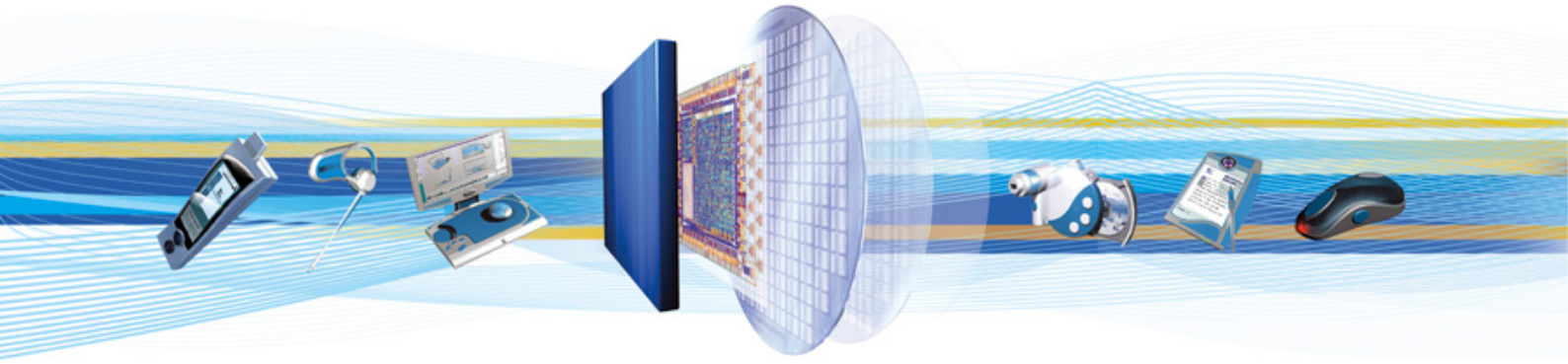
CSR Synergy Framework 3.1.1

Socket

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 API between CSR IP and applications running in the CSR scheduler. The API is called CSR IP Socket as it describes the socket interface on top of CSR IP (Transport layer in the TCP/IP model).

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- The applications using the CSR IP Socket API are using sockets to establish a connection.
- The lower layers are implemented
- Only one instance of CSR IP is active at any time
- Configuration of the interfaces is handled by CSR IP Ifconfig API

It is furthermore assumed that the reader is familiar with [IP_ARCH].

2 Description

This section will briefly describe the purpose of the CSR IP Socket API. After reading this section, the reader should be familiar with the location of CSR IP Socket API in the overall architecture as well as the services provided through it.

2.1 Introduction

The CSR IP Socket API makes it possible for components running within CSR Synergy to interface with a TCP/IP stack for application level communication.

The CSR IP Socket API provides the following functionality:

- Hostname-to-address translation e.g. using DNS lookups
- Connection control
- Data exchange

The CSR IP Socket API is defined to support both IPv4 and IPv6 as described in section 2.3

2.2 Reference Model

Figure 1: CSR IP Socket API shown relative to TCP/IP illustrates the CSR IP Socket API and its location relative to applications and CSR IP. This section describes the functionality provided by CSR IP Socket API as presented in Section 2.1. The socket communication is designed and implemented to be as similar to BSD sockets as possible. The socket communication is defined for both the UDP and TCP protocols as well as raw sockets.

Applications can use the DNS query functionality to resolve DNS names to IP address. It is permitted for an implementation to use site- or device-specific databases for name resolution, e.g. a local hosts file.

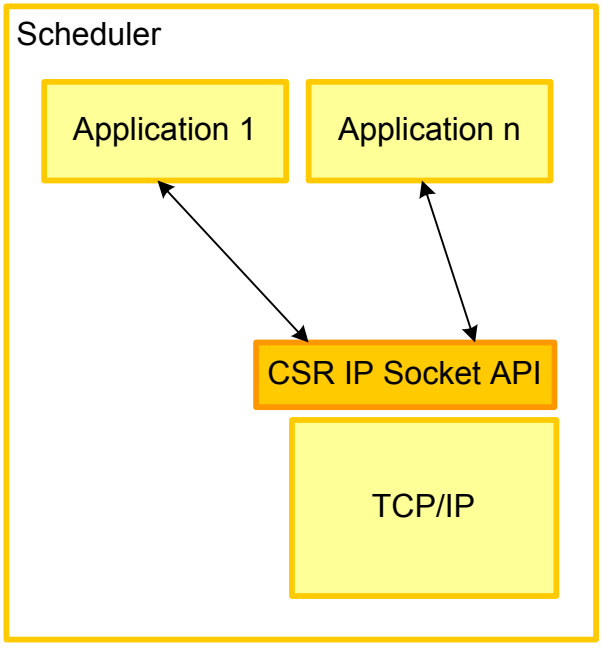


Figure 1: CSR IP Socket API shown relative to TCP/IP

2.3 IPv6 support

The CSR IP Socket API provides an interface that supports both IPv4 and IPv6. However, it is important to note that it is optional for implementations to implement IPv6 support. If an implementation does not support IPv6, it must inform the application of this limitation by returning a specific result value (defined later in this document) if it requests an IPv6 socket. An application has no way of knowing in advance whether the CSR IP Socket

implementation supports IPv6, and it is thus legal for an application to attempt to create IPv6 sockets even if the implementation is IPv4-only. Applications are required to handle the lack of IPv6 support by falling back to IPv4 where possible.

The API has been designed in a way that makes it possible to write an IP protocol version agnostic application that opportunistically uses IPv6 to make seamless migration from IPv4 to IPv6 possible. This has been done by providing an interface in which the same macros are used for IPv4 and IPv6 operations, and by not requiring the application to store any state for controlling the socket concerning the current protocol version for a given socket.

IP addresses are of a fixed length (16 byte array) no matter which protocol version is in use which also means the application can just save the entire array when it receives an address from the CSR IP Socket interface and pass it back to the stack in its entirety. The stack knows the type of socket family for a given socket, so it knows how to interpret the address field.

