S@T 01.20 v3.0.0 (Release 2007)

S@T Session Protocol

SSP

Published by simalliance now Trusted Connectivity Alliance

1 TABLE OF CONTENTS

2 Terminology

DE	S@TML/SBC Decoder / Encoder
DTD	Document Type Definition
GSM	Global System for Mobile Communication
HTTP	Hyper Text Transfer Protocol
S@T	SIM Alliance Toolbox
SB	S@T Browser
SBC	S@T Byte Code
SSP	S@T Session Protocol
SIM	Subscriber Identity Module
S@TML	S@T Markup Language
STK	SIM Application Toolkit
TLV	Tag Length Value encoding
URI	Unified Resource Identifier
URL	Unified Resource Locator
WAP	Wireless Application Protocol
WTAI	Wireless Telephony Application Interface
XML	Extensible Markup Language

3 List of documents

[1] GSM 03.38: "Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information".

[2] GSM 11.11. "Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

[3] GSM 11.14. "Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM – ME) interface".

[4] GSM 03.48. "Digital cellular telecommunications system (Phase 2+); Security Mechanisms for the SIM Application Toolkit; Stage 2".

[5] GSM 03.40 "Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS)"

[6] SBC 1.00 "SBC, S@T ML Byte Code (Technical Specification S@T 01.00)"

[7] SBC 1.22 " S@T Browsing commands (Technical Specification S@T 01.22)"

[8] SBC 1.21 " S@T Administrative commands (Technical Specification S@T 01.21)"

This document is part of a specification set, please refer to "S@T Release Note" for a comprehensive document list, including document versions.

4 Overview

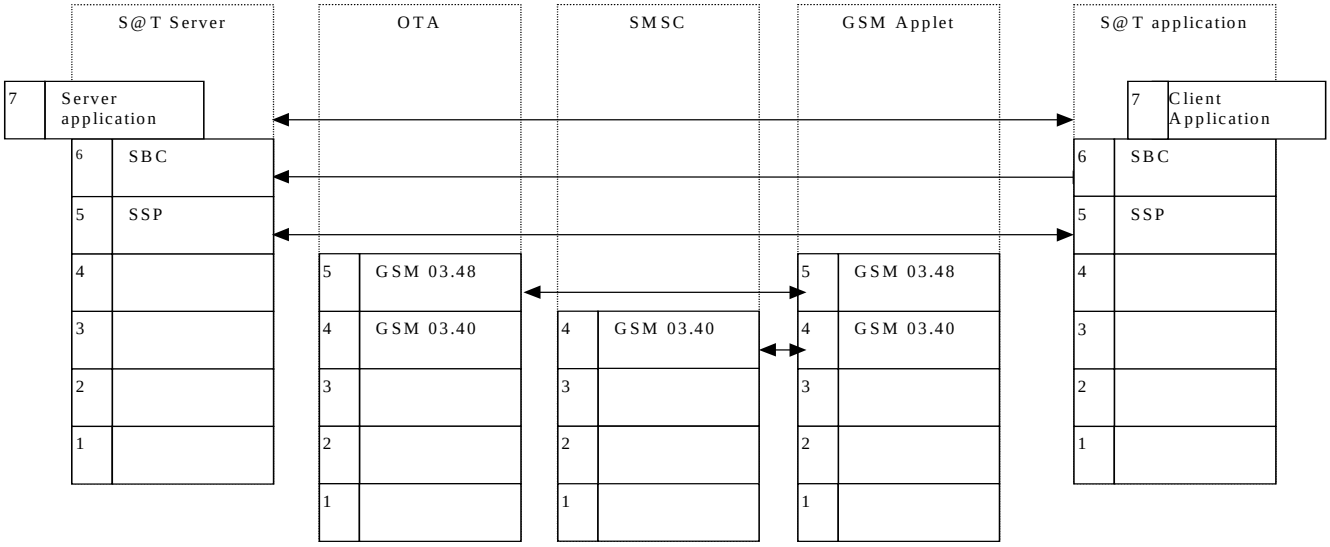


Fig 1. Layers Overview.

The way to present layers used in this document assumes that each layer will be fully independent from the layer above. The SSP layer protocol can be used on any connected or non connected media, and guarantees that upper layers are bearer independent by allowing the same functionalities regardless of the media used.

The SSP layer will be used to connect a Client application to a S@T Server (via a SMS-C and an OTA Entity for example) in order to send and receive data.

The SSP layer provides an efficient way to exchange messages and to prevent delayed messages from disrupting an implementation using SMS.

5 SSP layer definition

The coding is specified for a set of session commands an allows the management of a session layer between a Client Application and a S@T Server.

6 Command usage summary

A session command is issued in a ***XXX_REQ*** or ***XXX_RSP*** form, and is received in the associated ***XXX_IND*** or ***XXX_CNF*** form:
Client Application sends ***CONNECT_REQ***, S@T Server receives ***CONNECT_IND***. S@T Server sends back a Session ID with a ***CONNECT_RSP*** and Client Application receives a ***CONNECT_CNF***.

Session commands	Request (REQ)	Indication (IND)	Response (RSP)	Confirmation (CNF)	Signification
CONNECT	*X*	*X*	*X*	*X*	Establishment of a session.
DATA	*X*	*X*	*X*	*X*	Regular data transfer.
GET	*X*	*X*			Special data request with parameters.
POST	*X*	*X*			Special data request with parameters.
REPLY			*X*	*X*	Response to a special data request.
EXPRESS-DATA	*X*	*X*			Express data transfer.
DISCONNECT	*X*	*X*			Session release.
PAUSE	*X*	*X*			Suspend an established session
RESUME	*X*	*X*			Resume a suspended session

Table 1. Pool of possible commands.

Entity / Message	S@T Server	S@T Client
CONNECT_REQ		***X***
CONNECT_RSP	***X***	
DATA_REQ	***X***	***X***
DATA_RSP	***X***	***X***
GET_REQ		***X***[2]
POST_REQ		***X***[2]
REPLY_RSP	***X***	
EXPRESS_DATA_REQ	***X***	***X***
DISCONNECT_REQ	***X***	***X***[1]
PAUSE_REQ		***X***[2]
RESUME_REQ		***X***[2]

(***X*** indicates "entity can send this message")

Table 2. Command Usage.

[1] : Usage of this message is optional when used on connectionless network layers, as SMSs : S@T Client can optimize its disconnection phase regarding media type. The Server MUST handle time-out management to automatically terminate the corresponding session.
[2] : These commands can only be used when the server is the gateway and the client is the browser.

7 Generalized Message structure

All session messages will be formatted using the session format defined below:

SSC	SSP	SSC	SSP	

SSC : SSP Command. This field contains a session management command code.
Each command is coded on 1 byte, followed by its associated parameters.

Following notations will be used for further command description :

XXX_REQ : An entity sends a REQuest.
XXX_IND : Pending entity receives a request INDication.
XXX_RSP : Receiving entity sends a ReSPonse.
XXX_CNF : Requesting entity finally receives a CoNFirmation.

SSP : SSP command parameters.

A set of SSP commands MAY be present in every message. Moreover, correct implementation of the SSP protocol MUST be able to handle a set of commands in every message. Each SSP command must be handled in sequential order.
There is no size limitation imposed by the SSP protocol for a session message, but implementations MAY limit the size of this field.

8 Some specific payloads

8.1 Session ID

A session ID is a number identifying an unique session between a S@T Client and a S@T Server. The Session ID is initialized by the S@T Server entity on CONNECT_IND reception.

The Session ID is coded on 1 byte, the first bit indicates the direction of the CONNECT_RSP command. If CONNECT_RSP is Mobile Originated, the first bit of the session ID is set to 1 ; If CONNECT_RSP is Mobile Terminated, the first bit of the session ID is set to 0.

Session ID format for a CONNECT_RSP MO :

Strong bit weak bit

7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X
SessionID							

Session ID format for a CONNECT_RSP MT :

Strong bit weak bit

7	6	5	4	3	2	1	0
0	X	X	X	X	X	X	X
SessionID							

The session ID is allocated to a S@T Server / S@T Client couple and MUST remain valid and reserved until DISCONNECT_REQ command is sent, DISCONNECT_IND command is received, or a time-out occurs.

A server MAY accept or refuse simultaneous session establishment.

