PUBLIC

SAP HANA Platform 2.0 SPS 03

Document Version: 1.1 – 2018-10-31

SAP HANA SQL Command Network Protocol Reference

THE BEST RUN SAP

Content

1 SAP HANA SQL Command Network Protocol Reference

This guide describes the SQL Command Network Protocol that is used by SAP HANA clients to communicate with SAP HANA.

2 Introduction

The SQL Command Network Protocol Reference Guide describes the protocol used by SAP HANA database clients to communicate with the SAP HANA database.

This document describes the binary message format used in the communication and explains the purpose of the various protocol elements. It also discusses usage scenarios for the various messages defined in the SQL Command Network Protocol and defines usage sequences to utilize database server functionality, such as:

- The user authentication and connection process
- The execution of prepared statements
- The handling of large object data
- The usage of distributed transaction handling

2.1 Terminology

Within this document, specific terminology and abbreviations are used.

A **message** describes requests or replies exchanged between client and server. A **request** is always a message sent by the client, and a **reply** is always the message sent by the server.

The following abbreviations are used to describe data types in message formats:

Abbreviation	Data Type	C Data Type[1]
I1	1-byte integer value	`char`
UI1	1-byte unsigned integer value	`unsigned char`
B	1-byte unsigned integer value, or "one byte"	`unsigned char`
I2	2-byte integer in little-endian format	`short`
UI2	2-byte unsigned integer in little-endian format	`unsigned short`
BI2	2-byte integer in big-endian format	`unsigned short`
I4	4-byte integer in little-endian format	`int`
NI4	4-byte integer in client native (big/little-endian) format	`int`
UI4	4-byte unsigned integer in little-endian format	`unsigned int`
I8	8-byte integer in little-endian format	`long`
I12	12-byte integer in little-endian format	
I16	16-byte integer in little-endian format	

1 For a Linux GCC in an x86-64 architecture environment.

Abbreviation	Data Type	C Data Type[1]
NUI8	8-byte unsigned integer in client native (big/little-endian) format	`unsigned long`
FLOAT	IEEE single precision floating point value in little-endian format	`float`
DOUBLE	IEEE double precision floating point value in little-endian format	`double`
X[n]	Array of n elements of data type X.	

2.2 Protocol Modification

Additions to the protocol require careful modification and extension of the exchanged connect options.

The SQL Command Network Protocol must be kept stable so that clients are able to communicate with recent server software versions and vice versa. A more recent client may have to degrade the usage of the protocol depending on the server version while a more recent server may have to only use certain features if non-recent client software is detected.

Detection is performed during the connection by exchanging connect options, which specify the required behavior of the client and getting the supported feature set from the server. This detection does not perform global versioning but enables or disables feature flags, or modifies certain functionality.

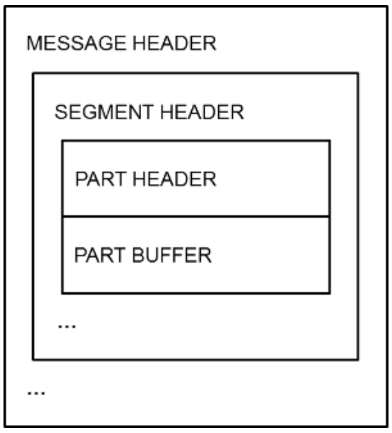
[1] For a Linux GCC in an x86-64 architecture environment.

3 Message Format

The communication between the client and the server is completely synchronous: the client can only send the next request once the reply to the previous request has been fully received.

A client is free to continue with its own processing or to communicate with other servers of an SAP HANA database system while waiting for the response.

A message consists of a fixed part, called the **message header** and a variable length **message buffer**. The message buffer contains **message segments**, which, in turn, consist of a **segment header** and a **segment buffer**. The segment buffer contains parts which have a fixed length **part header** and a variable length **buffer**.



The one exception to this format is during communication initialization when a different message pair is exchanged that is necessary to distinguish the current and former protocol variants.

3.1 Message Header

Specifies the fixed part of the message.

The message header is a 32 byte structure that consists of the following fields:

Field	Data Type	Description
SESSIONID	I8	Specifies a session identifier.
PACKETCOUNT	I4	Specifies a packet sequence number in this session. Packets with the same sequence number belong to one request/response pair.
VARPARTLENGTH	UI4	Specifies the amount of space used in the packet. The maximum size is 2G -1. The VARPARTLENGTH does not include the message header size itself, so the total packet size is VARPARTLENGTH + 32.

VARPARTSIZE	UI4	Specifies the total amount of space in the packet. The maximum size is 2G -1.
NOOFSEGM	I2	Specifies the number of segments in the packet.
PACKETOPTIONS	I1	Specifies 2 if the packet is compressed, 0 otherwise. If the packet is compressed, then the message header and the first segment header are uncompressed and the remainder of the packet is compressed.
RESERVED1	B	Reserved. Do not use.
COMPRESSIONVARPARTLENGTH	UI4	For a compressed packet, specifies the space used in the packet once it is decompressed, excluding the message header size itself.
RESERVED2	B[4]	Reserved. Do not use.

3.2 Segment Header

Specifies a part of the message segment in the message buffer.

The segment header has a length of 24 bytes. There are different segment header structures for request and reply, but both have the same definition for first 13-bytes of the structure. They are structured as follows:

Request Segment Header

SEGMENTLENGTH	SEGMENTOFS	NOOFPARTS	SEGMENTNO	SEGMENTKIND	...
...	MESSAGETYPE	COMMIT	COMMANDOPTIONS	RESERVED1	

Reply Segment Header

SEGMENTLENGTH	SEGMENTOFS	NOOFPARTS	SEGMENTNO	SEGMENTKIND	...
...	RESERVED2	FUNCTIONCODE	RESERVED3		

Field	Data Type	Description
SEGMENTLENGTH	I4	Specifies the length of the segment, including the header.
SEGMENTOFS	I4	Specifies an offset of the segment within the message buffer.
NOOFPARTS	I2	Specifies the number of contained parts.
SEGMENTNO	I2	Specifies the segment number within the packet.

SEGMENTKIND	I1	Specifies the segment kind, which further specifies the layout of the remaining segment header structure.
MESSAGETYPE	I1	Specifies the action requested from the database server.
COMMIT	I1	Specifies a value indicating whether or not the command is committed.
COMMANDOPTIONS	I1	Defines specific options for the sent message.
RESERVED1	B[8]	Reserved, do not use.
RESERVED2	I1	Reserved, do not use.
FUNCTIONCODE	I2	Defines the nature of the statement or functionality that has been prepared or executed.
RESERVED3	B[8]	Reserved, do not use.

3.2.1 Segment Kind

Specifies the last field of the segment header part, common to all types (kinds) of segments.

It defines the segment kind, which further specifies the layout of the remaining segment header structure.

The following values are defined in the order listed below:

Value	Description
0	Reserved for invalid segments. Do not use.
1	Specifies a request segment.
2	Specifies a reply segment.
5	Specifies an error segment (a reply segment containing error).

3.2.2 Message Type

Defines the action requested from the database server.

The following values are defined:

Value	Identifier	Description
0	NIL	Reserved for invalid messages. Do not use.

2	EXECUTEDIRECT	Directly executes an SQL statement.
3	PREPARE	Prepares an SQL statement.
4	ABAPSTREAM	Handles an ABAP stream parameter from a database procedure.
5	XA_START	Starts a distributed transaction in a multi-node SAP HANA system.
6	XA_JOIN	Joins a distributed transaction in a multi-node SAP HANA system.
7	XA_COMMIT	Commits a distributed transaction in a multi-node SAP HANA system.
13	EXECUTE	Executes a previously prepared SQL statement.
16	READLOB	Reads large object data.
17	WRITELOB	Writes large object data.
18	FINDLOB	Finds data in a large object.
25	PING	Reserved, do not use.
65	AUTHENTICATE	Sends authentication data.
66	CONNECT	Connects to the database.
67	COMMIT	Commits the current transaction.
68	ROLLBACK	Rolls back the current transaction.
69	CLOSERESULTSET	Closes a result set.
70	DROPSTATEMENTID	Drops a prepared statement identifier.
71	FETCHNEXT	Fetches the next result from the result set.
72	FETCHABSOLUTE	Moves the cursor to the given row number and fetches the data.
73	FETCHRELATIVE	Moves the cursor a number of rows relative to the position, either positive or negative, and fetches the data.
74	FETCHFIRST	Moves the cursor to the first row and fetches the data.
75	FETCHLAST	Moves the cursor to the last row and fetches the data.
77	DISCONNECT	Reserved (disconnects the session).
78	EXECUTEITAB	Executes a command in Fast Data Access mode.
79	FETCHNEXTITAB	Fetches the next result for an ITAB object in Fast Data Access mode.
80	INSERTNEXTITAB	Inserts the next result for an ITAB object in Fast Data Access mode.

81	BATCHPREPARE	Reserved, do not use.
82	DBCONNECTINFO	Requests and receives database connect information when connecting using the database name.
83	XOPEN_XASTART	Starts work on behalf of a given transaction.
84	XOPEN_XAEND	Ends work on behalf of a given transaction.
85	XOPEN_XAPREPARE	Prepares to commit the work done in the given transaction.
86	XOPEN_XACOMMIT	Commits the work done in the given transaction.
87	XOPEN_XAROLLBACK	Rolls back the work done in the given transaction.
88	XOPEN_XARECOVER	Returns a list of transactions that are in a prepared or heuristic state.
89	XOPEN_XAFORGET	Tells the server to forget about a heuristically completed transaction.

3.2.3 Command Options

Defines specific options for the sent message.

The options are as follows:

Bit	Identifier	Description
1	RESERVED_FIELD	Reserved field. Do not use.
3	HOLD_CURSORS_OVER_COMMIT	Keeps the result set created by this command over commit time.
4	EXECUTE_LOCALLY	Executes a command only on local partitions of affected partitioned table.
5	SCROLL_INSENSITIVE	Marks the result set created by this command as scroll insensitive.

3.2.4 Function Code

Identifies the nature of the statement or functionality that has been prepared or executed.

Value	Identifier	Description
0	NIL	Specifies the invalid command or function.
1	DDL	Specifies the DDL statement.

2	INSERT	Specifies the INSERT statement.
3	UPDATE	Specifies the UPDATE statement.
4	DELETE	Specifies the DELETE statement.
5	SELECT	Specifies the SELECT statement.
6	SELECTFORUPDATE	Specifies the SELECT...FOR UPDATE statement.
7	EXPLAIN	Specifies the EXPLAIN statement.
8	DBPROCEDURECALL	Specifies the CALL statement.
9	DBPROCEDURECALLWITHRESULT	Specifies the CALL statement returning one or more results.
10	FETCH	Specifies the FETCH message.
11	COMMIT	Specifies the COMMIT message or statement.
12	ROLLBACK	Specifies the ROLLBACK message or statement.
13	SAVEPOINT	Reserved, do not use.
14	CONNECT	Specifies the CONNECT or AUTHENTICATION message.
15	WRITELOB	Specifies the WRITELOB message.
16	READLOB	Specifies the READLOB message.
17	PING	Reserved, do not use.
18	DISCONNECT	Specifies the DISCONNECT message.
19	CLOSECURSOR	Specifies the CLOSECURSOR message.
20	FINDLOB	Specifies the FINDLOB message.
21	ABAPSTREAM	Specifies the ABAPSTREAM message for fast data access.
22	XASTART	Specifies the XA_START message.
23	XAJOIN	Specifies the XA_JOIN message.
24	ITABWRITE	Specifies the ITABWRITE message for fast data access.
25	XOPEN_XACONTROL	Specifies the XOPEN control message.
26	XOPEN_XAPREPARE	Specifies the XOPEN_XAPREPARE message.
27	XOPEN_XARECOVER	Specifies the XOPEN_XARECOVER message.

3.3 Part Header

The part header is 16-bytes in length.

The header fields are defined as follows:

Field	Data Type	Description
PARTKIND	I1	Specifies nature of part data.
PARTATTRIBUTES	I1	Specifies further attributes of part.
ARGUMENTCOUNT	I2	Specifies the argument count (the number of elements in part data). If ARGU-MENTCOUNT is -1, then BIGARGUMENTCOUNT specifies the argument count.
BIGARGUMENTCOUNT	I4	Specifies the argument count (the number of elements in part data) when ARGUMENTCOUNT is -1. This value is ignored unless ARGUMENTCOUNT is -1. BIGARGUMENTCOUNT is only supported by some part kinds.
BUFFERLENGTH	I4	Specifies the length of the part buffer in bytes (used space).
BUFFERSIZE	I4	Specifies the length remaining in the packet .

Related Information

Part Kind [page 14]
Part Attributes [page 17]
Part Kind [page 14]
Part Attributes [page 17]

3.3.1 Part Kind

Defines the first section of the part header.

The following values are defined for the PARTKIND field:

Value	Identifier	Description
0	NIL	Reserved for invalid part, do not use.
3	COMMAND	Specifies the SQL command data.

