TDS Protocol Documentation

This document attempts to cover the TDS protocol for:

TDS Version	Supported Products
4.2	Sybase SQL Server < 10 and Microsoft SQL Server 6.5
5.0	Sybase SQL Server >= 10
7.0	Microsoft SQL Server 7.0
7.1	Microsoft SQL Server 2000
7.2	Microsoft SQL Server 2005

Contents

Common Terms

```
TDS protocol versions
  TDS 5.0    tds version 5.0
  TDS 7.0    tds version 7.0
  TDS 7.0+   tds version 7.0, 7.1 and 7.2
  TDS 5.0-   tds version 5.0 and previous

Variable types used in this document:
  CHAR      8-bit char
    CHAR[6]     string of 6 chars
    CHAR[n]     variable length string
  XCHAR    single byte (TDS 5.0-) or ucs2le (TDS 7.0+) characters
  INT8      8-bit int
  INT16    16-bit int
  INT32    32-bit int
  UCS2LE   Unicode in UCS2LE format
```

Note: FreeTDS uses TDS_TINYINT for INT8 and TDS_SMALLINT for INT16.

Typical Usage sequences

These are TDS 4.2 and not meant to be 100% correct, but I thought they might be helpful to get an overall view of what goes on.

```
--> Login
<-- Login acknowledgement

--> INSERT SQL statement
<-- Result Set Done

--> SELECT SQL statement
<-- Column Names
<-- Column Info
<-- Row Result
<-- Row Result
<-- Result Set Done
```

```
--> call stored procedure
<-- Column Names
<-- Column Info
<-- Row Result
<-- Row Result
<-- Done Inside Process
<-- Column Names
<-- Column Info
<-- Row Result
<-- Row Result
<-- Done Inside Process
<-- Return Status
<-- Process Done
```

The packet format

Every informations in TDS protocol (query, RPCs, responses and so on) is splitted in packets.

All packets start with the following 8 byte header.

```
 INT8       INT8          INT16      4 bytes
+----------+-------------+----------+---------------------+
|  packet  | last packet |  packet  |    unknown          |
|   type   |  indicator  |   size   |                     |
+----------+-------------+----------+---------------------+

Fields:
packet type
     0x01 TDS 4.2 or 7.0 query
     0x02 TDS 4.2 or 5.0 login packet
     0x03 RPC
     0x04 responses from server
     0x06 cancels
     0x07 Used in Bulk Copy
     0x0F TDS 5.0 query
     0x10 TDS 7.0 login packet
     0x11 TDS 7.0 authentication packet
     0x12 TDS 8 prelogin packet
last packet indicator
     0x00 if more packets
     0x01 if last packet
packet size
     (in network byte order)
unknown?
     always 0x00
     this has something to do with server to server communication/rpc stuff
```

The remainder of the packet depends on the type of information it is providing. As noted above, packets break down into the types query, login, response, and cancels. Response packets are further split into multiple sub-types denoted by the first byte (a.k.a. the token) following the above header.

Note: A TDS packet that is longer than 512 bytes is split on the 512 byte boundary and the "more packets" bit is set. The full TDS packet is reassembled from its component 512 byte packets with the 8-byte headers stripped out. 512 is the block_size in the login packet, so it could be set to a different values. In Sybase you can configure a range of valid block sizes. TDS 7.0+ use a default of 4096 as block size.

TDS 4.2 & 5.0 Login Packet

Packet type (first byte) is 2. The numbers on the left are decimal offsets *including* the 8 byte packet header.

```
byte   var type    description
------------------------------
   8   CHAR[30]    host_name
  38   INT8        host_name_length
  39   CHAR[30]    user_name
  69   INT8        user_name_length
  70   CHAR[30]    password
```

```
100   INT8        password_length
101   CHAR[30]    host_process
131   INT8        host_process_length
132   ?           magic1[6]          /* mystery stuff */
138   INT8        bulk_copy
139   ?           magic2[9]          /* mystery stuff */
148   CHAR[30]    app_name
178   INT8        app_name_length
179   CHAR[30]    server_name
209   INT8        server_name_length
210   ?           magic3[1]          /* 0, don't know this one either */
211   INT8        password2_length
212   CHAR[30]    password2
242   CHAR[223]   magic4
465   INT8        password2_length_plus2
466   INT16       major_version      /* TDS version */
468   INT16       minor_version      /* TDS version */
470   CHAR        library_name[10]   /* "Ct-Library" or "DB-Library" */
480   INT8        library_length
481   INT16       major_version2     /* program version */
483   INT16       minor_version2     /* program version */
485   ?           magic6[3]          /* ? last two octets are 13 and 17 */
                                     /* bdw reports last two as 12 and 16 here  */
                                     /* possibly a bitset flag  */
488   CHAR[30]    language           /* e.g. "us-english" */
518   INT8        language_length
519   ?           magic7[1]          /*  mystery stuff */
520   INT16       old_secure         /* explanation? */
522   INT8        encrypted          /*  1 means encrypted all password fields blank */
523   ?           magic8[1]          /*  no clue... zeros */
524   CHAR        sec_spare[9]       /* explanation? */
533   CHAR[30]    char_set           /* e.g. "iso_1" */
563   INT8        char_set_length
564   INT8        magic9[1]          /* 1 */
565   CHAR[6]     block_size         /*  in text */
571   INT8        block_size_length
572   ?           magic10[25]        /* lots of stuff here...no clue */
```

Any help with the magic numbers would be most appreciated.