If manually writing addresses into an IP address field, the order of the bytes is identical to the order in the IP header (i.e. most significant byte first in array offset zero). IPv4 addresses are stored in the first four bytes of the array with the remaining 12 bytes unused. The local loopback address is shown below for both IPv4 (127.0.0.1) and IPv6 (::1).

IP address array offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IPv4	127	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X
IPv6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

2.4 Library interface

While the CSR IP Socket API is a message based API in which primitives (defined later in this document) are sent between CSR Synergy tasks, applications should not be manually allocating and filling out primitives because future specifications of the CSR IP Socket API may be amended that require subtle changes to the interpretation. Backwards compatibility is provided through macros that allocate and fill out primitives to ensure that a certain invocation of a given macro will lead to the identical behaviour across different CSR IP Socket implementations. The macros are defined in the header file csr_ip_socket_lib.h, and generally are named as the corresponding primitive name with ``Send'' appended.

In the current interface, two revisions exist of this macro interface. The difference between the two interface revisions is that the first supports IPv4 only, while the second supports both IPv4 and IPv6. The second interface is named similarly to the first except all macros have ``Send2'' appended instead of just ``Send''. New code should use the second revision of the CSR IP Socket interface to simplify the transition to IPv6. Revision 1 of the macros is considered deprecated and will be removed or changed in future revisions of the CSR Synergy Framework.

2.5 Interface Description

This section will present examples on usage of the API in terms of message sequence charts.

2.6 Socket Communication Message Sequence Charts

This section presents message sequence charts for raw sockets, UDP and TCP socket communication.

2.6.1 UDP Socket Communication

Figure 2 shows a message sequence chart for creating a UDP socket and sending some data using it.

A UDP socket is created by issuing a CSR_IP_SOCKET_UDP_NEW_REQ, which must contain an appHandle specifying the CSR scheduler queue id of the application. The CSR IP Socket layer uses the application handle

to send a CSR_IP_SOCKET_UDP_NEW_CFM message to the application when the socket has been created. The confirm message contains a unique socketHandle, which will be used in subsequent socket communication. The usage of socketHandle is similar to the usage of file descriptors in BSD sockets.

When the socket has been created, a port can be bound to the socket. The CSR_IP_SOCKET_UDP_BIND_REQ must contain the port and IP address, which will be bound to the socket. The CSR_IP_SOCKET_UDP_BIND_CFM indicates that the UDP socket is ready for use.

Data can be transmitted using CSR_IP_SOCKET_UDP_DATA_REQ which must contain an address and a port number of the intended receiver, as well as length of and a pointer to the payload, which is about to be send.

Data is received by a CSR_IP_SOCKET_UDP_DATA_IND which contains the IP address and port number of the sender.

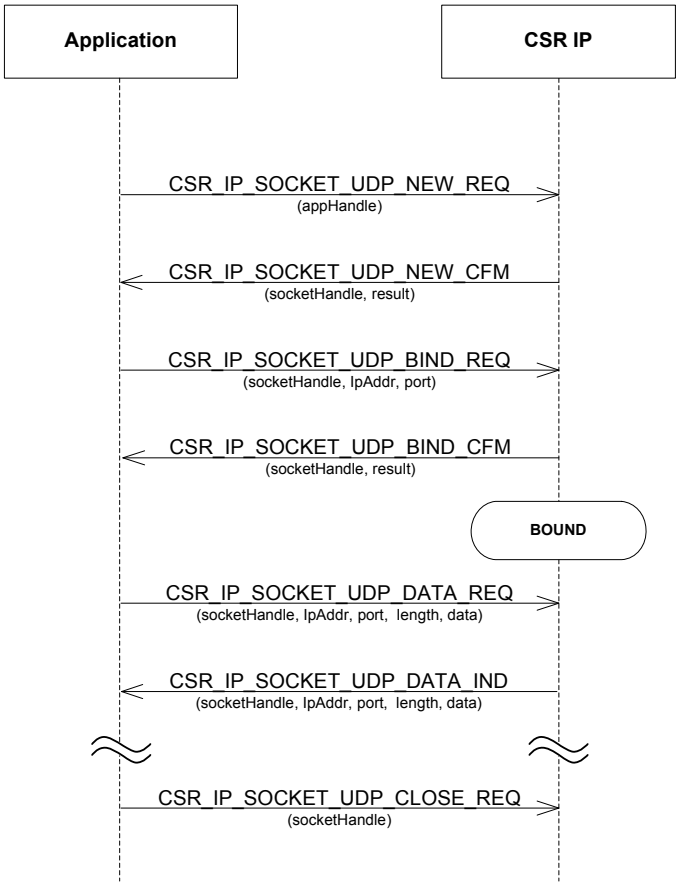


Figure 2: MSC for sending UDP data via CSR IP Socket

2.6.2 Raw Socket Communication

Figure 3 illustrate a message sequence chart (MSC) for creating a RAW socket and send some data over it. The sequence is equal to that of UDP but differs in that a protocol number is send at creation time of the socket and no port number are given when binding a socket and send data over it. The protocol number corresponds to the protocol number in an IP packet header.

