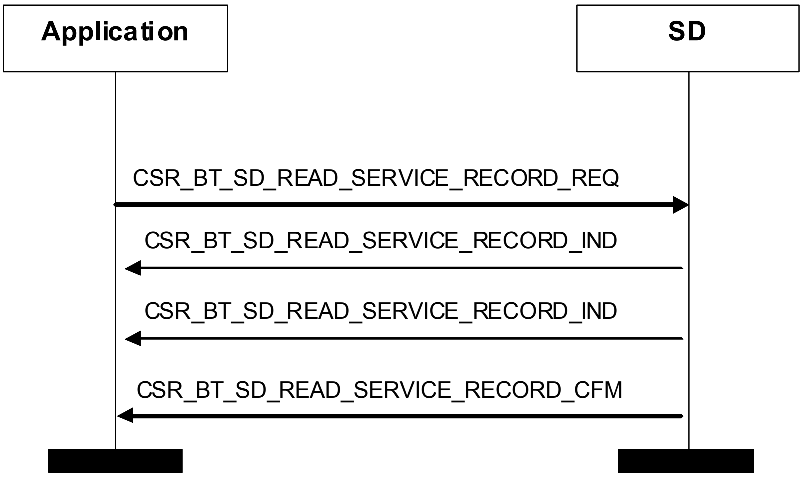

Application
SD
CSR_BT_SD_READ_SERVICE_RECORD_REQ
CSR_BT_SD_READ_SERVICE_RECORD_IND
CSR_BT_SD_READ_SERVICE_RECORD_IND
CSR_BT_SD_READ_SERVICE_RECORD_CFM

Figure 5: Read DI service record sequence

It is strongly recommended to use the helper functions available in CSR_BT_SD_lib.h for these operations.

Read DI service record:

```
CsrBool CsrBtSdReadDiServiceRecordV13(CsrSchedQid     apphandle,
                                      deviceAddr_t  deviceAddr);
```

The helper function will start the read DI service record sequence, and retrieve the DI service records as defined in the 'Device Identification Profile Specification' version 1.3.

3.4.1 Decoding of the DI service Record Format

When the application receives the DI service records, one at a time, in the CSR_BT_SD_READ_SERVICE_RECORD_IND primitives, the below helper function helps extract the parameters, and places them in the SdDiServiceRecordV13Struct structure, which is defined in CSR_BT_SD_lib.h.

```
void CsrBtSdExtractDiServiceRecordV13Data(CsrUint8 *data,
                                           CsrUint16 dataLen,
                                           CsrBtSdDiServiceRecordV13Struct *v13);
```

3.5 Cancel Read Device Information (DI)

This command allows the application to cancel the read service record sequence as described in section 3.4.

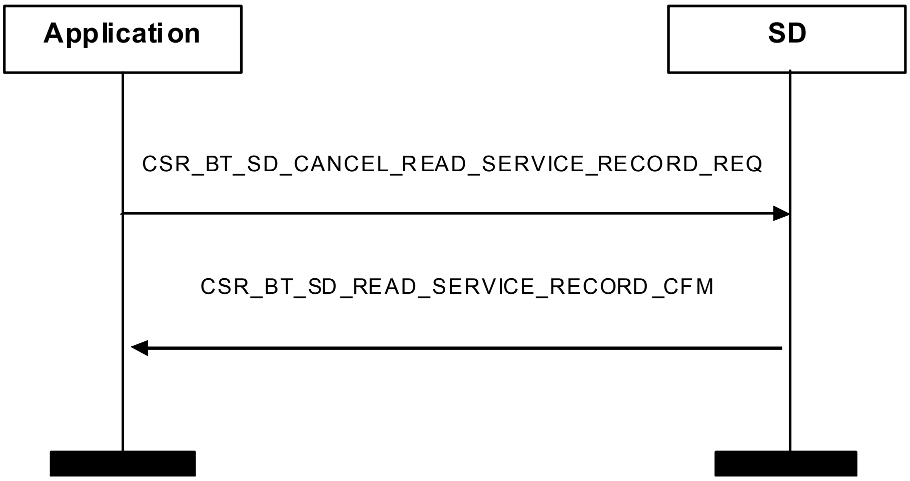

Application
SD
CSR_BT_SD_CANCEL_READ_SERVICE_RECORD_REQ
CSR_BT_SD_READ_SERVICE_RECORD_CFM

Figure 6: Cancel Read DI service record sequence

3.6 Typical Use Case of the DI functionality

In order to give an idea of how to use the DI functionality, a typical use case is depicted below.

