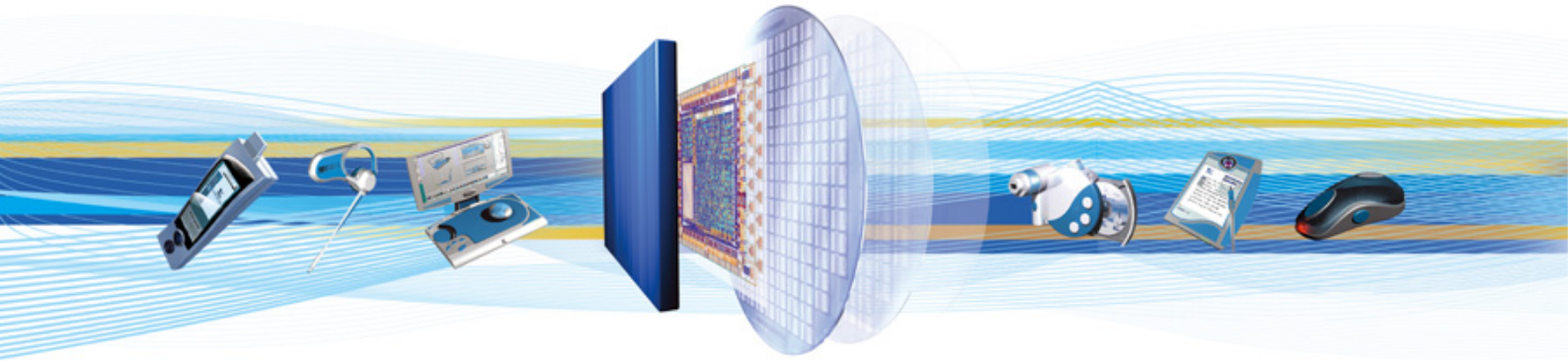
CSR Synergy Framework 3.1.0

FSAL

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 needs file system access and CSR FSAL. The API is called CSR FSAL.

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- Only one instance of CSR FSAL is active at any time
- All strings sent between the application and CSR FSAL is encoded as UTF8 and path separators are always specified as '/' (forward slash)
- CSR FSAL is running in a separate thread or scheduler instance so that if this task blocks 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 FSAL API. After this section the reader should be familiar with the location of CSR FSAL API in the overall architecture and the reason for introducing the API.

2.1 Introduction

The CSR FSAL API provides asynchronous file system access needed by other Synergy tasks.

API provides the following functionality:

- Basic file handling
- Basic directory handling
- Removal and creation of files and folders
- The interface is able to handle interaction with multiple tasks simultaneously and running in different directories on the file system

2.2 Reference Model

CSR FSAL API and its location.

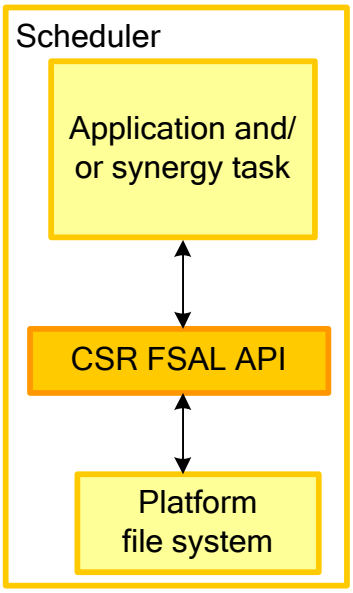


Figure 1: The CSR FSAL API shown relative to the platform file system

3 Interface Description

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

3.1 Session Creation and Closure

Figure 2 shows an application requesting permission to do file system operations through CSR FSAL by sending CSR_FSAL_SESSION_CREATE_REQ and answered by a CSR_FSAL_SESSION_CREATE_CFM which contains a “sessionId” that must be used in all future transactions with the CSR FSAL. After the session is opened any number of file operations and dir operations can occur until the session is closed again by sending CSR_FSAL_SESSION_DESTROY_REQ after which no file or directory operations are allowed for this specific sessionId. It is important to note that same application (scheduler task queue) can have multiple simultaneous sessions with the CSR FSAL and that a directory location is bound to the sessionId meaning that file and directory operations can take place in different locations for different sessionId’s.