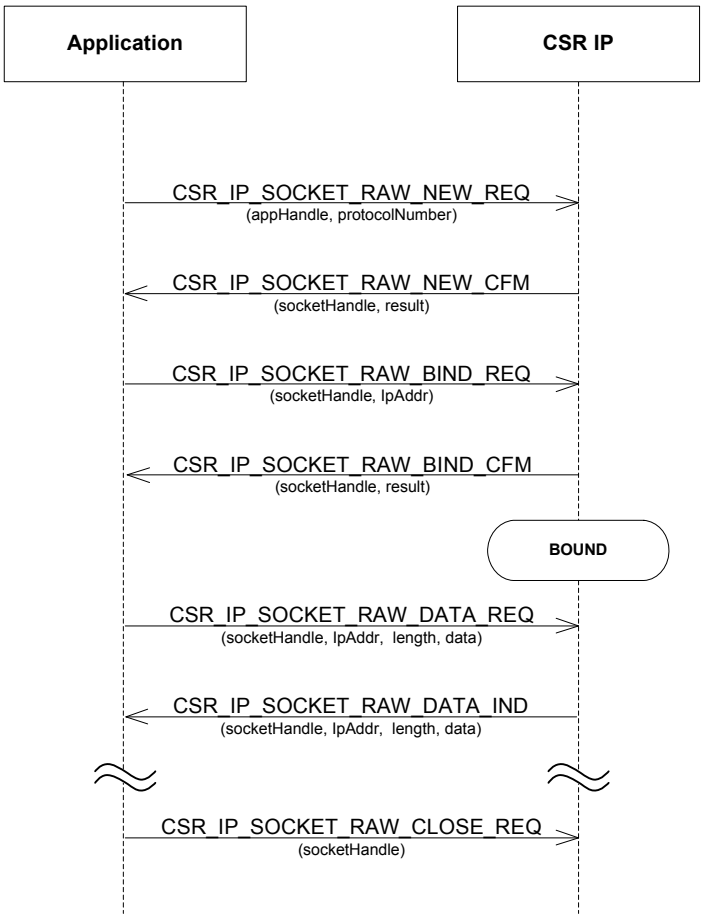


Figure 3: MSC for sending RAW data via CSR IP Socket

2.6.3 TCP Socket Server Communication

The TCP signal flow is similar to the UDP socket communication and thus is omitted.

Figure 4 shows a MSC for a socket server implementation using TCP. An application has created a TCP socket and bound a port to the socket. CSR_IP_SOCKET_TCP_LISTEN_REQ will cause the TCP/IP stack to wait for incoming connections. These will be accepted and a CSR_IP_SOCKET_TCP_ACCEPT_IND will be sent to the application owning the listening socket.

The CSR_IP_SOCKET_TCP_SEND_REQ will send TCP data. A CSR_IP_SOCKET_TCP_CFM will be received when the TCP/IP stack buffer is ready to handle next chunk of data.

2.6.4 TCP Socket Client Communication

Figure 5 illustrates how a client connects to a remote server. A CSR_IP_SOCKET_TCP_CONNECT_REQ is sent, and when the server has accepted the connection, a CSR_IP_SOCKET_TCP_CONNECT_CFM is received.

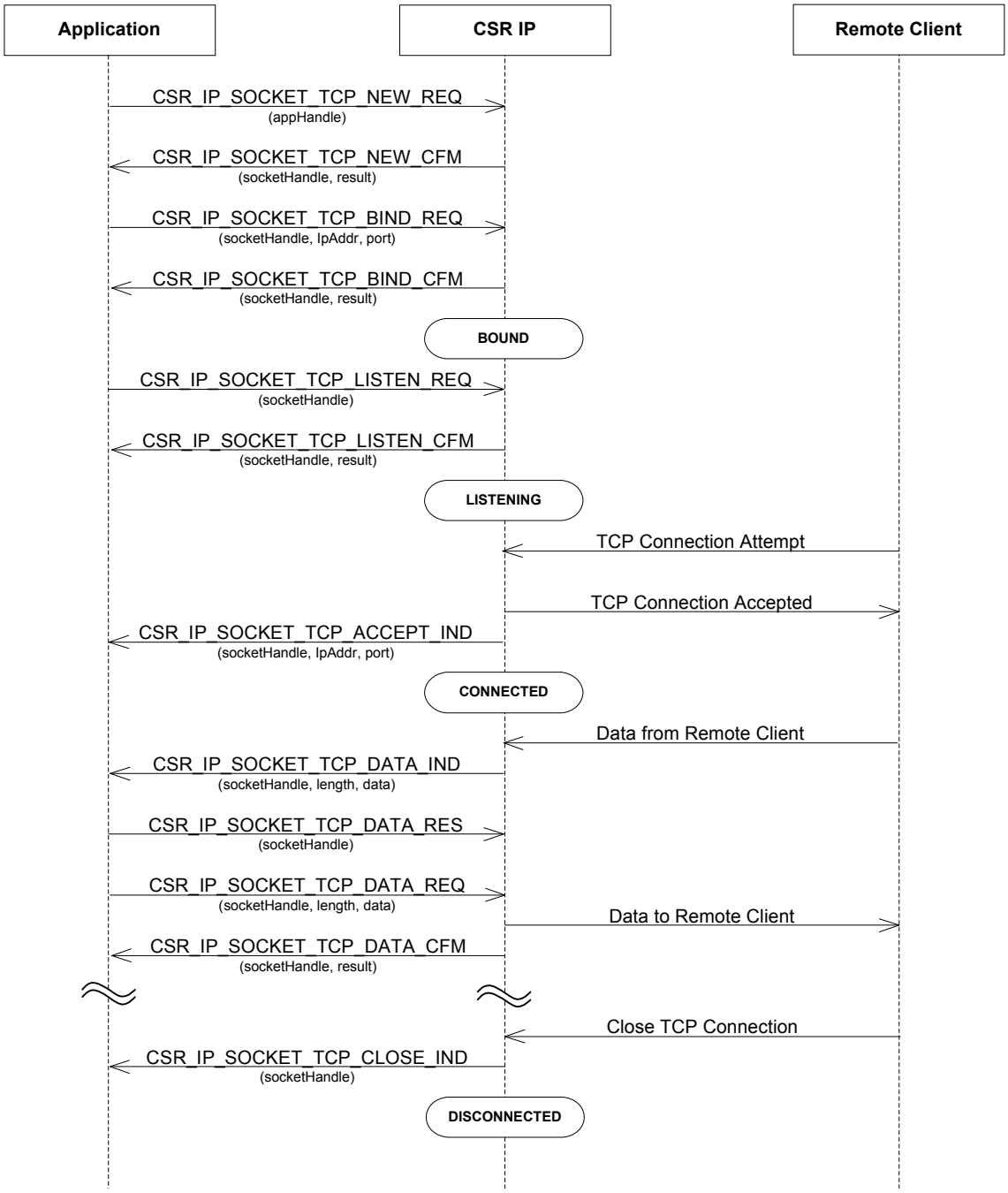


Figure 4: MSC for a TCP server socket implementation, which will transmit and receive data