The registration of a service record is typically done during initialisation. During normal operation the reading of remote service records is done, and when the application does a deinitialisation, the unregistering of the service record is done.

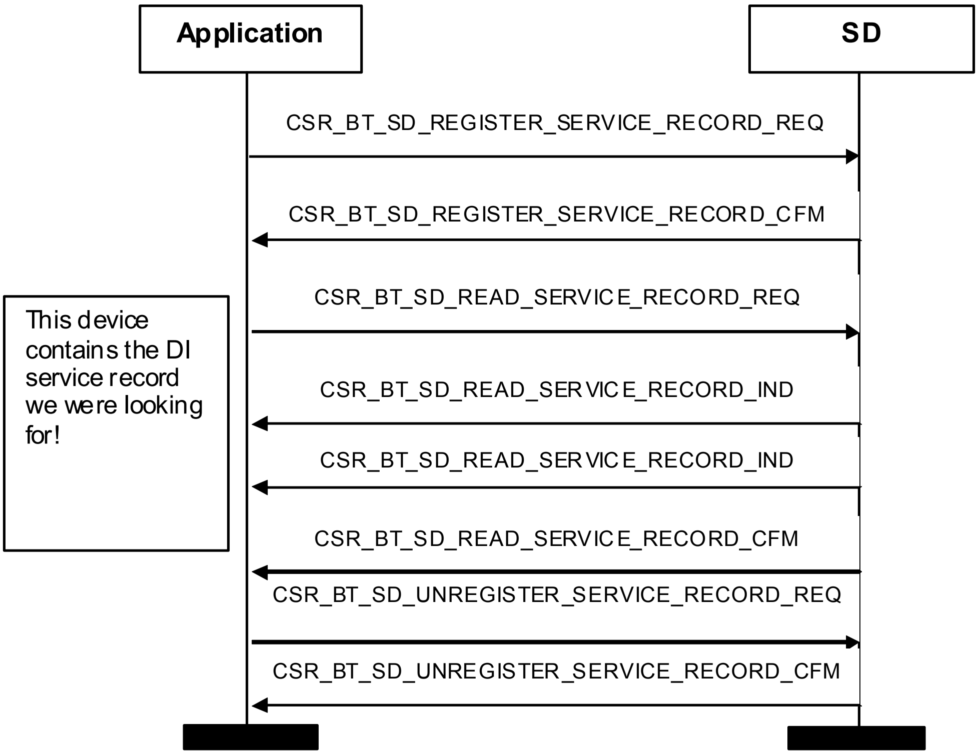

Application
SD
CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ
CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM
CSR_BT_SD_READ_SERVICE_RECORD_REQ
CSR_BT_SD_READ_SERVICE_RECORD_IND
CSR_BT_SD_READ_SERVICE_RECORD_IND
CSR_BT_SD_READ_SERVICE_RECORD_CFM
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_CFM
This device contains the DI service record we were looking for!

Figure 7: Typical use case of the DI functionality

The local DI service record is first registered, then the DI record from remote devices are read, and at the end, the local DI service record can be unregistered.

4 DI Functionality Primitives

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

For other primitives starting with the CSR_BT_SD_ prefix, please see the api-0103-sd documentation.