5	RESULTSET	Specifies the tabular result set data.
6	ERROR	Specifies the error information.
10	STATEMENTID	Specifies the prepared statement identifier.
11	TRANSACTIONID	Specifies the transaction identifier.
12	ROWSAFFECTED	Specifies the number of affected rows of a DML statement.
13	RESULTSETID	Specifies the identifier of a result set.
15	TOPOLOGYINFORMATION	Specifies the topology information.
16	TABLELOCATION	Specifies the location of the table data.
17	READLOBREQUEST	Specifies the request data of the READLOB message.
18	READLOBREPLY	Specifies the reply data of the READLOB message.
25	ABAPISTREAM	Specifies the ABAP input stream identifier.
26	ABAPOSTREAM	Specifies the ABAP output stream identifier.
27	COMMANDINFO	Specifies the command information.
28	WRITELOBREQUEST	Specifies the request data of the WRITELOB message.
29	CLIENTCONTEXT	Specifiies the client context information (client version, type, and application name).
30	WRITELOBREPLY	Specifies the reply data of the WRITELOB message.
32	PARAMETERS	Specifies the parameter data.
33	AUTHENTICATION	Specifies the authentication data.
34	SESSIONCONTEXT	Specifies the session context information.
35	CLIENTID	Specifies the client ID `("<PID>@<<hostname>/<computer name>>")`.
38	PROFILE	Specifies the client last send and receive time (in microseconds, -1 means unknown).
39	STATEMENTCONTEXT	Specifies the statement visibility context.
40	PARTITIONINFORMATION	Specifies the table partitioning information.
41	OUTPUTPARAMETERS	Specifies the output parameter data.
42	CONNECTOPTIONS	Specifies the protocol connection options, which are often used to version protocol features.

43	COMMITOPTIONS	Specifies the commit options.
44	FETCHOPTIONS	Specifies the fetch options.
45	FETCHSIZE	Specifies the number of rows to fetch.
47	PARAMETERMETADATA	Specifies the parameter metadata (type and length information).
48	RESULTSETMETADATA	Specifies the result set metadata (type, length, and name information).
49	FINDLOBREQUEST	Specifies the request data of the FINDLOB message.
50	FINDLOBREPLY	Specifies the reply data of the FINDLOB message.
51	ITABSHM	Reserved, do not use.
53	ITABCHUNKMETADATA	Reserved, do not use.
55	ITABMETADATA	Specifies the information on the ABAP ITAB structure for an ITAB transfer for FDA.
56	ITABRESULTCHUNK	Specifies the ABAP ITAB data chunk for FDA.
57	CLIENTINFO	Specifies the client information values (sets session variable values on the server).
58	STREAMDATA	Specifies the ABAP stream data.
59	OSTREAMRESULT	Specifies the ABAP output stream result information.
60	FDAREQUESTMETADATA	Specifies the information on memory and request details for an FDA request.
61	FDAREPLYMETADATA	Specifies the information on memory and request details for an FDA reply.
62	BATCHPREPARE	Reserved, do not use.
63	BATCHEXECUTE	Reserved, do not use.
64	TRANSACTIONFLAGS	Specifies the transaction handling flags.
65	ROWSLOTIMAGEPARAMMETADATA	Reserved, do not use.
66	ROWSLOTIMAGERESULTSET	Reserved, do not use.
67	DBCONNECTINFO	Specifies the location by database name.
68	LOBFLAGS	Specifies the LOB flags.
69	RESULTSETOPTIONS	Specifies the additional context data for result sets.
70	XATRANSACTIONINFO	Specifies the XOPEN XA transaction information.

71	SESSIONVARIABLE	Specifies the delta information from session variable changes.
72	WORKLOADREPLAYCONTEXT	Specifies the context information for workload replays.
73	SQLREPLYPOTIONS	Specifies the parsed information from certain client side encryption-related SQL statements.

3.3.2 Part Attributes

Indicates special features of a certain part.

The following values are defined:

Bit	Identifier	Description
0	LASTPACKET	Specifies the last part in a sequence of parts (FETCH, array command EXECUTE).
1	NEXTPACKET	Specifies a part in a sequence of parts.
2	FIRSTPACKET	Specifies the first part in a sequence of parts.
3	ROWNOTFOUND	Specifies an empty part, which is caused by a row not found error.
4	RESULTSETCLOSED	Specifies the result set that produced this part is closed.
5	RESERVED5	Reserved. Do not use.
6	RESERVED6	Reserved. Do not use.
7	RESERVED7	Reserved. Do not use.

3.4 Part Support in Request Messages

Parts can be used in a variety of request messages.

In the following table, letters denote the relationship between the part and the message:

X	Specifies the default relationship between the part and the message.
A	Specifies that this part can always be sent if distributed transaction handling requires it.
B	Specifies that this part is supported when the prepared statement has input parameters.
C	Specifies that this part is supported when the client application set information flags.

D Specifies that this part is supported when the query contains cached view information.

Message (top) Part Kind (below)	EXECUTEDIRECT	PREPARE	ABAPSTREAM	XA_START	XA_JOIN	EXECUTE	WRITELOB	READLOB	FINDLOB	AUTHENTICATE	CONNECT	COMMIT	ROLLBACK	FETCHNEXT	DISCONNECT	CLOSERESULTSET	DROPSTATEMENTID	EXECUTEITAB	FETCHNEXTITAB	INSERTNEXTITAB
NIL																				
COMMAND	X	X																		
RESULTSET																				
ERROR																				
STATEMENTID						X											X			
TRANSACTIONID					X															
ROWSAFFECTED																				

RESULTSETID			X	X
TOPOLOGYINFORMATION				
TABLELOCATION				
READLOBREQUEST		X		
READLOBREPLY				
ABAPISTREAM	X			
ABAPOSTREAM	X			

COMMANDINFO	X	X				X														
WRITELOBREQUEST							X													
WRITELOBREPLY																				
PARAMETERS						B														
AUTHENTICATION										X	X									
SESSIONCONTEXT	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
STATEMENTCONTEXT	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A

PARTITIONINFORMATION																				
OUTPUTPARAMETERS																				
CONNECTOPTIONS											X									
COMMITOPTIONS												X	X							
FETCHOPTIONS																				
FETCHSIZE														X						

PARAMETERMETADATA			X	
RESULTSETMETADATA				
FINDLOBREQUEST		X		
FINDLOBREPLY				
ITABSHM				
ITABCHUNKMETADATA				
ITABMETADATA				

ITABRESULTCHUNK																				
CLIENTINFO	C	C				C								C						
STREAMDATA			X																	
OSTREAMRESULT			X																	
FDAREQUESTMETADATA																				
FDAREPLYMETADATA																				
BATCHPREPARE																				

BATCHEXECUTE	
TRANSACTIONFLAGS	
RESULTSETOPTIONS	D

3.5 Part Support in Reply Messages

Parts can be used in a variety of reply messages.

In the following table, the letters denote the relationship between the part and the message:

X	Specifies the default relationship between the part and the message.
A	Specifies that this part is supported when the message yields one or more result sets.
B	Specifies a reply in case of errors while processing the statement.
C	Specifies that this part is supported when the statement generates a row count of affected rows.
D	Specifies if the statement caused an action in transaction handling (commit, rollback, or start of a new write transaction).
E	Specifies that this part is available if the server has information that is useful to applying statement routing (see the corresponding Usage Scenario Statement Routing).
F	Specifies that this part is supported when the statement has input and/or output parameters.
G	Specifies that this part is supported when the statement has output parameters.
H	Indicates a possible change of visible version.

I	Specifies that this part is supported when the parameter metadata has changed due to schema changes.

Message (top) Part Kind (below)	EXECUTEDIRECT	PREPARE	ABAPSTREAM	XA_START	XA_JOIN	EXECUTE	WRITELOB	READLOB	FINDLOB	AUTHENTICATE	CONNECT	COMMIT	ROLLBACK	FETCHNEXT	DISCONNECT	CLOSERESULTSET	DROPSTATEMENTID	EXECUTEITAB	FETCHNEXTITAB	INSERTNEXTITAB
NIL																				
COMMAND																				
RESULTSET	A					A														
ERROR	B	B	B	B	B	B	B	B	B											
STATEMENTID		X																		
TRANSACTIONID				X																
ROWSAFFECTED	C					C														

RESULTSETID	A					A		
TOPOLOGYINFORMATION								
TABLELOCATION		E						
READLOBREQUEST								
READLOBREPLY								X
ABAPISTREAM			X					
ABAPOSTREAM			X					

COMMANDINFO																				
WRITELOBREQUEST																				
WRITELOBREPLY							X													
PARAMETERS						B														
AUTHENTICATION										X										
SESSIONCONTEXT																	A	A		
STATEMENTCONTEXT	H	H	H	H	H	H	H	H	H											

PARTITIONINFORMATION	E	
OUTPUTPARAMETERS		G
CONNECTOPTIONS		
COMMITOPTIONS		
FETCHOPTIONS		
FETCHSIZE		

PARAMETERMETADATA		F	I	
RESULTSETMETADATA	A	A	A	
FINDLOBREQUEST				
FINDLOBREPLY				X
ITABSHM				
ITABCHUNKMETADATA				
ITABMETADATA				

ITABRESULTCHUNK				
CLIENTINFO	C		C	C
STREAMDATA		X		
OSTREAMRESULT		X		
FDAREQUESTMETADATA				
FDAREPLYMETADATA				
BATCHPREPARE				

BATCHEXECUTE								
TRANSACTIONFLAGS	D	D	D	D	D	D	D	D
LOBFLAGS						X		

3.6 Type Codes

Identify the type of a field transferred to or from the database.

Not all type codes known and processed internally are used in the protocol and clients may support a different level of type support, depending on their version. The protocol mechanism (connect options) for connecting to the database ensures that the client sends the server data it is able to process and that the client receives only data that it can process from the server.

Type codes are in the range of 0-127, as the most significant bit is used on input to signal a NULL value of a certain type.

Value	Identifier	Description	Support Level
0	NULL	Specifies a NULL value.	-
1	TINYINT	Specifies a TINYINT value.	1
2	SMALLINT	Specifies a SMALLINT value.	1
3	INT	Specifies an INTEGER value.	1
4	BIGINT	Specifies a BIGINT value.	1
5	DECIMAL	Specifies a DECIMAL and DECIMAL(p,s) value.	1
6	REAL	Specifies a REAL value.	1

7	DOUBLE	Specifies a DOUBLE value.	1
8	CHAR	Specifies a CHAR value.	1
9	VARCHAR	Specifies a VARCHAR value.	1
10	NCHAR	Specifies an NCHAR (Unicode character type) value.	1
11	NVARCHAR	Specifies an NVARCHAR (Unicode character type) value.	1
12	BINARY	Specifies a BINARY value.	1
13	VARBINARY	Specifies a VARBINARY value.	1
14	DATE	Specifies a DATE (deprecated type) value.	1 (deprecated with 3)
15	TIME	Specifies a TIME (deprecated type) value.	1 (deprecated with 3)
16	TIMESTAMP	Specifies a TIMESTAMP (millisecond precision) value.	1 (deprecated with 3)
17	TIME_TZ	Reserved, do not use.	-
18	TIME_LTZ	Reserved, do not use.	-
19	TIMESTAMP_TZ	Reserved, do not use.	-
20	TIMESTAMP_LTZ	Reserved, do not use.	-
21	INTERVAL_YM	Reserved, do not use.	-
22	INTERVAL_DS	Reserved, do not use.	-
23	ROWID	Reserved, do not use.	-
24	UROWID	Reserved, do not use.	-
25	CLOB	Specifies a character LOB value.	1
26	NCLOB	Specifies a Unicode character LOB value.	1
27	BLOB	Specifies a binary LOB value.	1
28	BOOLEAN	Specifies a BOOLEAN value.	7
29	STRING	Specifies a character string value.	1
30	NSTRING	Specifies a Unicode character string value.	1
31	BLOCATOR	Specifies a binary locator value.	1

32	NLOCATOR	Specifies a Unicode character locator value.	1
33	BSTRING	Specifies a binary string value.	1
34	DECIMAL_DIGIT_ARRAY	Reserved, do not use.	-
35	VARCHAR2	Specifies a VARCHAR value.	-
45	TABLE	Reserved, do not use.	-
48	ABAPITAB	Specifies an ABAPSTREAM procedure parameter value.	1
49	ABAPSTRUCT	Specifies an ABAP structure procedure parameter value.	1
50	ARRAY	Specifies an ARRAY value.	7
51	TEXT	Specifies a TEXT data type value.	3
52	SHORTTEXT	Specifies a SHORTTEXT data type value.	3
53	BINTEXT	Reserved, do not use.	-
55	ALPHANUM	Specifies an ALPHANUM data type value.	3
61	LONGDATE	Specifies a TIMESTAMP data type value.	3
62	SECONDDATE	Specifies a TIMESTAMP type with second precision value.	3
63	DAYDATE	Specifies a DATE data type value.	3
64	SECONDTIME	Specifies a TIME data type value.	3
70	CLOCATOR	Reserved, do not use.	-
71	BLOB_DISK	Reserved, do not use.	-
72	CLOB_DISK	Reserved, do not use.	-
73	NCLOB_DISK	Reserved, do not use.	-
74	GEOMETRY	In general, this is reserved and should not be used. It should only be used when SPATIALTYPES=1 or SPATIALTYPES=2 in SQLDBC.	-
75	POINT	In general, this is reserved and should not be used. It should only be used when SPATIALTYPES=1 or SPATIALTYPES=2 in SQLDBC.	-
76	FIXED16	Reserved, do not use.	-

77	ABAP_ITAB	Reserved, do not use.	-

3.7 Part Data Format

Parts are transmitted in messages between the client and the SAP HANA database server.