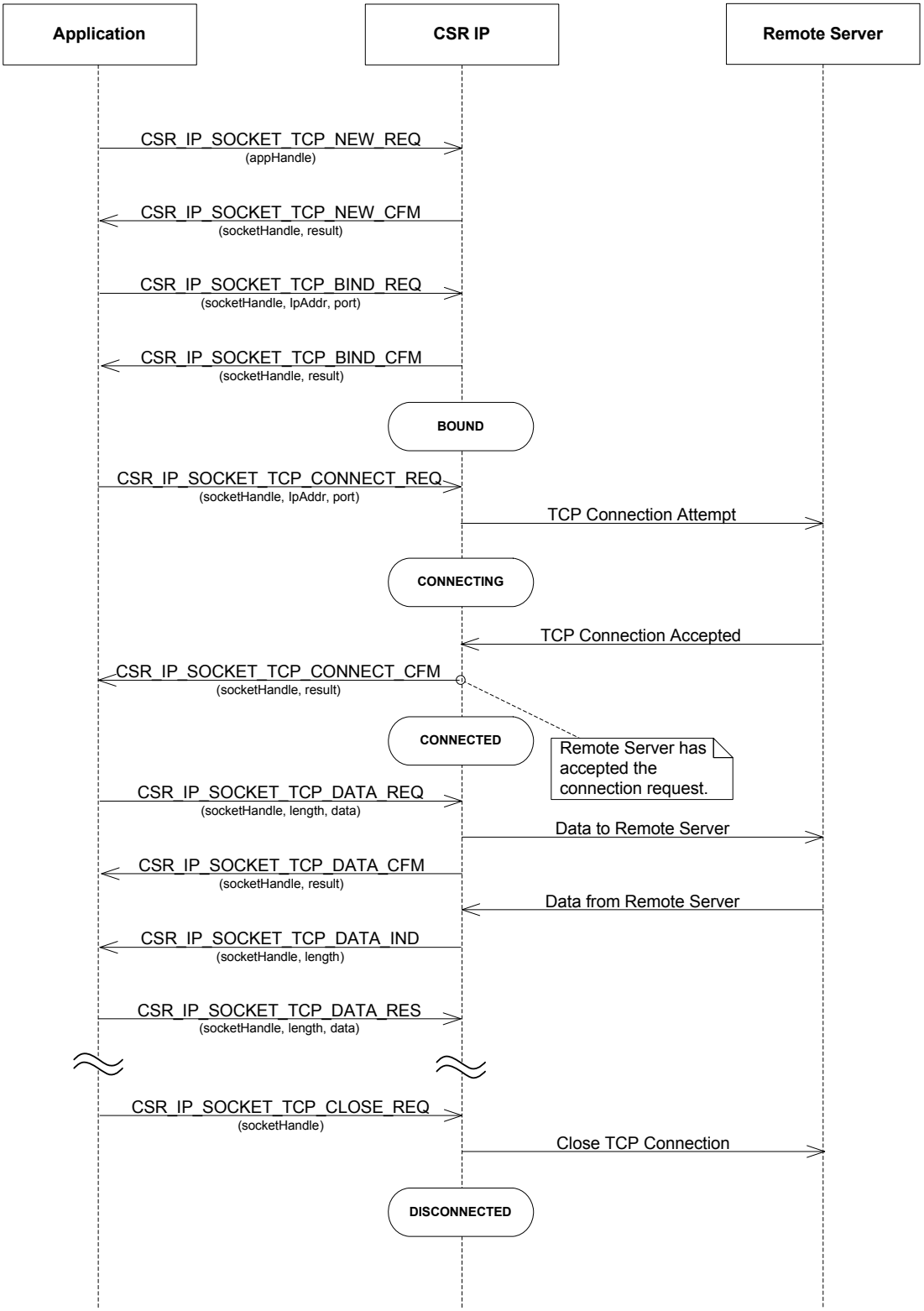


Figure 5: TCP client socket message sequence

2.7 DNS Message Sequence Chart

Figure 6 shows a MSC for a DNS query. An application wish to resolve a DNS name, which must supplied in the CSR_IP_SOCKET_DNS_RESOLVE_NAME_REQ. The resolved IP address is returned to the application in the CSR_IP_SOCKET_DNS_RESOLVE_NAME_CFM.

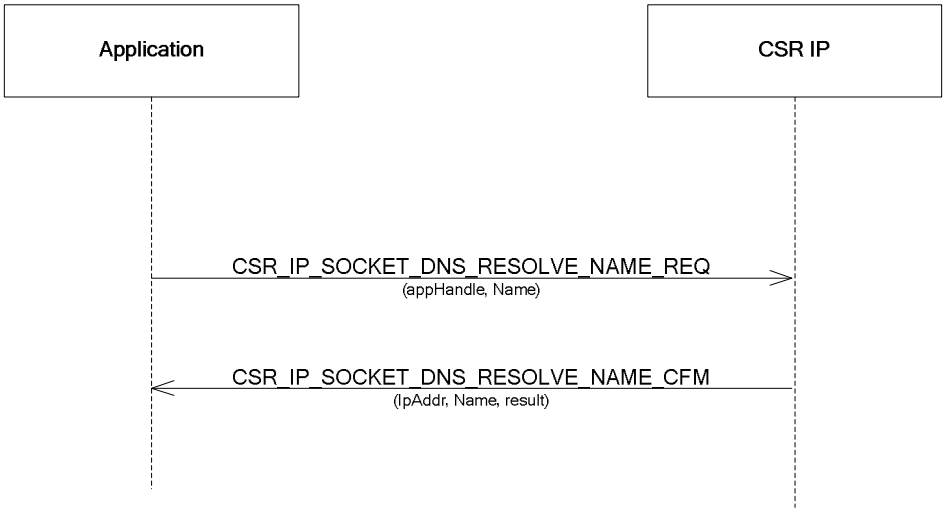


Figure 6: MSC for a DNS query

3 CSR IP Socket Primitives

This section introduces all the primitives and parameters used in the CSR IP Socket API. Detailed information can be found in the csr_ip_socket_prim.h file.

