ssI14229 Client User's Manual

Created by the UDS Experts!
Version 1.0
Revised June 17th, 2014

ssI14229 Protocol Stack License

READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY BEFORE OPENING THE PACKAGE CONTAINING THE PROGRAM DISTRIBUTION MEDIA (DISKETTES, CD, ELECTRONIC MAIL), THE COMPUTER SOFTWARE THEREIN, AND THE ACCOMPANYING USER DOCUMENTATION. THIS SOURCE CODE IS COPYRIGHTED AND LICENSED (NOT SOLD). BY OPENING THE PACKAGE CONTAINING THE SOURCE CODE, YOU ARE ACCEPTING AND AGREEING TO THE TERMS OF THIS LICENSE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY THE TERMS OF THIS LICENSE AGREEMENT, YOU SHOULD PROMPTLY RETURN THE PACKAGE IN UNOPENED FORM, AND YOU WILL RECEIVE A REFUND OF YOUR MONEY. THIS LICENSE AGREEMENT REPRESENTS THE ENTIRE AGREEMENT CONCERNING THE ssI14229 PROTOCOL STACK BETWEEN YOU AND SIMMA SOFTWARE, INC. (REFERRED TO AS "LICENSOR"), AND IT SUPERSEDES ANY PRIOR PROPOSAL, REPRESENTATION, OR UNDERSTANDING BETWEEN THE PARTIES.

1. Corporate License Grant. Simma Software hereby grants to the purchaser (herein referred to as the “Client”), a royalty free, non-exclusive license to use the ssI14229 protocol stack source code (collectively referred to as the "Software”) as part of Client’s product. Except as provided above, Client agrees to not assign, sublicense, transfer, pledge, lease, rent, or share the Software Code under this License Agreement.

2. Simma Software's Rights. Client acknowledges and agrees that the Software and the documentation are proprietary products of Simma Software and are protected under U.S. copyright law. Client further acknowledges and agrees that all right, title, and interest in and to the Software, including associated intellectual property rights, are and shall remain with Simma Software. This License Agreement does not convey to Client an interest in or to the Software, but only a limited right of use revocable in accordance with the terms of this License Agreement.

3. License Fees. The Client in consideration of the licenses granted under this License Agreement will pay a one-time license fee.

4. Term. This License Agreement shall continue until terminated by either party. Client may terminate this License Agreement at any time. Simma Software may terminate this License Agreement only in the event of a material breach by Client of any term hereof, provided that such shall take effect 60 days after receipt of a written notice from Simma Software of such termination and further provided that such written notice allows 60 days for Client to cure such breach and thereby avoid termination. Upon termination of this License Agreement, all rights granted to Client will terminate and revert to Simma Software. Promptly upon termination of this Agreement for any reason or upon discontinuance or abandonment of Client’s possession or use of the Software, Client must return or destroy, as requested by Simma Software, all copies of the Software in Client’s possession, and all other materials pertaining to the Software (including all copies thereof). Client agrees to certify compliance with such restriction upon Simma Software’s request.

5. Limited Warranty. Simma Software warrants, for Client’s benefit alone, for a period of one year (called the “Warranty Period”) from the date of delivery of the software, that during this period the Software shall operate substantially in accordance with the functionality described in the User's Manual. If during the Warranty Period, a defect in the Software appears, Simma Software will make all reasonable efforts to cure the defect, at no cost to the Client. Client agrees that the foregoing constitutes Client’s sole and exclusive remedy for breach by Simma Software of any warranties made under this Agreement. Simma Software is not responsible for obsolescence of the Software that may result from changes in Client’s requirements. The foregoing warranty shall apply only to the most current version of the Software issued from time to time by Simma Software. Simma Software assumes no responsibility for the use of superseded, outdated, or uncorrected versions of the licensed software. EXCEPT FOR THE WARRANTIES SET FORTH ABOVE, THE SOFTWARE, AND THE SOFTWARE CONTAINED THEREIN, ARE LICENSED "AS IS," AND SIMMA SOFTWARE DISCLAIMS ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

6. Limitation of Liability. Simma Software's cumulative liability to Client or any other party for any loss or damages resulting from any claims, demands, or actions arising out of or relating to this License Agreement shall not exceed the license fee paid to Simma Software for the use of the Software. In no event shall Simma Software be liable for any indirect, incidental, consequential, special, or exemplary damages or lost profits, even if Simma Software has been advised of the possibility of such damages. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO CLIENT.