The format of the data transmitted is uniquely identified by the part kind. Some part kinds share the same format as the contained data.

3.7.1 Option Part Data Format

Specifies a common format for transmitting options (typed key-value pairs).

An option part contains elements with the following structure:

OPTIONNAME	OPTIONTYPE	OPTIONVALUE

The fields are defined as follows:

Field	Data Type	Description
OPTIONNAME	I1	The option key.
OPTIONTYPE	I1	The type code of the option value.
OPTIONVALUE	B[...]	The option data.

Only some type codes apply to the option type field. The format of the option data is:

Type Code	Value Format
BOOLEAN	1-byte: zero for false and non-zero for a true value.
INT	A 4-byte signed integer (int) in little-endian format.
BIGINT	An 8-byte signed integer (long) in little-endian format.
STRING	A 2-byte signed integer (short) in little-endian format. This value is followed by the string in CESU-8 encoding (the number of bytes is specified by the length information).
BSTRING	A 2-byte signed integer (short) in little-endian format. This value is followed by the binary string (the number of bytes is specified by the length information).
DOUBLE	An IEEE double precision value in little-endian format.

3.7.2 Multi-Line Option Part Data Format

Specifies a common format to transmit collections of options (typed key-value pairs).

An option part contains rows having the following structure (the number of rows is the part argument count value):

ARGCOUNT	OPTION	OPTION	. . .

The fields are defined as follows:

Field	Data Type	Description
ARGCOUNT	I2	Specifies the number of options in a row.
OPTION		Specifies the option element as described in the option part format.

Different rows can have a different argument count, so the format is not completely tabular.

3.7.3 COMMAND Part Data Format

Contains the text of an SQL command.

For JDBC, the text is always encoded in CESU-8.

3.7.4 RESULTSET Part Data Format

Contains the row data of a result set.

The fields of the individual rows are formatted as defined in the output field format section. One result set part contains as many rows as the argument count field indicates:

ROW1/COL1	ROW1/COL2	. . .	ROW1/COLn
ROW2/COL1	. . .		ROW2/COLn
. . .			
ROWn/COL1	ROWn/COL2	. . .	ROWn/COLn

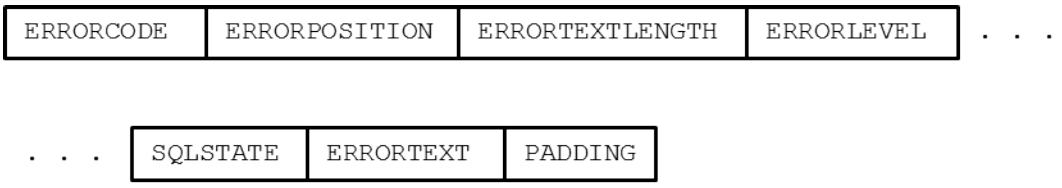
i Note

The field values may have variable lengths, depending on the data type. Therefore, all the values in the current row must be scanned before moving to the next row.

3.7.5 ERROR Part Data Format

Contains one or more errors returned by the database server to the client.

The error part contains elements with the following structure:



The fields are defined as follows:

Field	Data Type	Description
ERRORCODE	I4	Specifies the error code as sent from the database server.
ERRORPOSITION	I4	Specifies the position of the error.
ERRORTEXTLENGTH	I4	Specifies the length of the error text.
ERRORLEVEL	I1	Specifies the error level.
SQLSTATE	B[5]	Specifies the SQLSTATE as defined by the SQL standard for this message.
ERRORTEXT	B[ERRORTEXTLENGTH]	Specifies the error text or error message in CESU-8 encod-ing.
PADDING	B[0..7]	Provides padding so the length of structure is aligned to 8.

The error level defines the level of the message as a warning, error, or fatal session-terminating error using the following values:

Value	Identifier	Description
0	WARNING	Specifies the warning message.
1	ERROR	Specifies the error message: this statement (or part of it) has failed.
2	FATALERROR	Specifies the fatal error message: session is in an unusable state now.

3.7.6 STATEMENTID Part Data Format

Contains a statement identifier, which is a byte array of 8-bytes in length.

3.7.7 TRANSACTIONID Part Data Format

Contains the identifier of a transaction.

This is a byte array of variable length and is stored in the same way as a VARBINARY output value.

The transaction identifier is opaque to the client program.

3.7.8 ROWSAFFECTED Part Data Format

Contains one or multiple (as many as the part's ARGUMENTCOUNT field defines) 32-bit integers.

Indicates the number of rows affected by a statement or a row of an array command.

The following special values are defined to indicate certain situations:

Value	Identifier	Description
-2	SUCCESS_NO_INFO	Specifies that a statement or row has been processed but the number of affected rows cannot be determined.
-3	EXECUTION_FAILED	Specifies that the execution of a statement or processing of a row has failed.

3.7.9 RESULTSETID Part Data Format

Contains the identifier of a result set, which is a byte array of 8-bytes in length.

The result set identifier is opaque to the client program.

3.7.10 TOPOLOGYINFORMATION Part Data Format

Contains information about the server topology.

This is a multi-line option part with the following options defined:

Value	Identifier	Data Type	Description

1	HOSTNAME	STRING	Specifies the externally visible TCP/IP address of the server host name.[2]
2	HOSTPORT-NUMBER	INT	Specifies the SQL port number.
3	TENANT-NAME	STRING	Reserved, do not use.
4	LOADFAC-TOR	DOUBLE	Specifies the load factor in round-robin scheduling. The default is 1.0.
5	SITEVOLU-MEID	INT	Specifies a SITE ID when the byte count is high (0-255 bytes) and a Volume ID when the byte count is lower.
6	ISMASTER	BOOLEAN	Specifies the master server flag. This field contains 1 (true) if the server is the master server of the SAP HANA database system.
7	ISCURRENT-SESSION	BOOLEAN	Specifies the current session flag. This field contains 1 (true) if the server that manages the session has returned this topology information.
8	SERVICE-TYPE	INT	Specifies the service type.
9	NETWORK-DOMAIN	STRING	Specifies the network domain of the server if a non-empty domain is defined.[1]
10	ISSTANDBY	BOOLEAN	Specifies the standby server flag. This field contains a 1 (true) if the server is a standby server.
11	ALLIPAD-DRESSES	STRING	Specifies all TCP/IP addresses assigned to the node.[1]
12	ALLHOST-NAMES	STRING	Specifies all host names assigned to the node.[1]
13	SITETYPE	INTEGER	Specifies the site role in an SAP HANA System Replication scenario. This field contains a 0 if there is no SAP HANA System Replication enabled, 1 for the primary site, and 2 for the secondary site.

i Note

1 Deprecated: not sent by the server.

2	Externally-visible IP address/name.

3.7.11 TABLELOCATION Part Data Format

Contains one or more 4-byte integers in little-endian format, which identify the data volume IDs.

The argument count of the part is specified by the number of elements. The returned volume IDs signal preferred execution locations for the statement and depend on the following:

- The nature of the SQL command that is prepared (some commands can only be executed on a certain server)
- The number of table partitions that are affected by the statement
- The location of the tables that are affected by the statement

3.7.12 READLOBREQUEST Part Data Format

Requests BLOB/CLOB/NCLOB data to be read.

The structure has the following layout:

LOCATORID	READOFFSET	READLENGTH	FILLER

The fields are defined as follows:

Field	Data Type	Description
LOCATORID	B[8]	Specifies the identifier of the BLOB/CLOB/NCLOB locator.
READOFFSET	I8	Specifies the offset within a large object for reading, starting with 1.
READLENGTH	I4	Specifies the length to read.
FILLER	B[4]	Reserved, do not use.

For byte-based large object types, the READOFFSET and READLENGTH define byte positions and length values. For character-based large object types, these fields define character positions and length values within the large object.

3.7.13 READLOBREPLY Part Data Format

Answers a read LOB request from the client.

The READLOBREPLY part has the following format:

LOCATORID	OPTIONS	CHUNKLENGTH	FILLER	DATA

The fields are defined as follows:

Field	Data Type	Description
LOCATORID	B[8]	Specifies the identifier of BLOB/CLOB/NCLOB locator.
OPTIONS	I1	Specifies the result options (see table below).
CHUNKLENGTH	I4	Specifies the length of the data in a part.
FILLER	B[3]	Reserved, do not use.
DATA	B[CHUNKLENGTH]	Specifies large object data.

The OPTIONS field is a bit set that has the following defined values:

Bit	Identifier	Description
0	NULLINDICATOR	Specifies that the large object value is NULL (not used for READLOBREPLY).
1	DATAINCLUDED	Specifies that data is included (not used for READLOBREPLY).
2	LASTDATA	Specifies that there is no additional data remaining in the large object.

3.7.14 ABAPISTREAM Part Data Format

Requests an ABAP input stream for C++ procedure processing and is sent as a response from the client when sending the stream data.

The part content is structured as follows:

ABAPTABID	MASK

The fields are defined as follows:

Field	Data Type	Description
ABAPTABID	I4	Specifies the identifier of the ABAP table.
MASK	B[...]	Specifies the bit field that is used to mask ABAP table entries so that they are not transferred.

The MASK field may be omitted completely. In this case, all columns of the ABAP table are transferred. Otherwise, only the marked values are transferred, enabling a projection of the ABAP table in the server processing. The number of rows requested is supplied in the argument count of the ABAPISTREAM part. In the case that the client sends this part to identify stream data transmitted to the database server, only the ABAPTABID field is used.

3.7.15 ABAPOSTREAM Part Data Format

Contains one 4-byte integer value identifying the ABAPTABID of the table.

The data is sent in the corresponding STREAMDATA part.

3.7.16 COMMANDINFO Part Data Format

Identifies the location of a statement in the source program for debugging and analysis purposes. This is an options part.

The COMMANDINFO part supports the following option values:

Value	Identifier	Data Type	Description
1	LINENUMBER	INT	Specifies the line number in the source.
2	SOURCEMODULE	STRING	Specifies the name of the source module.

3.7.17 WRITELOBREQUEST Part Data Format

Writes data, piece-wise, into a large object.

WRITELOBREQUEST contains elements with the following structure:

LOCATORID	OPTIONS	WRITEOFFSET	CHUNKLENGTH	DATA

The fields are defined as follows:

Field	Data Type	Description
LOCATORID	B[8]	The BLOB/CLOB/NCLOB locator.
OPTIONS	I1	The request options.
WRITEOFFSET	I8	The offset within the large object data. An offset of -1 issues a request to append the data to an existing large object data.
CHUNKLENGTH	I4	The length of the data that follows.
DATA	B[CHUNKLENGTH]	The large object data.

A number of large objects can be written using this request. The argument count of the part describes how many elements are contained in the part.

The OPTIONS field is a bit set which has the following defined values:

Bit	Identifier	Description
0	NULLINDICATOR	Specifies that the large object value is NULL (not used for WRITELOBRE-QUEST).
1	DATAINCLUDED	Specifies that the data is included.
2	LASTDATA	Specifies that there is no more data remaining.

3.7.18 WRITELOBREPLY Part Data Format

Provides a reply from the server to a WRITELOBREQUEST part.

WRITELOBREPLY contains the LOCATORID identifiers for all large object locators that can still receive data (if LASTDATA has not been set). The argument count defines the number of locator identifiers contained in the data.

3.7.19 PARAMETERS Part Data Format

Contains input parameters.

The parameters are densely packed and use the input field format. The argument count of the part defines how many parameter rows are included.

Related Information

Input and Output Field Formats [page 61]
Input and Output Field Formats [page 61]

3.7.20 AUTHENTICATION Part Data Format

Contains information exchanged to perform client authentication.

AUTHENTICATION has the following general structure:

FIELDCOUNT	FIELD1	FIELD2	. . .

This structure is defined as follows:

Field	Data Type	Description
FIELDCOUNT	BI2	Specifies the number of fields.
FIELD		Specifies the field data (see the tables below).

Field data has the following structure, depending on its length:

Less than or Equal to 250-Bytes

FIELDLENGTH	FIELDDATA

Field	Data Type	Description
FIELDLENGTH	I1	Specifies the length of the data.
FIELDDATA	B[FIELDLENGTH]	Specifies the field data.

Larger than 250-Bytes

FIELDINDICATOR	FIELDLENGTH	FIELDDATA

Field	Data Type	Description
FIELDINDICATOR	I1	Always specifies 0xFF.
FIELDLENGTH	BI2	Specifies the field length.
FIELDDATA	B[FIELDLENGTH]	Specifies the field data.

Initial Request

The initial request contains the database user name and field pairs (method name/value) for each authentication method requested by the client.

Method Name	Description
PLAINPASSWORD	Reserved. Do not use.
SCRAMMD5	Reserved. Do not use.
SCRAMSHA256	Provides password-based authentication.
GSS	Provides GSS/Kerberos authentication.
SAML	Provides SAML authentication.

Additional Requests

Following the initial request, subsequent requests contain a number of field pairs for authentication methods that are performed by exchanging messages.

3.7.21 SESSIONCONTEXT Part Data Format

The SESSIONCONTEXT part is an option part.