8.2 Connection ID

Connection ID is coded on 1 byte. It is used to identify messages exchanged during the handshake phase (session establishment). Connection ID MUST be persistent.

8.3 Protocol ID

Protocol ID is coded on 1 byte. It is used to enable transported data to be typed, and allow above layers to handle correctly this encapsulated data. Associated value must be reserved (see table in annex).

8.4 Transaction ID

Transaction ID is coded on 1 byte. The role of the Transaction ID is to identify a message.

Management of the Transaction ID is application dependent. It MAY be incremented at each request command sent (XXX_REQ), except when sending a DATA_EXPRESS_REQ.

8.5 Parameter fields group

8.5.1 Total Parameter Size (TPS)

The field "Total Parameter Size" (TPS) is used in most commands. It is inserted before parameter value. It indicates the size of the rest of the command in bytes. TPS is coded on one or several bytes.
The standard BER-TLV coding will be used.

8.5.2 Parameter Value (V)

The field "Parameter Value " (V) is used in most commands. V is coded on one or several bytes.
Parameter Value contains application specific data.

9 Description of the commands

Each command type is coded on the first Byte of session command field.

9.1 Coding of connection commands

Two commands MUST be used to set up a session, a request by a S@T client and a response from the server.

9.1.1 CONNECT_REQ / CONNECT_IND

The CONNECT_REQ command is used by the S@T Client to ask the S@T Server to open a session.

For optimization purposes, GET_REQ, POST_REQ, DATA_REQ, DATA_EXPRESS_REQ commands MAY follow a CONNECT_REQ command in the same message. In this case, the server MUST ignore the Session ID parameter of the commands following the CONNECT_REQ command if it is equal to 0x00 and consider such commands as being part of the same session. If the Session ID is different from 0x00, the server should reject the command.

Commands other than the ones quoted MUST NOT follow a CONNECT_REQ in the same message.

CONNECT_REQ / CONNECT_IND (1 byte)
Protocol ID (1 Byte)
Connection ID (1 Byte)
Server Address (3 Bytes)
Application Address (3 Bytes)

- Protocol ID : 1 byte ; ID to identify data encapsulated in the messages exchanged during this session.
- Connection ID : 1 byte ; ID to identify connection messages during the handshake.
- Server Address : 3 bytes ; defines the targeted S@T Server. (will access the default S@T Server if set to zeroes)
- Application Address : 3 bytes ; defines the originating application. (will indicate default S@T Application if set to zeroes)

9.1.2 CONNECT_RSP / CONNECT_CNF

The CONNECT_RSP command is used by an application to respond to a connection request.
The entity receiving a CONNECT_IND and responding with a CONNECT_RSP is named S@T Server. It is in charge of allocating a *Session ID*. If *Status* is successful, the connection initiator will get this *Session ID* and each entity will insert this number in following messages.

If connection failed to achieve, each entity MUST return in IDLE state (c.f. figure 2 and figure 3).
There is no acknowledgement to be sent to the S@T Server.

For optimization purposes, other SSP commands SHOULD follow this command.

CONNECT_RSP / CONNECT_CNF (1 byte)
Connection ID (1 Byte)
Session ID (1 Byte)
Status / Cause (1 Byte)

- Connection ID : 1 byte ; ID to identify connection messages during the handshake. It MUST be the same as the one used in the corresponding CONNECT_REQ / CONNECT_IND.
- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Status / Cause : 1 byte ; Indicates a successful session allocation or its failure and the reason why the session could not be allocated. Possible values for this field are described in the paragraph *binary implementation*.

9.1.3 PAUSE_REQ / PAUSE_IND

The PAUSE_REQ command is used by the S@T Client to inform the S@T Server that a period of inactivity may follow, and then requests the Server to keep the session alive.
The S@T server MAY increase the time out period for this session, and MAY disconnect the session after this period if no RESUME_REQ is received.

All messages exchanged after a PAUSE_REQ/ PAUSE_IND MUST be ignored until a RESUME_REQ/ RESUME_IND is sent by the Client.

PAUSE_REQ / PAUSE_IND (1 byte)
Session ID (1 Byte)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.

9.1.4 RESUME_REQ / RESUME_IND

The RESUME_REQ command is used by the S@T Client to request the S@T Server to reactivate a suspended session.

If the server has not disconnected this session, it MUST restore the default time out period for this session

For optimization purposes, GET_REQ, POST_REQ, DATA_REQ, DATA_EXPRESS_REQ commands MAY follow a RESUME_REQ command in the same message.
Commands other than the ones quoted MUST NOT follow a RESUME_REQ in the same message.

If the session was disconnected the server MAY send a DISCONNECT_REQ message.

RESUME_REQ / RESUME_IND (1 byte)
Session ID (1 Byte)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.

9.2 Coding of Disconnection commands

Only one command is used to hang a session ; it is not confirmed.

9.2.1 DISCONNECT_REQ / DISCONNECT_IND

The DISCONNECT_REQ command MAY be used by S@T Server or S@T Client to stop an established session. There is no acknowledgement of this message. Furthermore, this message is NOT MANDATORY to terminate a session. Implementations MAY use time out to discard unused sessions.

For optimization purposes, other SSP commands MAY follow this command.

DISCONNECT_REQ / DISCONNECT_IND (1 byte)
Session ID (1 Byte)
Cause Value (1 Byte)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.

Cause Value : 1 byte ; To indicate the reason of the disconnection. Possible values for this field are described in paragraph *binary implementation*.

9.3 Coding of Data commands

9.3.1 DATA_REQ / DATA_IND

The DATA_REQ command is used to exchange regular data.

For optimization purposes, other SSP commands MAY follow this command.

DATA_REQ / DATA_IND (1 byte)
Session ID (1 Byte)
Transaction ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Transaction ID : 1 byte ; Identifies the message.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; application specific data.

9.3.2 DATA_RSP / DATA_CNF

The DATA_RSP command is used to exchange regular data, it is sent in response to a received DATA_IND.

For optimization purposes, other SSP command MAY follow this command.

DATA_RSP / DATA_CNF (1 byte)
Session ID (1 Byte)
Transaction ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Transaction ID : 1 byte ; Identifies the message. It MUST be the same as the one used in the corresponding DATA_REQ / DATA_IND.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; application specific data.

9.3.3 GET_REQ / GET_IND

The GET_REQ command is to implement the "GET" as specified in *RFC 2068 (Hypertext Transfer Protocol -- HTTP/1.1).*

For optimization purposes, other SSP commands MAY follow this command.

GET_REQ / GET_IND (1 byte)
Session ID (1 Byte)
Transaction ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Transaction ID : 1 byte ; Identifies the message.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; application specific data.

9.3.4 POST_REQ / POST_IND

The POST_REQ command is to implement the "POST" as specified in *RFC 2068 (Hypertext Transfer Protocol -- HTTP/1.1).*

For optimization purposes, other SSP commands MAY follow this command.

POST_REQ / POST_IND (1 byte)
Session ID (1 Byte)
Transaction ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Transaction ID : 1 byte ; Identifies the message.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; application specific data.

9.3.5 REPLY_RSP / REPLY_CNF

The REPLY_RSP command MUST be used by upper layers to reply to a POST_IND or a GET_IND message.

For optimization purposes, other SSP commands MAY follow this command.

REPLY_RSP / REPLY_CNF (1 byte)
Session ID (1 Byte)
Transaction ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Transaction ID : 1 byte ; Identifies the message. It MUST be the same as the one used in the corresponding POST_REQ / POST_IND or GET_REQ / GET_IND.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; A parameter used by the upper layer.

9.4 Coding of Express-Data commands

9.4.1 EXPRESS_DATA_ REQ / EXPRESS_DATA_ IND

This message MUST be used to send out of band data. Receiver MUST accept express data if the given Session ID corresponds to an active session.

For optimization purposes, other SSP commands MAY follow this command.

EXPRESS_DATA_ REQ / EXPRESS_DATA_ IND (1 byte)
Session ID (1 Byte)
Total Parameters Size (TPS)(1..n byte)
Parameter Value (V) (variable)

- Session ID : 1 byte ; Identifies a session between a S@T Server and a S@T Client.
- Total Parameter Size (TPS) : no fixed size ; Indicates the size in bytes of the rest of the command.
- Parameter Value (V) : no fixed size ; A parameter used by the upper layer.