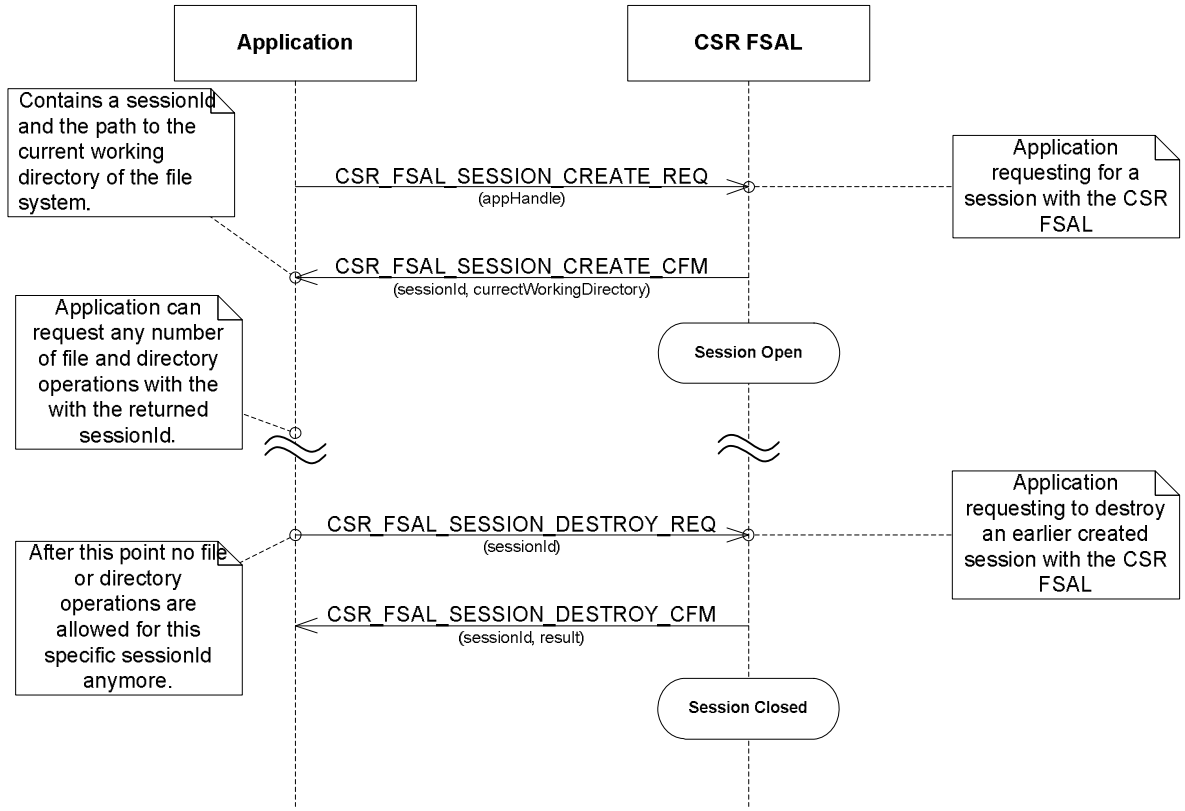


Figure 2: A session life cycle with CSR FSAL

3.2 Standard File Operations Example

Figure 3 illustrates how the application can perform standard file system operations. First a new file is created with CSR_FSAL_FILE_OPEN_REQ, then the file is written to with CSR_FSAL_FILE_WRITE_REQ, then seek’ed in with CSR_FSAL_FILE_SEEK_REQ, then read from with CSR_FSAL_FILE_READ_REQ and finally closed again with CSR_FSAL_FILE_CLOSE_REQ.