The following options are defined:

Value	Identifier	Data Type	Description
1	PRIMARYCONNECTIONID	INT	Specifies the ID of the primary connection.
2	PRIMARYHOSTNAME	STRING	Specifies the host name of the primary connection host.
3	PRIMARYHOSTPORTNUMBER	INT	Specifies the port number of the SQL port for the primary connection.
4	MASTERCONNECTIONID	INT	Specifies the connection ID of the transaction master.
5	MASTERHOSTNAME	STRING	Specifies the host name of the transaction master connection host.
6	MASTERHOSTPORTNUMBER	INT	Specifies the port number of the SQL port for the transaction master connection.

The **primary** connection is the first connection that is opened by the client program to the system in a distributed scenario. The **master** connection is the connection that starts the current distributed transaction.

3.7.22 STATEMENTCONTEXT Part Data Format

Provides previously received statement context information.

The STATEMENTCONTEXT part is an option part containing the following:

Value	Identifier	Data Type	Description
1	STATEMENTSEQUENCEINFO	BINARY	Specifies the information on the statement sequence within the transaction.
2	SERVERPROCESSINGTIME	BIGINT	Specifies the server processing time, in microseconds.

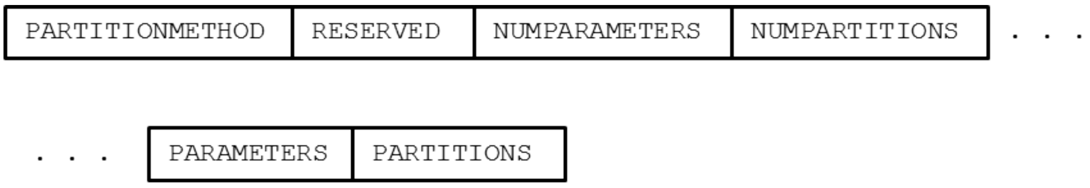
3	SCHEMANAME	STRING	Specifies the current schema after the schema change.
4	FLAGSET	B[1]	Specifies the flags that determine client routing behavior.
5	QUERYTIMEOUT	I8	Specifies the number of seconds before the query is timed out by the server.

The binary option content is opaque to the client application.

3.7.23 PARTITIONINFORMATION Part Data Format

Allows a specific pruning of input data as it pertains to the partitioning of data in the database server.

The PARTITIONINFORMATION part is structured as follows:



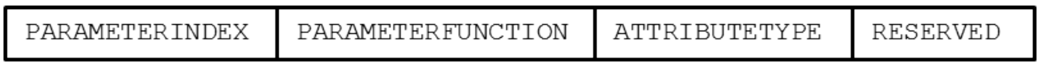
The fields are defined as follows:

Field	Data Type	Description
PARTITIONMETHOD	I1	Specifies the method of partitioning.
RESERVED	B[7]	Reserved. Do not use.
NUMPARAMETERS	I4	Specifies the number of subsequent parameter descriptors.
NUMPARTITIONS	I4	Specifies the number of subsequent partition descriptors.

The PARTITIONMETHOD field supports the following values:

Value	Identifier	Description
0	INVALID	Invalid method. Do not use.
1	ROUNDROBIN	Specifies that partitions are assigned using a round-robin specification.
2	HASH	Specifies that partitions are assigned using a hash specification.

The PARAMETERS field contains NUMPARAMETERS elements having the following structure:



The fields of this structure are defined as follows:

Field	Data Type	Description
PARAMETERINDEX	I4	Specifies the parameter index in the statement.
PARAMETERFUNCTION	I1	Specifies the function to apply to the parameter before the hash.
ATTRIBUTETYPE	I1	Specifies the storage type to be used for the hash calculation.
RESERVED	B[2]	Reserved, do not use.

The PARAMETERFUNCTION field supports the following values:

Value	Identifier	Description
0	INVALID	Invalid method, do not use.
1	YEAR	Specifies the extracted year part of the DATE.
2	MONTH	Specifies the extracted month part of the DATE.

The ATTRIBUTETYPE field supports the following values:

Value	Identifier	Description	SQL Data Type[2]
0	INVALID	Specifies an invalid type.	
65	ALPHANUM	Specifies a character string consisting of only numbers.	ALPHANUM
83	STRING	Specifies a character string in CESU-8 format.	NVARCHAR
67	FIXEDSTRING	Specifies a character string in CESU-8 format.	NCHAR
86	TEXT	Specifies the text.	SHORTTEXT
73	INT	Specifies a 32-bit integer value.	INTEGER
70	FLOAT	Specifies the IEEE float type.	
68	DATE	Specifies the date.	
64	LONGDATE	Specifies the LONGDATE value.	LONGDATE
116	TIME	Specifies the time.	
84	TEXT_OLD	Specifies the old text.	

[2] This reflects the default mapping of SQL data types to column store types.

Value	Identifier	Description	SQL Data Type[2]
66	FIXED	Specifies the DECIMAL value.	DECIMAL
100	DOUBLE	Specifies the IEEE double value.	DOUBLE
69	UNITDECFLOAT	Specifies the UNITDECFLOAT value.	
82	RAW	Specifies a binary string.	VARBINARY
77	DECIMAL_FLOAT	Specifies the decimal float.	
76	SDFLOAT	Specifies the SDFLOAT value.	
115	SECONDDATE	Specifies the SECONDDATE value.	SECONDDATE
101	DAYDATE	Specifies the DAYDATE value.	DAYDATE
117	SECONDTIME	Specifies the SECONDTIME value.	SECONDTIME

3.7.24 OUTPUTPARAMETERS Part Data Format

Specifies that the format of the OUTPUTPARAMETERS part is the same as the format of the RESULTSET part.

The part contains the scalar output parameter values returned by the statement. An empty output parameter part can be sent if there are no scalar output parameters in the procedure definition, but non-scalar output parameters (tables) instead. The argument count is always 1.

3.7.25 CONNECTOPTIONS Part Data Format

Specifies the connection options.

The connect option part includes the following:

Value	Identifier	Data Type	Description
1	CONNECTIONID	INT	Specifies the connection ID This field is filled by the server when the connection is established. This number can be used in DISCONNECT/KILL commands for command or session cancellation.

[2] This reflects the default mapping of SQL data types to column store types.

Value	Identifier	Data Type	Description
2	COMPLETEARRAYEXECUTION	BOOLEAN	Specifies the complete array execution feature. This field is set if array commands continue to process remaining input when detecting an error in an input row. This is always set for the current client and server.
3	CLIENTLOCALE	STRING	Specifies the locale set by the client. The locale is used in language-dependent handling within the SAP HANA database calculation engine.
4	SUPPORTSLARGEBULKOPERATIONS	BOOLEAN	Set by the server to process array commands.
5	DATAFORMATVERSION2	INTEGER	Sends spatial types ST_POINT, ST_POINTZ, and ST_GEOMETRY to the client. There is no change in behavior for the applications unless the connect option SPATIALTYPES is set. If it is set, the ParameterMetadata and ResultMetadata types are used.
6			Reserved. Do not use.
7			Reserved. Do not use.
8			Reserved. Do not use.
9			Reserved. Do not use.
10	LARGENUMBEROFPARAMETERSSUPPORT	BOOLEAN	Contains the host name of the server without any domain part. This identifier is filled by the server with the host name and resolves so that it does not contain a database server alias.
11	SYSTEMID	STRING	Specifies the ID of the system. This option is set by the server and filled with the SAPSYSTEMNAME of the connected instance for tracing and supportability purposes.
12			Reserved, do not use.

Value	Identifier	Data Type	Description
13	ABAPVARCHARMODE	BOOLEAN	Specifies the ABAP VARCHAR mode flag. This field is set by the client to indicate that the connection should honor the ABAP character handling, that is: • Trailing space of character parameters and column values is not significant. • Trailing space in character literals is not relevant. For example, the character literal '' is identical to the character literal' '.
14	SELECTFORUPDATESUPPORTED	BOOLEAN	Indicates that the client is able to handle the special function code for SELECT ... FOR UPDATE commands.
15	CLIENTDISTRIBUTIONMODE	INT	Specifies the mode of distribution in the client. The server sets this field to the appropriate support level depending on the client value and its own configuration. For more information, see the table below.
16	ENGINEDATAFORMATVERSION	INT	Specifies the level of data type format support. The server sets this field to the maximum version that it can support. The possible values correspond to the DATAFORMATVERSION flag.
17	DISTRIBUTIONPROTOCOLVERSION	INT	Specifies the level of distribution protocol support. The server may choose to disable distribution if the support level is not sufficient. CLIENTDISTRIBUTIONMODE is OFF if a value less than one (<1) is returned by the server. For more information, see the table below.
18	SPLITBATCHCOMMANDS	BOOLEAN	Specifies if configuration allows splitting batch (array) commands for parallel execution.

Value	Identifier	Data Type	Description
19	USETRANSACTIONFLAGSONLY	BOOLEAN	Specifies the transaction flag usage. This field is sent by the server to indicate that the client should gather the state of the current transaction only from the TRANSACTIONFLAGS command, not from the nature of the command (DDL, UPDATE, and so on).
20	ROWANDCOLUMNOPTIMIZEDFORMAT	BOOLEAN	Reserved, do not use.
21	IGNOREUNKNOWNPARTS	BOOLEAN	Specifies that the server should ignore unknown parts of the communication protocol instead of raising a fatal error.
22	TABLEOUTPUTPARAMETER	BOOLEAN	Specifies the table type as the supported output parameter type. This field is sent by the client to indicate that it understands output parameters described by type code TABLE in result sets.
23	DATAFORMATVERSION2	INTEGER	Specifies the data format version. The client indicates this set of understood type codes and field formats and then the server defines the value according to its own capabilities and sends it back. For more information, see the table below.
24	ITABPARAMETER	BOOLEAN	Specifies the support of ABAP ITAB parameter in statements. This field is sent by the server to signal that it understands the ABAP ITAB parameters of SQL statements (For-All-Entries Optimization).
25	DESCRIBETABLEOUTPUTPARAMETER	BOOLEAN	Specifies the request table output parameter metadata in the session. The returned type of the table output parameter is either STRING or TABLE, depending on the TABLEOUTPUTPARAMETER connect option.
26	COLUMNARRESULTSET	BITVECTOR	Specifies the list of types for which columnar result set format is supported.

Value	Identifier	Data Type	Description
27	SCROLLABLERESULTSET	INTEGER	Specifies the supported scrollable result sets.
28	CLIENTINFONULLVALUESSUPPORTED	BOOLEAN	Specifies that the server can handle null values in client information.
29	ASSOCIATEDCONNECTIONID	INTEGER	Specifies the connection ID of the associated connection. This is used only in dynamic tiering scenarios.
30	NONTRANSACTIONALPREPARE	BOOLEAN	Specifies what the server can handle and uses a non-transactional prepare operation.
31	FDAENABLED	BOOLEAN	Specifies that Fast Data Access (FDA) is enabled.
32	OSUSER	STRING	Specifies the user name of the current user on the client operating system.
33	ROWSLOTIMAGERESULT	BITVECTOR	Specifies the list of types for which row-slot image result set format is supported.
34	ENDIANNESS	INTEGER	Specifies the Endianness (big or little endian) of the server.
35			Reserved. Do not use.
36			Reserved. Do not use.
37	IMPLICITLOBSTREAMING	BOOLEAN	Indicates whether the server supports implicit LOB streaming when autocommit is on instead of raising an error.

For DATAFORMATVERSION2, the following values are supported:

Value	Description
1	Specifies the baseline data type support.
3	Specifies the extended data type support. ALPHANUM, TEXT, SHORTTEXT, LONGDATE, SECONDDATE, DAYDATE, SECONDTIME are supported without translation. Deprecated, do not use.
4	Specifies support for ALPHANUM, TEXT, SHORTTEXT, LONGDATE, SECONDDATE, DAYDATE, and SECONDTIME.
6	Specifies to send the data type BINTEXT to the client.

For CLIENTDISTRIBUTIONMODE, the following values are supported:

Value	Description
0	Specifies OFF. There is no routing or distributed transaction handling.
1	Specifies CONNECTION. The client can connect to any master/slave server in the topology. Connections are enabled so that the connection load on the nodes is balanced.
2	Specifies STATEMENT. The server returns information about which node is preferred for executing the statement. If possible, clients execute on that node.
3	Specifies STATEMENT_CONNECTION for both the STATEMENT and CONNECTION level.

For DISTRIBUTIONPROTOCOLVERSION, the following values are supported:

Value	Description
0	Specifies the baseline version.
1	Specifies that the client handles the statement sequence number information (statement context part handling).

CLIENTDISTRIBUTIONMODE is OFF if a value less than one (<1) is returned by the server.

Connect Option Usage

The following table further illustrates the use of the connect options. An option can depend on:

- The client parameter (set in the client to change server behavior).
- The server parameter (set in the server configuration to enable or disable the option).
- The server and client version (if a feature was introduced that needs to be in sync between a client and server).

More than one of these statements can be true for any option.