7. Governing Law. This License Agreement shall be construed and governed in accordance with the laws of the State of Indiana.

8. Severability. Should any court of competent jurisdiction declare any term of this License Agreement void or unenforceable, such declaration shall have no effect on the remaining terms hereof.

9. No Waiver. The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breach

TABLE OF CONTENTS

Chapter 1: Introduction

ssI14229 is a high performance ISO 14229 (UDS) protocol stack written in ANSI C. It adheres to both the ISO 14229 specification and to the software development best practices described in the MISRA C guidelines.

The ssI14229 protocol stack is a modularized design with an emphasis on software readability and performance. It is easy to understand and platform independent allowing it to be used on any CPU or DSP with or without an RTOS.

Filenames	File Description
uds.h	Core header file. Do not modify.
uds.c	Core source file. Do not modify.
udsapp.h	Application header file. Modification allowed.
udsapp.c	Application source file. Modification allowed.

Table 1-1: ssI14229 files

Chapter 2: Integration of ssI14229

This chapter describes how to integrate ssI14229 into your application. After this is complete, you will be able to receive and transmit ISO 14229 messages over CAN. For implementation details, please see the chapters covering the API for ssI14229.

Integration Steps:

1. Purchase a CAN device driver and integrate it in accordance with the integration specifications for ssI15765.

2. Implement the udsapp_process_seed method as required to process the seed into the key for creating a secured session.

3. As needed adjust the number and size of the message buffer.

Chapter 3: ssI14229 API

This chapter describes the application program interface (API) for the ssI14229 module.

Function Prototypes	Function Descriptions
void uds_init (void)	Initializes protocol stack
void udsapp_init (void)	Called on startup
void uds_update (void)	Provides periodic time base
void udsapp_update (void)	Called at periodic tick rate
void udsapp_process (i15765_t *msg)	Processes received messages.
void udsapp_nrsp_process (uint8_t svcid, uint8_t nrsp)	Processes negative responses
void uds_req_diagnostic_session (uint8_t func, uint8_t sprsp)	Requests a higher level diagnostic session
void uds_req_ecu_reset (uint8_t func, uint8_t sprsp)	Requests a reset of the ECU
void uds_req_security_access (uint8_t func, uint8_t sprsp, uint8_t *key, uint8_t klen)	Requests a secure session
void uds_req_communication_control (uint8_t func, uint8_t sprsp, uint8_t type)	Enable/Disable certain messages
void uds_req_tester_present (uint8_t sprsp)	Inform server that a tester is present
void uds_req_access_timing_parameter (uint8_t func, uint8_t sprsp, uint8_t *key, uint8_t klen)	Read and change link timing
void uds_req_secured_data_transmission (uint8_t *sd, uint8_t sdlen);	Transmit data in a secure manner
void uds_req_control_dtc_setting (uint8_t func, uint8_t sprsp, uint8_t *key, uint8_t klen)	Halt/resume setting of DTCs
void uds_req_response_on_event (uint8_t evntype, uint8_t wintime, uint8_t *rec, uint8_t reclen, uint8_t *rsp, uint8_t rsplen)	Automatically respond to certain events with a defined request
void uds_req_link_control_predef (uint8_t sprsp, uint8_t baud)	Check to see if transition of link baud rate to predefined rate is possible
void uds_req_link_control_user (uint8_t sprsp, uint32_t baud)	Check to see if transition of link baud rate to specified rate is possible
void uds_req_link_control (uint8_t func, uint8_t sprsp)	Transition to previously discussed rate
void uds_req_read_data_by_id (uint16_t *did, uint8_t dlen)	Reads data by defined dataIdentifier
void uds_req_read_memory_by_address (uint32_t addr, uint16_t size)	Read memory by address
void uds_req_read_scaling_by_id (uint16_t id)	Read scaling data by dataIdentifier
void uds_req_read_data_by_periodic_id (uint8_t txmode, uint16_t *did, uint8_t dlen)	Read data by periodic identifier

