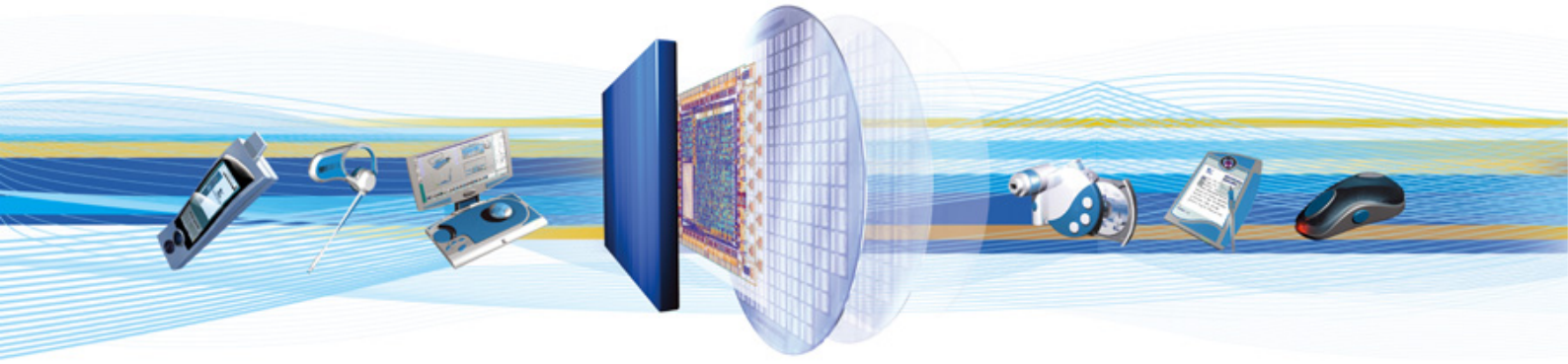
# CSR Synergy™

# Synergy Framework 3.1.0

# DHCP Server

# 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

Bluetooth®

# Contents

**List of Figures**



**List of Tables**

# 1 Introduction

## 1.1 Introduction and Scope

This document describes the API for the CSR Dynamic Host Configuration Protocol (DHCP) Server.

The CSR DHCP Server described in this document is implemented as defined in RFC 2132. The following should be noted about the implementation:

- In section 4.1 in RFC 2131 it states that the server unicast DHCPOFFER and DHCPACK messages to the client's hardware address and 'yiaddr' address if the broadcast bit is not set and 'giaddr' is zero and 'ciaddr' is zero. This is not possible, as the CSR DHCP Server implementation uses the API described in [1], that API does not make that possible, and the messages are broadcast instead.

- The CSR DHCP Server will not offer a client on a different subnet an IP address, i.e. if the 'giaddr' is not zero in a DHCPDISCOVER message.

- The CSR DHCP Server does not probe the offered address with an ICMP Echo request message before a client is offered the IP address. It is in the interest of the client to check if the IP address offered is not used by anyone.

.

# 2 Description

This section will give an overview of the CSR DHCP Server API.

## 2.1 Reference Model

Figure 1 illustrates the CSR DHCP Server API and its location relative to applications and CSR IP Socket API.