10 Server State diagram

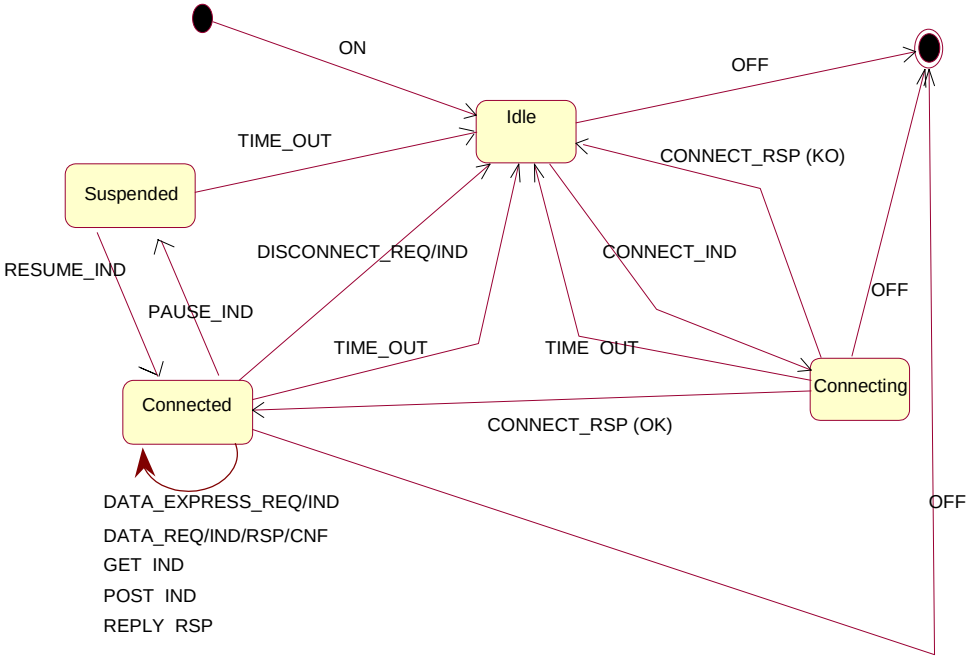


figure 2

11 Application State diagram

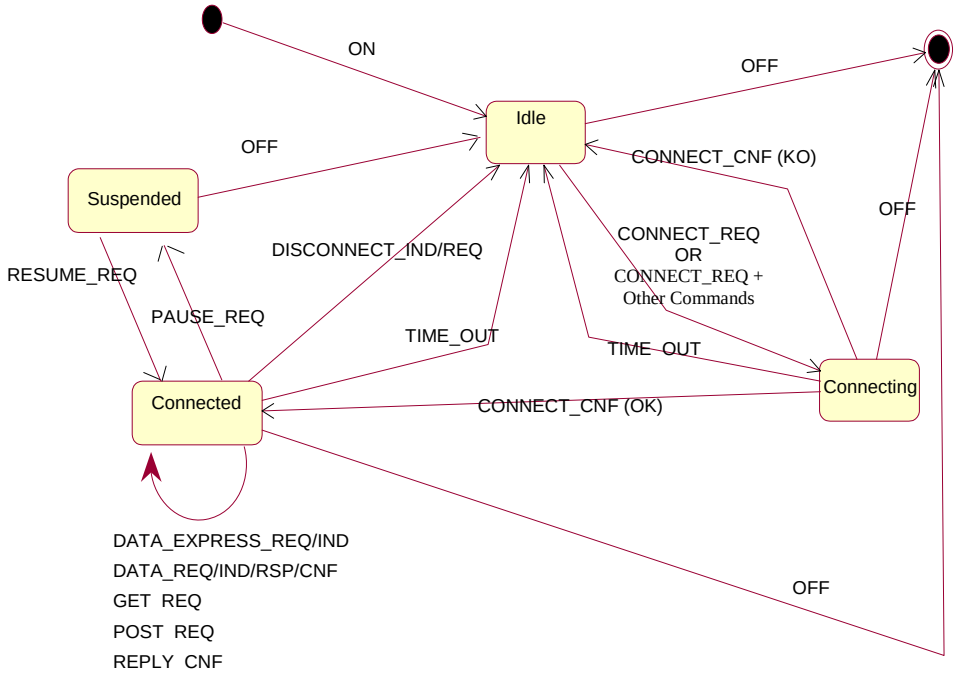


figure 3

12 Main session management flow charts

The aim of these charts is to give simple examples of SSP Layer Protocol use. According and in reference to the S@T Administrative Commands S@T 01.21 specification § "9-CONNECTION PROCEDURE FOR AN ADMINISTRATION SESSION"

FC 1.1	Connecting session layer
FC 1.2	Canceling connection
FC 1.3	Canceling connection with reconnection
FC 1.4	Server refusing connection
FC 1.5	Simultaneous Session Establishment
FC 2.1	Exchanging data with Data_XXX (Unidirectional)
FC 2.2	Request Collision
FC 2.3	Exchanging data with Data_XXX (Bi-directional)
FC 2.4	Use of Data Express
FC 2.5	Use of GET_REQ and REPLY_RSP
FC 2.6	Use of POST_REQ and REPLY_RSP
FC 3.1	Client initiating disconnection (using DISCONNECT_REQ)
FC 3.2	Client initiating disconnection (Without using DISCONNECT_REQ)
FC 3.3	Server Initiating Disconnection

This list of examples is far from being exhaustive. Plus, they do not define an implementation direction / obligation (Transaction ID, Session ID and Connection ID management.) Some mandatory steps of implementation are described in paragraph *Implementation Requirements*

Following notation will be used in these charts:

C : Connection ID.

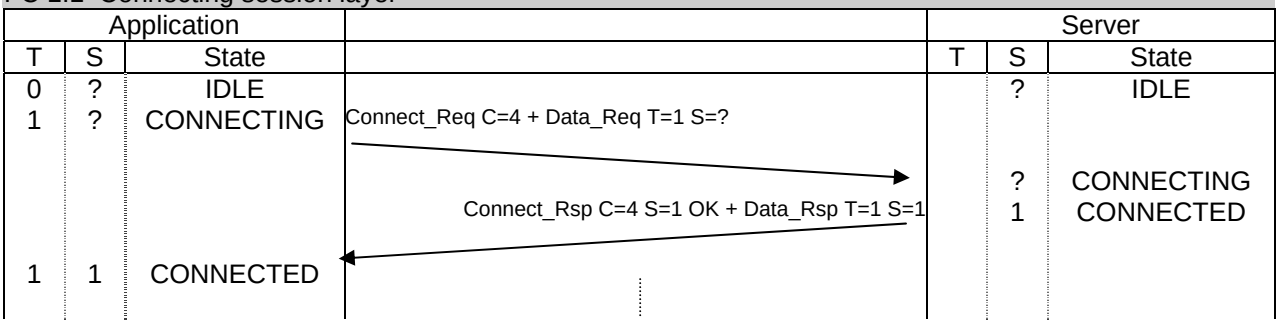
S : Session ID

St : Diagram State.

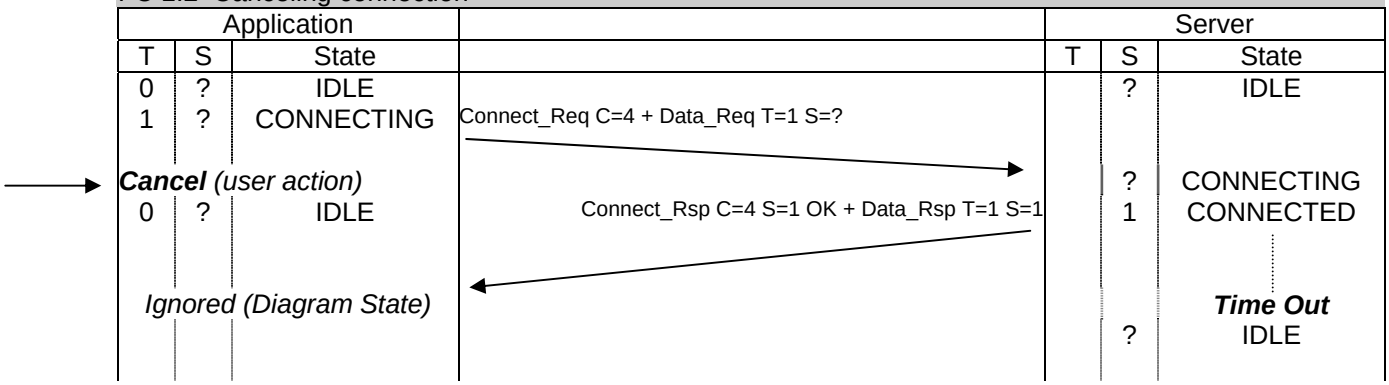
T : Transaction ID. Management of the Transaction ID being implementation specific on the application side, management used in the charts is not mandatory.