void uds_req_dynamically_define_by_data_id (uint8_t sprsp, uint16_t ddid, uint16_t *sdid, uint8_t *pos, uint8_t *msize, uint8_t len)	Define a dataIdentifier for reading
void uds_req_dynamically_define_by_memory_address (uint8_t sprsp, uint16_t ddid, uint32_t *addr, uint16_t *size, uint8_t len)	Define a dataIdentifier using a memory address for reading
void uds_req_clear_dynamically_defined_data_id (uint8_t sprsp, uint16_t ddid)	Clear a dynamically defined dataIdentifier
void uds_req_write_data_by_id (uint16_t did, uint8_t *drec, uint8_t len)	Write data specified by dataIdentifier
void uds_req_write_memory_by_address (uint32_t addr, uint16_t size, uint8_t *drec)	Write data to memory by address
void uds_req_clear_diagnostic_information (uint32_t dtc)	Clear DTC information
void uds_req_read_dtc_info_mask (uint8_t func, uint8_t sprsp, uint8_t mask)	Read DTC info by status mask
void uds_req_read_dtc_info_report_snapshot_id (uint8_t sprsp)	Request report of all DTCSnapshots
void uds_req_read_dtc_info_report_snapshot_by_dtc (uint8_t sprsp, uint32_t mask, uint8_t recnum)	Request report of DTCSnapshots related to specified DTC
void uds_req_read_dtc_info_report_snapshot_by_record (uint8_t sprsp, uint8_t recnum)	Read data from specified DTCSnapshot
void uds_req_read_dtc_info_edr_dtc (uint8_t func, uint8_t sprsp, uint32_t mask, uint8_t recnum)	Read extended data about specified DTC
void uds_req_read_dtc_info_severity_info(uint8_t func, uint8_t sprsp, uint16_t mask)	Request list of DTCs which match specified severity level
void uds_req_read_dtc_info_severity_dtc (uint8_t sprsp, uint32_t mask)	Read severity info of specified DTC
void uds_req_read_dtc_info_misc (uint8_t func, uint8_t sprsp)	Catchall function for remaining DTC info sub-functions
void uds_req_io_control (uint16_t did, uint8_t *opt, uint8_t optlen, uint8_t *mask, uint8_t masklen)	Emulate value for input/output signal
void uds_req_routine_control (uint8_t rtype, uint16_t rid, uint8_t *ropt, uint8_t optlen)	Start/stop stored routine
void uds_req_download (uint8_t dfmtid, uint8_t *maddr, uint8_t maddrlen, uint8_t *msize, uint8_t msizelen)	Request a download to the server
void uds_req_upload (uint8_t dfmtid, uint8_t *maddr, uint8_t maddrlen, uint8_t *msize, uint8_t msizelen)	Request an upload from the server
int uds_req_transfer_data (uint8_t blckcnt, uint8_t *txparam, uint8_t paramlen)	Transfer data after request has been made
void uds_req_transfer_exit (uint8_t *sd, uint8_t sdlen)	Exit data transfer mode
void uds_create_secured_session (uint8_t level)	Automatically create secured session
void udsapp_process_seed (uint8_t *seed, uint16_t *seedlen)	Processes seed into key for secure session creation

Table 3-1: API functions

3.1 uds_init

Function Prototype:

```
void uds_init (
    void
);
```

Description:

Initializes the ssI14229 module.

Parameters:

void

Return Value:

void

3.2 udsapp_init

Function Prototype:

```
void udsapp_init (
    void
);
```

Description:

Initializes the ssI14229 software. The implementation of this method is left for the user to add any necessary functions.

Parameters:

void

Return Value:

void

3.3 uds_update

Function Prototype:

```
void uds_update (
    void
);
```

Description:

Provides the periodic time base for the ssI14229 module.