TDS 7.0+ Login Packet

```
byte  var type  description
---------------------------
  0   INT32     total packet size
  4   INT8[4]   TDS Version
                        0x00000070 7.0
                        0x01000071 7.1
                        0x02000972 7.2 (7.2.9?)
  8   INT32     packet size (default 4096)
 12   INT8[4]   client program version
 16   INT32     PID of client
 20   INT32     connection id (usually 0)
 24   INT8      option flags 1
                0x80 enable warning messages if SET LANGUAGE issued
                0x40 change to initial database must succeed
                0x20 enable warning messages if USE <database> issued
                0x10 enable BCP
                0x08 use ND5000 floating point format (untested)
                0x04 use VAX floating point format (untested)
                0x02 use EBCDIC encoding (untested)
                0x01 use big-endian byte order (untested)
 25   INT8      option flags 2
                0x80 enable domain login security
                0x40 "USER_SERVER - reserved"
                0x20 user type is "DQ login"
                0x10 user type is "replication login"
                0x08 "fCacheConnect"
                0x04 "fTranBoundary"
                0x02 client is an ODBC driver
                0x01 change to initial language must succeed
 26   INT8      0x04 spawn user instance (TDS 7.2)
                0x02 XML data type instances are returned as binary XML (TDS 7.2)
                0x01 password change requested (TDS 7.2)
 27   INT8       0x01 SQL Type: 0 = use default, 1 = use T-SQL (TDS 7.2)
 28   INT8[4]    time zone (0x88ffffff ???)
 32   INT8[4]    collation information
 36   INT16      position of client hostname (86)
 38   INT16      hostname length
 40   INT16      position of username
 42   INT16      username length
 44   INT16      position of password
 46   INT16      password length
 48   INT16      position of app name
 50   INT16      app name length
 52   INT16      position of server name
 54   INT16      server name length
 56   INT16      position of remote server/password pairs
 58   INT16      remote server/password pairs length
 60   INT16      position of library name
 62   INT16      library name length
 64   INT16      position of language
 66   INT16      language name (for italian "Italiano", coded UCS2)
 68   INT16      position of database name
 70   INT16      database name length
 72   INT8[6]    MAC address of client
 78   INT16      position of auth portion
 80   INT16      NT authentication length
 82   INT16      next position (same as total packet size)
 84   INT16      0
 86   UCS2LE[n] hostname
      UCS2LE[n] username
      UCS2LE[n] encrypted password
      UCS2LE[n] app name
      UCS2LE[n] server name
      UCS2LE[n] library name
      UCS2LE[n] language name
      UCS2LE[n] database name
      NT Authentication packet
 NT Authentication packet
   0   CHAR[8]   authentication id "NTLMSSP\0"
   8   INT32     1  message type
  12   INT32     0xb201 flags
  16   INT16     domain length
  18   INT16     domain length
  20   INT32     domain offset
  24   INT16     hostname length
  26   INT16     hostname length
  28   INT32     hostname offset
  32   CHAR[n]   hostname
       CHAR[n]   domain
 See documentation on Samba for detail (or search ntlm authentication for IIS)

 For mssql 2005 before hostname (byte 86) you have
  86   INT16     next position,  or
                position of file name for a database to be
                attached during the connection process
  88   INT16     database filename length
  90   INT16     new password position
  92   INT16     new password length
  94   UCS2LE[n] hostname
       ... (as above)
```

"current pos" is the starting byte address for a Unicode string within the packet. The length of that Unicode string immediately follows. That implies there are at least 2 more strings that could be defined. (character set??)

Username and password are empty if domain authentication is used.

If the client uses an authentication packet, the server replies with an Authentication token followed by an Authentication packet.

TDS 7.0 Authentication Packet

```
 varies
+-------+
| auth |
+-------+

auth   authentication data
       for NTLM this message 3
```

This packet usually follows Authentication token.