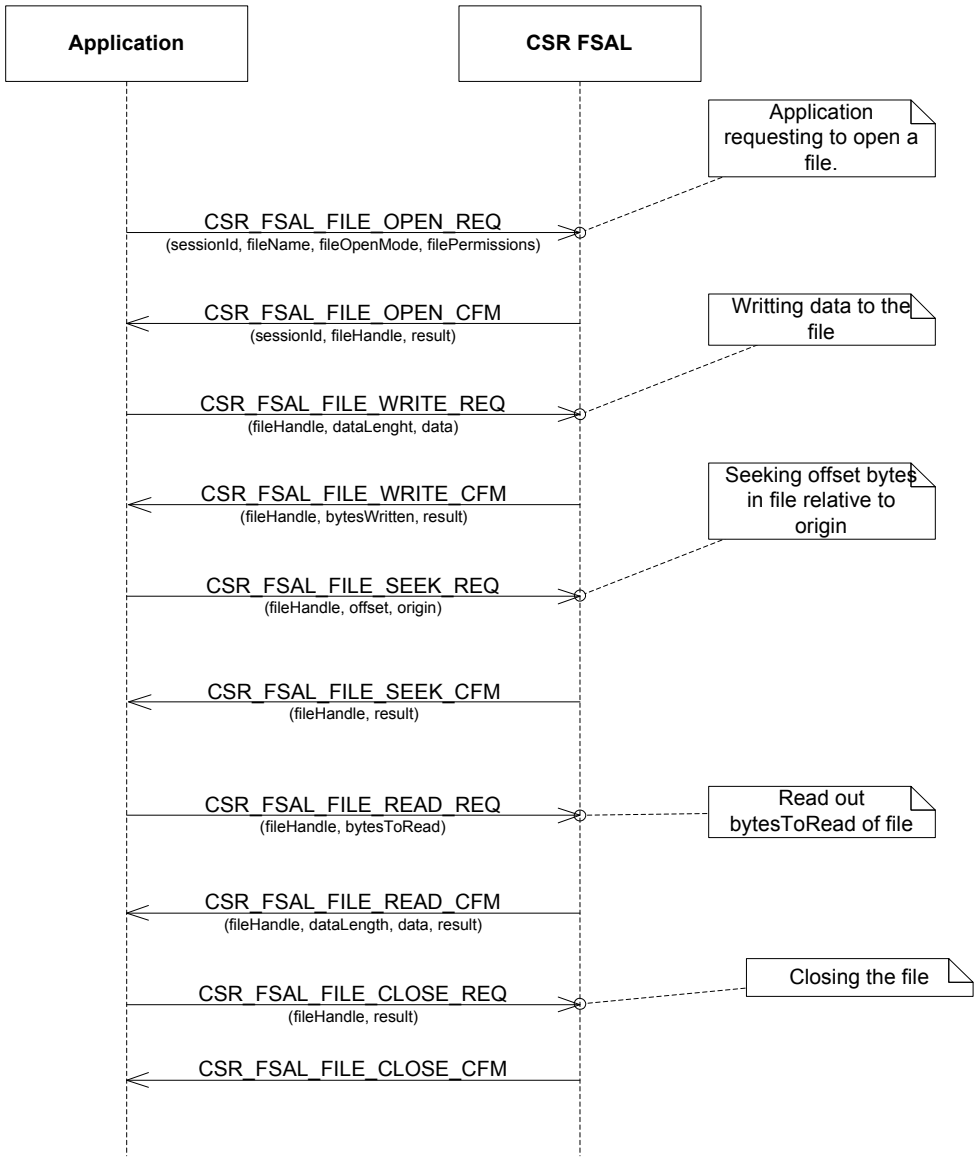


Figure 3: A standard set of file operations with CSR FSAL

3.3 Standard Directory Operations Example

Figure 4 illustrates how the application can perform standard directory operations. First a directory listing is performed with CSR_FSAL_DIR_FIND_FIRST_REQ/CSR_FSAL_FIND_NEXT_REQ, when the first directory is found the search is closed with CSR_FSAL_FIND_CLOSE_REQ and after that the current working directory is changed with CSR_FSAL_DIR_CHANGE_REQ. Finally, is a new directory made with CSR_FSAL_DIR_MAKE_REQ.