Parameters:

void

Return Value:

void

3.4 udsapp_update

Function Prototype:

```
void udsapp_update (
    void
);
```

Description:

Provides a regularly called method stub for the user to add methods, which need to be called regularly.

Parameters:

void

Return Value:

void

3.5 udsapp_process

Function Prototype:

```
void udsapp_process (
    i15765_t *msg
);
```

Description:

Processes received UDS messages. This function is called by the ssI14229 module with a complete message and is the intended location for the application to handle received messages.

For multi-frame messages this function isn't called until all frames have been received and assembled into a valid and complete message.

Parameters:

msg: Pointer to the received message

Return Value:

void

3.6 udsapp_nrsp_process

Function Prototype:

```
void udsapp_nrsp_process (
    uint8_t svcid,
    uint8_t nrsp
);
```

Description:

Processes negative response messages from the server.

Parameters:

svcid: Service Id of failed service request
nrsp: Negative response code

Return Value:

void

3.7 uds_req_diagnostic_session

Function Prototype:

```
void uds_req_diagnostic_session (
    uint8_t func,
    uint8_t sprsp
);
```

Description:

This service requests a diagnostic session from the server.

Parameters:

func: Sub-function code
sprsp: Suppress response from server

Return Value:

void

3.8 uds_req_ecu_reset

Function Prototype:

```
void uds_req_ecu_reset (
    uint8_t func,
    uint8_t sprsp
);
```

Description:

The ECUReset service is used by the client to request a server reset.

Parameters:

func: Sub-function code
sprsp: Suppress response from server

Return Value:

void

3.9 uds_req_security_access

Function Prototype:

```
void uds_req_security_access (
   uint8_t func,
   uint8_t sprsp,
   uint8_t *key,
   uint8_t klen
);
```

Description:

The purpose of this service is to provide a means to access data and/or diagnostic services which have restricted access for security, emissions or safety reasons.

Parameters:

func: Sub-function code
sprsp: Suppress response from server
key: Pointer to key for authorization
klen: Length of key array in bytes

Return Value:

void

3.10 uds_req_communication_control

Function Prototype:

```
void uds_req_communication_control (
    uint8_t func,
    uint8_t sprsp,
    uint8_t type
);
```

Description:

The purpose of this service is to switch on/off the transmission and/or the reception of certain messages of (a) server(s).

Parameters:

func: Sub-function code
sprsp: Suppress response from server
type: Reference to the kind of communication to be controlled

Return Value:

void

3.11 uds_req_tester_present

Function Prototype:

```
void uds_req_tester_present (
    uint8_t sprsp
);
```

Description:

This service is used to indicate to a server (or servers) that a client is still connected to the vehicle and that certain diagnostic services and/or communications that have been previously activated remain active.

Parameters:

sprsp: Suppress response from server

Return Value:

void

3.12 uds_req_access_timing_parameter

Function Prototype:

```
void uds_req_access_timing_parameter (
    uint8_t func,
    uint8_t sprsp,
    uint8_t *key,
    uint8_t klen
);
```

Description:

This service is used to read and change the default timing parameters of a communication link for the duration that this communication link is active.

Parameters:

func: Sub-function code
sprsp: Suppress response from server
*key: Timing parameter values to be set in the server
klen: Length of key array in bytes

Return Value:

void

3.13 uds_req_secured_data_transmission

Function Prototype:

```
void uds_req_secured_data_transmission (
    uint8_t *sd,
    uint8_t sdlen
);
```

Description:

The purpose of this service is to transmit data that is protected against attacks from third parties, which could endanger data security, according to ISO 15764.

Parameters:

sd: Data to be transferred in a secure manner
sdlen: Length of sd in bytes

Return Value:

void

3.14 uds_req_control_dtc_setting

Function Prototype:

```
void uds_req_control_dtc_setting (
    uint8_t func,
    uint8_t sprsp,
    uint8_t *key,
    uint8_t klen
);
```

Description:

This service shall be used by a client to stop or resume the setting of diagnostic trouble codes (DTCs) in the server(s).