Types

HEX	DEC	type	protocol	nullable	size	collate
0x1F	31	SYBVOID	7+	no	0	
0x22	34	SYBIMAGE		yes	4	
0x23	35	SYBTEXT		yes	4	yes
0x24	36	SYBUNIQUE	7+	yes	1	
0x25	37	SYBVARBINARY		yes	1	
0x26	38	SYBINTN		yes	1	
0x27	39	SYBVARCHAR		yes	1	
0x2D	45	SYBBINARY		yes	1	
0x2F	47	SYBCHAR		yes	1	
0x30	48	SYBINT1		no	0	
0x32	50	SYBBIT		no	0	
0x34	52	SYBINT2		no	0	
0x38	56	SYBINT4		no	0	

0x3A	58	SYBDATETIME4		no	0	
0x3B	59	SYBREAL		no	0	
0x3C	60	SYBMONEY		no	0	
0x3D	61	SYBDATETIME		no	0	
0x3E	62	SYBFLT8		no	0	
0x40	64	SYBSINT1	5	no	0	
0x41	65	SYBUINT2	5	no	0	
0x42	66	SYBUINT4	5	no	0	
0x43	67	SYBUINT8	5	no	0	
0x62	98	SYBVARIANT	7+	yes	4	
0x63	99	SYBNTEXT	7+	yes	4	yes
0x67	103	SYBNVARCHAR	7+	yes	1	
0x68	104	SYBBITN		yes	1	
0x6A	106	SYBDECIMAL		yes	1	
0x6C	108	SYBNUMERIC		yes	1	
0x6D	109	SYBFLTN		yes	1	
0x6E	110	SYBMONEYN		yes	1	
0x6F	111	SYBDATETIMN		yes	1	
0x7A	122	SYBMONEY4		no	0	
0x7F	127	SYBINT8		no	0	
0xA5	165	XSYBVARBINARY	7+	yes	2 *	
0xA7	167	XSYBVARCHAR	7+	yes	2 *	yes
0xAD	173	XSYBBINARY	7+	yes	2	
0xAF	175	XSYBCHAR	7+	yes	2	yes

0xE1	225	SYBLONGBINARY	5	yes	4	
0xE7	231	XSYBNVARCHAR	7+	yes	2 *	yes
0xEF	239	XSYBNCHAR	7+	yes	2	yes

* Under TDS 7.2+ these types allow size to be -1, representing varchar(max), varbinary(max) and nvarchar(max). Data representation for them changes:

- size is 64 (not 16) bits
- size of -1 means NULL
- size of -2 means the size is unknown
- the data are split in chunks, where each chunk starts with a 32-bit size
- a chunk with size <= 0 is the terminal chunk

Collation type - TDS 7.1

The collation structure contains information about the character set encoding and comparison method.

```
 INT16      INT16    INT8
+----------+--------+------------+
| codepage | flags  | charset_id |
+----------+--------+------------+

codepage    windows codepage (see http://www.microsoft.com/globaldev/nlsweb/)
            also specified in lcid column of master..syslanguages
flags       sort flags
            0x100 binary compare
            0x080 width insensitive
            0x040 Katatype insensitive
            0x020 accent insensitive
            0x010 case insensitive
            If binary flag is specified other flags are not present
            Low nibble of flags is a charset specifier (like chinese dialect)
```

```
charset_id  charset id in master..syscharsets table or zero for no SQL collations
```

Collations names can be obtained from `select name from ::fn_helpcollations()` query

Column Metadata

```
 INT8          XCHAR[n]       INT8    INT32   INT8
+-------------+--------------+-------+-------+---------+
| column name | column name  | flags |  user | column  |
|   length    |              |       |  type |  type   |
+-------------+--------------+-------+-------+---------+

 varies        INT8       INT8       INT16      XCHAR[n]     INT8     varies
+-------------+----------+----------+----------+------------+--------+--------+
| column size |precision |   scale  | t length | table name | locale | locale |
|             |          |          |          |            | length |  info  |
| (optional)  |(optional)|(optional)|(optional)| (optional) | (opt)  | (opt)  |
+-------------+----------+----------+----------+------------+--------+--------+

column name length
column name        column name in result set, not necessarily db column name
flags              bit flags
                   0x1  hidden (TDS 5.0)
                   0x2  key
                   0x10 writable
                   0x20 can be NULL
                   0x40 identity
user type          usertype column from syscolumns
column type        column type
column size        not present for fixed size columns
precision          present only for SYBDECIMAL and SYBNUMERIC
scale              present only for SYBDECIMAL and SYBNUMERIC
t length           present only for SYBTEXT and SYBIMAGE, length of table name
table name         present only for SYBTEXT and SYBIMAGE
locale length      length of locale info (in bytes)
```

```
                   only for TDS 5.0 results (not for parameters)
locale info        unknown
                   only for TDS 5.0 results (not for parameters)
```

Client request

Normal tokens (contained in packets 0xF)

```
TODO
```

Special packets

- 0x1 1 Language
- 0x3 3 RPC TDS 4.6+
- 0x7 7 BCP TDS 5.0+

Language packet (0x1 1)

This sample packet contain just SQL commands. It's supported by all TDS version (although TDS 5.0 have others token with similar use)

```
  XCHAR[n]
+---------+
| string  |
+---------+

string   SQL text
```