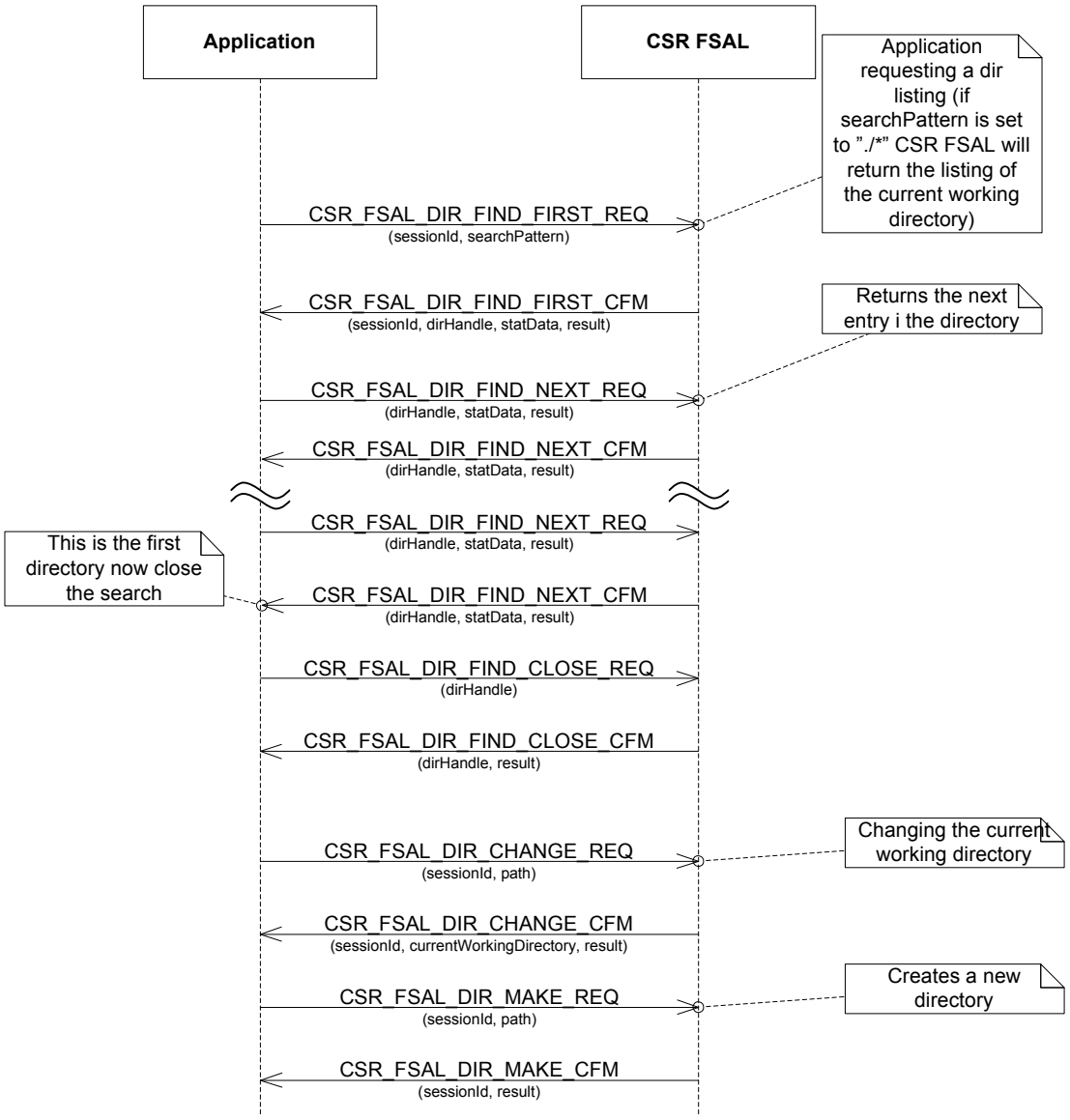


Figure 4: A standard set of directory operations with CSR FSAL

4 CSR FSAL Primitives

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

Primitives	Reference
CSR_FSAL_SESSION_CREATE	Section 4.1
CSR_FSAL_SESSION_DESTROY	Section 4.2
CSR_FSAL_FILE_OPEN	Section 4.3
CSR_FSAL_FILE_CLOSE	Section 4.4
CSR_FSAL_FILE_READ	Section 4.5
CSR_FSAL_FILE_WRITE	Section 4.6
CSR_FSAL_FILE_SEEK	Section 4.7
CSR_FSAL_FILE_TELL	Section 4.8
CSR_FSAL_STAT	Section 4.9
CSR_FSAL_REMOVE	Section 4.10
CSR_FSAL_RENAME	Section 4.11
CSR_FSAL_PERMISSIONS_SET	Section 4.12
CSR_FSAL_DIR_MAKE	Section 4.13
CSR_FSAL_DIR_CHANGE	Section 4.14
CSR_FSAL_DIR_FIND_FIRST	Section 4.15
CSR_FSAL_DIR_FIND_NEXT	Section 4.16
CSR_FSAL_ DIR_FIND_CLOSE	Section 4.17
CSR_FSAL_REMOVE_RECURSIVELY	Section 4.18

Table 1: List of CSR FSAL Primitives

4.1 CSR_FSAL_SESSION_CREATE

Primitives \ Parameters	type	appHandle	sessionId	*currentWorkingDir	result
CSR_FSAL_SESSION_CREATE_REQ	✓	✓			
CSR_FSAL_SESSION_CREATE _CFM	✓		✓	✓	✓

Table 2: CSR_FSAL_SESSION_CREATE Primitives

Description

Creates a new session with CSR FSAL.

Please note that it is possible to have multiple sessions associated with the same appHandle. It is the responsibility of the CSR FSAL to keep track of the current working directory for a given sessionId.

Parameters

type	`CSR_FSAL_SESSION_CREATE_REQ/CFM`
appHandle	The identity of the calling task.
sessionId	The assigned sessionId to be used for all future interactions with the FSAL
*currentWorkingDir	The path to the current working directory for this sessionId.
result	The outcome of the operation. If successful this will be set to `CSR_RESULT_SUCCESS` all other values indicates errors but are currently not used. The application should assume the unused values as reserved for future usage and hence disregard them.

4.2 CSR_FSAL_SESSION_DESTROY

Parameters / Primitives	type	sessionId	result
CSR_FSAL_ SESSION_DESTROY_REQ	✓	✓	
CSR_FSAL_ SESSION_DESTROY_CFM	✓	✓	✓

Table 3: CSR_FSAL_SESSION_DESTROY Primitives

Description

Destroys an earlier created session with CSR FSAL. After this all other operations other than CSR_FSAL_SESSION_CREATE_REQ towards the CSR FSAL is illegal with the sessionId in this request.

Parameters

type `CSR_FSAL_SESSION_DESTROY_REQ/CFM`

sessionId The sessionId to destroy.

result The outcome of the operation.

If successful this will be set to `CSR_RESULT_SUCCESS` all other values indicates errors but are currently not used. The application should assume the unused values as reserved for future usage and hence disregard them.

4.3 CSR_FSAL_FILE_OPEN

Primitives \ Parameters	type	sessionId	flags	permissions	*fileName	handle	size	result
CSR_FSAL_FILE_OPEN_REQ	✓	✓	✓	✓	✓			
CSR_FSAL_FILE_OPEN_CFM	✓					✓	✓	✓

Table 4: CSR_FSAL_FILE_OPEN Primitives

Description

Open a file.

Parameters

type — `CSR_FSAL_FILE_OPEN_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

flags — Bitmask that specifies the type of operations allowed on the file.