Parameters:

func: Sub-function code
sprsp: Suppress response from server
*key: User optional argument, may contain list of DTCs to be turned on or off.
klen: Length of key array in bytes

Return Value:

void

3.15 uds_req_response_on_event

Function Prototype:

```
void uds_req_response_on_event (
    uint8_t evntype,
    uint8_t wintime,
    uint8_t *rec,
    uint8_t reclen,
    uint8_t *rsp,
    uint8_t rsplen,
);
```

Description:

This service requests a server to execute a given service in responses to a specified event.

Parameters:

evntype: Type of event to respond to
wintime: Amount of time the response logic should persist
*rec: Additional parameters for given response type
reclen: Length of rec array in bytes
*rsp: Service parameters of the service to be executed in response
rsplen: Length of rsp array in bytes

Return Value:

void

3.16 uds_req_link_control_predef

Function Prototype:

```
void uds_req_link_control_predef (
    uint8_t sprsp,
    uint8_t baud,
);
```

Description:

This service is used to control the communication link baud rate between the client and the server(s) for exchange of diagnostic data. This service optionally applies to those data link layers, which allow for a baud rate transition during an active diagnostic session. This function specifically verified the ability to transition to a baud rate, which has been pre-defined on the server.

Parameters:

sprsp: Suppress response from server
baud: Identifier of pre-defined baud rate to switch to

Return Value:

void

3.17 uds_req_link_control_user

Function Prototype:

```
void uds_req_link_control_user (
    uint8_t sprsp,
    uint32_t baud,
);
```

Description:

This service is used to control the communication link baud rate between the client and the server(s) for exchange of diagnostic data. This service optionally applies to those data link layers, which allow for a baud rate transition during an active diagnostic session. This function verifies the ability to transition to a baud rate, which is user-specified in the message.

Parameters:

sprsp: Suppress response from server
baud: Baud rate to be switched to

Return Value:

void

3.18 uds_req_link_control

Function Prototype:

```
void uds_req_link_control (
    uint8_t func,
    uint8_t sprsp
);
```

Description:

This service is used to control the communication link baud rate between the client and the server(s) for exchange of diagnostic data. This service optionally applies to those data link layers, which allow for a baud rate transition during an active diagnostic session. This function transitions to the new baud rate, which was verified in the previous command.

Parameters:

func: Sub-function code (most likely 3)
sprsp: Suppress response from server

Return Value:

void

3.19 uds_req_read_data_by_id

Function Prototype:

```
void uds_req_read_data_by_id (
    uint16_t *did,
    uint8_t dlen,
);
```

Description:

This service allows the client to request data record values from the server as identified by one or more dataIdentifiers.

Parameters:

*did: Array of dataIdentifiers
dlen: Number of dataIdentifiers

Return Value:

void

3.20 uds_req_read_memory_by_address

Function Prototype:

```
void uds_req_read_memory_by_address (
    uint32_t addr,
    uint16_t size
);
```

Description:

This service allows the client to request memory data from the sever via a provided starting address and to specify the size of memory to be read.

Parameters:

addr: Address to start reading from
size: Size of memory block to be read

Return Value:

void

3.21 uds_req_read_scaling_by_id

Function Prototype:

```
void uds_req_scaling_by_id(
    uint16_t id
);
```

Description:

This service allows the client to request scaling data record information from the server identified by a dataIdentifier.

Parameters:

id: dataIdentifier

Return Value:

void

3.22 uds_req_read_data_by_periodic_id

Function Prototype:

```
void uds_req_read_data_by_periodic_id (
    uint8_t txmode,
    uint16_t *did,
    uint8_t dlen
);
```

Description:

This service allows the client to request the periodic transmission of data record values from the server identified by one or more periodicDataIdentifiers.