The CSR IP Socket API/layer implements two sets of primitives (or functionalities):

- BSD similar socket primitives
- DNS primitives

3.1 Socket Primitives

Primitives	Reference
CSR_IP_SOCKET_UDP_NEW_REQ	See Section 3.3
CSR_IP_SOCKET_UDP_NEW_CFM	See Section 3.3
CSR_IP_SOCKET_UDP_BIND_REQ	See Section 3.4
CSR_IP_SOCKET_UDP_BIND_CFM	See Section 3.4
CSR_IP_SOCKET_UDP_DATA_REQ	See Section 3.5
CSR_IP_SOCKET_UDP_DATA_CFM	See Section 3.5
CSR_IP_SOCKET_UDP_DATA_IND	See Section 3.5
CSR_IP_SOCKET_UDP_CLOSE_REQ	See Section 3.6
CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE_REQ	See Section 3.7
CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE_CFM	See Section 3.7
CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_REQ	See Section 3.8
CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_CFM	See Section 3.8
CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE_REQ	See Section 3.9
CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE_CFM	See Section 3.9
CSR_IP_SOCKET_TCP_NEW_REQ	See Section 3.10
CSR_IP_SOCKET_TCP_NEW_CFM	See Section 3.10
CSR_IP_SOCKET_TCP_BIND_REQ	See Section 3.11
CSR_IP_SOCKET_TCP_BIND_CFM	See Section 3.11
CSR_IP_SOCKET_TCP_LISTEN_REQ	See Section 3.12
CSR_IP_SOCKET_TCP_LISTEN_CFM	See Section 3.12
CSR_IP_SOCKET_TCP_CONNECT_REQ	See Section 3.13
CSR_IP_SOCKET_TCP_CONNECT_CFM	See Section 3.13
CSR_IP_SOCKET_TCP_ACCEPT_IND	See Section 3.14
CSR_IP_SOCKET_TCP_DATA_REQ	See Section 3.15
CSR_IP_SOCKET_TCP_DATA_CFM	See Section 3.15
CSR_IP_SOCKET_TCP_DATA_IND	See Section 3.15
CSR_IP_SOCKET_TCP_DATA_RES	See Section 3.15
CSR_IP_SOCKET_TCP_CLOSE_REQ	See Section 3.16
CSR_IP_SOCKET_TCP_CLOSE_CFM	See Section 3.16
CSR_IP_SOCKET_TCP_ABORT_REQ	See Section 3.17
CSR_IP_SOCKET_RAW_NEW_REQ	See Section 3.18
CSR_IP_SOCKET_RAW_NEW_CFM	See Section 3.18
CSR_IP_SOCKET_RAW_BIND_REQ	See Section 3.19
CSR_IP_SOCKET_RAW_BIND_CFM	See Section 3.19

CSR_IP_SOCKET_RAW_DATA_REQ	See Section 3.20
CSR_IP_SOCKET_RAW_DATA_IND	See Section 3.20
CSR_IP_SOCKET_RAW_DATA_CFM	See Section 3.20
CSR_IP_SOCKET_RAW_CLOSE_REQ	See Section 3.21
CSR_IP_SOCKET_OPTIONS_REQ	See Section 3.22
CSR_IP_SOCKET_OPTIONS_CFM	See Section 3.22

Table 1: List of Socket Primitives

3.2 DNS Primitives

Primitives	Reference
CSR_IP_SOCKET_DNS_RESOLVE_NAME_REQ	See Section 3.23
CSR_IP_SOCKET_DNS_RESOLVE_NAME_CFM	See Section 3.23

Table 2: List of DNS Primitives

3.3 CSR_IP_SOCKET_UDP_NEW

Parameters / Primitives	type	appHandle	socketFamily	socketHandle	result
CSR_IP_SOCKET_UDP_NEW_REQ	✓	✓	✓		
CSR_IP_SOCKET_UDP_NEW_CFM	✓			✓	✓

Table 3: CSR_IP_SOCKET_UDP_NEW Primitives

Description

Create a new UDP socket and receive a handle for it.

Parameters

type	CSR_IP_SOCKET_UDP_NEW_REQ/CFM
appHandle	The identity of the calling task.
socketFamily	CSR_IP_SOCKET_FAMILY_IP4 or CSR_IP_SOCKET_FAMILY_IP6
socketHandle	A unique socket handle that is used subsequent socket communication
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE, CSR_IP_SOCKET_RESULT_NO_MORE_SOCKETS, or CSR_IP_SOCKET_RESULT_IP6_NOT_SUPPORTED.

3.4 CSR_IP_SOCKET_UDP_BIND

Parameters / Primitives	type	socketHandle	ipAddress	port	result
CSR_IP_SOCKET_UDP_BIND_REQ	✓	✓	✓	✓	
CSR_IP_SOCKET_UDP_BIND_CFM	✓	✓		✓	✓

Table 4: CSR_IP_SOCKET_UDP_BIND Primitives

Description

Bind an IP address and port to a socket.