RPC packet (0x3 3)

Do not confuse an RPC packet with an RPC token. The RPC packet is supported by all version of TDS; the RPC token is supported only by TDS 5.0 (and has different format). This is the oldest (and the only one in mssql) way to call directly an RPC. Sybase also documents it, but as 0xE.

```
  INT16          XCHAR[n]   INT16
+-------------+----------+-------+----------+
| name length | rpc name | flags | params   |
+-------------+----------+-------+----------+

name length   length of RPC name in characters.
              mssql2k+ support some core RPC using numbers
              If a number is used instead of name name length is marked as -1
              (null) and a INT16 is used for the name.
                0x1  1  sp_cursor
                0x2  2  sp_cursoropen
                0x3  3  sp_cursorprepare
                0x4  4  sp_cursorexecute
                0x5  5  sp_cursorprepexec
                0x6  6  sp_cursorunprepare
                0x7  7  sp_cursorfetch
                0x8  8  sp_cursoroption
                0x9  9  sp_cursorclose
                0xA  10 sp_executesql
                0xB  11 sp_prepare
                0xC  12 sp_execute ???
                0xD  13 sp_prepexec
                0xE  14 sp_prepexecrpc
                0xF  15 sp_unprepare
              sp_execute seems to have some problems, even MS ODBC use name
              version instead of number.
rpc name      name of RPC.
flags         bit flags.
               0x1 1 recompile procedure (TDS 7.0+/TDS 5.0)
               0x2 2 no metadata (TDS 7.0+)
```

```
                     (I don't know meaning of "no metadata" -- freddy77)
params         parameters. See below
```

Every parameter has the following structure

```
+-----------+------+
| data info | data |
+-----------+------+
data info    data information. See below
data         data. See results for detail
```

Data info structure

```
  INT8          XCHAR[n]     INT8    INT32
+-------------+------------+-------+--------------------+
| name length | param name | flags | usertype (TDS 5.0) |
+-------------+------------+-------+--------------------+
  INT8   varies  varies     INT8[5]         INT8
+------+-------+----------+---------------+------------------+
| type | size  | optional | collate       | locale           |
|      | (opt) | (opt)    | info(TDS 7.1) | length (TDS 5.0) |
+------+-------+----------+---------------+------------------+

name length   parameter name length (0 if unused)
param name    parameter name
flags         bit Name           Meaning
              0x1 TDS_RPC_OUTPUT output parameter
              0x2 TDS_RPC_NODEF  output parameter has no default value.
                                 Valid only with TDS_RPC_OUTPUT.

usertype      usertype
type          param type
size          see Results
optional      see Results. Blobs DO NOT have
              optional on input parameters (output blob parameters
              are not supported by any version of TDS).
```

```
collate info  only for type that want collate info and using TDS 7.1
locale length locale information length. Usually 0 (if not locale
              information follow, the structure is unknown)
```

Chained RPCs

Under TDS 7.0+ is possible to chain multiple RPCs together. This is useful to limit packets and round-trips with server. RPCs can be chained using byte 0x80 (TDS 7.0/TDS 7.1) or 0xFF (TDS 7.2).

```
       INT8                     INT8
+-----+------------------------+----------------+-----+
| RPC | 0x80 (TDS 7.0/TDS 7.1) | 0xFF (TDS 7.2) | RPC | ...
+-----+------------------------+----------------+-----+
```

Bulk Copy packet (0x7 7)

This documents a TDS 5.0 packet. It might be true for others....

BCP Packet Structure

```
 INT8       INT16          INT32
+----------+-------------+--------------------+
|  packet  | last packet |  packet            |
| type = 7 |  indicator  |   size             |
+----------+-------------+--------------------+

                followed by N row buffers,
                where N is computed by exhausting the packet size
```

BCP Packet Row Buffer

```
 INT16          INT8       INT8       INT16
+--------------+----------+----------+--------------+
| size         |  ncols   |  zero    | size (again) |
+--------------+----------+----------+--------------+

        followed by column buffers (data), where

                first, the fixed-size datatype columns

                 fixed size and count (determined by column definition)
                +--------------+
                |    data .... | [repeats once for each mandatory column]
                +--------------+

                then, the variable-size (including nullable) datatype columns

                 variable size and count
                +--------------+
                |    data .... | [repeats ncol times]
                +--------------+

                followed by two tables (!) to describe the column buffers
```

Adjustment Table (optional)

```
 INT8       INT8
+----------+----------+
| 1 + ncols| offset   | [repeats ncol + 1 times]
+----------+----------+
```

Offset Table (mandatory)

```
 INT8       INT8
+----------+----------+
| 1 + ncols| offset   | [repeats ncol + 1 times]
+----------+----------+
```

The BCP packet has a slightly different Packet header!?