Parameters:

txmode: Transmission rate of data to be passed
*did: One or more dataIdentifiers
dlen: Number of dataIdentifiers

Return Value:

void

3.23 uds_req_dynamically_define_by_data_id

Function Prototype:

```
void uds_req_dynamically_define_by_data_id (
    uint8_t sprsp,
    uint16_t ddid,
    uint16_t *sdid,
    uint8_t *pos,
    uint8_t *msize,
    uint8_t len
);
```

Description:

This service allows the client to dynamically define in a server a data identifier that can be read via the ReadDataByIdentifier service at a later time.

Parameters:

sprsp: Suppress response from this server to this message
ddid: dataIdentifer to be defined
*sdid: One or more dataIdentifiers which specifies the source to be loaded into the defined dataId
*pos: Reference to the starting byte position of the excerpt to be defined.
*msize: Number of bytes to be loaded into the defined dataId
len: Number of dataIdentifiers to be defined into ddid

Return Value:

void

3.24 uds_req_dynamically_define_by_memory_address

Function Prototype:

```
void uds_req_dynamically_define_by_memory_address (
    uint8_t sprsp,
    uint16_t ddid,
    uint32_t *addr
    uint8_t *size,
    uint8_t len
);
```

Description:

This service allows the client to dynamically define in a server a data identifier, using a memory address that can be read via the ReadDataByIdentifier service at a later time.

Parameters:

sprsp: Suppress response from this server to this message
ddid: dataIdentifer to be defined
*addr: One or more memory addresses which specifies the source to be loaded into the defined dataId
*size: Number of bytes to be loaded into the defined dataId
len: Number of dataIdentifiers to be defined

Return Value:

void

3.25 uds_req_clear_dynamically_defined_data_id

Function Prototype:

```
void uds_req_clear_dynamically_defined_data_id (
    uint8_t sprsp,
    uint16_t ddid
);
```

Description:

This service allows the client to clear a dynamically defined dataIdentifier.

Parameters:

sprsp: Suppress response from this server to this message
ddid: dataIdentifier to be cleared

Return Value:

void

3.26 uds_req_write_data_by_id

Function Prototype:

```
void uds_req_write_data_by_id (
    uint16_t did,
    uint8_t *drec,
    uint8_t len
);
```

Description:

This service allows the client to write information into the server at an internal location specified by the provided dataIdentifier.

Parameters:

did: dataIdentifer to write to
*drec: Data to be written
len: Number of bytes of data to be written

Return Value:

void

3.27 uds_req_write_memory_by_address

Function Prototype:

```
void uds_req_write_memory_by_address (
    uint32_t addr,
    uint16_t size,
    uint8_t *drec
);
```

Description:

This service allows the client to write information into the server at an internal location specified by the provided memory address.

Parameters:

addr: Starting memory address
size: Size of memory block to be written to
*drec: Data to be written

Return Value:

void

3.28 uds_req_clear_diagnostic_information

Function Prototype:

```
void uds_req_clear_diagnostic_information (
    uint32_t dtc
);
```

Description:

This service allows the client to clear diagnostic information from (a) server(s) memory.

Parameters:

dtc: Indicates the group of DTCs or individual DTC to be cleared.

Return Value:

void

3.29 uds_req_read_dtc_info_mask

Function Prototype:

```
void uds_req_read_dtc_info_mask (
    uint8_t func,
    uint8_t sprsp,
    uint8_t mask
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function supports the various sub-functions, which take a DTC status mask.

Parameters:

func: Sub-function code
sprsp: Suppress response from the server to this message
mask: Status to be matched to all DTCs

Return Value:

void

3.30 uds_req_read_dtc_info_report_snapshot_id

Function Prototype:

```
void uds_req_read_dtc_info_report_snapshot_id (
    uint8_t sprsp
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function reports all DTCSnaptshot data record identifications.

Parameters:

sprsp: Suppress response from the server to this message

Return Value:

void

3.31 uds_req_read_dtc_info_report_snapshot_by_dtc

Function Prototype:

```
void uds_req_read_dtc_info_report_snapshot_by_dtc (
    uint8_t sprsp,
    uint32_t mask,
    uint8_t recnum
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function returns all Snapshot records associated with a defined DTC number and Snapshot number.

Parameters:

sprsp: Suppress response from the server to this message
mask: Unique identifier of DTC
recnum: Client-defined Snapshot record number

Return Value:

void

3.32 uds_req_read_dtc_info_report_snapshot_by_record

Function Prototype:

```
void uds_req_read_dtc_info_report_snapshot_by_record (
    uint8_t sprsp,
    uint8_t recnum
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function returns only the Snapshot record associated with a defined Snapshot number.