4.1 List of All Primitives

Primitives	Reference
CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ	See section 4.2
CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM	See section 4.2
CSR_BT_SD_READ_SERVICE_RECORD_REQ	See section 4.3
CSR_BT_SD_READ_SERVICE_RECORD_IND	See section 4.3
CSR_BT_SD_READ_SERVICE_RECORD_CFM	See section 4.3
CSR_BT_SD_CANCEL_READ_SERVICE_RECORD_REQ	See section 4.4
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ	See section 4.5
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_CFM	See section 4.5

Table 1: List of all primitives

4.2 CSR_BT_SD_REGISTER_SERVICE_RECORD

Primitives \ Parameters	type	phandle	flags	dataLen	*data	serviceHandle	resultCode	resultSupplier
CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ	✓	✓	✓	✓	✓			
CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM	✓					✓	✓	✓

Table 2: CSR_BT_SD_REGISTER_SERVICE_RECORD Primitives

Description

This signal is used for the registration of a DI service record.

Parameters

type Signal identity, CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ / _CFM.

phandle The identity of the calling process.

flags Reserved for future use. MUST be set to 0 (zero), to avoid backwards compatibility problems in the future

dataLen The length of the data

*data The data, which contains the DI service record which needs to be registered.

serviceHandle An identifier of the newly registered service record, which needs to be used if the service record needs to be unregistered later on

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. All values which are currently not specified in the respective prim.h files 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

The function:

```
    CsrBool CsrBtSdRegisterDiServiceRecordV13(CsrSchedQid apphandle,
                                        CsrUint16  vendorId,
                                        CsrUint16  productId,
                                        CsrUint16  version,
                                        CsrBool    primaryRecord,
                                        CsrUint16  vendorIdSource,
                                        CsrUint8  *serviceDescription,
                                        CsrUint16  serviceDescriptionLen,
                                        CsrUint8  *clientExecutableUrl,
                                        CsrUint16  clientExecutableUrlLen,
                                        CsrUint8  *documentationUrl,
                                        CsrUint16  documentationUrlLen);
```

defined in csr_bt_sd_lib.h, builds and sends the CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ primitive to SD. Please note that if this function returns false no CSR_BT_SD_REGISTER_SERVICE_RECORD_REQ is sent and hence no CSR_BT_SD_REGISTER_SERVICE_RECORD_CFM should be expected.

4.3 CSR_BT_SD_READ_SERVICE_RECORD

Parameters Primitives	type	phandle	deviceAddr	flags	dataLen	data	resultCode	resultSupplier
CSR_BT_SD_READ_SERVICE_RECORD_REQ	✓	✓	✓	✓	✓	✓		
CSR_BT_SD_READ_SERVICE_RECORD_IND	✓				✓	✓		
CSR_BT_SD_READ_SERVICE_RECORD_CFM	✓						✓	✓

Table 3: CSR_BT_SD_READ_SERVICE_RECORD Primitives

Description

These signals are used for retrieving DI service Records from a remote device.

Parameters

type — Signal identity, CSR_BT_SD_READ_SERVICE_RECORD_REQ / _IND / _CFM.

phandle — The identity of the calling process.

deviceAddr — The Bluetooth address of the remote device

flags — Reserved for future use. MUST be set to 0 (zero), to avoid backwards compatibility problems in the future

dataLen — The length of the data

data — The data, which contains the empty DI service record which needs to be read in the requirement signal and the filled out read DI service record in the indication signal. Data received in a CSR_BT_SD_READ_SERVICE_RECORD_IND must be freed by the application.

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. All values which are currently not specified in the respective prim.h files 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

The function:

```
    CsrBool CsrBtSdReadDiServiceRecordV13(CsrSchedQid     apphandle,
                                          deviceAddr_t  deviceAddr);
```

defined in csr_bt_sd_lib.h, builds and sends the CSR_BT_SD_READ_SERVICE_RECORD_REQ primitive to SD. Please note that if this function returns false no CSR_BT_SD_READ_SERVICE_RECORD_REQ is sent and hence no CSR_BT_SD_READ_SERVICE_RECORD_CFM should be expected.