Computation of Offset and Adjustment tables

The offset and adjustment tables describe the postion of the first byte of each variable-size column. The first element holds the count of elements in the offset/adjustment table. Thereafer, the offsets are arranged in reverse order: the last element — which is also the last byte of the row buffer — holds the offset from the start of the row of the first variable-size column. The next-to-last offset table element holds the starting position of the second variable-size column, and so on.

Offset Table Example

1. 5
2. 31
3. 22
4. 21
5. 8
6. 4

The first element is 5 because there are five elements in the list. There are 4 column data buffers with 5 endpoints. The first column's data begins at offset 4. Computations:

column 1
    offset 4
    length: 4 = 8 - 4
column 2
    offset 8
    length: 13 = 21 - 8
column 3
    offset 21

        length: 1 = 22 - 21
column 4
        offset 22
        length: 9 = 31 - 22

Any column not accounted for is implicitly NULL. To represent a NULL column between two dataful columns, the offset table will have adjacent entries of the same value.

Adjustment table

The so-called adjustment table provides for longer rows. The reader will note the Offset table has 8-bit elements, which would limit the width of the table: the last variable column would have to end less than 256 bytes from the start fo the row. Rather than changing the definition of the Offset table, a second table, the Adjustment table, was introduced. It holds high-order bytes for the column offsets.

In other words, to compute a variable column's offset from the start of the row buffer, the server looks up its offset table value, then consults the same position in the adjustment table, and splices them together.

Offset table commentary

The BCP packet is very dense. The data formats are governed by the table definition. Non-NULL columns of course must be present; there is no need to count them or compute their size. The NULL columns are undelimited; their boundaries are defined by the minimalist offset table.

The Adjustment table seems silly at first glance. Why not just make the offset table's elements 16 or or even 32 bits? The reason is overhead. Most rows will have less than 256 bytes of variable column data. By using the adjustment table, the BCP packet avoids adding one or even three empty bytes per column per row.

Why is the table in reverse order? Because that places the first offset at a known location: the end of each *row* gives the start of the first column. The server can work its way down the offset table and compute the column sizes. If they don't add up — if there are data between the (presumed) end of the last column and the start of the offset table — the server knows it should look for an adjustment table. Because the scheme is infinitely repeatable, rows could one day grow to terabyte widths without redefining the packet structure.

Server Responses

Responses from the server start with a single octet (token) identifying its type. If variable length, they generally have the length as the second and third bytes

Tokens encountered thus far:

HEX	DEC	name	note
0x20	32	Param Format 2	5.0 only
0x21	33	Language	5.0 only, client-side
0x22	34	OrderBy 2	5.0 only??
0x61	97	Row Format 2	5.0 only
0x71	113	"Logout"	5.0? ct_close(), client-side?
0x79	121	Return Status	
0x7C	124	Process ID	4.2 only
0x80	128	Cursor Close	5.0 only
0x81	129	Cursor Delete	5.0 only
0x81	129	7.0 Result	7.0 only
0x82	130	Cursor Fetch	5.0 only
0x83	131	Cursor Info	5.0 only
0x84	132	Cursor Open	5.0 only

0x86	134	Cursor Declare	5.0 only
0x88	136	7.0 Compute Result	7.0 only
0xA0	160	Column Name	4.2 only
0xA1	161	Column Format	4.2 only
0xA3	163	Dynamic 2	5.0 only
0xA4	164	Table names	name of tables in a FOR BROWSE select
0xA5	165	Column Info	column information in a FOR BROWSE select
0xA6	166	Option Cmd	5.0 only
0xA7	167	Compute Names	
0xA8	168	Compute Result	
0xA9	169	Order By	
0xAA	170	Error Message	
0xAB	171	Info Message	
0xAC	172	Output Parameters	
0xAD	173	Login Acknowledgement	
0xAE	174	Control	
0xD1	209	Data --- Row Result	
0xD3	211	Data --- Compute Result	
0xD7	215	Params	5.0 only
0xE2	226	Capability	5.0 only. Information on server
0xE3	227	Environment Change	(database change, packet size, etc...)
0xE5	229	Extended Error Message	
0xE6	230	DBRPC	5.0 only RPC calls
0xE7	231	Dynamic	5.0 only

0xEC	236	Param Format	5.0 only
0xED	237	Authentication	7.0 only
0xEE	238	Result Set	5.0 only
0xFD	253	Result Set Done	
0xFE	254	Process Done	
0xFF	255	Done inside Process	

Param Format 2 - TDS 5.0 (0x20 32)

TODO.

Language - TDS 5.0 (0x21 33)

```
 INT32    INT8     CHAR[n]
+--------+--------+-------+
| length | status | query |
+--------+--------+-------+

length  total token length
status  0 no args
        1 has args (followed by PARAMFMT/PARAMS)
query   query (total length - 1)
```

Order By 2 (0x22 34)

TODO.