Possible bit flags are:

```
CSR_FSAL_OPEN_FLAGS_CREATE
CSR_FSAL_OPEN_FLAGS_READ_ONLY
CSR_FSAL_OPEN_FLAGS_WRITE_ONLY
CSR_FSAL_OPEN_FLAGS_READ_WRITE
CSR_FSAL_OPEN_FLAGS_APPEND
CSR_FSAL_OPEN_FLAGS_TRUNCATE
CSR_FSAL_OPEN_FLAGS_EXCL
```

permissions — Bitmask specifying the allowed permissions to the file.

Possible bit flags are:

```
CSR_FSAL_OPEN_PERMS_NOT_APPLICABLE
CSR_FSAL_OPEN_PERMS_READ
CSR_FSAL_OPEN_PERMS_WRITE
```

NB: This parameter is only applicable if the `CSR_FSAL_OPEN_FLAGS_CREATE` is set in the flags parameter.

*fileName — The path to the file which should be opened.

Examples of valid file names are:

“foo[.extention]”: A file specified in this way should be opened relative to the current working directory which CSR FSAL knows through the sessionId. The [.extention] part is not mandatory.

./[directory1/][directory2/]foo[.extention]: A file specified in this way should be opened in the specified directory path but still relative to the current working directory which CSR FSAL knows through the sessionId. The [.extention] part is not mandatory. The number of [directory/] in the path can be any number but the FSAL is allowed to impose a maximum length in bytes to any path (including the path of the current working directory). It is recommended not to impose this limit to be less than 255

bytes.

/[directory1/][directory2/]foo[.extention]: A file specified in this way should be opened in the specified directory path but from the root of the filesystem. The [.extention] part is not mandatory. The number of [directory/] in the path can be any number but the FSAL is allowed to impose a maximum length in bytes to any path (including the path of the current working directory). It is recommended not to impose this limit to be less than 255 bytes.

handle — The file handle to use in any subsequent operations on the file.

size — The size of the file opened.

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_FSAL_FILE_OP_FAILURE
CSR_FSAL_FILE_OP_EOF
CSR_FSAL_FILE_OP_READ_ONLY
CSR_FSAL_FILE_OP_NOT_EXIST
CSR_FSAL_FILE_OP_NOT_ALLOWED
CSR_FSAL_FILE_OP_ALREAD_EXISTS
CSR_FSAL_FILE_OP_NO_SPACE
```

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

4.4 CSR_FSAL_FILE_CLOSE

Primitives \ Parameters	type	handle	result
CSR_FSAL_FILE_CLOSE_REQ	✓	✓	
CSR_FSAL_FILE_CLOSE_CFM	✓		✓

Table 5: CSR_FSAL_FILE_CLOSE Primitive

Description

Close a file.

Parameters

type `CSR_FSAL_UP_REQ/CFM`

handle The file handle to perform the operation on.

result The outcome of the operation.

If successful this will be set `to CSR_RESULT_SUCCESS` all other values indicates errors but are currently not used. The application should assume the unused values as reserved for future usage and hence disregard them.

4.5 CSR_FSAL_FILE_READ

Parameters / Primitives	type	handle	bytesToRead	dataLen	*data	result
CSR_FSAL_FILE_READ_REQ	✓	✓	✓			
CSR_FSAL_FILE_READ_CFM	✓	✓		✓	✓	✓

Table 6: CSR_FSAL_FILE_READ Primitives

Description

Reads data from file.

Parameters

type — `CSR_FSAL_FILE_READ_REQ/CFM`

handle — The file handle to perform the operation on.

bytesToRead — The amount of bytes to read out of file

dataLen — The amount of bytes actually read out of file

*data — The pointer to the data.

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_FSAL_FILE_OP_FAILURE
CSR_FSAL_FILE_OP_EOF
CSR_FSAL_FILE_OP_READ_ONLY
CSR_FSAL_FILE_OP_NOT_EXIST
CSR_FSAL_FILE_OP_NOT_ALLOWED
CSR_FSAL_FILE_OP_ALREAD_EXISTS
CSR_FSAL_FILE_OP_NO_SPACE
```

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

4.6 CSR_FSAL_FILE_WRITE

Primitives \ Parameters	type	handle	bytesWritten	dataLen	*data	result
CSR_FSAL_FILE_WRITE _REQ	✓	✓		✓	✓	
CSR_FSAL_FILE_WRITE_CFM	✓	✓	✓			✓

Table 7: CSR_FSAL_DHCP_INFORM Primitives

Description

Write data to file.

Parameters

type — `CSR_FSAL_FILE_WRITE_REQ/CFM`

handle — The file handle to perform the operation on.

bytesWritten — The amount of bytes actually written to file

dataLen — The amount of bytes to write to file

*data — The pointer to the data.