Only Significant parameters will be noted on exchanged messages. In the examples, the messages exchanged after a successful handshake consist of a single command, this is for presentation purposes only.

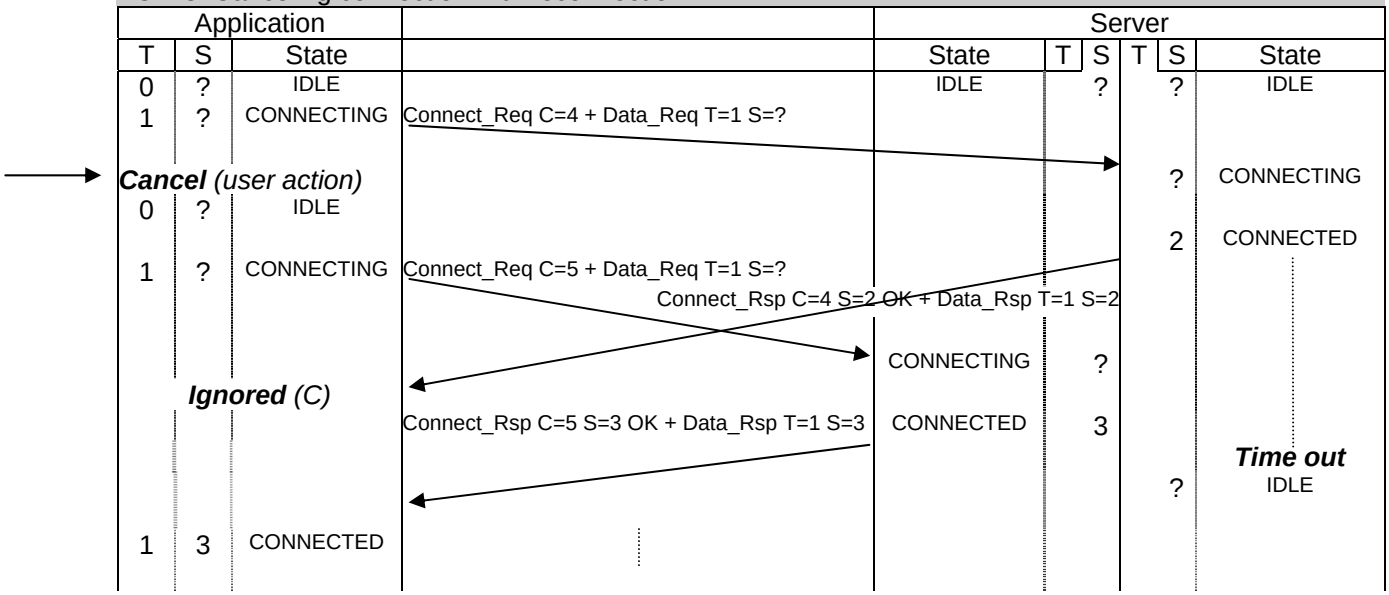
FC 1.1 Connecting session layer



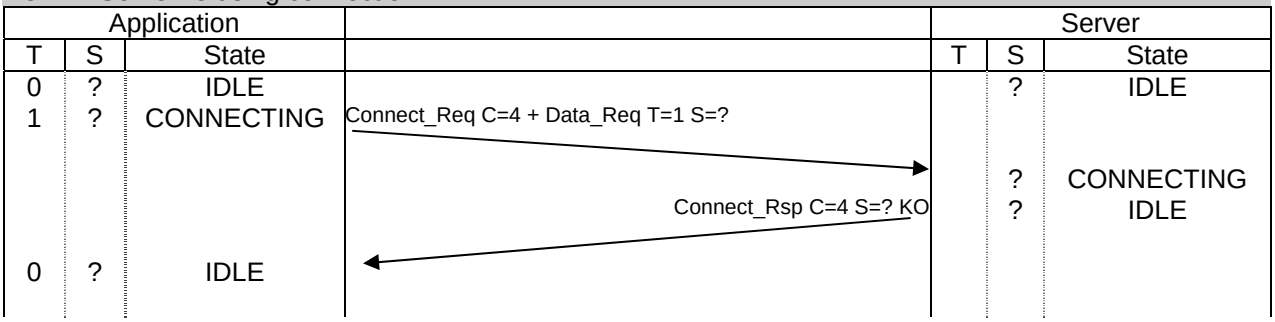
FC 1.2 Canceling connection



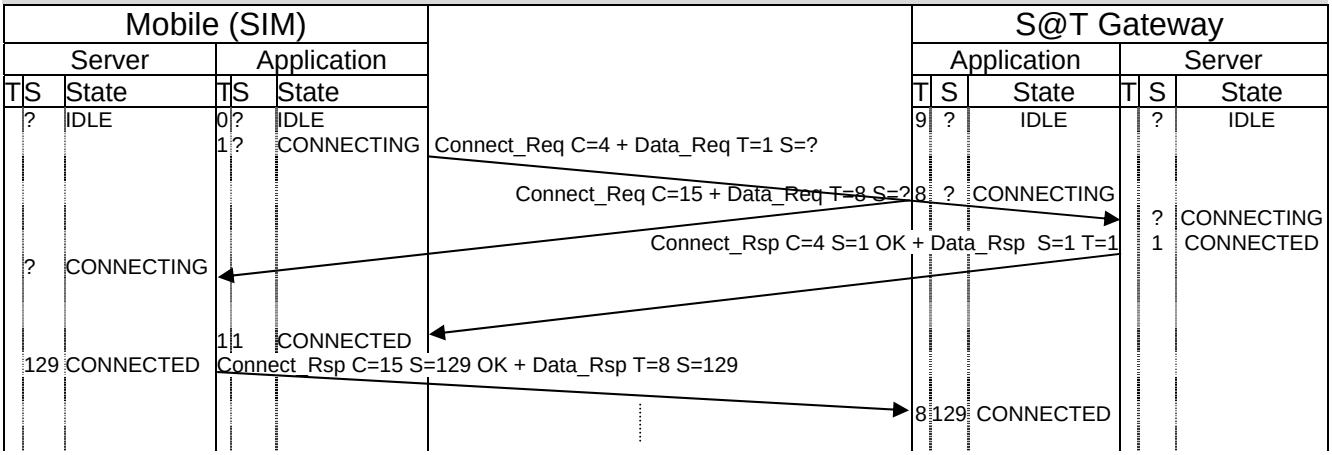
FC 1.3 Canceling connection with reconnection