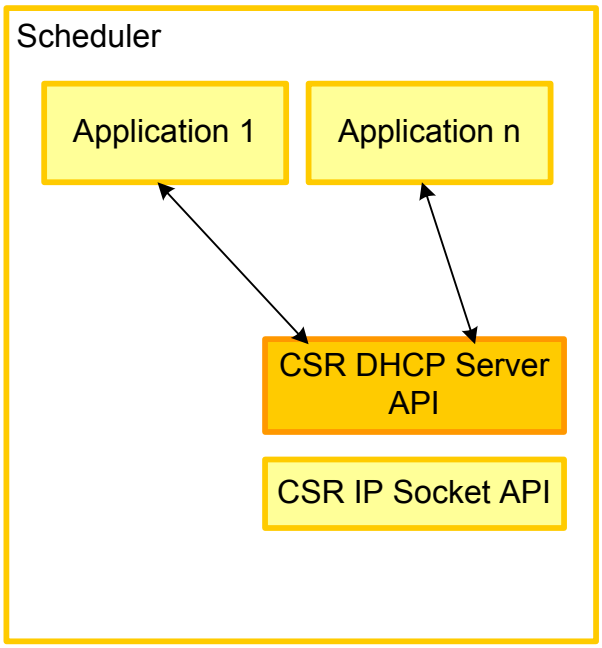



**Figure 1: The CSR DHCP Server API shown relative to the applications**

## 2.2 Sequence Overview

Figure 2 illustrates how the API of the CSR DHCP Server can be used. First, the application shall create a server with a CSR_DHCP_SERVER_CREATE_REQ. When the server has been created, it can be started with a CSR_DHCP_SERVER_START_REQ. The server can be stopped again with a CSR_DHCP_SERVER_STOP_REQ. When the server has been stopped, it will ignore all messages it receives from any clients. When the application does not want to use the server anymore it shall destroy the server. It can do so by sending a CSR_DHCP_SERVER_DESTROY_REQ, which will remove the server and all its leases again.

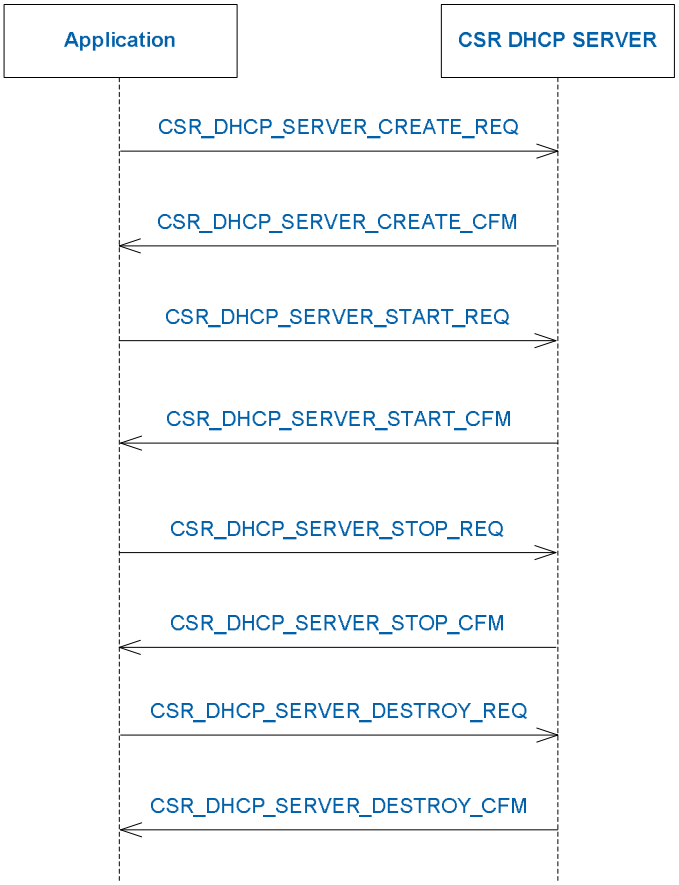



**Figure 2: MSC for using the DHCP Server**

If the application wants to change the leases, the application shall stop the server and then get the current leases with a CSR_DHCP_SERVER_LEASES_GET_REQ. When the leases has been changed they can be sent to the server with a CSR_DHCP_SERVER_LEASES_SET_REQ. When the server receives a CSR_DHCP_SERVER_LEASES_SET_REQ it will clear all the known leases and then use the leases received. For this reason, it is important that the server is not running from when the application gets the current leases to it sets the new one. This could give an inconsistency between the server's leases and the clients' leases as all leases updated in between the CSR_DHCP_SERVER_LEASES_GET_REQ and CSR_DHCP_SERVER_LEASES_SET_REQ will be lost. An MSC for this can be seen in Figure 3.


**Figure 3: MSC for changing leases**

The CSR_DHCP_SERVER_LEASES_GET_REQ can also be used by the application to get the current leases and e.g. show them to the user. In this case, the application does not need to send a CSR_DHCP_SERVER_STOP_REQ before the application sends the CSR_DHCP_SERVER_LEASES_GET_REQ signal.