Parameters

type	CSR_IP_SOCKET_UDP_BIND_REQ/CFM
socketHandle	A unique handle for the socket
ipAddress	IP address to bind to. Interpretation of this field depends on the socket family.
port	The port to bind to. If port is zero, the network stack picks a free port and informs the application of the port in the confirmation signal.
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_SOCKET_RESULT_PORT_IN_USE

3.5 CSR_IP_SOCKET_UDP_DATA

Primitives \ Parameters	type	socketHandle	ipAddress	port	dataLength	*data
CSR_IP_SOCKET_UDP_DATA_REQ	✓	✓	✓	✓	✓	✓
CSR_IP_SOCKET_UDP_DATA_CFM	✓	✓				
CSR_IP_SOCKET_UDP_DATA_IND	✓	✓	✓	✓	✓	✓

Table 5: CSR_IP_SOCKET_UDP_DATA Primitives

Description

CSR_IP_SOCKET_UDP_DATA_REQ sends UDP data via a socket. The application must wait for the CSR_IP_SOCKET_UDP_DATA_CFM before sending another CSR_IP_SOCKET_UDP_DATA_REQ.

CSR_IP_SOCKET_UDP_DATA_IND indicates that data has been received and is ready to be read by the application.

Parameters

Type	CSR_IP_SOCKET_UDP_DATA_REQ/CFM/IND
socketHandle	A unique handle for the socket
ipAddress	For data requests, this field contains the destination IP address to send the data to. For data indications, the source IP address from which the data was sent. Interpretation of this field depends on the socket family.
port	For data requests, this field contains the destination port. For data indications, this field contains the source port the data originates from.
dataLength	Length of data to be sent
*data	Pointer to data to be sent

3.6 CSR_IP_SOCKET_UDP_CLOSE

Primitives \ Parameters	type	socketHandle
CSR_IP_SOCKET_UDP_CLOSE_REQ	✓	✓

Table 6: CSR_IP_SOCKET_UDP_CLOSE Primitives

Description

Close a UDP socket.

Parameters

type	CSR_IP_SOCKET_UDP_CLOSE_REQ
socketHandle	A unique handle for the socket

3.7 CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE

Parameters / Primitives	type	socketHandle	group	result
CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE_REQ	✓	✓	✓	
CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE _CFM	✓	✓	✓	✓

Table 7: CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE Primitives

Description

Subscribe to a multicast group.

Parameters

type	CSR_IP_SOCKET_UDP_MULTICAST_SUBSCRIBE_REQ/CFM
socketHandle	A unique handle for the socket
interfaceIp	IP address of the interface to join the group on. Interpretation of this field depends on the socket family.
group	The multicast group to subscribe to. Interpretation of this field depends on the socket family.
result	Result code showing whether the request was fulfilled

3.8 CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE

Primitives \ Parameters	type	socketHandle	group	result
CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_REQ	✓	✓	✓	
CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_CFM	✓	✓	✓	✓

Table 8: CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE Primitives

Description

Unsubscribe from a multicast group.

Parameters

type	CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_REQ/CFM
socketHandle	A unique handle for the socket
interfaceIp	IP address of the interface to leave the group on. Interpretation of this field depends on the socket family.
group	The multicast group to unsubscribe from. Interpretation of this field depends on the socket family.
result	Result code showing whether the request was fulfilled

3.9 CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE

Primitives \ Parameters	type	socketHandle	interfaceIp	result
CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE_REQ	✓	✓	✓	
CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE_CFM	✓	✓		✓

Table 9: CSR_IP_SOCKET_UDP_MULTICAST_INTERFACE Primitives

Description

Set default network interface to use for multicast transmissions on the given socket.

Parameters

type	CSR_IP_SOCKET_UDP_MULTICAST_UNSUBSCRIBE_REQ/CFM
socketHandle	A unique handle for the socket
interfaceIp	IP address of the interface to use. Interpretation of this field depends on the socket family.
result	Result code showing whether the request was fulfilled

3.10 CSR_IP_SOCKET_TCP_NEW

Primitives \ Parameters	type	appHandle	socketFamily	socketHandle	result	nagle
CSR_IP_SOCKET_TCP_NEW_REQ	✓	✓	✓			✓
CSR_IP_SOCKET_TCP_NEW_CFM	✓			✓	✓	

Table 10: CSR_IP_SOCKET_TCP_NEW Primitives

Description

Create a new TCP socket and receive a handle for that socket.

Parameters

type	CSR_IP_SOCKET_TCP_NEW_REQ /CFM
appHandle	The identity of the calling task.
socketFamily	CSR_IP_SOCKET_FAMILY_IP4 or CSR_IP_SOCKET_FAMILY_IP6
socketHandle	A unique socket handle that is used subsequent socket communication
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE, CSR_IP_SOCKET_RESULT_NO_MORE_SOCKETS, or CSR_IP_SOCKET_RESULT_IP6_NOT_SUPPORTED
nagle	CSR_IP_SOCKET_NAGLE_PREFERRED or CSR_IP_SOCKET_NAGLE_DISABLED

3.11 CSR_IP_SOCKET_TCP_BIND

Parameters / Primitives	type	socketHandle	ipAddress	port	result
CSR_IP_SOCKET_TCP_BIND_REQ	✓	✓	✓	✓	
CSR_IP_SOCKET_TCP_BIND_CFM	✓	✓		✓	✓

Table 11: CSR_IP_SOCKET_TCP_BIND Primitives

Description

Bind an IP address and port to a TCP socket.

Parameters

type	CSR_IP_SOCKET_TCP_BIND_REQ/CFM
socketHandle	A unique handle for the socket
ipAddress	IP address to bind to. Interpretation of this field depends on the socket family.
port	The port to bind to. If port is zero, the network stack picks a free port and informs the application of the port in the confirmation signal.
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_SOCKET_RESULT_PORT_IN_USE

3.12 CSR_IP_SOCKET_TCP_LISTEN

Primitives \ Parameters	type	socketHandle	result
CSR_IP_SOCKET_TCP_LISTEN_REQ	✓	✓	
CSR_IP_SOCKET_TCP_LISTEN_CFM	✓	✓	✓

Table 12: CSR_IP_SOCKET_TCP_LISTEN Primitives

Description

Listen on a TCP socket.