FC 1.4 Server refusing connection

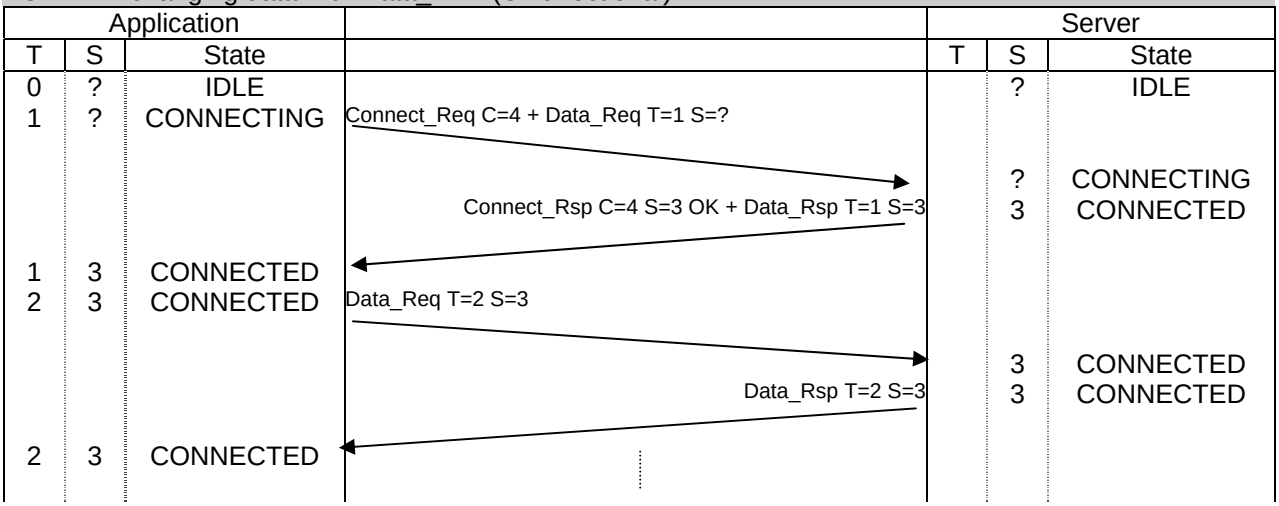


FC 1.5 Simultaneous Session Establishment

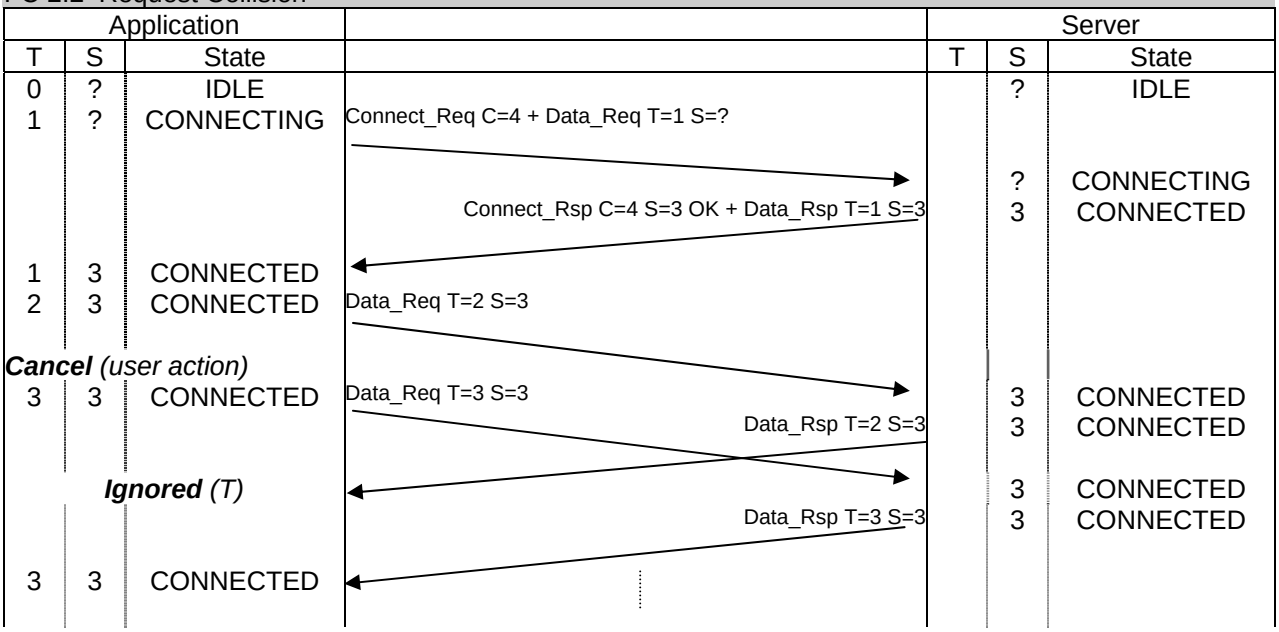


This example shows the utility of the Session ID Format.

FC 2.1 Exchanging data with Data_XXX (Unidirectional)

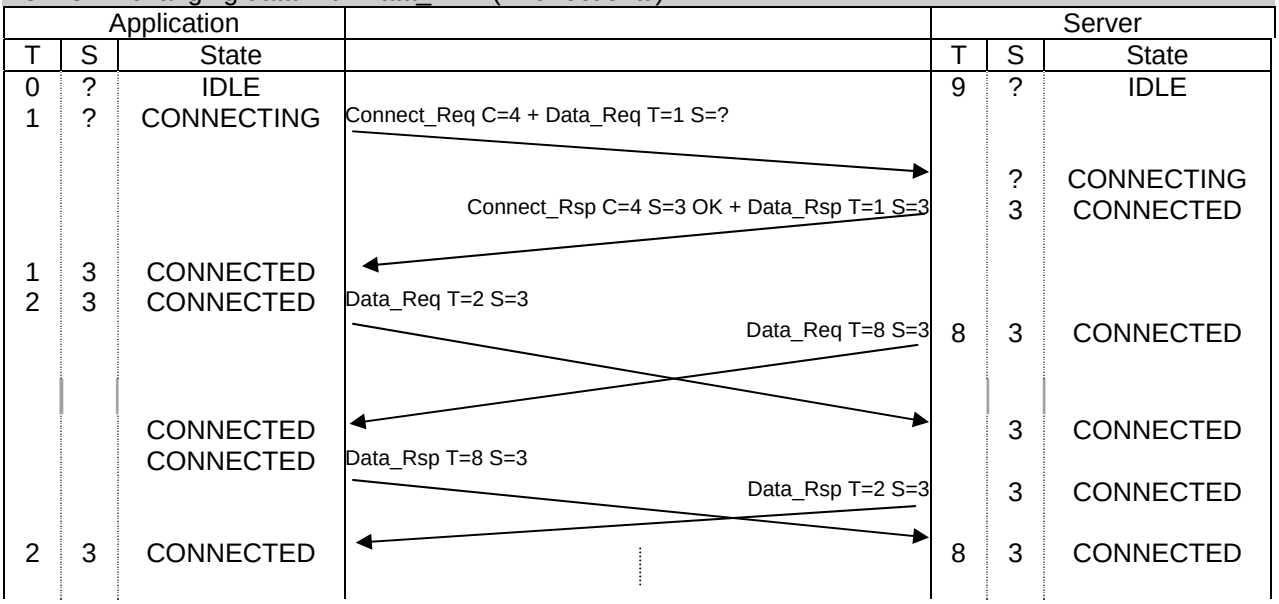


FC 2.2 Request Collision



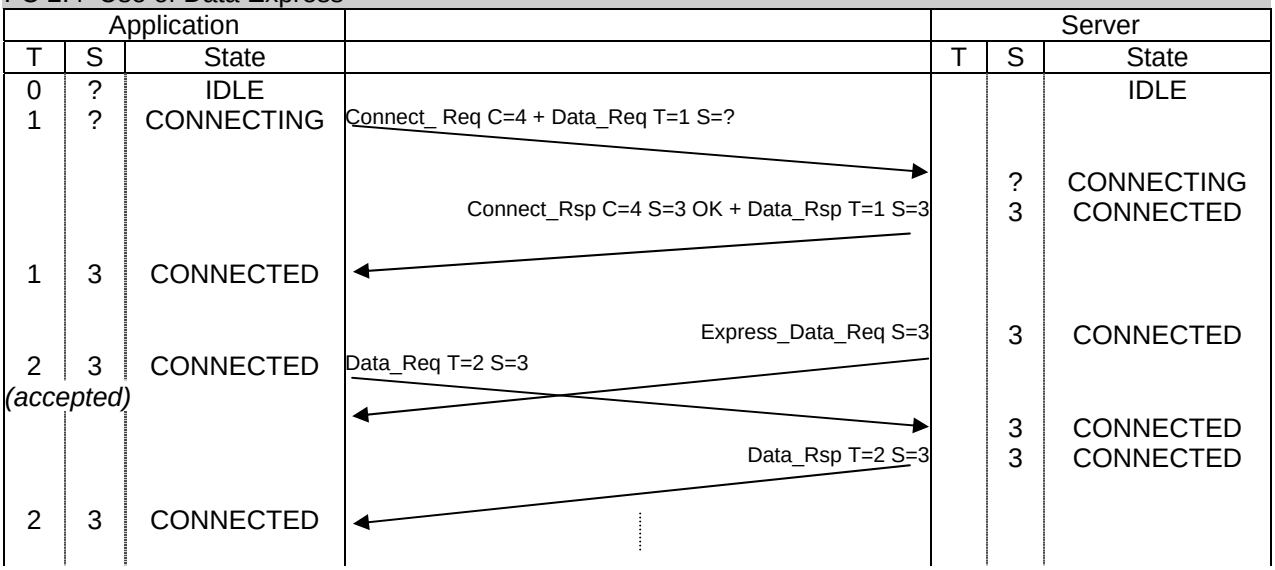
(In this case the S@T Client is not multi-request.)