The CSR_DHCP_SERVER_LEASES_SET_REQ can also be used for creating a binding between a client and an IP address. This is described in Section 2.4

## 2.3 Client State

The server uses the following internally states of a client.

- IDLE: The server has no knowledge of the client.
- OFFERED: If the client has been offered an IP address and a lease time.
- ACTIVE: If the client has an IP address and a lease time that has not expired.
- INACTIVE: If the client has an IP address but a lease time that has expired.

These states will decide which IP addresses clients will be offered by the server.

## 2.4 IP Address

A client will be offered an IP address depending of the state of the client.

- OFFERED: The client will be offered the same IP address as already offered to the client.

- ACTIVE: The client will be offered the same IP address as it already has
- INACTIVE: The client will be offered the same IP address as it had earlier if the IP address is not already used by another client, i.e. in OFFERED or ACTIVE state. If no IP addresses can be found it will try to find an IP address as if the client was in IDLE stat.
- IDLE: If the client has requested an IP address the client will be offered that IP address if not used, else the server will find an IP address not used. A used IP address is an address used by a client in OFFERED, ACTIVE or INACTIVE state. If no IP addresses can be found it will use an IP address from a client in INACTIVE state if one exists.

The application can create a binding between a client and an IP address with the signal CSR_DHCP_SERVER_LEASES_SET_REQ. By doing this the client will be in either ACTIVE or INACTIVE state depending of the lease time given in the signal. The client will then be offered an IP address as described above. If the lease time is infinite, the client will be guaranteed to be offered the IP address specified in the signal.

One thing that should be noted is that internally the CSR DHCP Server implementation is using a socket listing on the broadcast address 255.255.255.255 on the DHCP server port 67 to receive broadcast messages. When this socket receives, a broadcast message it will send the message to all the servers that has been created and is started. It does this, as the CSR DHCP Server does not know which interface the message was received on. The server will then decide how to handle this message. For this reason if multiple servers is running they could all offer the client an IP address and it is then up to the client which one to accept.

## 2.5 Lease Time

The expiration time for the lease is chosen as follows:

- If the client has requested a specific lease, the client will get that lease rounded down to whole hours. Except if the lease is below 1 hour the lease will be 1 hour and if the lease is above 255 hours, the lease will be 255 hours. If the requested lease time is infinite, the client will get an infinite lease time.
- If the client has not requested a specific lease time and is in ACTIVE state then the client will get the expiration time previously assigned to the client.
- If the client has not requested a specific lease time and the client is not in ACTIVE state the client will get the default lease time specified in a CSR_DHCP_SERVER_CREATE_REQ.

## 2.6 Rogue IP addresses

When the server discovers that a suggested IP address is already in use a CSR_DHCP_SERVER_ROGUE_IND signal is sent to the application. The server will discover this when a client sends a DHCPDECLINE message to the server. This indicates of a possible configuration problem. The server will mark the address as not available and not offer the address to a client again. The list of not available addresses will be cleared again when the server receives a CSR_DHCP_SERVER_START_REQ after the server has been stopped.

# 3 CSR DHCP Server Primitives

This section introduces all the primitives and parameters used in the CSR DHCP Server API. Detailed information can be found in the csr_dhcp_server_prim.h file.

## 3.1 Socket Primitives

| Primitives | Reference |
|---|---|
| CSR_DHCP_SERVER_CREATE_REQ | See Section 3.2 |
| CSR_DHCP_SERVER_CREATE_CFM | See Section 3.2 |
| CSR_DHCP_SERVER_DESTROY_REQ | See Section 3.3 |
| CSR_DHCP_SERVER_DESTROY_CFM | See Section 3.3 |
| CSR_DHCP_SERVER_START_REQ | See Section 3.4 |
| CSR_DHCP_SERVER_START_CFM | See Section 3.4 |
| CSR_DHCP_SERVER_STOP_REQ | See Section 3.5 |
| CSR_DHCP_SERVER_STOP_CFM | See Section 3.5 |
| CSR_DHCP_SERVER_LEASES_GET_REQ | See Section 3.6 |
| CSR_DHCP_SERVER_LEASES_GET_CFM | See Section 3.6 |
| CSR_DHCP_SERVER_LEASES_SET_REQ | See Section 3.7 |
| CSR_DHCP_SERVER_LEASES_SET_CFM | See Section 3.7 |
| CSR_DHCP_SERVER_ROGUE_IP_ADDRESS_IND | See Section 3.8 |

**Table 1: List of DHCP Primitives**

## 3.2 CSR_DHCP_SERVER_CREATE

| Primitives \ Parameters | type | qid | ipAddress | network | networkMask | gateway | dns1 | dns2 | leaseTime | serverHandle | result |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CSR_DHCP_SERVER_CREATE_REQ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| CSR_DHCP_SERVER_CREATE_CFM | ✓ | | | | | | | | | ✓ | ✓ |

**Table 2: CSR_DHCP_SERVER_CREATE Primitives**

**Description**

Create a new DHCP server and receive a handle for it.

**Parameters**

| | |
|---|---|
| type | CSR_DHCP_SERVER_CREATE_REQ/CFM |
| qid | The identity of the calling task. |
| ipAddress | Address to listen to -- if part of network it won't be announced |
| network | Network to announce, e.g. 192.168.0.0 |
| networkMask | Subnet Mask e.g. 255.255.255.0 |
| gateway | Default gateway |
| dns1 | Domain Name Server |
| dns2 | Domain Name Server |
| leaseTime | Default lease lifetime in hours, 0 == infinite. If the client does not suggest a lease, then this lease will be used. The lease offered to a client will always be between 1 hour and 255 hours, or infinite. |
| serverHandle | A unique server handle that is used in subsequent server communication |
| result | • CSR_RESULT_SUCCESS: If the server has been created<br>• CSR_RESULT_FAILURE: If e.g. no more sockets is available, the port 67 is already in use, not enough permissions to listen on port 67, the IP address is not valid |

**Function prototype**
The following function is used for constructing and sending this primitive:

```
CsrDhcpServerCreateReqSend(CsrSchedQid qid,
                           CsrUint8 ipAddress[4],
                           CsrUint8 network[4],
                           CsrUint8 networkMask[4],
                           CsrUint8 gateway[4],
                           CsrUint8 dns1[4],
                           CsrUint8 dns2[4],
                           CsrUint8 leaseTime);
```

## 3.3 CSR_DHCP_SERVER_DESTROY

| Primitives \ Parameters | type | serverHandle | result |
|---|---|---|---|
| CSR_DHCP_SERVER_DESTROY_REQ | ✓ | ✓ | |
| CSR_DHCP_SERVER_DESTROY_CFM | ✓ | ✓ | ✓ |

**Table 3: CSR_DHCP_SERVER_DESTROY Primitives**

**Description**

Destroy the server again.

**Parameters**

| | |
|---|---|
| type | CSR_DHCP_SERVER_DESTROY_REQ/CFM |
| serverHandle | A unique handle for the server |
| result | CSR_RESULT_SUCCESS |

**Function prototype**
The following function is used for constructing and sending this primitive:

```
CsrDhcpServerDestroyReqSend(CsrDhcpServerHandle serverHandle);
```

## 3.4 CSR_DHCP_SERVER_START

| Parameters / Primitives | type | serverHandle | result |
|---|---|---|---|
| CSR_DHCP_SERVER_START_REQ | ✓ | ✓ | |
| CSR_DHCP_SERVER_START_CFM | ✓ | ✓ | ✓ |

**Table 4: CSR_DHCP_SERVER_START Primitives**

**Description**

Start the server.

**Parameters**

Type — CSR_DHCP_SERVER_START_REQ/CFM

serverHandle — A unique handle for the server

result
- CSR_RESULT_SUCCESS: If the server has been started
- CSR_RESULT_FAILURE: If the server is already started

**Function prototype**
The following function is used for constructing and sending this primitive:

```
CsrDhcpServerStartReqSend(CsrDhcpServerHandle serverHandle);
```

## 3.5 CSR_DHCP_SERVER_STOP_REQ

| Primitives \ Parameters | type | serverHandle | result |
|---|---|---|---|
| CSR_DHCP_SERVER_STOP_REQ | ✓ | ✓ | |
| CSR_DHCP_SERVER_STOP_CFM | ✓ | ✓ | ✓ |

**Table 5: CSR_DHCP_SERVER_STOP Primitives**

**Description**

Stop the server.