Option	Client Parameter	Server Parameter	Version-Dependent
CONNECTIONID	-	-	-
COMPLETEARRAYEXECUTION	-	-	X
CLIENTLOCALE	X	-	-
SUPPORTSLARGEBULKOPERATIONS	-	-	X
LARGENUMBEROFPARAMETERSSUPPORT	-	X	X

Option	Client Parameter	Server Parameter	Version-Dependent
SYSTEMID	-	-	-
DATAFORMATVERSION	X	-	X
ABAPVARCHARMODE	X	-	X
SELECTFORUPDATESUPPORTED	-	-	X
CLIENTDISTRIBUTIONMODE	X	X	X
ENGINEDATAFORMATVERSION	-	-	X
DISTRIBUTIONPROTOCOLVERSION	-	-	X
SPLITBATCHCOMMANDS	X	X	X
USETRANSACTIONFLAGSONLY	-	-	X
IGNOREUNKNOWNPARTS	-	-	X
TABLEOUTPUTPARAMETER	X	-	X
ITABPARAMETER	-	X	X
DESCRIBETABLEOUTPUTPARAMETER	X	-	X

3.7.26 COMMITOPTIONS Part Data Format

Specifies an option part.

The COMMITOPTIONS part uses the following option:

Value	Identifier	Data Type	Description
1	HOLDCURSORSOVERCOMMIT	BOOLEAN	Specifies the hold cursors.

If HOLDCURSORSOVERCOMMIT is set by the client on commit, then all cursors are held, including those marked explicitly as HOLD. This is not currently used by any client interface implementation.

3.7.27 FETCHOPTIONS Part Data Format

Specifies an option part.

The FETCHOPTIONS part uses the following identifier:

Value	Identifier	Data Type	Description
1	RESULTSETPOS	INT	Specifies the position for FETCH.

The RESULTSETPOS field can be used to skip over entries when fetching. Currently not used by any client interface implementation.

3.7.28 FETCHSIZE Part Data Format

Contains one 4-byte integer value (little-endian) that defines the number of rows that are being fetched by the application.

3.7.29 PARAMETERMETADATA Part Data Format

Contains the input parameters of prepared statements and additional output parameters of stored procedure call statements.

A PARAMETERMETADATA part starts with an array of entries with the following structure:

PARAMETEROPTIONS	DATETYPE	MODE	FILLER1	NAMEOFFSET	LENGTH	FRACTION	FILLER2

The fields of this structure are defined as follows:

Field	Data Type	Description
PARAMETEROPTIONS	I1	Specifies options that further refine parameter details.
DATATYPE	I1	Specifies the data type parameter (type code).
MODE	I1	Specifies whether the parameter is an input or an output parameter.
FILLER1	I1	Reserved, do not use.
NAMEOFFSET	UI4	Specifies the offset of the parameter name in part. Set to 0xFFFFFFFF to signal no name.
LENGTH	I2	Specifies the length/precision of the parameter.
FRACTION	I2	Specifies the scale of the parameter.

Field	Data Type	Description
FILLER2	I4	Reserved, do not use.

The PARAMETEROPTIONS field is a bit set that has the following defined values:

Bit	Identifier	Description
0	MANDATORY	Specifies that the parameter is not nullable (must not be set to NULL).
1	OPTIONAL	Specifies that the parameter is nullable.
2	DEFAULT	Specifies that the parameter has a defined DEFAULT value.
3		Reserved, do not use.
4		Reserved, do not use.
5		Reserved, do not use.

The MODE field is a bit set that defines the direction of a parameter:

Bit	Identifier	Description
0	IN	Specifies that the parameter direction is IN.
1	INOUT	Specifies that the parameter direction is INOUT.
2	OUT	Specifies that the parameter direction is OUT.

The array of parameter descriptions can be followed by the names of the parameters. Each name is written in the following format:

LENGTH	NAME

Field	Data Type	Description
LENGTH	UI1	Specifies the length of the name in bytes.
NAME	B[LENGTH]	Specifies the name.

3.7.30 RESULTSETMETADATA Part Data Format

Contains result column metadata (type information).

A RESULTSETMETADATA part starts with an array of entries with the following structure:

```
| COLUMNOPTIONS | DATATYPE | FRACTION | LENGTH | FILLER | TABLENAMEOFFSET | . . .

. . . | SCHEMANAMEOFFSET | COLUMNNAMEOFFSET | COLUMNDISPLAYNAMEOFFSET |
```

The fields of this structure are defined as follows:

Field	Data Type	Description
COLUMNOPTIONS	I1	Specifies the options that further refine column details.
DATATYPE	I1	Specifies the parameter data type (type code).
FRACTION	I2	Specifies the scale of the column.
LENGTH	I2	Specifies the length and precision of the column.
FILLER	I2	Reserved, do not use.
TABLENAMEOFFSET	UI4	Specifies the offset of the table name in the part. Set to 0xFFFFFFFF to signal no available name.
SCHEMANAMEOFFSET	UI4	Specifies the offset of the schema name in part. Set to 0xFFFFFFFF to signal no available name.
COLUMNNAMEOFFSET	UI4	Specifies the offset of the column name in the part. Set to 0xFFFFFFFF to signal no available name.
COLUMNDISPLAYNAMEOFFSET	UI4	Specifies the offset of the column display name (label) in part. Set to 0xFFFFFFFF to signal no available name.

The COLUMNOPTIONS field is a bit set that has the following defined values:

Bit	Identifier	Description
0	MANDATORY	Specifies that the column is defined as NOT NULL.
1	OPTIONAL	Specifies that the column can be a NULL value.
2		Reserved, do not use.
3		Reserved, do not use.

Bit	Identifier	Description
4		Reserved, do not use.
5		Reserved, do not use.

The array of column descriptions can be followed by the individual schema names, table names, column names, and column display names. Here, each name is written in the following format:

LENGTH	NAME

Field	Data Type	Description
LENGTH	UI1	Specifies the length of the name in bytes.
NAME	B[LENGTH]	Specifies the name.

3.7.31 FINDLOBREQUEST Part Data Format

Searches for a substring of a BLOB, CLOB, NCLOB, or TEXT value.

The FINDLOBREQUEST part has the following structure:

LOCATORID	STARTOFFSET	PATTERNLENGTH	PATTERN

The fields are defined as follows:

Field	Data Type	Description
LOCATORID	B[8]	Specifies the identifier of the BLOB/CLOB/NCLOB/ TEXT locator.
STARTOFFSET	I8	Specifies the start offset for the search. The offset is a character or byte position dependent on the loca-tor data type.
PATTERNLENGTH	I4	Specifies the length of the pattern in bytes.
PATTERN	B[PATTERNLENGTH]	Specifies the pattern data, which is binary data or character data, depending on the locator data type.

The length of the pattern must not exceed 256-bytes.

3.7.32 FINDLOBREPLY Part Data Format

Responds to a FINDLOBREQUEST part.

FINDLOBREPLY contains one 8-byte integer value (little-endian) defining the position within the locator of the search pattern. A value of -1 indicates that the pattern has not been found.

3.7.33 ITABSHM Part Data Format

Describes how the memory used for an ITAB transfer is sent to the SAP HANA database.

An ITABSHM part has the following structure:

TRANSPORTTYPE	SHMID	OFFSET	LENGTH

The fields of this structure are defined as follows:

Field	Data Type	Description
TRANSPORTTYPE	NI4	Specifies the transport mechanism used.
SHMID	NI4	Reserved, do not use.
OFFSET	NUI8	Reserved, do not use.
LENGTH	NUI8	Reserved, do not use.

The following values are defined for the TRANSPORTTYPE field:

Value	Identifier	Description
0		Reserved, do not use.
1	SOCKET	Specifies that the ITAB is sent over a socket connection.

3.7.34 CLIENTINFO Part Data Format

Contains key/value pairs sent by the client for additional information.

The fields are formatted as variable length strings (similar to an NSTRING value in result set part). The argument count is the number of strings in the part. Since these are key/value pairs, the argument count is always an even number.

3.7.35 STREAMDATA Part Data Format

Contains stream data read or written by a C++ database procedure.

The STREAMDATA structure depends on the field information described by the corresponding ABAPITAB parameter.

3.7.36 BATCHPREPARE Part Data Format

This part is deprecated and not used.

3.7.37 BATCHEXECUTE Part Data Format

This part is deprecated and not used.

3.7.38 TRANSACTIONFLAGS Part Data Format

Specifies an option part.

The TRANSACTIONFLAGS part can return one or more of the following defined options:

Value	Identifier	Data Type	Description
0	ROLLEDBACK	BOOLEAN	Specifies that the transaction is rolled back.
1	COMMITED	BOOLEAN	Specifies that the transaction is committed.
2	NEWISOLATIONLEVEL	INTEGER	Specifies that the transaction isolation level has changed.
3	DDLCOMMITMODECHANGED	BOOLEAN	Specifies that the DDL auto-commit mode has been changed.
4	WRITETRANSACTIONSTARTED	BOOLEAN	Specifies that a write transaction has been started.
5	NOWRITETRANSACTIONSTARTED	BOOLEAN	Specifies that no write transaction has been started.
6	SESSIONCLOSINGTRANSACTIONERROR	BOOLEAN	Specifies that an error occurred that requires the session to be terminated.

The part is sent from the server to signal changes to the current transaction status (for example, committed, rolled back, or the start of a write transaction) and changes of the general session state, and whether the transaction isolation level has been changed or whether DDL statements are automatically committed or not. Also, the server can signal that it has detected a state that makes it impossible to continue processing the session.

3.7.39 DBCONNECTINFO Part Data Format

Specifies an option part.

The DBCONNECTINFO part has the following defined options:

Value	Identifier	Data Type	Description
1	DATABASENAME	STRING	Specifies the database name, sent from the client to the server.
2	HOST	STRING	Specifies the returned host name of the da-tabase (master).
3	PORT	INT4	Specifies the returned SQL port number of the master index server.
4	ISCONNECTED	BOOLEAN	Specifies the returned status, if the database is connected.

In the request message, only the database name is sent. The replied message includes the ISCONNECTED value and may include HOST and PORT values. The DATABASENAME value is not sent back.

3.7.40 LOBFLAGS Part Data Format

Specifies an option part.

The LOBFLAGS part can return one or more of the following defined options:

Value	Identifier	Data Type	Description
0	IMPLICITSTREAMING	BOOLEAN	Specifies that the implicit streaming has started.

The part is sent from the client to signal whether the implicit LOB streaming has started so that the server does not commit the current transaction, even with auto-commit on while LOB streaming.

3.7.41 RESULTSETOPTIONS Part Data Format

Contains query metadata for a result set from a prepared statement.

The fields are defined as follows:

Field	Data Type	Description
MAXAGE	I4	Specifies the scheduled refresh interval of the cached view.
LASTREFRESHTIMESTAMP	I8	Specifies the timestamp of the last time that the cached view was refreshed.

3.7.42 PRINTOPTIONSPART Part Data Format

Contains output from the SQLSCRIPT_PRINT library.

This is a multi-line option part. Each row consists of a single STRING option with an output line from the SQLSCRIPT_PRINT library.

3.8 Input and Output Field Formats

Input fields consist of type code information, followed by field data if the value is not the NULL value.

TYPECODE	VALUE

The value is left blank if the type code indicates a NULL value (the MSB of the type code field is set). The following sections only describe the VALUE format.

Output fields contain no special type code field. The type information is supplied in the respective PARAMETERDATA (for output parameters) or RESULTSETMETADATA parts. All output data is densely packed, there are no gaps between individual values, so some values may not be aligned in memory as required for the native type.

3.8.1 TINYINT Input and Output Field Format

Specifies the TINYINT input and output field format.

Input TINYINT

A TINYINT is sent as a 1-byte unsigned integer.

Output TINYINT

A TINYINT value is formatted as follows:

NULLIND	VALUE

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	UI1	Specifies the TINYINT value. This is only present if the NUL-LIND is not 0.

The field has a length of 2-bytes if the value is not NULL, and 1-byte if the value is NULL.

3.8.2 SMALLINT Input and Output Field Format

Specifies the SMALLINT input and output field format.

Input SMALLINT

A SMALLINT is sent as a 2-byte signed integer in little-endian format.

Output SMALLINT

A SMALLINT value is formatted as follows:

NULLIND	VALUE

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I2	Specifies the SMALLINT value. This is only present if NUL-LIND is not 0.

The field has a length of 3-bytes if the value is not NULL and 1-byte if the value is NULL.

3.8.3 INT Input and Output Field Format

Specifies the INT input and output field format.

Input INT

An INT is sent as a 4-byte signed integer in little-endian format.

Output INT

An INT value is formatted as follows:

`NULLIND`	`VALUE`

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I4	Specifies the INT value. This is only present if NULLIND is not 0.

The field has a length of 5-bytes if the value is not NULL and 1-byte if the value is NULL.

3.8.4 BIGINT Input and Output Field Format

Specifies the BIGINT input and output field format.

Input BIGINT

A BIGINT is sent as 8-byte signed integer in little-endian format.

Output BIGINT

A BIGINT value is formatted as follows:

NULLIND	VALUE

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I8	Specifies the BIGINT value. This is only present if NULLIND is not 0.

The field has a length of 9-bytes if the value is not NULL and 1-byte if the value is NULL.

3.8.5 DECIMAL Input and Output Field Format

Specifies the DECIMAL input and output field format.

Input DECIMAL

A decimal value is a 128-bit (16-byte) value formatted as follows:

SIGN	EXPONENT	MANTISSA

Field	Length	Description
SIGN	1-bit	Specifies that 0 is positive and 1 is negative.
EXPONENT	14-bit	Specifies the exponent (biased with 6176 leading to a range -6143 to +6144).
MANTISSA	113-bit	Specifies the integer mantissa.