FC 2.3 Exchanging data with Data_XXX (Bi-directional)



This example shows that Transaction IDs are locally managed, and that their management is implementation dependant.

FC 2.4 Use of Data Express



This sample shows that Express Data doesn't require a Transaction ID management.

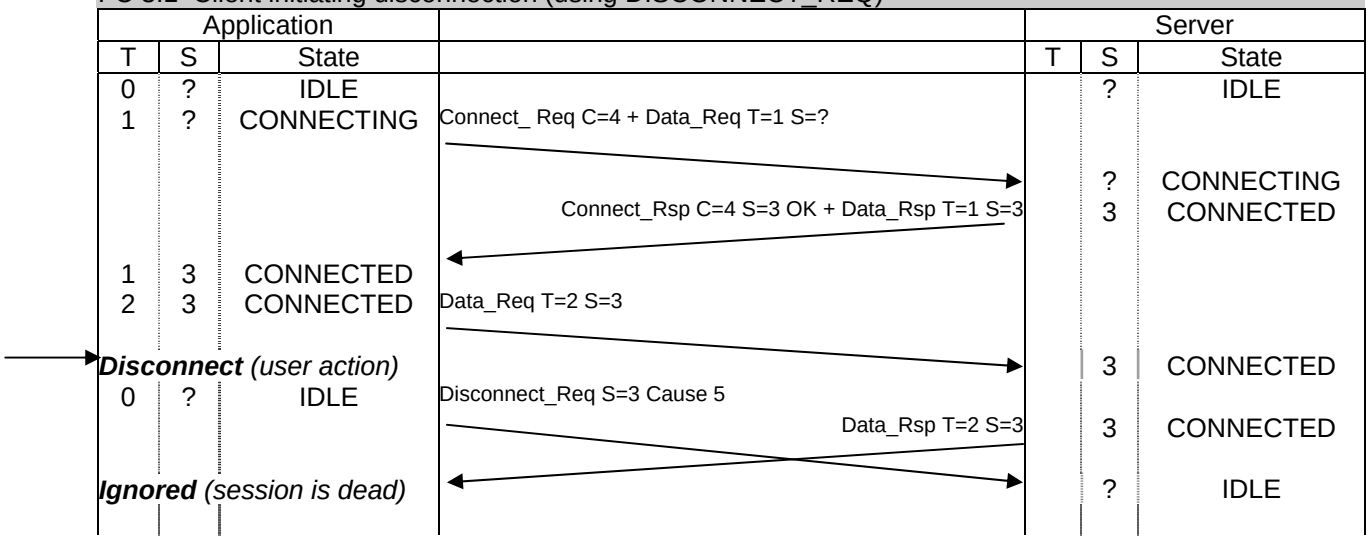
FC 2.5 Use of GET_REQ and REPLY_RSP

Application				Server		
T	S	State		T	S	State
0	?	IDLE			?	IDLE
1	?	CONNECTING	Connect_Req C=4 + Get_Req T=1 S=? →			
					?	CONNECTING
			← Connect_Rsp C=4 S=3 OK + Reply_Rsp T=1 S=3		3	CONNECTED
1	3	CONNECTED				
2	3	CONNECTED	Get_Req T=2 S=3 →			
					3	CONNECTED
			← Reply_Rsp T=2 S=3		3	CONNECTED
2	3	CONNECTED				

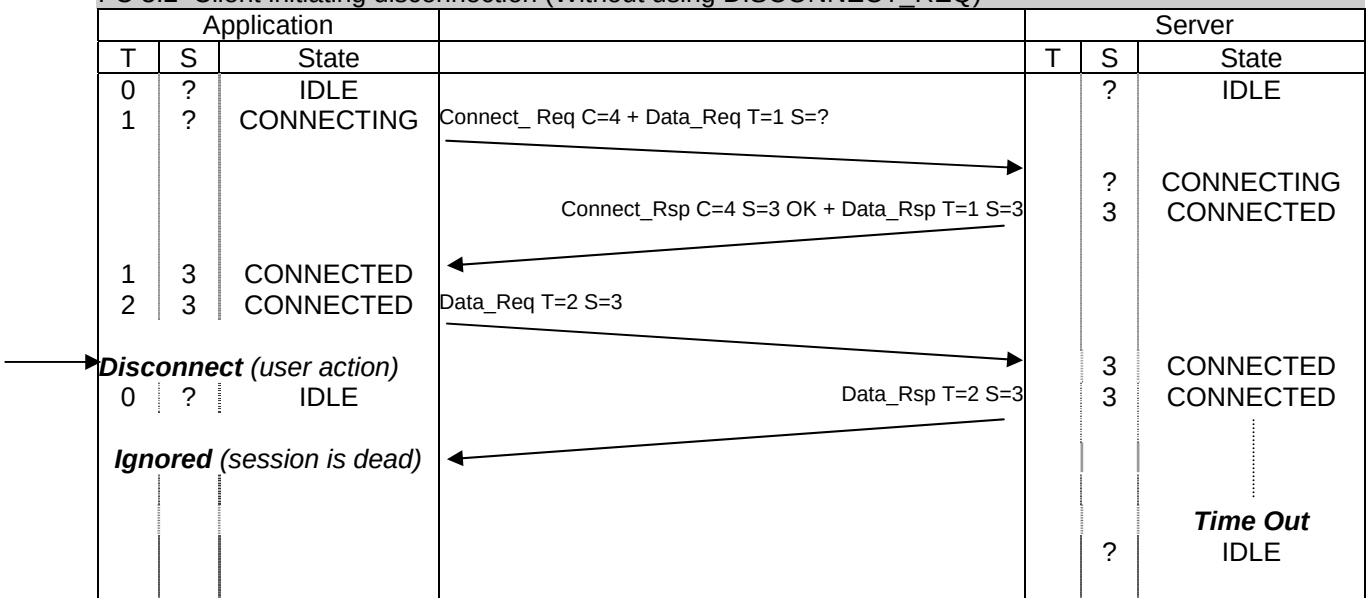
FC 2.6 Use of POST_REQ and REPLY_RSP

Application				Server		
T	S	State		T	S	State
0	?	IDLE			?	IDLE
1	?	CONNECTING	Connect_Req C=4 + Post_Req T=1 S=? →			
					?	CONNECTING
			← Connect_Rsp C=4 S=3 OK + Reply_Rsp T=1 S=3		3	CONNECTED
1	3	CONNECTED				
2	3	CONNECTED	Post_Req T=2 S=3 →			
					3	CONNECTED
			← Reply_Rsp T=2 S=3		3	CONNECTED
2	3	CONNECTED				

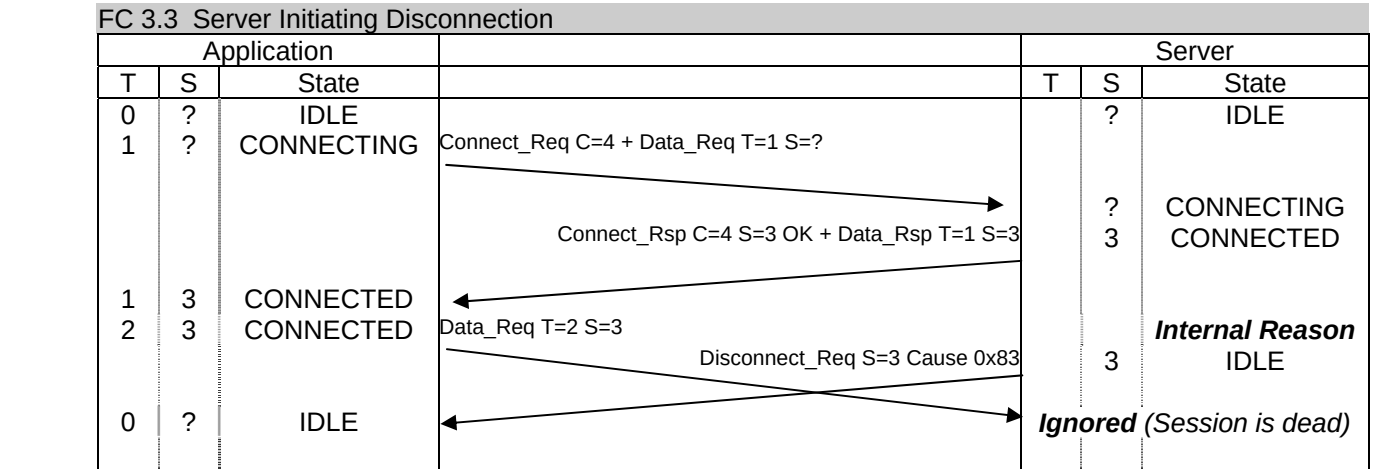
FC 3.1 Client initiating disconnection (using DISCONNECT_REQ)