Row Format 2 - TDS 5.0 (0x61 97)

TODO.

"Logout" (0x71 113)

No information. (1 byte, value=0 ?)

Return Status (0x79 121)

```
 INT32
+----------------+
| Return status |
+----------------+
```

The return value of a stored procedure.

Process ID (0x7C 124)

```
 8 bytes
+-----------------+
| process number |
+-----------------+
```

Presumably the process ID number for an executing stored procedure. (I'm not sure how this would ever be used by a client. *mjs*)

Cursor Close - TDS 5.0 (0x80 128)

TODO.

Cursor Delete - TDS 5.0 (0x81 129)

TODO.

Result - TDS 7.0+ (0x81 129)

```
 INT16
+----------+------------+
| #columns | column_info |
+----------+------------+
```

The TDS 7.0 column_info is formatted as follows for each column:

```
 INT16      INT16   INT8   varies  varies     INT8[5]          INT8          UCS2LE[n]
+----------+-------+------+-------+----------+----------------+-------------+---------+
| usertype | flags | type | size  | optional | collate        | name length | name    |
|          |       |      | (opt) | (opt)    | info(TDS 7.1)  |             |         |
+----------+-------+------+-------+----------+----------------+-------------+---------+

usertype        type modifier
flags           bit flags
                0x1  can be NULL
                0x8  can be written (it's not an expression)
                0x10 identity
type            data type, values >128 indicate a large type
size            none for fixed size types
```

```
                4 bytes for blob and text
                2 bytes for large types
                1 byte for all others
optional
                                INT8        INT8
                               +-----------+-------+
  numeric/decimal types:       | precision | scale |
                               +-----------+-------+

                                INT16              UCS2LE[n]
                               +-------------------+------------+
  blob/text types:             | table name length | table name |
                               +-------------------+------------+

  collate info are available only using TDS 7.1 and for characters types (but not
  for old type like short VARCHAR, only 2byte length versions)
```

Cursor Fetch - TDS 5.0 (0x82 130)

TODO.

Cursor Info - TDS 5.0 (0x83 131)

TODO.

Cursor Open - TDS 5.0 (0x84 132)

TODO.

Cursor Declare - TDS 5.0 (0x86 134)

TODO.

Compute Result - TDS 7.0+ (0x88 136)

TODO.

Column Name (0xA0 160)

```
 INT16          INT8      CHAR[n]                INT8      CHAR[n]
+--------------+---------+--------------+------+---------+--------------+
| total length | length1 | column1 name | .... | lengthN | columnN name |
+--------------+---------+--------------+------+---------+--------------+
```

This token is the first token that contain result informations. Is usually followed by Column Format token (0xA1 161)

Column Format (0xA1 161)

```
 INT16
+--------------+-------------+
| total length | column_info |
+--------------+-------------+
```

The number of columns is the same of previous Column Name token.

The TDS 4.2 column_info is formatted as follows for each column:

```
 INT8[4]     INT8   varies  varies
+-----------+------+-------+----------+
| usertype/ | type | size  | optional |
|   flags   |      | (opt) | (opt)    |
+-----------+------+-------+----------+
usertype/flags for Sybase

   INT32
  +----------+
  | usertype |
  +----------+

usertype/flags for MSSQL

   INT16
  +----------+-------+
  | usertype | flags |
  +----------+-------+

usertype        type modifier
flags           bit flags (only MSSQL)
                0x1  can be NULL
                0x8  can be written (it's not an expression)
                0x10 identity
type            data type
size            none for fixed size types
                4 bytes for blob and text
                1 byte for all others
                (TDS 4.2 do not support large types)
optional
                               INT8        INT8
                              +-----------+-------+
  numeric/decimal types:      | precision | scale |
  (supported??)               +-----------+-------+

                               INT16              CHAR[n]
                              +-------------------+------------+
  blob/text types:            | table name length | table name |
                              +-------------------+------------+
```

Dynamic 2 - TDS 5.0 (0xA3 163)

TODO.

Option Cmd - TDS 5.0 (0xA6 166)

TODO.

Compute Result (0xA8 168)

```
 INT16          INT16        INT8       varies        INT8      INT8[n]
+--------------+------------+----------+-------------+---------+-------+
| total length | compute id | #columns | column info | #bycols | bycol |
+--------------+------------+----------+-------------+---------+-------+
```

column info:

```
 INT8       INT8      INT32      INT8     varies  INT8                varies
+----------+---------+----------+--------+-------+------------------+----------------+
| operator | operand | usertype | column | size  | locale length    | locale info    |
|          |         |          |  type  | (opt) |  info (TDS 5.0)  | (TDS 5.0)      |
+----------+---------+----------+--------+-------+------------------+----------------+

operator      operator
              0x4b COUNT
              0x4c UNSIGNED? COUNT
              0x4d SUM
              0x4e UNSIGNED? SUM
              0x4f AVG
              0x50 UNSIGNED? AVG
              0x51 MIN
              0x52 MAX
              0x09 COUNT_BIG (mssql2k)
              0x30 STDEV (mssql2k)
              0x31 STDEVP (mssql2k)
              0x32 VAR (mssql2k)
              0x33 VARP (mssql2k)
              0x72 CHECKSUM_AGG (mssql2k)
operand       ???
usertype      usertype
column type   data type
size          data size
locale length length of locale informations
locale info   locale informations (unknown)
```

Each bycol information contains column info for a specific column.