The number represented is `(10^EXPONENT)*MANTISSA`. It is expected that the MANTISSA is not a multiple of 10.

Output DECIMAL

A DECIMAL value is formatted similar to the input format, with the NULL value indicated by having bits 4, 5, and 6 set to 1 in the last byte.

3.8.6 REAL Input and Output Field Format

Specifies the REAL input and output field format.

Input REAL

A REAL value is sent as float value (IEEE single precision floating point), in little-endian format.

Output REAL

A REAL value is sent as float value (IEEE single precision floating point), in little-endian format. The NULL value is indicated by all bit sets in the value sent from the server (equal to 0xFFFFFFFF as unsigned int).

3.8.7 DOUBLE Input and Output Field Format

Specifies the DOUBLE input and output field format.

Input DOUBLE

A REAL value is sent as double value (IEEE double precision floating point), in little-endian format.

Output DOUBLE

A DOUBLE value is sent as double value (IEEE double precision floating point), in little-endian format. The NULL value is indicated by all bit sets in the value sent from the server (equal to 0xFFFFFFFFFFFFFFFFul as unsigned long).

3.8.8 STRING/NSTRING Input and Output Field Format

Specifies the STRING/NSTRING input and output field format.

Input STRING/NSTRING

A STRING/NSTRING input value is formatted as follows:

`LENGTHINDICATOR`	`VALUE`

LENGTHINDICATOR is a field up to 5-bytes containing the following:

- The length in bytes of VALUE, if the VALUE length is less than or equal to 245
- 246, followed by a 2-byte integer (little-endian), which contains the actual length of VALUE
- 247, followed by a 4-byte integer (little-endian), which contains the actual length of VALUE

The VALUE field contains the input string in CESU-8 encoding.

Output STRING/NSTRING

The formatting is similar to the input with a LENGTHINDICATOR value of 255 indicating a NULL value.

3.8.9 BINARY Input and Output Field Format

Specifies the BINARY input and output field format.

Input BINARY

A BINARY input value is formatted similarly as the STRING/NSTRING value, with respect to a LENGTHINDICATOR and a VALUE part. The VALUE contains the binary data.

Output BINARY

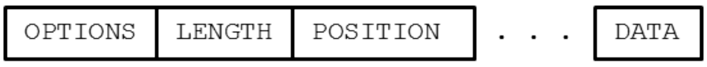
The formatting is similar to the input value with a LENGTHINDICATOR value of 255 indicating a NULL value.

3.8.10 BLOB/CLOB/NCLOB Input and Output Field Format

Specifies the BLOB/CLON/NCLOB input and output field format.

Input BLOB/CLOB/NCLOB

A BLOB/CLOB/NCLOB field is indicated using an input LOB descriptor, followed by the LOB data after the end of the record.



Field	Data Type	Description
OPTIONS	I1	Specifies the options that further refine the descriptor.
LENGTH	I4	Specifies the length, in bytes, of the data.
POSITION	I4	Specifies the position (1-based) of the data in the part.
DATA	B[LENGTH]	Specifies the BLOB/CLOB/NCLOB data.

The DATA field does not immediately follow the descriptor. First, all parameters of a row are transferred and then the BLOB/CLOB/NCLOB data follows.

The OPTIONS field is a bit set defined as follows:

Bit	Identifier	Description
0	NULLINDICATOR	Specifies the large object value is NULL (not used in the input).
1	DATAINCLUDED	Specifies that the data is included.
2	LASTDATA	Specifies that there is no more data remaining.

Output BLOB/CLOB/NCLOB

A BLOB/CLOB/NCLOB field is formatted using an output LOB descriptor, followed by the LOB data:

Field	Data Type	Description
TYPE	I1	Specifies the type of data.
OPTIONS	I1	Specifies the options that further refine the descriptor.
FILLER	I2	Reserved, do not use.
TOTALCHARLENGTH	I8	Specifies the total number of characters for NCLOB or bytes for CLOB or BLOB.
TOTALBINARYLENGTH	I8	Specifies the total number of bytes in LOB.
LOCATORID	B[8]	Specifies the identifier of BLOB/CLOB/NCLOB locator.
DATALENGTH	I4	Specifies the length of the DATA field in bytes.
DATA	B[DATALENGTH]	Specifies the large object data.

The TYPE field further refines the LOB source type code received in the result set metadata. It is defined as follows:

Value	Data Type
0	Undefined
1	BLOB
2	CLOB
3	NCLOB

The OPTIONS field is a bit set defined as follows:

Bit	Identifier	Description
0	NULLINDICATOR	Specifies that the large object value is NULL.
1	DATAINCLUDED	Specifies that the data is included.
2	LASTDATA	Specifies that there is no more data remain-ing.

3.8.11 DATE Input and Output Field Format

Specifies the DATE input and output field format.

A DATE field is formatted as follows:[3]

YEAR	MONTH	DAY

Field	Data Type	Description
YEAR	I2	Specifies the year.
MONTH	I1	Specifies the month.
DAY	I2	Specifies the day.

A NULL value is indicated by the MSB (0x8000) set in the YEAR field.

3.8.12 TIME Input and Output Field Format

Specifies the TIME input and output field format.

A TIME value is formatted as follows:[4]

HOUR	MINUTE	MILLISEC

Field	Data Type	Description
HOUR	I1	Specifies the hour.
MINUTE	I1	Specifies the minute.

[3] This format is retained for purpose of compatibility, DAYDATE is used where possible and the endorsed format.
[4] This format is retained for compatibility purposes. SECONDTIME is used where possible and is the endorsed format.

Field	Data Type	Description
MILLISEC	UI2	Specifies the milliseconds.

A NULL value is indicated by setting the MSB (0x80) in the HOUR field.

3.8.13 TIMESTAMP Input and Output Field Format

Specifies the TIMESTAMP input and output field format.

Input TIMESTAMP

A TIMESTAMP is formatted as a DATE value followed by a TIME value.[5]

Output TIMESTAMP

A TIMESTAMP is formatted as a DATE value followed by a TIME value. A NULL value is indicated by setting the NULL value in both components.

Related Information

DATE Input and Output Field Format [page 69]
TIME Input and Output Field Format [page 69]
DATE Input and Output Field Format [page 69]
TIME Input and Output Field Format [page 69]

[5] This format is retained for compatibility purposes. The LONGDATE or SECONDDATE formats are used where possible and are the endorsed formats.

3.8.14 ABAPITAB Input and Output Field Format

This value does not occur in the input or output data.

3.8.15 ABAPSTRUCT Input and Output Field Format

Specifies the ABAPSTRUCT input and output field format.

Input ABAPSTRUCT

An ABAPSTRUCT is formatted similar to a BINARY value. The layout of the structure depends on the meta-information known only to the processing liveCache C++ procedure and the ABAP client.

Output ABAPSTRUCT

The value is formatted similar to a BINARY value.

3.8.16 LONGDATE Input and Output Field Format

Specifies the LONGDATE input and output field format.

A LONGDATE is a 64-bit integer value computed as follows:

```
LONGDATE = (DAYDATE-1) * DAYFACTOR + TIMEVALUE + 1
```

The variables are computed as follows:

- DAYDATE – The date value as a DAYDATE data type
- DAYFACTOR – 10000000 * 60 * 60* 24
- TIMEVALUE – `((((Hours * 60) + Minutes) * 60 + Seconds) + Nanoseconds /100) * 10000000`

The value 3155380704000000001 is the NULL value.

3.8.17 SECONDDATE Input and Output Field Format

Specifies the SECONDDATE input and output field format.

A SECONDDATE is a 64-bit integer value computed as follows:

```
SECONDDATE = (DAYDATE-1) * DAYFACTOR + TIMEVALUE + 1
```

The variables are computed as follows:

- DAYDATE – The date value as a DAYDATE data type.
- DAYFACTOR – `60 * 60* 24`
- TIMEVALUE – `(((Hours * 24) + Minutes) * 60 + Seconds) * 60`

The value 315538070401 is the NULL value.

3.8.18 DAYDATE Input and Output Field Format

Specifies the DAYDATE input and output field format.

A DAYDATE is computed by taking the Julian Day Number of the specified date and subtracting 1721423.

The value 3652062 is the NULL value.

3.8.19 SECONDTIME Input and Output Field Format

Specifies the SECONDTIME input and output field format.

A SECONDTIME is a 32-bit integer value, which is computed as follows:

```
(((Hours * 24) + Minutes) * 60 + Seconds) * 60
```

The value 86401 is the NULL value.

3.8.20 ST_POINT/ST_GEOMETRY Input and Output Field Format

Specifies the ST_POINT and ST_GEOMETRY input field format.

Input ST_POINT/ST_POINTZ/ST_GEOMETRY

ST_POINT and ST_GEOMETRY are sent as BINARY. They use unique type codes when SPATIALTYPES=1 or SPATIALTYPES=2.

Related Information

Type Codes [page 31]

3.8.21 BOOLEAN Input and Output Field Format

Specifies the BOOLEAN input and output field format.

Input BOOLEAN

A BOOLEAN is sent as a byte with a value of 0x00 for FALSE and 0x02 for TRUE.

Output BOOLEAN

A BOOLEAN is formatted as follows:

Field	Data Type	Description
VALUE	B	Sets bit 0 if NULL and bit 1 if TRUE. (0x00 = FALSE, 0x01 = NULL, 0x02 = TRUE)

3.8.22 FIXED8 Input and Output Field Format

Specifies the FIXED8 input and output field format.

Input FIXED8

A FIXED8 is sent as an 8-byte signed integer in little-endian format.

Output FIXED8

A FIXED8 value is formatted as follows:

`NULLIND`	`VALUE`

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I8	Specifies the FIXED8 value. This is only present if the NUL-LIND is not 0.

The field has a length of 9 bytes if the value is not NULL and 1 byte if the value is NULL.

3.8.23 FIXED12 Input and Output Field Format

Specifies the FIXED12 input and output field format.

Input FIXED12

A FIXED12 is sent as a 12 byte signed integer in little-endian format.

Output FIXED12

A FIXED12 value is formatted as follows:

`NULLIND`	`VALUE`

The fields are defined as follows:

Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I12	Specifies the FIXED12 value. This is only present if the NUL-LIND is not 0.

The field has a length of 13 bytes if the value is not NULL and 1 byte if the value is NULL.

3.8.24 FIXED16 Input and Output Field Format

Specifies the FIXED16 input and output field format.

Input FIXED16

A FIXED16 is sent as 16-byte signed integer in little-endian format.

Output FIXED12

A FIXED16 value is formatted as follows:

NULLIND	VALUE

The fields are defined as follows:

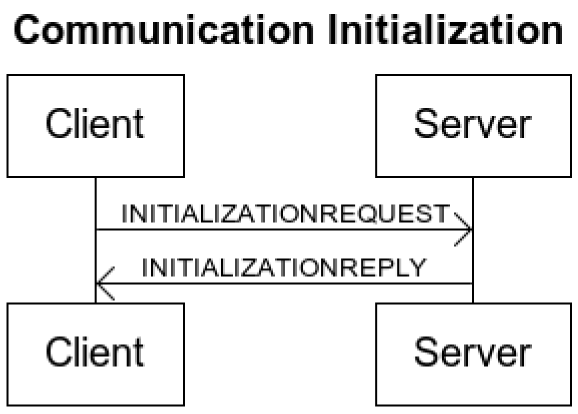
Field	Data Type	Description
NULLIND	I1	Specifies the NULL value indicator. The value is NULL if this is 0.
VALUE	I16	Specifies the FIXED16 value. This is only present if the NUL-LIND is not 0.

The field has a length of 17 bytes if the value is not NULL, and 1 byte if the value is NULL.

3.9 Usage Scenarios

The following section depicts common scenarios and the parameters that apply.

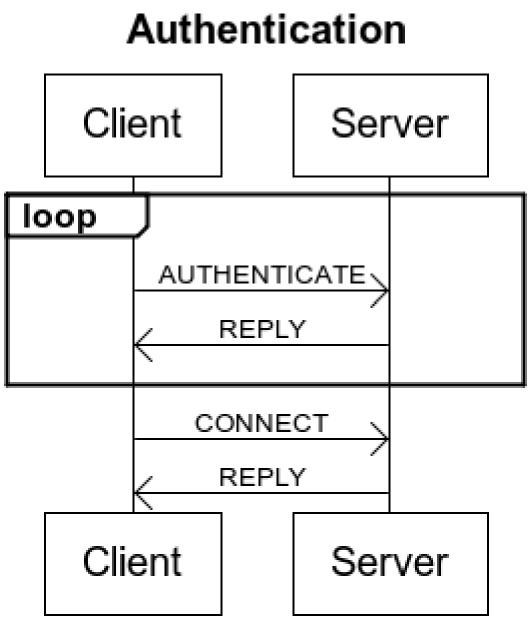
3.9.1 Communication Initialization



3.9.2 Authentication and Connect

The authentication and connection process consists of a number of AUTHENTICATE messages which result in a CONNECT message.