FC 3.2 Client initiating disconnection (Without using DISCONNECT_REQ)



FC 3.3 Server Initiating Disconnection



13 SSP layer binary implementation

SSP commands are coded with the following values :

SSP Commands	
CONNECT_REQ / CONNECT_IND	0x01
CONNECT_RSP / CONNECT_CNF	0x10
DATA_REQ / DATA_IND	0x08
DATA_RSP / DATA_CNF	0x80
GET_REQ / GET_IND	0x02
POST_REQ / POST_IND	0x04
REPLY_RSP / REPLY_CNF	0x20
EXPRESS_DATA_ REQ / EXPRESS_DATA_ IND	0x06
DISCONNECT_REQ / DISCONNECT_IND	0x05
PAUSE_REQ / PAUSE_IND	0x07
RESUME_REQ / RESUME_IND	0x09

Status fields must be initialized using following error codes :

Error Value Set	
SSE_NO_ERROR	0x00
SSE_COMMAND_SYNTAX_ERROR	0x81
SSE_SERVER_NOT_FOUND	0x82
SSE_SERVER_BUZY	0x83
SSE_CONNECTION_REFUSED	0x84
SSE_TIME_OUT	0x85
SSE_TOO_MANY_SESSIONS	0x86
SSE_SESSION_ERROR	0x88
SSE_NOT_SPECIFIED	0x8F

14 Implementation Requirements.

This paragraph states implementation requirements. They do concern verifications that have to be made when receiving/sending a message. Most of these requirements can be deduced from the previous paragraphs. Changes of diagram state are not indicated here, they correspond to the state diagrams "figure 2" and "figure 3".

14.1 Reception of CONNECT_IND / Transmission of a CONNECT_RSP.

Upon reception of a CONNECT_IND, the S@T Server verifies if it can allocate a session to communicate with the S@T Client.

If the S@T Server accepts a connection with the S@T Client, the S@T Server MUST return a CONNECT_RSP with:
- Connection ID : Identical to the one received in the CONNECT_IND ;
- Session ID : set to a new allocated value. It will be used through the whole session ;
- Status set to SSE_NO_ERROR ;
- if other commands follow the received CONNECT_IND in the same message, they MUST be handled after state change, if their Session ID are set to 0x00, they are considered to be the same as the newly allocated one.
  Other response commands SHOULD be added in the response message after the CONNECT_RSP.

If the S@T Server does not accept a connection with the S@T Client, the S@T Server MAY return a CONNECT_RSP with:
- Connection ID : Identical to the one received in the CONNECT_IND ;
- Session ID : unset ;
- Status set to an error code ;
- commands following CONNECT_IND in the same message MUST be ignored if Session ID is set to 0x00.

14.2 Reception of CONNECT_CNF

Upon reception of a CONNECT_CNF, the S@T Client
- MUST verify if the diagram state corresponding to the Connection ID received is in "CONNECTING" state ;
- then the Connection ID will be verified, it has to be equal to the one sent in the corresponding CONNECT_REQ ;
- if the Status of the CONNECT_RSP is SSE_NO_ERROR, the S@T Client gets the Session ID from the CONNECT_RSP and goes into the state "CONNECTED".
  Else, the S@T Client goes back to IDLE ;
- if other commands follow this command in the same message, they MUST be handled.

14.3 Reception of a DATA_IND / Transmission of a DATA_RSP

Upon reception of a DATA_IND, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- manage the Parameter Value to give to the upper layer ;
- if other commands follow this command in the same message, they MUST be handled.

The receiver of a DATA_IND MAY send a response.
To send a response :
- it MUST use the DATA_RSP command ;
- Transaction ID and Session ID will be the same as in the DATA_IND received ;
- PVL / PV MUST be present.

14.4 Reception of a DATA_CNF

Upon reception of a DATA_CNF, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- verify the Transaction ID ;

- manage the Parameter Value to give to the upper layer ;
- if other commands follow this command in the same message, they MUST be handled.

14.5 Reception of a GET_IND / Transmission of a REPLY_RSP

Upon reception of a GET_IND, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- manage the Parameter Values to give to the upper layer ;
- if other commands follow this command in the same message, they MUST be handled.

The receiver of a GET_IND MAY send a response.
To send a response :
- it MUST use the REPLY_RSP command ;
- Transaction ID and Session ID will be the same as in the received GET_IND ;
- PVL / PV MUST be present.

14.6 Reception of a POST_IND / Transmission of a REPLY_RSP

Upon reception of a POST_IND, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- manage the Parameter Values to give to the upper layer;
- if other commands follow this command in the same message, they MUST be handled.

The receiver of a POST_IND MAY send a response.
To send a response :
- it MUST use the REPLY_RSP command ;
- Transaction ID and Session ID will be the same as in the POST_IND received ;
- PVL / PV MUST be present.

14.7 Reception of a REPLY_CNF

Upon reception of a REPLY_CNF, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- verify the Transaction ID ;
- manage the Parameter Value to give to the upper layer ;
- if other commands follow this command in the same message, they MUST be handled.

14.8 Reception of EXPRESS_DATA_IND

Upon reception of EXPRESS_DATA_IND, the receiver MUST
- verify the Session_ID to see if this session is alive ;
- check the diagram state (state MUST be "CONNECTED") ;
- manage the Parameter Value to give to the upper layer ;
- if other commands follow this command in the same message, they MUST be handled.

14.9 Reception of a DISCONNECT_IND

Upon reception of a DISCONNECT_IND, the receiver MUST :
- verify the Session_ID to see if this session is alive ;
- delete the session corresponding to the received Session ID.
- if other commands follow this command in the same message, they MUST be handled.

15 Reception of a PAUSE_IND

Upon reception of a PAUSE_IND, the receiver MUST :

- verify the Session_ID to see if this session is alive ;
- suspend the session corresponding to the received Session ID.
- if other commands are received on this suspended session they MUST be ignored.

16 Reception of a RESUME_IND

Upon reception of a RESUME_IND, the receiver MUST :

- verify the Session_ID to see if this session is suspended ;
- resume the session corresponding to the received Session ID.
- if other commands are received on this resumed session they MUST be handled.

17 Protocol ID to be used in CONNECT_REQ message

When establishing a connection, the client MUST specify the protocol needing to exchange informations in SSP messages. A list of possible values is proposed below :

Protocol ID Value	Protocol ID Name	Description
00	Not specified	Free for testing
01	S@T	S@T Browsing protocol
02	S@Ta	S@T Administration protocol
03	S@Tp	S@T High Priority Push protocol
04-FF	Reserved	Reserved for future use

18 Annex : Implementation of SSP and STLS over SMS

S@T browsing application will consider two levels of security, associated to the received message. One will allow only operational commands, the other will allow administrative commands.

18.1 Implementation using SMS bearer

When using SMS bearer, S@T applications will use standard protocols to guarantee interoperability and code sharing with other card applications.

GSM 03.48 layer will be used as secure transport layer under SSP layer protocol, and GSM 03.40 will be used as standard transport layer under GSM 03.48.

18.2 GSM 03.48 support

GSM 03.48 must be used in both directions (client to server and server to client) to be able to perform secure message exchanges.

The SMS-PP part of the GSM 03.48 v 8.0.0 is taken into account.