TODO: optional possible?? collate infos ??

TabName (0xA4 164)

TDS4/5/7:

```
 INT16          INT16         XCHAR[n]
+--------------+-------------+------------+
| total length | name length | table name | ...
+--------------+-------------+------------+

name length   table name length
table name    table name
```

TDS 7.1:

```
 INT16          varies
+--------------+-------------+
| total length | table names |
+--------------+-------------+
table name:
 INT8              INT16              XCHAR[n]
+-----------------+------------------+----------------+
| # of components | component length | component name |
+-----------------+------------------+----------------+

ie:
  name        -> 01  04 00  ucs2le "name"
  db..name    -> 03  02 00  ucs2le "db"  00 00  04 00  ucs2le "name"
  db.dbo.name -> 03  02 00  ucs2le "db"  03 00  ucs2le "dbo"  04 00  ucs2le "name"
```

Column Info (0xA5 165)

```
 INT16          varies
+--------------+--------------+
| total length | column infos |
+--------------+--------------+
```

column info:

```
 INT8    INT8          INT8    INT8          XCHAR[n]
+-------+-------------+-------+-------------+-------------+
| index | table index | flags | name length | column name |
|       |             |       |  (opt)      |  (opt)      |
+-------+-------------+-------+-------------+-------------+

index        index in result format (1-based)
table index  index in previous TabName (1-based)
             0 means no table (ie computed)
flags        set of flags
             0x04 expression
             0x08 key
             0x10 hidden
             0x20 column name present
name length  length of following column
column name  real column name (result contain the label)
```

This token follow TabName token

---

compute "control" ? (0xA7 167)

"control" (0xAE 174)

Miscellaneous note (from *bdw* ?) found with 0xAE:

```
  has one byte for each column,
  comes between result(238) and first row(209),
  I believe computed column info is stored here, need to investigate
```

Order By (0xA9 169)

```
 INT16    variable (1 byte per col)
+--------+---------+
| length | orders  |
+--------+---------+

length          Length of packet(and number of cols)
orders          one byte per order by indicating the
                column # in the output matching the
                order from Column Info and Column Names
                and data in following Row Data items.
                A 0 indicates the column is not in the
                resulting rows.

an example:
select first_name, last_name, number from employee
order by salary, number
assuming the columns are returned in the order
```

```
queried:
first_name then last_name, then number. we would have:
----------------
|  2   | 0 | 3 |
----------------
where length = 2 then the orders evaluate:
0 for salary, meaning there is no salary data returned
3 for number, meaning the 3rd data item corresponding
to a column is the number
```

Error Message (0xAA 170)

Non-error Message (0xAB 171)

Extended Error Message (0xE5 229)

```
 INT16    INT32        INT8    INT8
+--------+------------+-------+-------+
| length | msg number | state | level |
+--------+------------+-------+-------+

 INT16      XCHAR[n]  INT8       XCHAR[n]  INT8       XCHAR[n]  INT16             INT32
+----------+---------+----------+---------+----------+---------+-----------------+-----------------+
| m length | message | s length | server  | p length | process | line#(TDS 7.1-) | line# (TDS 7.2) |
+----------+---------+----------+---------+----------+---------+-----------------+-----------------+

length          Length of packet
msg number      SQL message number
state           ?
level           An error if level > 10, a message if level <= 10
```

```
m length        Length of message
message         Text of error/message
s length        Length of server name
server          Name of "server" ?
p length        Length of process name
process name    Stored procedure name, if any
line#           Line number of input which generated the message
```

Output Parameters (0xAC 172)

Output parameters of a stored procedure.

```
 INT16    INT8       XCHAR[n]  INT8    INT32      INT8
+--------+----------+---------+-------+----------+----------+------+
| length | c length | colname | flags | usertype | datatype | .... |
+--------+----------+---------+-------+----------+----------+------+

length          Length of packet
c length        Length of colname
colname         Name of column
flags           0x1 Nullable
usertype        cf. systypes table in database
datatype        Type of data returned

The trailing information depends on whether the datatype is
a fixed size datatype.
                                 N bytes
                                +----------+
  Datatype of fixed size N      | data     |
                                +----------+

                                 INT8          INT8            N bytes
                                +-------------+---------------+--------+
  Otherwise                     | column size | actual size N | data   |
                                +-------------+---------------+--------+
```