The function:

```
    void CsrBtSdExtractDiServiceRecordV13Data(CsrUint8 *data,
                                                  CsrUint16 dataLen,
                                                  CsrBtsdDiServiceRecordV13Struct *v13);
```

defined in csr_bt_sd_lib.h, extracts the DI service record data (as defined in the version 1.3 of the Device Identification Profile Specification) and places the data in the `CsrBtsdDiServiceRecordV13Struct` structure, where it is also possible to verify if the data in the structure is valid or not (if something was retrieved or not from the remote device).

```
typedef struct
{
    CsrBool   specificationIdValid;
    CsrUint16 specificationIdValue;

    CsrBool   vendorIdValid;
    CsrUint16 vendorIdValue;

    CsrBool   productIdValid;
    CsrUint16 productIdValue;

    CsrBool   versionValid;
    CsrUint16 versionValue;

    CsrBool   primaryRecordValid;
    CsrBool   primaryRecordValue;

    CsrBool   vendorIdSourceValid;
    CsrUint16 vendorIdSourceValue;

    CsrBool    clientExecutableUrlValid;
    CsrUint8  *clientExecutableUrlValue;
    CsrUint16  clientExecutableUrlValueLen;

    CsrBool    serviceDescriptionValid;
    CsrUint8  *serviceDescriptionValue;
    CsrUint16  serviceDescriptionValueLen;

    CsrBool    documentationUrlValid;
    CsrUint8  *documentationUrlValue;
    CsrUint16  documentationUrlValueLen;
}CsrBtSdDiServiceRecordV13Struct;
```

4.4 CSR_BT_SD_CANCEL_READ_SERVICE_RECORD

Parameters / Primitives	type	phandle
CSR_BT_SD_CANCEL_READ_SERVICE_RECORD_REQ	✓	✓

Table 4: CSR_BT_SD_CANCEL_READ_SERVICE_RECORD Primitive

Description

This signal is used for cancelling the reading of the DI Service Records from a remote device.

Parameters

type	Signal identity, CSR_BT_SD_CANCEL_READ_SERVICE_RECORD_REQ.
phandle	The identity of the calling process.

The function:

```
    void CsrBtSdCancelReadServiceRecordReqSend(CsrSchedQid phandle);
```

defined in csr_bt_sd_lib.h, builds and sends the CSR_BT_SD_CANCEL_READ_SERVICE_RECORD_REQ primitive to SD.

4.5 CSR_BT_SD_UNREGISTER_SERVICE_RECORD

Parameters Primitives	type	phandle	flags	serviceHandle	resultCode	resultSupplier
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ	✓	✓	✓	✓		
CSR_BT_SD_UNREGISTER_SERVICE_RECORD_CFM	✓			✓	✓	✓

Table 5: CSR_BT_SD_UNREGISTER_SERVICE_RECORD Primitives

Description

This signal allows an application to unregister the service record, identified by the serviceHandle.

Parameters

type Signal identity, CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ / _CFM.

phandle The identity of the calling process. The response is returned to phandle.

flags Reserved for future use. MUST be set to 0 (zero), to avoid backwards compatibility problems in the future

serviceHandle An identifier of the registered service record, which the application requests to be unregistered.

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. All values which are currently not specified in the respective prim.h files 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

The function:

```
CsrBtSdUnregisterServiceRecordReqSend(CsrSchedQid appHandle,
                                      CsrUint32  flags,
                                      CsrUint32  serviceHandle);
```

defined in csr_bt_sd_lib.h, builds and sends the CSR_BT_SD_UNREGISTER_SERVICE_RECORD_REQ primitive to SD.

5 Document References

Document	Reference
Specification of the Bluetooth System Version 1.1, 1.2 and 2.0	[BT]
CSR Synergy Bluetooth, SD - Service Discovery API Description, document no. api-0103-sd	[SD]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc	[SC]

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
DI	Device Identification
SD	Service Discovery
SC	Security Controller
SIG	Special Interest Group
UniFi™	Group term for CSR’s range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

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.