To ensure interoperability of different manufacturers, some GSM 03.48 mechanism have to be supported :

DES Algorythm in CBC mode with 8 bytes key
3DES Algorythm in CBC mode with 16 bytes keys

Some 03.48 fields must also be defined as explained below.

18.3 GSM 03.48 TAR management

A special TAR will be reserved to identify S@T Applications.
The reserved value will be :

TAR Byte 1	TAR Byte 2	TAR Byte 3
0x53	0x40	0x54
'S'	'@'	'T'

18.4 GSM 03.48 security parameters management

The security levels will be mapped into the SIM card to a 03.48 security parameter set, that may be customizable to answer particular needs.

When used on SMS bearer, the security parameters will be differents regarding SMS direction.

Mobile Terminated SMS
The 'proposed' way to map logical security levels is :

Security level	**SPI value**
Operational	0x00 : no security used
Administrative	0x12 : CC and Counter mode 2

The CC is defined to be 8 bytes long, padding is done using 0x00 bytes.
The initial counter value to be used by the gateway is defined to be 0x0000000032

Mobile Originated SMS

Security level	SPI value
Operational	0x00 : no security used
Administration	0x00 : no security used

These levels are mandatory supported to ensure interoperability.

If operational security layer is changed and implies a key usage, two different keys must be stored.

The second byte of the SPI field must be set to 0, to deactivate any proof of receipt.

In case a wrong security level is used in MT messages, then the browser MUST ignore the messages.

18.5 Underlying layer requirements

If necessary, layer under SSP should implement concatenation to allow sending messages 'without' size restrictions in Server to browser direction, but has not been considered as mandatory for this release in browser to server direction.

This means that when using SMS bearer, standard GSM 03.40 concatenation will be used in SMS MT direction, but not in MO direction.

The SMS-SUBMIT and SMS-DELIVER only have to be supported and to be used.

The number of concatenated SMS to be handled in the card must be enough to allow receiving at least a 1024 bytes SSP buffer (i.e. several SSP commands stored consecutively in a SMS concatenation sequence).

The SSP buffer size is accessible by an environment variable specified in the S@T browsing command document (*ReceptionBufferSize*). This size can also be sent at connection time on its own by the browser if it is found usefull.

Except the last, all the SMS part of a concatenated message must use the full TP-UD space (i.e. must contain 140 bytes).

19 Annex : Optional Features [informative]

The following features of SSP are optional:

- Pause/Resume mechanism

20 Annex : Example of using S@T Protocols [informative]

This example shows a full request issued by the browser and an arbitrary response.
The URL requested is "http://machine/path/file.satml" and the deck sent back to the browser contains a get environment macro to get the browser version stored in a variable, which is then sent out of the browser with the exit macro.

FIRST REQUEST SAMPLE (1 MO)

03.40	XX XX.. XX 0x4F 0x02 0x70 0x00	Standard SMS 03.40 header Standard SMS 03.40 UDL UDHL UDH : IEI 03.48 03.48 header length (IEIL)
03.48	0x00 0x4A 0x0D 0x00 0x00 0x00 0x00 0x53 0x40 0x54 0x00 0x00 0x00 0x00 0x00 0x00	Secure data length (Length of command packet) CHL SPI (no security) Kic Kid TAR S@T Counter (not used) Padding counter
SSP	0x01 0x01 0x08 0x00 0x00 0x00 0x00 0x00 0x00	CONNECT_REQ Protocol ID (S@T browsing protocol) Connection ID (persistent browser incremented value) Server Address (Default gateway) Application Address (Default application)
	0x06 0x00 0x08 0x50 0x03 0xFF 0x12 0x04 0x03 0x01 0x00	EXPRESS_DATA_REQ Session ID (unknwon: use the same as the connect_req) TPS (total parameter size) BrowserInfo TAG BrowserInfo Length SIMBrowserSupplier (other) BrowserVersion (S@T v1.x, browser v2) ReceptionBufferSize (4*256Bytes) BrowserProfile TAG BrowserProfile Length Browser Profile Value (no option supported)
	0x02 0x00 0x01 0x24	GET_REQ Session ID (unknown: use the same as the connect_req) Transaction ID (volatile browser incremented value) TPS (total parameter size)
S@T Operational commands	0x40 0x22 0x0D	Browser Request TAG Browser Request Length URL Reference TAG (example without

	0x20 0x0E 0x1E "http://machine/path/file.satml"	parameters) URL Reference Length Address Reference TAG Address Reference Length Address

(parameters may follow, impacting lengths)

RESPONSE SAMPLE (2 MT)

03.40	XX XX.. XX 0x28 0x07 0x00 0x03 0x20 0x02 0x01 0x70 0x00	Standard SMS 03.40 header Standard SMS 03.40 UDL (MUST BE 0xFF outside this example) UDHL UDH : IEI Concatenation Concatenation header length (IEIL) Concatenation Reference number (arbitrary) Maximum Packet Number (2) Current Sequence Number (2nd) UDH : IEI 03.48 03.48 header length (IEIL)
03.48	0x00 0x1E 0x0D 0x00 0x00 0x00 0x00 0x53 0x40 0x54 0x00 0x00 0x00 0x00 0x00 0x00	Secure data length (Length of command packet) CHL SPI (no security) Kic Kid TAR S@T Counter (not used) Padding counter
SSP	0x10 0x08 0x10 0x00	CONNECT_RSP Connection ID (sent by the browser) Session ID (Server allocated) Status (SSE_NO_ERROR)
	0x20 0x10 0x01 0x0D	REPLY_RSP Session ID Transaction ID (sent by the browser) TPS (total parameter size)
S@T Operational commands	None	None
SBC (first Part)	0x01 0x0B 0x05 0x09 0x22 0x02 0x01 0x02	Deck Tag Deck Length Card Tag Card Length GetEnvironment Tag GetEnvironment Length DestVarId (example) EnvVarRef (BrowserVersion)

(the point where the deck is cut is only for example purpose: in a real case the 1st SMS must be full and a Byte Code may be cut)

03.40	XX XX.. XX 0x0B 0x05 0x00	Standard SMS 03.40 header Standard SMS 03.40 UDL UDHL UDH : IEI Concatenation

	0x03 0x20 0x02 0x02	Concatenation header Length Concatenation Reference number (arbitrary) Maximum Packet Number (2) Current Sequence Number (2nd)
SBC (second part)	0x2C 0x03 0x09 0x01 0x01	Exit Tag Exit length VarRefListTag VarRefList Length Variable ID (the one containing the browser version)

21 History

Document history		
Release	Approved by	Comment
V1.0.0	SIM Alliance TDG	First internal release
V1.0.1	SIM Alliance TDG	Update Layout
V1.0.2	SIM Alliance TDG	Final release
V1.0.3	SIM Alliance TDG	Example of using S@T Protocols added. Annex describing Optional features added
V1.0.4	SIM Alliance TDG	Working document
V1.0.5	SIM Alliance TDG	Editorial changes
V1.0.6	SIM Alliance TDG	Editorial changes for publication
V2.0.0	SIM Alliance S@T Group	Clarifications on CR and editorial changes for publication June 2004 CR 2004-13 CR 2004-22 CR 2004-27
V3.0.0	SIM Alliance TDG	Editorial modifications Release 2007 publications

21.1 Annex : LIST OF CHANGE REQUESTS [informative]

CR Number	CR Identifier	Subject	Document Reference	Status / Meeting No.
10105	SCHLUMBERGER-WG1 -MAR-2001#1	Security LEVEL ERROR MANAGEMENT	S@T 1.20 V1.0.3	Accepted#27
10119	GEMPLUS-WG1-MAY-2001#28-2	Protocol ID declaration for high priority PUSH	S@T 1.20 V1.0.4	Accepted #29
2004-13	AXALTO DECEMBER 2003 #013	Session ID when commands follow a CONNECT_REQ	S@T 1.20 V2.0.0	Accepted (email vote 9th March 2004)
2004-22	GEMPLUS-FEBRUARY-2004#5	Security level for administrative acknowledgement	S@T 1.20 V2.0.0	Accepted (email vote 9th March 2004)
2004-27	G&D 02.2004 #5	12. Session diagram admin reference	S@T 1.20 V2.0.0	Accepted (email vote 9th March 2004)