Login Acknowledgement (0xAD 173)

```
 INT16    INT8    4 bytes   INT8       XCHAR[n] 4 bytes
+--------+-------+---------+----------+--------+----------+
| length |  ack  | version | t length |  text  | ser_ver  |
+--------+-------+---------+----------+--------+----------+

length          length of packet
ack             0x01 success    4.2
                0x05 success    5.0
                0x06 failure    5.0
version         TDS version 4 bytes:  major.minor.?.?
t length        length of text
text            server name (ie 'Microsoft SQL Server')
ser_ver         Server version
                (with strange encoding, differring from TDS version)
```

Data - Row Result (0xD1 209)

Data - Compute Result (0xD3 211)

```
 INT8       variable size
+----------+--------------------+
|  token   |   row data         |
+----------+--------------------+
```

Row data starts with one byte (decimal 209), for variable length types, a one byte length field precedes the data, for fixed length records just the data appears.

Note: nullable integers and floats are variable length.

For example: sp_who

The first field is spid, a smallint
The second field is status a char(12), in our example "recv sleep "

The row would look like this:

```
  byte  0 is the token
  bytes 1-2 are a smallint in low-endian
  byte  3 is the length of the char field
  bytes 4-15 is the char field

byte  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
hex  D1  01  00  0C  72  65  63  76  20  73  6C  65  65  70  20  20
    209   1   0  12   r   e   c   v ' '   s   l   e   e   p ' ' ' '
```

Params - TDS 5.0 (0xD7 215)

TODO.

Capability - TDS 5.0 (0xE2 226)

```
 INT16    variable
+--------+--------------+
| length | capabilities |
+--------+--------------+

length          Length of capability string
capabilities    Server capabilities?  Related to login magic?
```

Environment change (0xE3 227)

```
 INT16    INT8       INT8        CHAR[n]   INT8        CHAR[n]
+--------+----------+-----------+---------+-----------+---------+
| length | env code | t1 length |  text1  | t2 length |  text2  |
+--------+----------+-----------+---------+-----------+---------+

env code        Code for what part of environment changed
        0x01  database context
        0x02  language
        0x03  character set
        0x04  packet size
        0x05  TDS 7.0+ LCID
        0x06  TDS 7.0+ ??? (sort method? sql server encoding?)
        0x07  Collation info
text1           Old value
text2           New value

text1 and text2 are text information (coded in ucs2 in TDS 7.0+) except
collation info that's a structure (see collation structure)
```

DBRPC - TDS 5.0 (0xE6 230)

TODO.

Dynamic - TDS 5.0 (0xE7 231)

TODO.

Param Format - TDS 5.0 (0xEC 236)

```
 INT16     INT16       variable size
```

```
+---------+------------+-------------------+
| length  | number of  | parameter info    |
|         | parameters |                   |
+---------+------------+-------------------+

length                  length of message following this field
number of parameters    number of parameter formats following
list of formats         I (*bdw*) imagine it uses the column format structure.
```

Authentication - TDS 7.0 (0xED 237)

```
 INT16     varies
+---------+------+
| length  | auth |
+---------+------+

length   length of authentication data following this field
auth     authentication data
         for NTLM this is message 2
```

Client reply with Authentication packet.

Result Set - TDS 5.0 (0xEE 238)

```
 INT16     INT16        variable size
+---------+------------+-----------------+
| length  | number of  | column info     |
|         | columns    |                 |
+---------+------------+-----------------+


Fields:
length               length of message following this field
number of columns    number of columns in the result set, this many column
```

```
                   information fields will follow.
column info        column info
```

Done Packets

Result Set Done (0xFD 253)
Process Done (0xFE 254)
Done Inside Process (0xFF 255)

```
 INT16       INT16     INT32                  INT64
+-----------+---------+----------------------+---------------------+
| bit flags | unknown | row count (TDS 7.1-) | row count (TDS 7.2) |
+-----------+---------+----------------------+---------------------+

Fields:
bit flags          0x01 more results
                   0x02 error (like invalid sql syntax)
                   0x10 row count is valid
                   0x20 cancelled
unknown            2,0  /* something to do with block size perhaps */
row count          number of rows affected / returned in the result set.
                   row count is 64-bit using TDS 7.2.
                (FIXME check if "affected / returned" is correct)
```

"Result Set Complete" is the end of a query that doesn't create a process on the server. I.e., it doesn't call a stored procedure.
"Process Done" is the end of a stored procedure
"Done In Process" means that a query internal to a stored procedure has finished, but the stored procedure isn't done overall.

Acknowledgements

The following people have contributed to this document:

> Brian Bruns (first draft, protocol discovery)
> Brian Wheeler (protocol discovery)
> Mark Schaal (second draft)
> Frediano Ziglio

(short list)

Document Status

$Id: tds.html,v 1.41 2008-11-25 23:38:33 jklowden Exp $