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_FSAL_FILE_OP_FAILURE
CSR_FSAL_FILE_OP_EOF
CSR_FSAL_FILE_OP_READ_ONLY
CSR_FSAL_FILE_OP_NOT_EXIST
CSR_FSAL_FILE_OP_NOT_ALLOWED
CSR_FSAL_FILE_OP_ALREAD_EXISTS
CSR_FSAL_FILE_OP_NO_SPACE
```

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

4.7 CSR_FSAL_FILE_SEEK

Primitives \ Parameters	type	handle	offset	origin	result
CSR_FSAL_FILE_SEEK_REQ	✓	✓	✓	✓	
CSR_FSAL_FILE_SEEK_CFM	✓	✓			✓

Table 8: CSR_FSAL_FILE_SEEK Primitive

Description

Seek in file.

Parameters

type — `CSR_FSAL_FILE_SEEK_REQ/CFM`

handle — The file handle to perform the operation on.

offset — The offset to move in the file relative to origin

origin — The origin from where the seek operation should be performed.

Possible values are:

```
CSR_FSAL_SEEK_SET
CSR_FSAL_SEEK_CUR
CSR_FSAL_SEEK_END
```

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_FSAL_FILE_OP_FAILURE
CSR_FSAL_FILE_OP_EOF
CSR_FSAL_FILE_OP_READ_ONLY
CSR_FSAL_FILE_OP_NOT_EXIST
CSR_FSAL_FILE_OP_NOT_ALLOWED
CSR_FSAL_FILE_OP_ALREAD_EXISTS
CSR_FSAL_FILE_OP_NO_SPACE
```

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

4.8 CSR_FSAL_FILE_TELL

Primitives \ Parameters	type	handle	position	result
CSR_FSAL_FILE_TELL_REQ	✓	✓		
CSR_FSAL_FILE_TELL_CFM	✓	✓	✓	✓

Table 9: CSR_FSAL_FILE_TELL Primitive

Description

Tell the current position of the file pointer in file.

Parameters

Type	`CSR_FSAL_FILE_TELL_REQ/CFM`
Handle	The file handle to perform the operation on.
Position	The current position of the file pointer.

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_FSAL_FILE_OP_FAILURE`
	`CSR_FSAL_FILE_OP_EOF`
	`CSR_FSAL_FILE_OP_READ_ONLY`
	`CSR_FSAL_FILE_OP_NOT_EXIST`
	`CSR_FSAL_FILE_OP_NOT_ALLOWED`
	`CSR_FSAL_FILE_OP_ALREAD_EXISTS`
	`CSR_FSAL_FILE_OP_NO_SPACE`
	
	The application should assume the unused values as reserved for future usage and hence disregard them.

4.9 CSR_FSAL_STAT

Primitives \ Parameters	type	sessionId	path	stat	result
CSR_FSAL_STAT_REQ	✓	✓	✓		
CSR_FSAL_STAT_CFM	✓	✓		✓	✓

Table 10: CSR_FSAL_STAT Primitive

Description

Get status information on a file or directory.

Parameters

type `CSR_FSAL_STAT_REQ/CFM`

sessionId The sessionId for which the operation is performed.

*path The path to the file or directory. The same clauses applies to this parameter as the `filename` parameter in Section 4.3.

stat

The status information of the specified file or directory.

The information is filled into the following `CsrFsalDirEntry` struct

typedef struct

```
{
    CsrUtf8String  *name;   /* Name of entry */
    CsrFsalTm      time;    /* Time of last modification */
    CsrSize        size;    /* 0 if not file */
    CsrFsalMode    mode;    /* mode */
} CsrFsalDirEntry;
```

Where the the time parameter is specified but the `CsrFsalTm` struct below.

```
typedef struct
{
    CsrTime tm_sec;   /* Seconds: 0-59  */
    CsrTime tm_min;   /* Minutes: 0-59  */
    CsrTime tm_hour;  /* Hours since midnight: 0-23 */
    CsrTime tm_mday;  /* Day of the month: 1-31 */
    CsrTime tm_mon;   /* Months since january: 0-11 */
    CsrTime tm_year;  /* Years since 1900 */
    CsrTime tm_wday;  /* Days since Sunday (0-6) */
    CsrTime tm_yday;  /* Days since Jan. 1: 0-365 */
    CsrTime tm_isdst; /* +1 Daylight Savings Time, 0 No DST,
0xFFFF don't know    */
    CsrBool utcTime;  /* TRUE=UTC, FALSE=local time */
} CsrFsalTm;
```

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_FSAL_STAT_FAILURE`
`CSR_FSAL_STAT_NOT_EXIST`

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

4.10 CSR_FSAL_REMOVE

Parameters / Primitives	type	sessionId	*path	result
CSR_FSAL_REMOVE_REQ	✓	✓	✓	
CSR_FSAL_REMOVE_CFM	✓	✓		✓

Table 11: CSR_FSAL_REMOVE Primitive

Description

Remove a file or directory. It is the responsibility of CSR FSAL to find out if it is a file or directory that should be removed.