Parameters:

sprsp: Suppress response from the server to this message
recnum: Client-defined Snapshot record number

Return Value:

void

3.33 uds_req_read_dtc_info_edr_dtc

Function Prototype:

```
void uds_req_read_dtc_info_report_edr_dtc (
    uint8_t func,
    uint8_t sprsp,
    uint32_t mask,
    uint8_t recnum
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function returns the Extended Data Record associated with a given DTC and Snapshot record number either from main memory (func = 0x06) or from the mirror memory (func = 0x10).

Parameters:

func: Sub-function code
sprsp: Suppress response from the server to this message
mask: Unique identifier of DTC
recnum: Client-defined Snapshot record number

Return Value:

void

3.34 uds_req_read_dtc_info_severity_info

Function Prototype:

```
void uds_req_read_dtc_info_report_info_severity_info (
    uint8_t func,
    uint8_t sprsp,
    uint16_t mask
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function returns either the number of DTCs which match the given severity level (func = 0x07), or a list of them (func = 0x08).

Parameters:

func: Sub-function code
sprsp: Suppress response from the server to this message
mask: Severity and Status masks

Return Value:

void

3.35 uds_req_read_dtc_info_severity_dtc

Function Prototype:

```
void uds_req_read_dtc_info_report_severity_dtc (
    uint8_t sprsp,
    uint32_t mask
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function returns the severity info of a specific DTC.

Parameters:

sprsp: Suppress response from the server to this message
mask: Unique identifier of DTC

Return Value:

void

3.36 uds_req_read_dtc_info_misc

Function Prototype:

```
void uds_req_read_dtc_info_misc (
    uint8_t func,
    uint8_t sprsp
);
```

Description:

This service allows the client to read Diagnostic Trouble Code (DTC) information from any server or group of servers within a vehicle. This function serves as a catch all for the remaining sub-functions of this service, which take no arguments.

Parameters:

func: Sub-function code
sprsp: Suppress response from the server to this message

Return Value:

void

3.37 uds_req_io_control

Function Prototype:

```
void uds_req_io_control (
    uint16_t did,
    uint8_t *opt,
    uint8_t optlen,
    uint8_t *mask,
    uint8_t masklen
);
```

Description:

This service allows the client to substitute a value for an input signal, internal server function, and/or control an output (actuator) of an electronic system.

Parameters:

did: DataIdentifier of signal or parameter to be substituted
*opt: All information needed to be substituted
optlen: Length of opt in bytes
*mask: Optional mask which might be used to only control specific bits.
masklen: Length of mask in bytes

Return Value:

void

3.38 uds_req_routine_control

Function Prototype:

```
void uds_req_routine_control (
    uint8_t rtype,
    uint16_t rid,
    uint8_t *ropt,
    uint8_t optlen
);
```

Description:

This service allows the client to start, stop, or request details about a routine stored on the ECU.

Parameters:

rtype: Sub-function: start, stop, request results.
rid: Unique identifier of a routine
*ropt: Optional additional arguments to be passed to the routine
optlen: Length of ropt in bytes.

Return Value:

void

3.39 uds_req_download

Function Prototype:

```
void uds_req_download (
    uint8_t dfmtid,
    uint8_t *maddr,
    uint8_t maddrlen,
    uint8_t *msize
    uint8_t msizelen
);
```

Description:

This service allows the client to request a data transfer from the client to the server.

Parameters:

dfmtid: Encryption and compression format
*maddr: Starting address for transfer destination
maddrlen: Length of maddr in bytes
*msize: Amount of data to be transfered
msizelen: Length of msize in bytes

Return Value:

void

3.40 uds_req_upload

Function Prototype:

```
void uds_req_upload (
    uint8_t dfmtid,
    uint8_t *maddr,
    uint8_t maddrlen,
    uint8_t *msize
    uint8_t msizelen
);
```

Description:

This service allows the client to request a data transfer from the server to the client.

Parameters:

dfmtid: Encryption and compression format
*maddr: Starting address for transfer source
maddrlen: Length of maddr in bytes
*msize: Amount of data to be transfered
msizelen: Length of msize in bytes

Return Value:

void

3.41 uds_req_transfer_data

Function Prototype:

```
uint8_t uds_req_transfer_data (
    uint8_t blckcnt,
    uint8_t *txparam,
    uint8_t paramlen
);
```

Description:

This service allows the client to transfer data either to or from the server. This function will only allow one message packet to be transmitted at a time, and will release this lock when: a positive response is received, a negative response is received, or a timeout with no response. If no response is received the previous message will be re-sent automatically.

Parameters:

blckcnt: Block counter for data transfer
*txparam: Specific parameters for data transfer, during download this contains the data to be transfered
paramlen: Length of txparam in bytes

Return Value:

0: If message was sent.
1: If the stack is currently waiting for a positive response to a previous transfer message.

3.42 uds_transfer_exit

Function Prototype:

```
void uds_req_transfer_exit (
    uint8_t *sd,
    uint8_t sdlen
);
```

Description:

This service allows the client to end an in progress data transfer.

Parameters:

*sd: Optional, specific parameters for ending data transfer
sdlen: Length of sd in bytes

Return Value:

void

3.43 uds_create_secured_session

Function Prototype:

```
void uds_create_secured_session (
    uint8_t level
);
```

Description:

This method automates the call-response seed-key exchange of the client and server. When udsapp_process_seed is properly configured, this method will automatically request a seed, process the returned seed, and send the key back to the server to create a secured session.

Parameters:

level: Level of secured session to be created.

Return Value:

void

3.44 udsapp_process_seed

Function Prototype:

```
void uds_create_secured_session (
    uint8_t *seed,
    uint16_t *seedlen,
);
```

Description:

This method processes the seed into the correct key to be used to create a secured session. This method must be user configured for the automated uds_create_secured_session method to run successfully. Seed and seedlen should be modified in place to become key and keylen respectively.

Parameters:

*seed: Seed returned by the server
*seedlen: Length of seed in bytes.

Return Value:

void

Chapter 4: Configuration

This chapter describes all configurable items of the ssI14229 module. All of these configurations are defined in uds.h. Remember to also configure the other stack layers per the applicable user manuals.

UDS Message Buffer Length

The message buffer is used in the assembly of variable length UDS messages. If any of these messages will be longer than the default of 50 bytes this option should be changed.

```
#define UDS_MSG_BUF_LEN                 50
```

UDS Transfer Retry Limit

If a transfer data message must be resent due to failure to receive a positive response this constant will define how many retries should be made.

```
#define UDS_TX_RETRY_MAX                1
```

Chapter 5: Examples

This chapter gives examples of how to decode and transmit I14229 messages.

5.1 Decode I14229 Messages Example:

```
void
udsapp_process ( i15765_t *msg )
{
   switch( msg->buf[0] ) {

     /* OK ECU Reset */
     case UDS_PRSP_ECURESET:
         printf("ECU is reseting.");
         break;


      /* add other messages here */
      case XXX: {
      }
   }
}
```

5.2 Decode I14229 Negative Response Example:

```
void
udsapp_nrsp_process ( uint8_t svcid, uint8_t nrsp )
{
   switch( nrsp ) {

     /* Engine is Running */
     case UDS_NRSP_EIR:
         if( svcid == UDS_SVCID_ECURESET )
              printf("ECU Reset Failed: Engine is running.");
         break;


      /* add other messages here */
      case XXX: {
      }
   }
}
```

5.3 Transmit I14229 Message Example:

```
/*** Transmit an ECU Reset Request.*/

void
udsapp_example ( void )
{
  /* Perform a soft reset */
  uds_req_ecu_reset( 3, 0 );
}
```