Parameters

type	CSR_IP_SOCKET_TCP_LISTEN_REQ/CFM
socketHandle	A unique handle for the socket
result	CSR_RESULT_SUCCESS or CSR_RESULT_FAILURE.

3.13 CSR_IP_SOCKET_TCP_CONNECT

Parameters / Primitives	type	socketHandle	ipAddress	port	result
CSR_IP_SOCKET_TCP_CONNECT_REQ	✓	✓	✓	✓	
CSR_IP_SOCKET_TCP_CONNECT_CFM	✓	✓			✓

Table 13: CSR_IP_SOCKET_TCP_CONNECT Primitives

Description

Connect to a remote IP address and port. Failure to connect also means the socket is closed.

Parameters

type	CSR_IP_SOCKET_TCP_CONNECT_REQ/CFM
socketHandle	A unique handle for the socket
ipAddress	Remote IP address to connect to. Interpretation of this field depends on the socket family.
port	Remote port to connect to
result	CSR_RESULT_SUCCESS or CSR_RESULT_FAILURE.

3.14 CSR_IP_SOCKET_TCP_ACCEPT

Primitives \ Parameters	type	socketHandle	ipAddress	port
CSR_IP_SOCKET_TCP_ACCEPT_IND	✓	✓	✓	✓

Table 14: CSR_IP_SOCKET_TCP_ACCEPT Primitives

Description

An incoming connection has been accepted from IP address and port.

Parameters

type	CSR_IP_SOCKET_TCP_ACCEPT_ IND
socketHandle	A unique handle for the socket
ipAddress	IP address of the connecting client. Interpretation of this field depends on the listening socket family.
port	Port number of the connecting client

3.15 CSR_IP_SOCKET_TCP_DATA

Primitives \ Parameters	type	socketHandle	dataLength	*data	result
CSR_IP_SOCKET_TCP_DATA_REQ	✓	✓	✓	✓	
CSR_IP_SOCKET_TCP_DATA_CFM	✓	✓			✓
CSR_IP_SOCKET_TCP_DATA_RES	✓	✓			
CSR_IP_SOCKET_TCP_DATA_IND	✓	✓	✓	✓	

Table 15: CSR_IP_SOCKET_TCP_DATA Primitives

Description

CSR_IP_SOCKET_TCP_DATA_REQ is used for sending data on a TCP socket. The application must wait for the CSR_IP_SOCKET_TCP_DATA_CFM before sending another CSR_IP_SOCKET_TCP_DATA_REQ.

CSR_IP_SOCKET_TCP_DATA_IND indicates that data has been received and is ready to be read by the application. No further CSR_IP_SOCKET_TCP_DATA_IND will be sent to the application until the application has responded with a CSR_IP_SOCKET_TCP_DATA_RES.

Parameters

type	CSR_IP_SOCKET_TCP_DATA_REQ/CFM/IND/RES
socketHandle	A unique handle for the socket
dataLength	Length of the data to be send
*data	Pointer to the data to be send
result	CSR_RESULT_SUCCESS or CSR_RESULT_FAILURE.

3.16 CSR_IP_SOCKET_TCP_CLOSE

Primitives \ Parameters	type	socketHandle
CSR_IP_SOCKET_TCP_CLOSE_REQ	✓	✓
CSR_IP_SOCKET_TCP_CLOSE_IND	✓	✓

Table 16: CSR_IP_SOCKET_TCP_CLOSE Primitives

Description

CSR_IP_SOCKET_TCP_CLOSE_REQ closes a TCP socket and CSR_IP_SOCKET_TCP_CLOSE_IND indicates that a socket was closed by remote side. No confirmation signal is sent for CSR_IP_SOCKET_TCP_CLOSE_REQ.

Parameters

Type	CSR_IP_SOCKET_TCP_CLOSE_REQ/IND
socketHandle	A unique handle for the socket

3.17 CSR_IP_SOCKET_TCP_ABORT_REQ

Primitives \ Parameters	type	socketHandle
CSR_IP_SOCKET_TCP_ABORT_REQ	✓	✓

Table 17: CSR_IP_SOCKET_TCP_ABORT Primitives

Description

CSR_IP_SOCKET_TCP_ABORT_REQ abort the connection to the remote side.

Parameters

Type	CSR_IP_SOCKET_TCP_ABORT_REQ
socketHandle	A unique handle for the socket

3.18 CSR_IP_SOCKET_RAW_NEW

Primitives \ Parameters	type	appHandle	protocolNumber	socketHandle	result
CSR_IP_SOCKET_RAW_NEW_REQ	✓	✓	✓		
CSR_IP_SOCKET_RAW_NEW_CFM	✓			✓	✓

Table 18: CSR_IP_SOCKET_RAW_NEW Primitives

Description

Creates a RAW socket and receive a handle for it.

Parameters

type	CSR_IP_SOCKET_RAW_NEW_REQ/CFM
appHandle	The identity of the calling task.
protocolNumber	The numeric value to use in the protocol field in the IP header
socketHandle	A unique handle for the socket
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE, CSR_IP_SOCKET_RESULT_NO_MORE_SOCKETS, or CSR_IP_SOCKET_RESULT_IP6_NOT_SUPPORTED.

3.19 CSR_IP_SOCKET_RAW_BIND

Parameters / Primitives	type	socketHandle	ipAddress	result
CSR_IP_SOCKET_RAW_BIND_REQ	✓	✓	✓	
CSR_IP_SOCKET_RAW_BIND_CFM	✓	✓		✓

Table 19: CSR_IP_SOCKET_RAW_BIND Primitives

Description

Bind an IP address to a RAW socket.