**Parameters**

type — CSR_DHCP_SERVER_STOP_REQ/CFM

serverHandle — A unique handle for the server

result
- CSR_RESULT_SUCCESS: If the server has been stopped
- CSR_RESULT_FAILURE: If the server could not be stopped

**Function prototype**
The following function is used for constructing and sending this primitive:

```
CsrDhcpServerStopReqSend(CsrDhcpServerHandle serverHandle);
```

## 3.6 CSR_DHCP_SERVER_LEASES_GET

| Primitives \ Parameters | type | serverHandle | leasesCount | leases |
|---|---|---|---|---|
| CSR_DHCP_SERVER_LEASES_GET_REQ | ✓ | ✓ | | |
| CSR_DHCP_SERVER_LEASES_GET_CFM | ✓ | ✓ | ✓ | ✓ |

**Table 6: CSR_DHCP_SERVER_LEASES_GET Primitives**

**Description**

Get the current leases.

**Parameters**

| | |
|---|---|
| type | CSR_DHCP_SERVER_LEASES_GET_REQ/CFM |
| serverHandle | A unique handle for the server |
| leasesCount | The number of leases |
| leases | A pointer to the leases |

The leases are of the type CsrDhcpServerLease.

```
typedef struct
{
    CsrUint8    ip[4];
    CsrUint8    mac[6];
    CsrTimeUtc  expiryTime;

} CsrDhcpServerLease;
```

| | |
|---|---|
| ip | The IP address of the client |
| mac | The hardware address of the client |
| expiryTime | The time when the lease expire. If sec == 0xFFFFFFFF and msec == 0xFFFF the lease is infinite. If sec == 0x00 and msec == 0x00 the lease has expired |

**Function prototype**

The following function is used for constructing and sending this primitive:

```
CsrDhcpServerLeasesGetReqSend(CsrDhcpServerHandle serverHandle);
```

## 3.7 CSR_DHCP_SERVER_LEASES_SET

| Primitives \ Parameters | type | serverHandle | leasesCount | leases | result |
|---|---|---|---|---|---|
| CSR_DHCP_SERVER_LEASES_SET_REQ | ✓ | ✓ | ✓ | ✓ | |
| CSR_DHCP_SERVER_LEASES_SET_CFM | ✓ | ✓ | | | ✓ |

**Table 7: CSR_DHCP_SERVER_LEASES_SET Primitives**

**Description**

Set the leases.

**Parameters**

| | |
|---|---|
| type | CSR_DHCP_SERVER_LEASES_SET_REQ/CFM |
| serverHandle | A unique handle for the server |
| leasesCount | The number of leases |
| leases | A pointer to the leases. See section 3.6 for a description. |
| result | CSR_RESULT_SUCCESS |

**Function prototype**
The following function is used for constructing and sending this primitive:

```
CsrDhcpServerLeasesSetReqSend(CsrDhcpServerHandle serverHandle,
                              CsrUint16 leasesCount,
                              CsrDhcpServerLease *leases);
```

## 3.8 CSR_DHCP_SERVER_ROGUE_IP_ADDRESS

| Primitives \ Parameters | type | serverHandle | ipAddress |
|---|---|---|---|
| CSR_DHCP_SERVER_ROGUE_IP_ADDRESS_IND | ✓ | ✓ | ✓ |

**Table 8: CSR_DHCP_SERVER_ROGUE_IP_ADDRESS Primitives**

**Description**

Sent to the application when an offered IP address is already in use

**Parameters**

| | |
|---|---|
| type | CSR_DHCP_SERVER_ROGUE_IP_ADDRESS_IND |
| serverHandle | A unique handle for the server |
| ip | The IP address |

# 4 Document References

| Ref | Title |
|---|---|
| [1] | CSR Synergy Socket API Description |

# Terms and Definitions

| | |
|---|---|
| CSR | Cambridge Silicon Radio |
| DHCP | Dynamic Host Configuration Protocol |

**Table 9: Abbreviations and Definitions**

# Document History

| Revision | Date | History |
|---|---|---|
| 1 | Nov 2010 | Initial revision |
| 2 | Aug 1011 | 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.