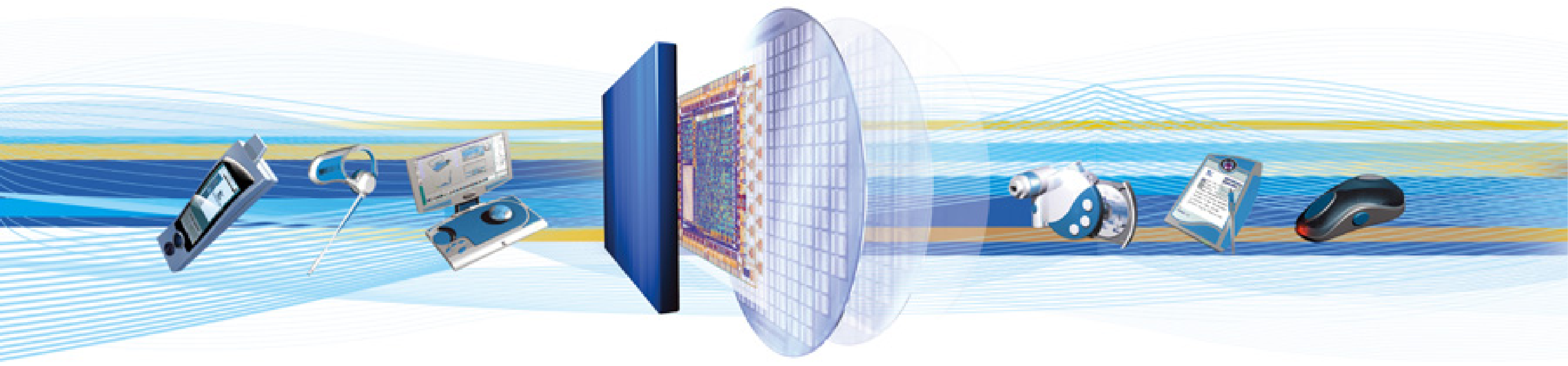
CSR Synergy®

CSR Synergy Framework 3.1.0

Data Store

API Description

August 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 API between an application task which need non volatile storage access and CSR DATA STORE. The API is called CSR DATA STORE.

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- Only one instance of CSR DATA STORE is active at any time
- All strings sent between the application and CSR DATA STORE are encoded as UTF8
- An application running on top of this API should always try to keep keys as short as possible
- If the port of CSR DATA STORE needs to block during its operations again the NVS, then it is running in a separate thread or scheduler instance so that it will not affect the performance of the other tasks running in the synergy scheduler

2 Description

This section will briefly describe the purpose of introducing the CSR DATA STORE API. After this section the reader should be familiar with the location of CSR DATA STORE API in the overall architecture and the reason for introducing the API.

2.1 Introduction

The CSR DATA STORE API provides asynchronous access to non volatile data storage needed by synergy tasks.

The API provides the following functionality:

- Creation, opening, closure and deletion of a data store.
- Reading, writing and removal of keys, with an associated record, in a data store.
- The interface is able to handle interaction with multiple tasks simultaneously.

2.2 Reference Model

CSR DATA STORE API and its location.

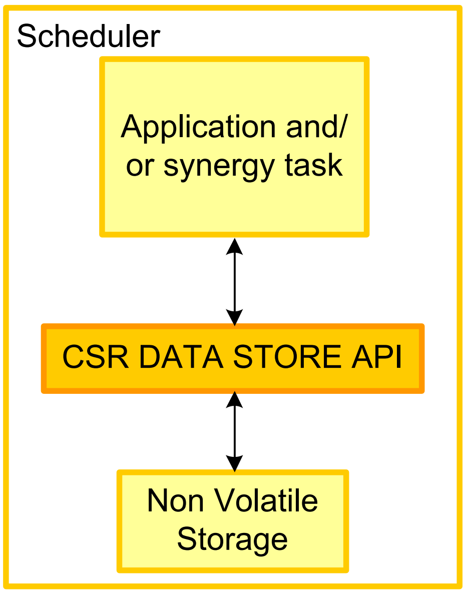


Figure 1: The CSR DATA STORE API shown relative to the platforms non volatile storage

3 Interface Description

The following sessions will describe typical usage scenarios of CSR DATA STORE through examples using MSCs.

3.1 Creation of a Data Store

Figure 2 illustrates how the Application can create a data store.

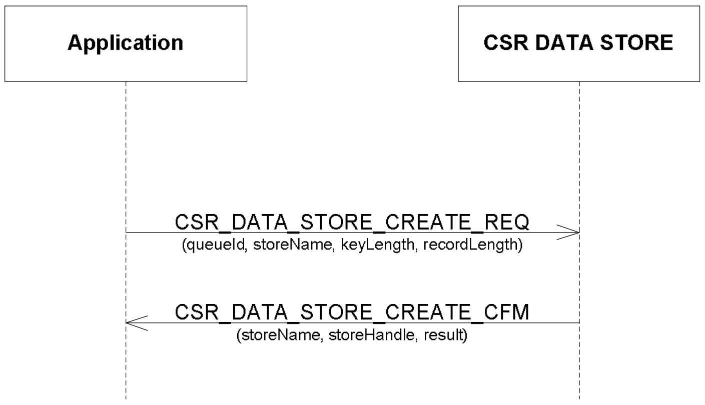

Application
CSR DATA STORE
CSR_DATA_STORE_CREATE_REQ
(queueId, storeName, keyLength, recordLength)
CSR_DATA_STORE_CREATE_CFM
(storeName, storeHandle, result)

Figure 2: Creation of a data store

3.2 Opening of a Data Store

Figure 2 illustrates how the Application can open an existing data store.

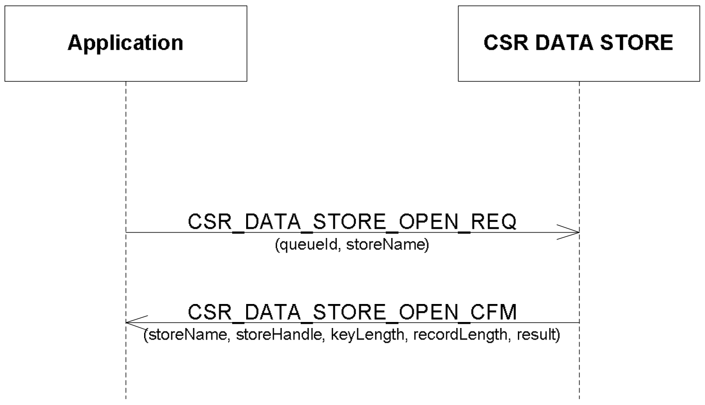


Figure 3: Opening of a data store

3.3 Closure of a Data Store

Figure 2 illustrates how the Application can close an open data store handle.

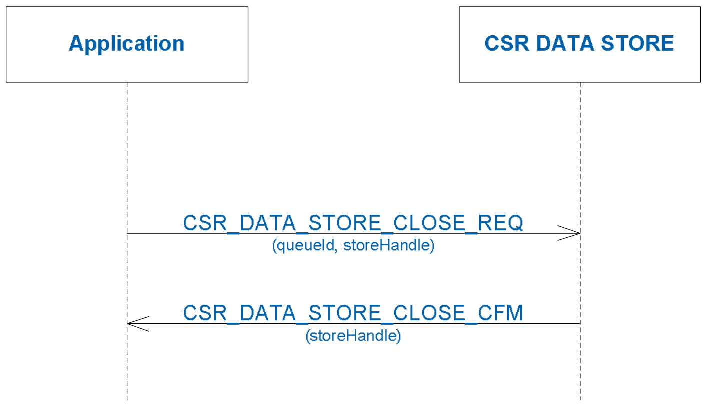


Figure 4: Closure of a data store

3.4 Deletion of a Data Store

Figure 2 illustrates how the Application can delete an existing data store.

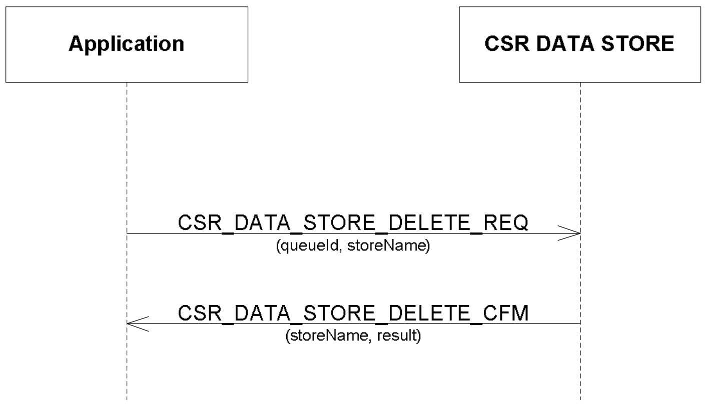


Figure 5: Deletion of a data store

3.5 Writing of a Record to a Data Store

Figure 2 illustrates how the Application can write a record to an open data store.

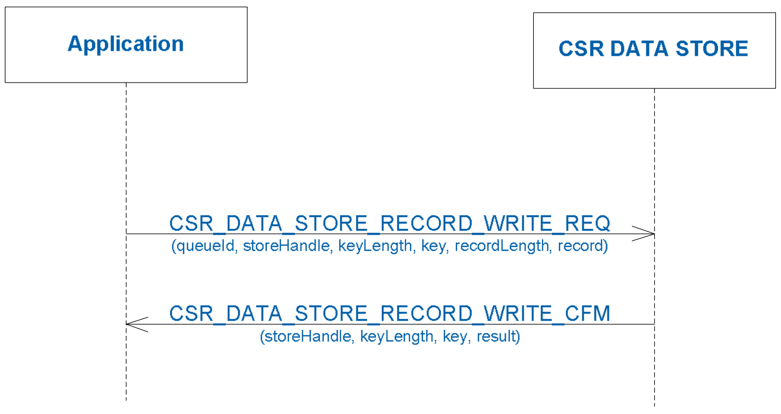

Application
CSR DATA STORE
CSR_DATA_STORE_RECORD_WRITE_REQ
(queueId, storeHandle, keyLength, key, recordLength, record)
CSR_DATA_STORE_RECORD_WRITE_CFM
(storeHandle, keyLength, key, result)

Figure 6: Writing of a record to a data store

3.6 Reading a Record in a Data Store

Figure 2 illustrates how the Application can read a record in an open data store.

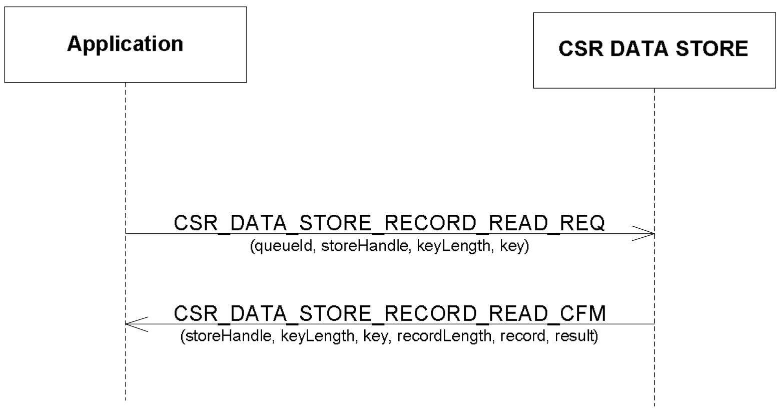

Application
CSR DATA STORE
CSR_DATA_STORE_RECORD_READ_REQ
(queueId, storeHandle, keyLength, key)
CSR_DATA_STORE_RECORD_READ_CFM
(storeHandle, keyLength, key, recordLength, record, result)

Figure 7: Reading a record in a data store

3.7 Deletion of a Record in a Data Store

Figure 2 illustrates how the Application can delete a record in an open data store.

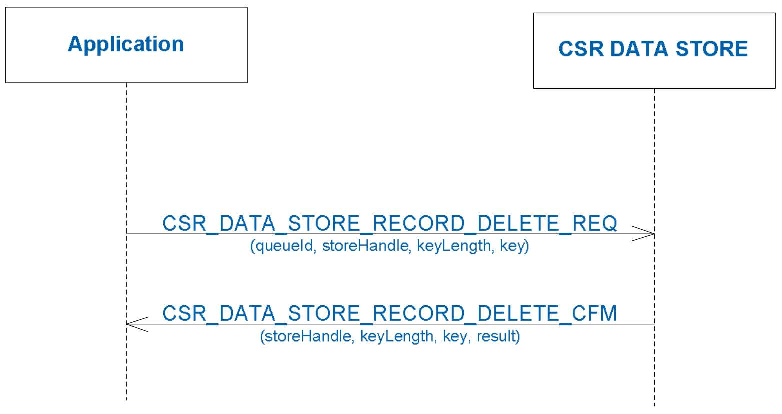

Application
CSR DATA STORE
CSR_DATA_STORE_RECORD_DELETE_REQ
(queueId, storeHandle, keyLength, key)
CSR_DATA_STORE_RECORD_DELETE_CFM
(storeHandle, keyLength, key, result)

Figure 8: Deletion of a record in a data store

4 CSR DATA STORE Primitives

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

Primitives	**Reference**
CSR_DATA_STORE_CREATE	Section 4.1
CSR_DATA_STORE_OPEN	Section 4.2
CSR_DATA_STORE_CLOSE	Section 4.3
CSR_DATA_STORE_DELETE	Section 4.4
CSR_DATA_STORE_RECORD_WRITE	Section 4.5
CSR_DATA_STORE_ RECORD_READ	Section 4.6
CSR_DATA_STORE_ RECORD_DELETE	Section 4.7

Table 1: List of CSR DATA STORE Primitives

4.1 CSR_DATA_STORE_CREATE

Primitives \ Parameters	type	queueId	*storeName	keyLength	recordLength	storeHandle	result
CSR_DATA_STORE _CREATE_REQ	✓	✓	✓	✓	✓		
CSR_DATA_STORE _CREATE_CFM	✓		✓			✓	✓

Table 2: CSR_DATA_STORE_CREATE Primitives

Description

Creates a new data store.

Please note that it is possible to have multiple data stores open with the same queueId.

Parameters

type	`CSR_DATA_STORE_CREATE_REQ/CFM.`
queueId	The identity of the calling task.
*storeName	The name of the new data store,
keyLength	The length of keys in the new data store.
recordLength	The maximum length of records in the new data store.
storeHandle	The handle to the newly created data store. This handle can be used for future transactions to this new data store, There is hence no need to open it first after this call.

result	The outcome of the operation.
	
	If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are:
	`CSR_DATA_STORE_CREATE_FAILURE`
	`CSR_DATA_STORE_CREATE_NOT_ALLOWED`
	
	The application should assume the unused values as reserved for future usage and hence disregard them

4.2 CSR_DATA_STORE_OPEN

Parameters / Primitives	queueId	*storeName	keyLength	recordLength	storeHandle	result
CSR_DATA_STORE_OPEN_REQ	✓	✓				
CSR_DATA_STORE_OPEN_CFM		✓	✓	✓	✓	✓

Table 3: CSR_DATA_STORE_SESSION_DESTROY Primitives

Description

Open an existing data store.

Please note that it is possible to have multiple data stores open with the same queueId.

Parameters

type	`CSR_DATA_STORE_CREATE_REQ/CFM`
queueId	The identity of the calling task.
*storeName	The name of the data store,
keyLength	The length of keys in the data store.
recordLength	The maximum length of records in the data store.
storeHandle	The handle to the newly created data store. This handle can be used for future transactions to this data store.

result	The outcome of the operation.
	
	If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are:
	`CSR_DATA_STORE_OEPN_FAILURE`
	`CSR_DATA_STORE_OPEN_NOT_EXIST`
	
	The application should assume the unused values as reserved for future usage and hence disregard them

4.3 CSR_DATA_STORE_CLOSE

Primitives \ Parameters	type	queueId	storeHandle
CSR_DATA_STORE_CLOSE_REQ	✓	✓	✓
CSR_DATA_STORE_CLOSE_CFM	✓		✓

Table 4: CSR_DATA_STORE_CLOSE Primitives

Description

Close a data store handle.

Parameters

type	`CSR_DATA_STORE_CLOSE_REQ/CFM`
queueId	The identity of the calling task.
storeHandle	The handle to the data store.

4.4 CSR_DATA_STORE_DELETE

Primitives \ Parameters	type	queueId	*storeName	result
CSR_DATA_STORE_DELETE_REQ	✓	✓	✓	
CSR_DATA_STORE_DELETE_CFM	✓		✓	✓

Table 5: CSR_DATA_STORE_DELETE Primitive

Description

Delete an existing data store.

Parameters

type	`CSR_DATA_STORE_UP_REQ/CFM`
queueId	The identity of the calling task.
storeName	The name of the data store.
result	The outcome of the operation. If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are: `CSR_DATA_STORE_DELETE_FAILURE` The application should assume the unused values as reserved for future usage and hence disregard them

4.5 CSR_DATA_STORE_RECORD_WRITE

Primitives \ Parameters	type	queueId	storeHandle	keyLength	*key	recordLength	*record	result
CSR_DATA_STORE_RECORD_WRITE_REQ	✓	✓	✓	✓	✓	✓	✓	
CSR_DATA_STORE_RECORD_WRITE_CFM	✓		✓	✓	✓			✓

Table 6: CSR_DATA_STORE_RECORD_WRITE Primitives

Description

Writes a record to a data store.

Parameters

type	`CSR_DATA_STORE_RECORD_WRITE_REQ/CFM`
queueId	The identity of the calling task.
storeHandle	The data store handle to perform the operation on.
keyLength	The length of the key used for identifying the record.
*key	The key used for identifying the record.
recordLength	The length of the record to write in the data store.
*record	The record to write.
result	The outcome of the operation.

If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are:
`CSR_DATA_STORE_RECORD_WRITE_FAILURE`
`CSR_DATA_STORE_RECORD_WRITE_INVALID_HANDLE`

The application should assume the unused values as reserved for future usage and hence disregard them.

4.6 CSR_DATA_STORE_RECORD_READ

Primitives \ Parameters	type	queueId	storeHandle	keyLength	*key	recordLength	*record	result
CSR_DATA_STORE_RECORD_READ_REQ	✓	✓	✓	✓	✓			
CSR_DATA_STORE_RECORD_READ_CFM	✓		✓	✓	✓	✓	✓	✓

Table 7: CSR_DATA_STORE_RECORD_READ Primitives

Description

Reads a record from a data store.

Parameters

type	`CSR_DATA_STORE_RECORD_READ_REQ/CFM`
queueId	The identity of the calling task.
storeHandle	The data store handle to perform the operation on.
keyLength	The length of the key used for identifying the record.
*key	The key used for identifying the record.
recordLength	The length of the record read out of the data store.
*record	The record that was read.

result	The outcome of the operation.
	
	If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are:
	`CSR_DATA_STORE_RECORD_READ_FAILURE`
	`CSR_DATA_STORE_RECORD_READ_NOT_EXIST`
	`CSR_DATA_STORE_RECORD_READ_INVALID_HANDLE`
	
	The application should assume the unused values as reserved for future usage and hence disregard them.

4.7 CSR_DATA_STORE_RECORD_DELETE

Primitives \ Parameters	type	queueId	storeHandle	keyLength	*key	result
CSR_DATA_STORE_RECORD_DELETE_REQ	✓	✓	✓	✓	✓	
CSR_DATA_STORE_RECORD_DELETE_CFM	✓		✓	✓	✓	✓

Table 8: CSR_DATA_STORE_RECORD_DELETE Primitives

Description

Deletes a record from an open data store.

Parameters

type	`CSR_DATA_STORE_RECORD_DELETE_REQ/CFM`
queueId	The identity of the calling task.
storeHandle	The data store handle to perform the operation on.
keyLength	The length of the key used for identifying the record.
*key	The key used for identifying the record.

result	The outcome of the operation.
	
	If successful this will be set to `CSR_RESULT_SUCCESS` if it fails the possible result codes are:
	`CSR_DATA_STORE_RECORD_DELETE_FAILURE`
	`CSR_DATA_STORE_RECORD_DELETEE_INVALID_HANDLE`
	
	The application should assume the unused values as reserved for future usage and hence disregard them.

5 Document References

Terms and Definitions

CSR	Cambridge Silicon Radio
MSC	Message Sequence Chart
NVS	Non Volatile Storage

Table 9: Abbreviations and Definitions

Document History

Revision	Date	History
1	27 NOV 09	Initial revision
2	30 NOV 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	OCT 10	Ready for release 2.2.0
5	DEC 10	Ready for release 3.0.0
6	Aug 11	Ready for release 3.1.0

TradeMarks, Patents and Licences

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.