Parameters

type	CSR_IP_SOCKET_RAW_BIND_REQ/CFM
socketHandle	A unique handle for the socket
ipAddress	IP address to bind to. Interpretation of this field depends on the socket family.
result	CSR_RESULT_SUCCESS or CSR_RESULT_FAILURE

3.20 CSR_IP_SOCKET_RAW_DATA

Parameters / Primitives	type	socketHandle	ipAddress	dataLength	*data
CSR_IP_SOCKET_RAW_DATA_REQ	✓	✓	✓	✓	✓
CSR_IP_SOCKET_RAW_DATA_IND	✓	✓	✓	✓	✓
CSR_IP_SOCKET_RAW_DATA_CFM	✓	✓			

Table 20: CSR_IP_SOCKET_RAW_DATA Primitives

Description

CSR_IP_SOCKET_RAW_DATA_REQ is used for sending data on a RAW socket. The application must wait for the CSR_IP_SOCKET_RAW_DATA_CFM before sending another CSR_IP_SOCKET_RAW_DATA_REQ.

CSR_IP_SOCKET_RAW_DATA_IND indicates that data has been received and is ready to be read by the application.

Parameters

type	CSR_IP_SOCKET_RAW_DATA_REQ/IND/CFM
socketHandle	A unique handle for the socket
ipAddress	For data requests, this field contains the destination IP address to send the data to. For data indications, the source IP address from which the data was sent. Interpretation of this field depends on the socket family.
dataLength	Length of data to be send
*data	Pointer to the data to be send

3.21 CSR_IP_SOCKET_RAW_CLOSE_REQ

Primitives \ Parameters	type	socketHandle
CSR_IP_SOCKET_RAW_CLOSE_REQ	✓	✓

Table 21: CSR_IP_SOCKET_RAW_CLOSE_REQ Primitive

Description

CSR_IP_SOCKET_RAW_DATA_REQ closes a RAW socket.

Parameters

type	CSR_IP_SOCKET_RAW_CLOSE_REQ
socketHandle	A unique handle for the socket

3.22 CSR_IP_SOCKET_OPTIONS

Primitives \ Parameters	type	socketHandle	txWindow	rxWindow	nagle	keepAlive	dscp	validOptions
CSR_IP_SOCKET_OPTIONS_REQ	✓	✓	✓	✓	✓	✓	✓	✓
CSR_IP_SOCKET_OPTIONS_CFM	✓	✓	✓	✓	✓	✓	✓	✓

Table 22: CSR_IP_SOCKET_OPTIONS Primitives

Description

CSR_IP_SOCKET_OPTIONS_REQ is used to set certain socket options listed in the table below:

nagle	Controls whether Nagle's algorithm is enabled for TCP-sockets.
keepAlive	Controls whether the IP stack should use TCP keepalive packets to detect connection loss.
broadcast	Controls whether the socket can send and receive broadcast datagrams.
dscp	Set the Differentiated Services Code Point bits.
rxWindow txWindow	Receive and transmit buffer sizes in bytes. An IP stack may use these as input to the TCP window announcements for the given socket.

Setting the options is not guaranteed to complete because the IP stack may not support the requested options or (e.g. keepalives) have the resources required (e.g. for socket buffers). If the application requests to have socket buffers increased to a certain value and the IP stack cannot fulfill this request, it is legal for the IP stack to ignore the request or increase the socket buffer size to less than what was requested.

The confirmation returns the current settings for the given socket. If an IP stack does not support a particular option, the value returned for the option is undefined and the corresponding flag in `validOptions` MUST be cleared.

Parameters

type	CSR_IP_SOCKET_OPTIONS_REQ/CFM
socketHandle	A unique handle for the socket
txWindow	Transmission buffer size in byte
rxWindow	Reception buffer size in byte
nagle	Nagle's algorithm enabled
keepAlive	Use TCP keepalives
broadcast	Enable broadcast transmission and reception
dscp	DSCP value
validOptions	A bit mask indicating which of the above values are valid

3.23 CSR_IP_SOCKET_DNS_RESOLVE_NAME

Primitives \ Parameters	type	appHandle	socketFamilyMax	socketFamily	*name	ipAddress	result
CSR_IP_SOCKET_DNS_RESOLVE_NAME_REQ	✓	✓	✓		✓		
CSR_IP_SOCKET_DNS_RESOLVE_NAME_CFM	✓			✓	✓	✓	✓

Description

Resolve a DNS name given by the application and return the IP address to the application. It is possible to specify whether to attempt to obtain an IPv6 address or only an IPv4 address. The confirmation signal specifies which type of address is returned, and this value can be used directly as the socket family type in a request for a new socket to make it possible to write IP protocol agnostic code.

Parameters

Type	CSR_IP_SOCKET_DNS_RESOLVE_NAME_REQ/CFM
appHandle	The identity of the calling task.
socketFamilyMax	The maximum socket family requested – either CSR_IP_SOCKET_FAMILY_IP4 or CSR_IP_SOCKET_FAMILY_IP6. If CSR_IP_SOCKET_FAMILY_IP6 is given, an IPv4 address may be returned only if there is no IPv6 record (AAAA) for the given name. If CSR_IP_SOCKET_FAMILY_IP4 is given, only IPv4 addresses may be returned.
socketFamily	Returns the socket type that corresponds to the type of address returned – either CSR_IP_SOCKET_FAMILY_IP4.
*name	DNS name to be resolved
ipAddress	Resolved IP address
result	CSR_RESULT_SUCCESS or CSR_RESULT_FAILURE

4 Document References

Ref	Title
[IP_ARCH]	gu-0002-ip-architecture

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
BSD	Berkeley Software Distribution
CSR	Cambridge Silicon Radio
DNS	Domain Name System
IP	Internet Protocol
MSC	Message Sequence Chart
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table 23: Abbreviations and Definitions

Document History

Revision	Date	History
1	22 FEB 2009	Initial revision
2	30 NOV 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	DEC 10	Ready for release 3.0.0
5	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.