The authentication and connection process is laid out in the following sequence diagram:



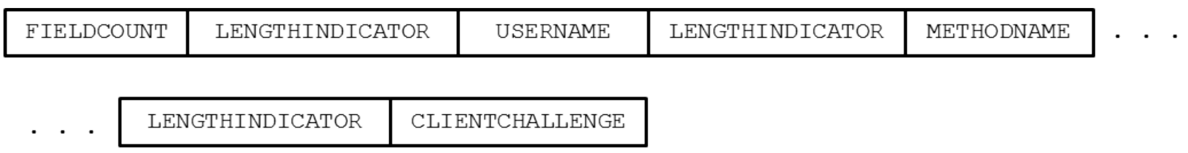
The AUTHENTICATE message performs a handshake on supported authentication methods and also is used repeatedly in case the authentication requires multiple messages to be exchanged until completion (this is only necessary for GSS authentication).

For several fields in the authentication request a LENGTHINDICATOR is used. It can be up to 5-bytes:

- The length in bytes of the following value, if the length is less 250
- A length of 250 is followed by a 2-byte integer (little-endian) which contains the actual length of the value

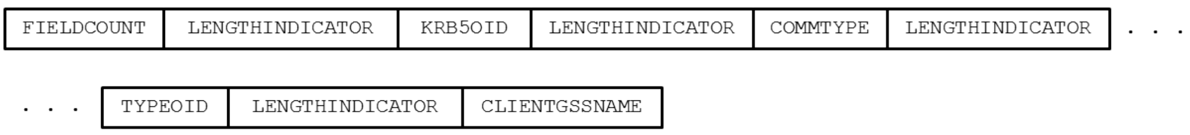
3.9.2.1 GSS Authentication

Initial Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Number of fields within this request
LENGTHINDICATOR	B1	Length of the following field
USERNAME	B[DATALENGTH]	User name
LENGTHINDICATOR	B1-5	Length of the following field
METHODNAME	B[DATALENGTH]	Method name
LENGTHINDICATOR	B1	Length of the following field
CLIENTCHALLENGE	B[DATALENGTH]	Client challenge

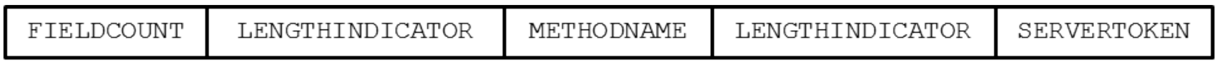
CLIENTCHALLENGE has the following format:



Field	Data Type	Description
FIELDCOUNT	I2	Number of fields within this request
LENGTHINDICATOR	B1	Length of the following field
KRB5OID	B[DATALENGTH]	KRB5 object ID
LENGTHINDICATOR	B1	Length of the following field

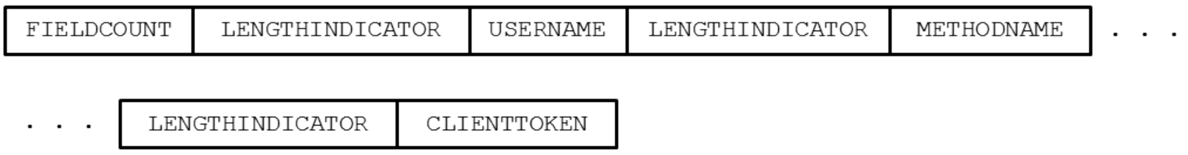
COMMTYPE	B1	Communication type
LENGTHINDICATOR	B1	Length of the following field
TYPEOID	B[DATALENGTH]	Type object ID
LENGTHINDICATOR	B1	Length of the following field
CLIENTGSSNAME	B[DATALENGTH]	Client GSS Name

Authentication Reply



Field	Data Type	Description
FIELDCOUNT	I2	Number of fields within this request
LENGTHINDICATOR	B1	Length of the following field
METHODNAME	B[DATALENGTH]	Method name
LENGTHINDICATOR	B1	Length of the following field
SERVERTOKEN	B[DATALENGTH]	Server-specific Kerberos tokens

Follow-Up Authentication Reply

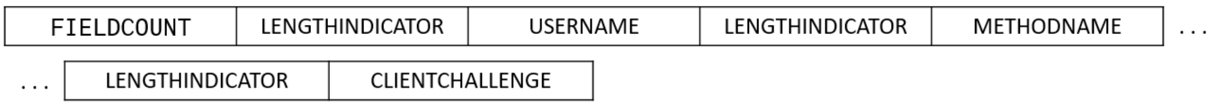


Field	Data Type	Description
FIELDCOUNT	I2	Number of fields within this request
LENGTHINDICATOR	B1	Length of the following field
USERNAME	B[DATALENGTH]	User name
LENGTHINDICATOR	B1-5	Length of the following field
METHODNAME	B[DATALENGTH]	Method name
LENGTHINDICATOR	B1	Length of the following field
CLIENTOKEN	B[DATALENGTH]	Client specific Kerberos tokens

3.9.2.2 LDAP Authentication

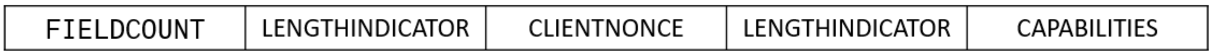
Securely transmits a client-specified password to an SAP HANA database, which then securely forwards the password to an LDAP server.

Initial Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within the request.
LENGTHINDICATOR	B1	Specifies the length of the USERNAME field.
USERNAME	B[DATALENGTH]	Specifies the name of the user.
LENGTHINDICATOR	B1	Specifies the length of the METHOD-NAME field.
METHODNAME	B[DATALENGTH]	Specifies the method name "LDAP".
LENGTHINDICATOR	B1	Specifies the length of the CLIENTCH-ALLENGE field.
CLIENTCHALLENGE	B[DATALENGTH]	Specifies the client challenge.

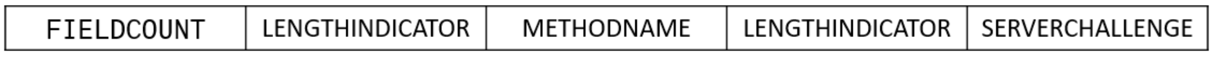
CLIENTCHALLENGE has the following format:



i Note

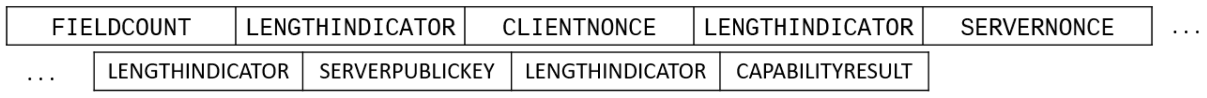
Using the username SYSTEM with LDAP authentication causes a failure.

Initial Authetication Reply



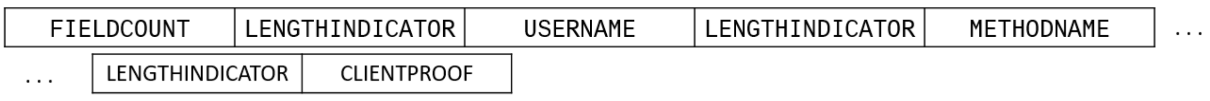
Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the METHOD-NAME field.
METHODNAME	B[DATALENGTH]	Specifies the method name "LDAP".
LENGTHINDICATOR	B1-2	Specifies the length of the SERVER-CHALLENGE field.
SERVERCHALLENGE	B[DATALENGTH]	Specifies the server challenge subpara-meters.

SERVERCHALLENGE has the following format:



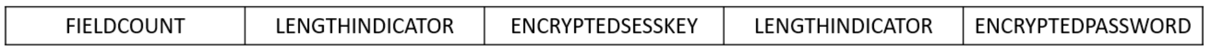
Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the CLIENT-NONCE field.
CLIENTNONCE	B[DATALENGTH]	Specifies the client nonce that was sent in the initial request.
LENGTHINDICATOR	B1	Specifies the length of the SERVER-NONCE field.
SERVERNONCE	B[DATALENGTH]	Specifies the server nonce.
LENGTHINDICATOR	B1-2	Specifies the length of the serialized RSA public key in the following field.
SERVERPUBLICKEY	B[DATALENGTH]	Specifies the server public key.
LENGTHINDICATOR	B1	Specifies the length of the CAPABILI-TYRESULT field.
CAPABILITYRESULT	B[DATALENGTH]	Specifies the capability, chosen by the server, from the client request.

Final Authentication Request



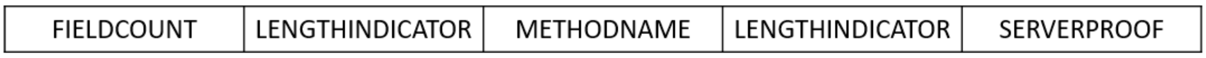
Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the USERNAME field.
USERNAME	B[DATALENGTH]	Specifies the username.
LENGTHINDICATOR	B1	Specifies the length of the METHOD-NAME field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1-2	Specifies the length of the CLIENT-PROOF field.
CLIENTPROOF	B[DATALENGTH]	Specifies the client proof.

CLIENTPROOF has the following format:



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1-2	Specifies the length of the ENCRYP-TEDSESSKEY field.
ENCRYPTEDSESSKEY	B[DATALENGTH]	Specifies the encrypted session key. This is specified as: RSAEncrypt(public key, SESSIONKEY + SERVERNONCE).
LENGTHINDICATOR	B1-2	Specifies the length of the ENCRYP-TEDPASSWORD field.
ENCRYPTEDPASSWORD	B[DATALENGTH]	Specifies the encrypted password. This is specified as: AES256Encrypt(SES-SIONKEY, PASSWORD + SERVER-NONCE).

Final Authentication Reply

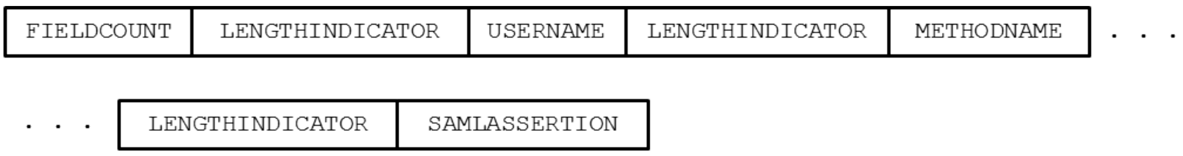


Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the METHOD-NAME field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the SERVER-PROOF field.
SERVERPROOF	B[1]	Specifies the authentication result from the LDAP server. This is specified as either SUCCESS or FAIL.

3.9.2.3 SAML Authentication

Performs two server roundtrips, an initial request, and a final request.

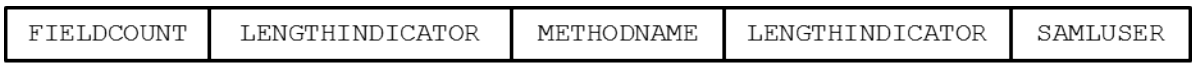
Initial Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field (this is always zero).
USERNAME	B[0]	Specifies the user name (always empty user name).

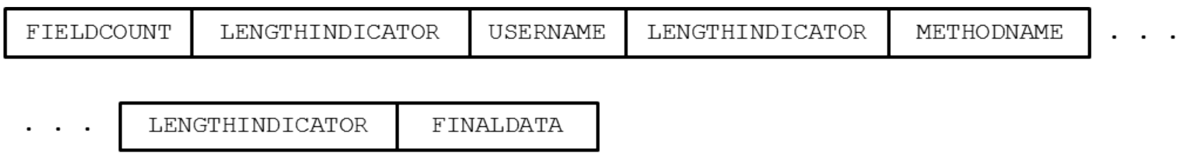
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
SAMLASSERTION	B[DATALENGTH]	Specifies the SAML assertion.

Initial Authentication Reply



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (this is always zero).
SAMLUSER	B[0]	Specifies the user name associated with the SAML assertion.

Final Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.

LENGTHINDICATOR	B1	Specifies the length of the following field.
USERNAME	B[DATALENGTH]	Specifies the user name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the ength of the following field (this is always zero).
FINALDATA	B[0]	Specifies the final data (this is empty).

Final Authentication Reply

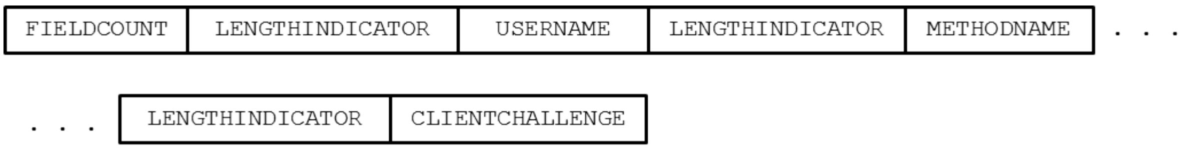
`FIELDCOUNT`	`LENGTHINDICATOR`	`METHODNAME`	`LENGTHINDICATOR`	`SESSIONCOOKIE`

Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (this is always zero).
SESSIONCOOKIE	B[0]	Specifies the session cookie used for the reconnect.

3.9.2.4 SCRAMSHA256 Authentication

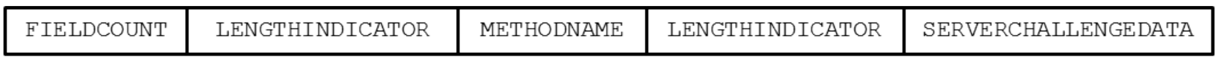
Following the SCRAMSHA256 authentication mechanism, two server roundtrips are necessary, an initial request and a final request.

Initial Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
USERNAME	B[DATALENGTH]	Specifies the user name.
LENGTHINDICATOR	B1-5	Specifies the length of the field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (always 64-bytes).
CLIENTCHALLENGE	B[64]	Specifies the client challenge.