Please note that if a directory is tried to be removed and this directory contains files or subdirectories this operation should fail. If the application wishes to remove a directory with content it should instead use `CSR_FSAL_REMOVE_RECURSIVELY_REQ` described in 4.18.

Parameters

type — `CSR_FSAL_REMOVE_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

*path — The path to the file or directory. The same clauses applies to this parameter as the `*filename` parameter in Section 4.3.

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_FSAL_DELETE_FAILURE
CSR_FSAL_DELETE_READ_ONLY
CSR_FSAL_DELETE_NOT_EXIST
CSR_FSAL_DELETE_NOT_EMPTY
```

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

4.11 CSR_FSAL_RENAME

Parameters / Primitives	type	sessionId	*oldPath	*newPath	result
CSR_FSAL_RENAME_REQ	✓	✓	✓	✓	
CSR_FSAL_RENAME_CFM	✓	✓			✓

Table 12: CSR_FSAL_RENAME Primitive

Description

Rename a file or directory.

Parameters

type — `CSR_FSAL_RENAME_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

*oldPath — The old path to the file or directory. The same clauses applies to this parameter as the `*filename` parameter in Section 4.3.

*newPath — The new path to the file or directory. The same clauses applies to this parameter as the `*filename` parameter in Section 4.3.

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_FSAL_RENAME_FAILURE`
`CSR_FSAL_RENAME_NOT_EXIST`
`CSR_FSAL_RENAME_NOT_ALLOWED`

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

4.12 CSR_FSAL_PERMISSIONS_SET

Primitives \ Parameters	type	sessionId	*path	permissions	result
CSR_FSAL_PERMISSIONS_SET_REQ	✓	✓	✓	✓	
CSR_FSAL_PERMISSIONS_SET_CFM	✓	✓			✓

Table 13: CSR_FSAL_PERMISSIONS_SET Primitive

Description

Set permissions for a file or directory.

Parameters

type — `CSR_FSAL_PERMISSIONS_SET_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

*path — The path to the file or directory. The same clauses applies to this parameter as the `*filename` parameter in Section 4.3.

permissions — The permissions which should apply to the specified file or directory.

Possible values are:

```
CSR_FSAL_PERMISSION_USER_READ
CSR_FSAL_PERMISSION_USER_WRITE
CSR_FSAL_PERMISSION_USER_EXECUTE
CSR_FSAL_PERMISSION_GROUP_READ
CSR_FSAL_PERMISSION_GROUP_WRITE
CSR_FSAL_PERMISSION_GROUP_EXECUTE
CSR_FSAL_PERMISSION_OTHERS_READ
CSR_FSAL_PERMISSION_OTHERS_WRITE
CSR_FSAL_PERMISSION_OTHERS_EXECUTE
```

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_FSAL_SET_PERMISSIONS_FAILURE
CSR_FSAL_SET_PERMISSIONS_NOT_EXIST
```

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

4.13 CSR_FSAL_DIR_MAKE

Parameters / Primitives	type	sessionId	*dirName	result
CSR_FSAL_DIR_MAKE_REQ	✓	✓	✓	
CSR_FSAL_DIR_MAKE_CFM	✓	✓		✓

Table 14: CSR_FSAL_DIR_MAKE Primitive

Description

Creates a new directory.

Parameters

type — `CSR_FSAL_DIR_MAKE_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

* dirName — The path to the new directory. The same clauses regarding a path in the directory name applies to this parameter as the `*filename` parameter in Section 4.3.

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_FSAL_DIR_MAKE_FAILURE
CSR_FSAL_DIR_MAKE_EXIST
CSR_FSAL_DIR_MAKE_INVALID_PATH
```

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

4.14 CSR_FSAL_DIR_CHANGE

Parameters / Primitives	type	sessionId	*path	*currentWorkingDir	result
CSR_FSAL_DIR_CHANGE_REQ	✓	✓	✓		
CSR_FSAL_DIR_CHANGE_CFM	✓	✓		✓	✓

Table 15: CSR_FSAL_DIR_CHANGE Primitive

Description

Change to a new directory.

Parameters

type `CSR_FSAL_DIR_CHANGE_REQ/CFM`

sessionId The sessionId for which the operation is performed.

*path The path to the new directory. The same clauses regarding a path in the directory name applies to this parameter as the `*filename` parameter in Section 4.3.

*currentWorkingDir The path to the current working directory for this sessionId.

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_FSAL_DIR_CHANGE_FAILURE`
`CSR_FSAL_DIR_CHANGE_NOT_EXIST`

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

4.15 CSR_FSAL_DIR_FIND_FIRST

Primitives \ Parameters	type	sessionId	*searchPattern	handle	entry	result
CSR_FSAL_DIR_FIND_FIRST_REQ	✓	✓	✓			
CSR_FSAL_DIR_FIND_FIRST_CFM	✓	✓		✓	✓	✓

Table 16: CSR_FSAL_DIR_FIND_FIRST Primitive

Description

Provides information about the first instance of a file or directory name that matches the searchPattern in a directory.

Please note that if this operation succeeds it is the responsibility of the application to close the search again with `CSR_FSAL_DIR_FIND_CLOSE_REQ`.

Parameters

type — `CSR_FSAL_DIR_FIND_FIRST_REQ/CFM`

sessionId — The sessionId for which the operation is performed.

*searchPattern — The search pattern to match. The search pattern are allowed to contain wildcards and paths like demonstrated in the clauses regarding the `*filename` parameter in Section 4.3.

handle — The directory handle to use in `CSR_FSAL_DIR_FIND_NEXT_REQ` and `CSR_FSAL_DIR_FIND_CLOSE_REQ`.

entry — The information of the first file or directory that matches searchPattern.

The information is filled into the following `CsrFsalDirEntry` struct

typedef struct

```
{
    CsrUtf8String  *name;  /* Name of entry */
    CsrFsalTm      time;   /* Time of last modification */
    CsrSize        size;   /* 0 if not file */
    CsrFsalMode    mode;   /* mode */
} CsrFsalDirEntry;
```

Where the the time parameter is specified but the `CsrFsalTm` struct below.

```
typedef struct
{
    CsrTime tm_sec;   /* Seconds: 0-59  */
    CsrTime tm_min;   /* Minutes: 0-59  */
    CsrTime tm_hour;  /* Hours since midnight: 0-23 */
    CsrTime tm_mday;  /* Day of the month: 1-31 */
    CsrTime tm_mon;   /* Months since january: 0-11 */
    CsrTime tm_year;  /* Years since 1900 */
    CsrTime tm_wday;  /* Days since Sunday (0-6) */
```

```
    CsrTime tm_yday;  /* Days since Jan. 1: 0-365 */
    CsrTime tm_isdst; /* +1 Daylight Savings Time, 0 No DST,
0xFFFF don't know    */
    CsrBool utcTime;  /* TRUE=UTC, FALSE=local time */
} CsrFsalTm;
```

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_FSAL_DIR_OP_FAILURE

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

4.16 CSR_FSAL_DIR_FIND_NEXT

Parameters / Primitives	type	handle	entry	result
CSR_FSAL_DIR_FIND_NEXT_REQ	✓	✓		
CSR_FSAL_DIR_FIND_NEXT_CFM	✓	✓	✓	✓

Table 17: CSR_FSAL_DIR_FIND_NEXT Primitive

Description

Searches for the next instance of a file or directory that matches the search patter provided in `CSR_FSAL_DIR_FIND_FIRST_REQ`.

Parameters

type — `CSR_FSAL_DIR_FIND_NEXT_REQ/CFM`

handle — The dir handle for which the operation is performed.

entry — The information of the next file or directory that matches searchPattern from `CSR_FSAL_DIR_FIND_FIRST_REQ`. This parameter is structured in the same way as the entry parameter described in 4.15

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_FSAL_DIR_OP_NO_MORE_MATCHING_ENTRIES`

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

4.17 CSR_FSAL_DIR_FIND_CLOSE

Parameters / Primitives	type	handle	result
CSR_FSAL_DIR_FIND_CLOSE_REQ	✓	✓	
CSR_FSAL_DIR_FIND_CLOSE_CFM	✓	✓	✓

Table 18: CSR_FSAL_DIR_FIND_CLOSE Primitive

Description

Closes and ongoing find session started with `CSR_FSAL_DIR_FIND_FIRST_REQ`.

Parameters

type `CSR_FSAL_DIR_FIND_CLOSE_REQ/CFM`

handle The dir handle for which the operation is performed.

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_FSAL_DIR_OP_FAILURE`

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

4.18 CSR_FSAL_REMOVE_RECURSIVELY

Parameters / Primitives	type	sessionId	*dir	result
CSR_FSAL_REMOVE_RECURSIVELY_REQ	✓	✓	✓	
CSR_FSAL_REMOVE_RECURSIVELY_CFM	✓	✓		✓

Table 19: CSR_FSAL_REMOVE_RECURSIVELY Primitive

Description

Remove a directory recursively, so that if it contain files or subfolders these are removed before the actual directory is removed.

Parameters

type `CSR_FSAL_REMOVE_RECURSIVELY_REQ/CFM`

sessionId The sessionId for which the operation is performed.

*dir The path to the directory. The dir string are allowed to contain paths like demonstrated in the clauses regarding the `*filename` parameter in Section 4.3.

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_FSAL_DELETE_FAILURE`

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

Table 20: 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

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.

No statements or representations in this document are to be construed as advertising, marketing, or offering for sale in the United States imported covered products subject to the Cease and Desist Order issued by the U.S. International Trade Commission in its Investigation No. 337-TA-602. Such products include SiRFstarIII™ chips that operate with SiRF software that supports SiRFInstantFix™, and/or SiRFLoc® servers, or contains SyncFreeNav functionality.

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.