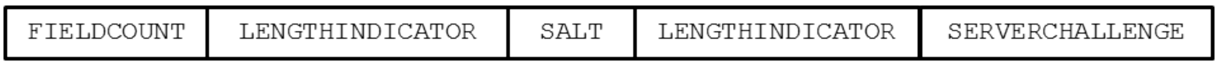
Initial Authentication Reply



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.

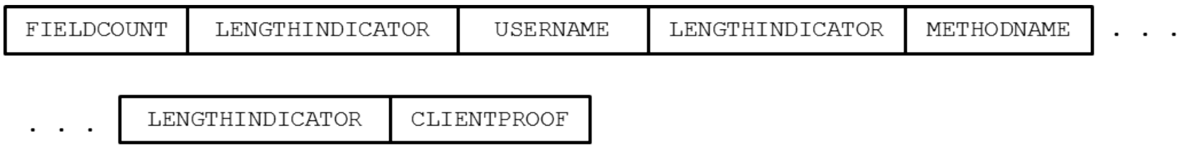
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (always 68-bytes).
SERVERCHALLENGEDATA	B[68]	Specifies the server challenge.

SERVERCHALLENGEDATA has the following format:



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
SALT	B[DATALENGTH]	Specifies the password salt.
LENGTHINDICATOR	B1	Specifies the length of the following field.
SERVERCHALLENGE	B[DATALENGTH]	Specifies the server challenge.

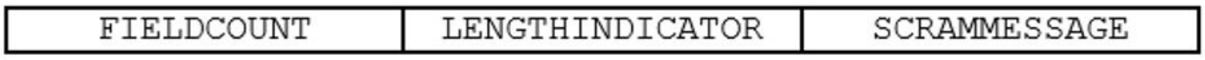
Final Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
USERNAME	B[DATALENGTH]	Specifies the user name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.

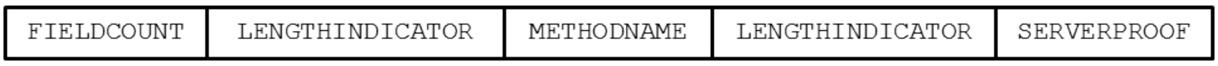
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (always 35-bytes).
CLIENTPROOF	B[35]	Specifies the client proof.

CLIENTPROOF has the following format:



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this subparameter.
LENGTHINDICATOR	B1	Specifies the length of the following (hash) field.
SCRAMMESSAGE	B[32]	Specifies the SCRAM HMAC message, the actual Client Proof that is sent to the server.

Final Authentication Reply



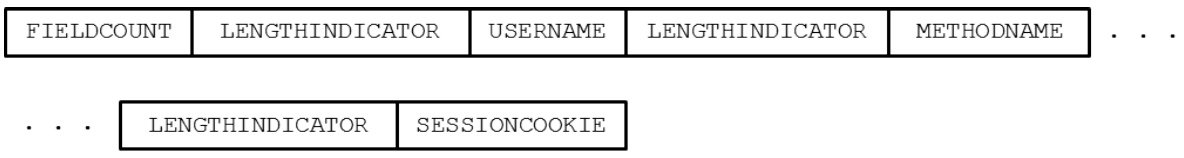
Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (0-bytes).
SERVERPROOF	B[DATALENGTH]	Specifies the server proof.

3.9.2.5 Session Cookie Authentication

Used in cases of a needed reconnection.

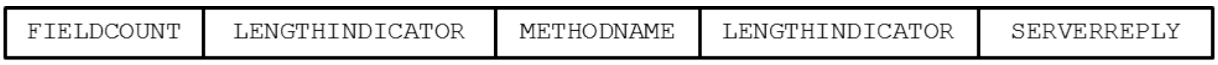
The cookie is obtained from a previous connection and makes two server roundtrips.

Initial Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
USERNAME	B[DATALENGTH]	Specifies the user name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
SESSIONCOOKIE	B[DATALENGTH]	Specifies the session cookie, process ID, and hostname.

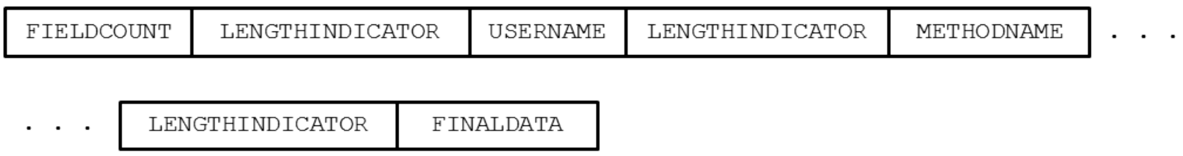
Initial Authentication Reply



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.

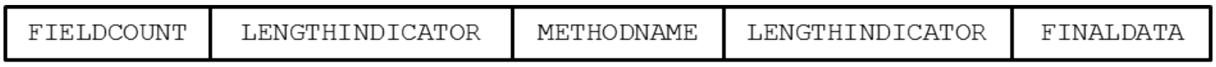
LENGTHINDICATOR	B1	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (always zero).
SERVERREPLY	B[0]	Specifies the server reply (this is empty).

Final Authentication Request



Field	Data Type	Description
FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
USERNAME	B[DATALENGTH]	Specifies the user name.
LENGTHINDICATOR	B1-5	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (this is always zero).
FINALDATA	B[0]	Specifies the final data (this is empty).

Final Authentication Reply



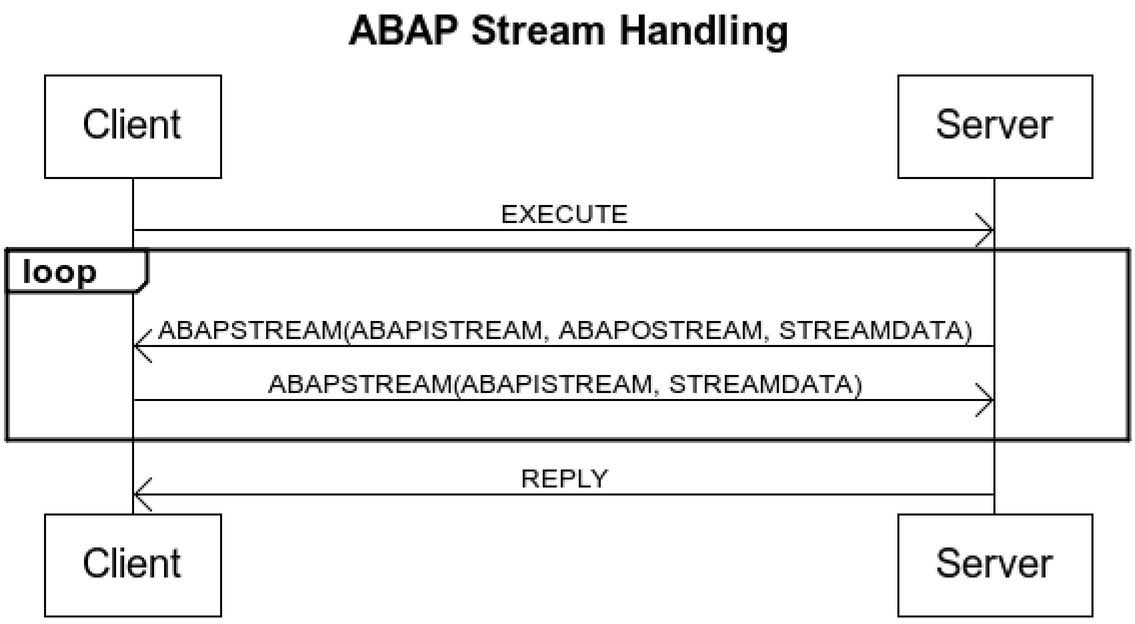
Field	Data Type	Description

FIELDCOUNT	I2	Specifies the number of fields within this request.
LENGTHINDICATOR	B1	Specifies the length of the following field.
METHODNAME	B[DATALENGTH]	Specifies the method name.
LENGTHINDICATOR	B1	Specifies the length of the following field (this is always zero).
FINALDATA	B[0]	Specifies the final data (this is empty).

3.9.3 ABAP Stream Handling

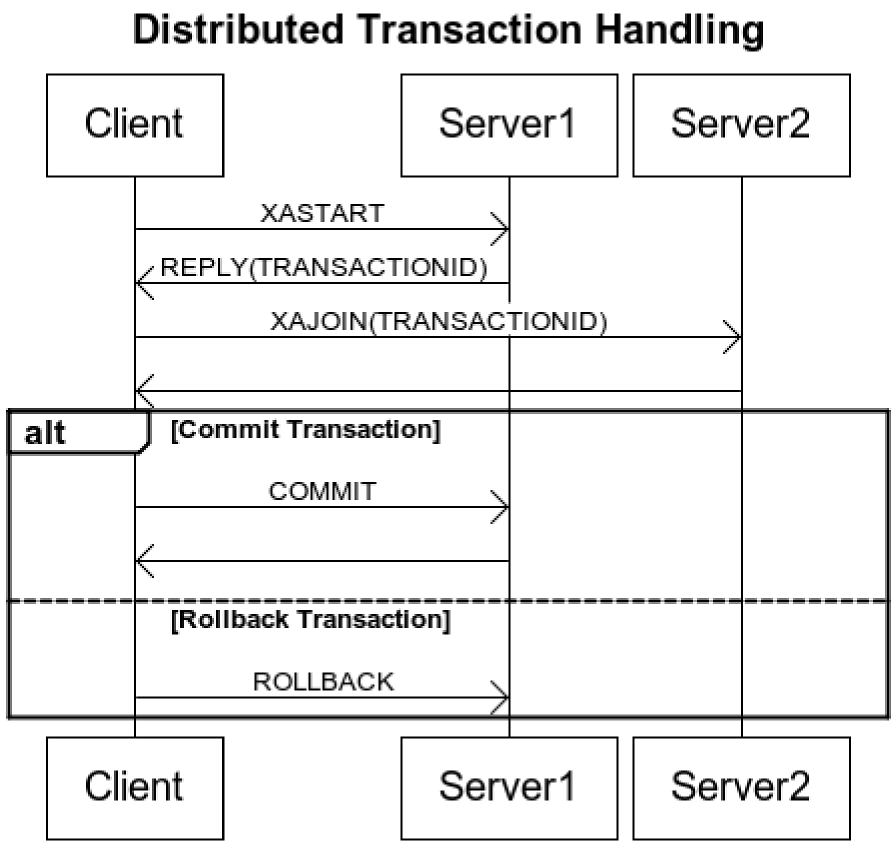
Specifies how ABAP streams data.

The following diagram shows how ABAP streams data between the client and the server:


ABAP Stream Handling
Client
Server
EXECUTE
loop
ABAPSTREAM(ABAPISTREAM, ABAPOSTREAM, STREAMDATA)
ABAPSTREAM(ABAPISTREAM, STREAMDATA)
REPLY
Client
Server

3.9.4 Distributed Transaction Handling

Provides information about how transactions are distributed.



3.9.5 Statement Routing

Evaluates the correct server node of a distributed system before statement execution.

Statement routing reduces the overhead in server processing and reduces communication between server nodes. For SQLDBC-based clients such as ODBC, ODBO, ADO.NET, Python DB API, and DBSL, the client library makes statement routing decisions to reduce server-side inter-node routing. To utilize this feature, the `optimized_routing_for_partition_table` property must be set to "on". This is done by default.

Utilization of Table Partitioning Information for Statement Routing Optimization

Utilization	JDBC	SQLDBC
Hash partitioning	Yes	Yes
Hash partitioning during split batch in-sert	Split batch not supported	Yes

Utilization	JDBC	SQLDBC
Range partitioning	No	Yes
Range partitioning during split batch insert	Split batch not supported	Yes

i Note

Active/Active (Read Enabled) hint-based routing uses the same mechanism as statement routing.

Preconditions

The server decides which statements are eligible for statement routing. When preparing a statement, the server returns a TABLELOCATION part or a PARTITIONINFORMATION part to describe the preferred nodes in detail.

JDBC and SQLDBC Statement Routing Settings

Server setting	Client Connection Property	Must Be Set To
indexserver.ini->distribution->client_distribution_mode	client_distribution_mode	Specifies "Statement" (default) or "All".

SQLDBC Split Batch Client-Side Routing Settings

Server setting	Client Connection Property	Must Be Set To
indexserver.ini->distribution->split_batch_commands	SPLITBATCHCOMMANDS	Specifies "1" (default).

4 Glossary

CESU-8	Compatibility Encoding Scheme for UTF-16: 8-Bit
DDL	Data Definition Language
DML	Data Modification Language
FAE	For All Entries. A specific ABAP Language construct, where a client-side table is joined with a server-side table.
FDA	Fast Data Access. A method to submit data for INSERT in the format used by the SAP ABAP Application Server (ABAP Table) to the server or retrieve SELECT results in the same format, to avoid field-wise copying and data conversion.
MSB	Most Significant Bit. The highest bit in an integer value, for example, Bit 7 in a byte.
SAML	Structured Authentication Markup Language

Important Disclaimer for Features in SAP HANA Platform

For information about the capabilities available for your license and installation scenario, refer to the Feature Scope Description (FSD) for your specific SAP HANA version on the SAP HANA Platform webpage.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP