1 Introduction

The Tabular Data Stream (TDS) protocol versions 7 and 8 is an application layer request/response protocol that facilitates interaction with a database server and provides for the following:

- Authentication and channel encryption.
- Specification of requests in SQL (including Bulk Insert).
- Invocation of a **stored procedure** or user-defined function, also known as a **remote procedure call (RPC)**.
- The return of data.
- **Transaction manager** requests.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Azure Active Directory Authentication Library (ADAL): A tool in Microsoft .NET Framework that allows application developers to authenticate users either to the cloud or to a deployed on-premises Active Directory and to then obtain tokens for secure access to API calls.

big-endian: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

bulk insert: A method for efficiently populating the rows of a table from the client to the server.

common language runtime user-defined type (CLR UDT): A data type that is created and defined by the user on a database server that supports SQL by using a Microsoft .NET Framework common language runtime assembly.

data classification: An information protection framework that includes sensitivity information about the data that is being returned from a query. The sensitivity information includes labels and information types and their identifiers.

data stream: A stream of data that corresponds to specific Tabular Data Stream (TDS) semantics. A single data stream can represent an entire TDS message or only a specific, well-defined portion of a TDS message. A TDS data stream can span multiple network data packets.

Distributed Transaction Coordinator (DTC): A Windows service that coordinates transactions across multiple resource managers, including databases. For more information, see [MSDN-DTC].

enclave: A protected region of memory that is used only on the server side. This region is within the address space of SQL Server, and it acts as a trusted execution environment. Only code that runs within the enclave can access data within that enclave. Neither the data nor the code inside the enclave can be viewed from the outside, even with a debugger.

enclave computations: Locally enabled cryptographic operations and other operations in Transact-SQL queries on encrypted columns that are performed inside an enclave.

federated authentication: An authentication mechanism that allows a security token service (STS) in one trust domain to delegate user authentication to an identity provider in another

trust domain, while generating a security token for the user, when there is a trust relationship between the two domains.

Global Transactions: A feature that allows users to execute transactions across multiple databases that are hosted in a shared service, such as Microsoft Azure SQL Database.

interface: A group of related function prototypes in a specific order, analogous to a C++ virtual interface. Multiple objects, of different object class, may implement the same interface. A derived interface may be created by adding methods after the end of an existing interface. In the Distributed Component Object Model (DCOM), all interfaces initially derive from IUnknown.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

Microsoft/Windows Data Access Components (MDAC/WDAC): With Microsoft/Windows Data Access Components (MDAC/WDAC), developers can connect to and use data from a wide variety of relational and nonrelational data sources. You can connect to many different data sources using Open Database Connectivity (ODBC), ActiveX Data Objects (ADO), or OLE DB. You can do this through providers and drivers that are built and shipped by Microsoft, or that are developed by various third parties. For more information, see [MSDN-MDAC].

Multiple Active Result Sets (MARS): A feature in Microsoft SQL Server that allows applications to have more than one pending request per connection. For more information, see [MSDN-MARS].

nullable column: A database table column that is allowed to contain no value for a given row.

out-of-band: A type of event that happens outside of the standard sequence of events. For example, an out-of-band signal or message can be sent during an unexpected time and will not cause any protocol parsing issues.

query notification: A feature in SQL Server that allows the client to register for notification on changes to a given query result. For more information, see [MSDN-QUERYNOTE].

remote procedure call (RPC): A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [C706].

result set: A list of records that results from running a stored procedure or query, or applying a filter. The structure and content of the data in a result set varies according to the implementation.

Security Support Provider Interface (SSPI): An API that allows connected applications to call one of several security providers to establish authenticated connections and to exchange data securely over those connections. It is equivalent to Generic Security Services (GSS)-API, and the two are on-the-wire compatible.

Session Multiplex Protocol (SMP): A multiplexing protocol that enables multiple logical client connections to share a single transport connection to a server. Used by **Multiple Active Result Sets (MARS)**. For more information, see [MC-SMP].

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO): An authentication mechanism that allows Generic Security Services (GSS) peers to determine whether their credentials support a common set of GSS-API security mechanisms, to negotiate different options within a given security mechanism or different options from several security mechanisms, to select a service, and to establish a security context among themselves using that service. **SPNEGO** is specified in [RFC4178].

SQL batch: A set of **SQL statements**.

SQL Server Native Client (SNAC): SNAC contains the SQL Server ODBC driver and the SQL Server OLE DB provider in one native dynamic link library (DLL) supporting applications using native-code APIs (ODBC, OLE DB, and ADO) to Microsoft SQL Server. For more information, see [MSDN-SNAC].

SQL Server User Authentication (SQLAUTH): An authentication mechanism that is used to support user accounts on a database server that supports SQL. The username and password of the user account are transmitted as part of the login message that the client sends to the server.

SQL statement: A character string expression in a language that the server understands.

stored procedure: A precompiled collection of SQL statements and, optionally, control-of-flow statements that are stored under a name and processed as a unit. They are stored in a SQL database and can be run with one call from an application. Stored procedures return an integer return code and can additionally return one or more result sets. Also referred to as sproc.

table response: A collection of data, all formatted in a specific manner, that is sent by the server to the client for the purpose of communicating the result of a client request. The server returns the result in a table response format for LOGIN7, SQL, and remote procedure call (RPC) requests.

TDS session: A successfully established communication over a period of time between a client and a server on which the Tabular Data Stream (TDS) protocol is used for message exchange.

transaction manager: The party that is responsible for managing and distributing the outcome of atomic transactions. A transaction manager is either a root transaction manager or a subordinate transaction manager for a specified transaction.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

Virtual Interface Architecture (VIA): A high-speed interconnect that requires special hardware and drivers that are provided by third parties.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[IANAPORT] IANA, "Service Name and Transport Protocol Port Number Registry", http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[IEEE754] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[IETF-AuthEncr] McGrew, D., Foley, J., and Paterson, K., "Authenticated Encryption with AES-CBC and HMAC-SHA", Network Working Group Internet-Draft, July 2014, http://tools.ietf.org/html/draft-mcgrew-aead-aes-cbc-hmac-sha2-05

[MS-BINXML] Microsoft Corporation, "SQL Server Binary XML Structure".

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference".

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989, http://www.rfc-editor.org/rfc/rfc1122.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, https://www.rfc-editor.org/rfc/rfc2119.html

[RFC2246] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999, https://www.rfc-editor.org/info/rfc2246

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.rfc-editor.org/rfc/rfc4234.txt

[RFC5246] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, https://www.rfc-editor.org/info/rfc5246

[RFC6101] Freier, A., Karlton, P., and Kocher, P., "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011, http://www.rfc-editor.org/rfc/rfc6101.txt

[RFC6234] Eastlake III, D., and Hansen, T., "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011, http://www.rfc-editor.org/rfc/rfc6234.txt

[RFC7301] Friedl, S., Popov, A., Langley, A., and Stephan, E., "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, July 2014, https://datatracker.ietf.org/doc/html/rfc7301

[RFC793] Postel, J., Ed., "Transmission Control Protocol: DARPA Internet Program Protocol Specification", RFC 793, September 1981, http://www.rfc-editor.org/rfc/rfc793.txt

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018, https://www.rfc-editor.org/info/rfc8446

[UNICODE] The Unicode Consortium, "The Unicode Consortium Home Page", http://www.unicode.org/

[VIA2002] Cameron, D., and Regnier, G., "The Virtual Interface Architecture", Intel Press, 2002, ISBN:0971288704.

1.2.2 Informative References

[MC-SMP] Microsoft Corporation, "Session Multiplex Protocol".

[MS-NETOD] Microsoft Corporation, "Microsoft .NET Framework Protocols Overview".

[MS-SSCLRT] Microsoft Corporation, "Microsoft SQL Server CLR Types Serialization Formats".

[MSDN-Autocommit] Microsoft Corporation, "Autocommit Transactions", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms187878(v=sql.105)

[MSDN-BEGIN] Microsoft Corporation, "BEGIN TRANSACTION (Transact SQL)", https://learn.microsoft.com/en-us/sql/t-sql/language-elements/begin-transaction-transact-sql

[MSDN-BOUND] Microsoft Corporation, "Using Bound Sessions", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms177480(v=sql.105)

[MSDN-BROWSE] Microsoft Corporation, "Browse Mode", in SQL Server 2000 Retired Technical documentation, p. 12261, https://www.microsoft.com/en-us/download/confirmation.aspx?id=51958

[MSDN-Collation] Microsoft Corporation, "Collation and Unicode Support", https://learn.microsoft.com/en-us/sql/relational-databases/collations/collation-and-unicode-support

[MSDN-ColSets] Microsoft Corporation, "Use Column Sets", https://learn.microsoft.com/en-us/sql/relational-databases/tables/use-column-sets

[MSDN-ColSortSty] Microsoft Corporation, "Windows Collation Sorting Styles", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms143515(v=sql.105)

[MSDN-COMMIT] Microsoft Corporation, "COMMIT TRANSACTION (Transact-SQL)", https://learn.microsoft.com/en-us/sql/t-sql/language-elements/commit-transaction-transact-sql

[MSDN-DTC] Microsoft Corporation, "Distributed Transaction Coordinator", https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms684146(v=vs.85)

[MSDN-INSERT] Microsoft Corporation, "INSERT (Transact-SQL)", https://learn.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql

[MSDN-ITrans] Microsoft Corporation, "ITransactionExport::GetTransactionCookie", https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms679869(v=vs.85)

[MSDN-MARS] Microsoft Corporation, "Using Multiple Active Result Sets (MARS)", https://learn.microsoft.com/en-us/sql/relational-databases/native-client/features/using-multiple-active-result-sets-mars

[MSDN-MDAC] Wilkes, R., Bunch, A., and Dove, D., "Microsoft Data Access Components (MDAC) Installation", May 2005, https://learn.microsoft.com/en-us/previous-versions/ms810805(v=msdn.10)

[MSDN-NamedPipes] Microsoft Corporation, "Creating a Valid Connection String Using Named Pipes", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189307(v=sql.105)

[MSDN-NP] Microsoft Corporation, "Named Pipes", https://learn.microsoft.com/en-us/windows/desktop/ipc/named-pipes

[MSDN-NTLM] Microsoft Corporation, "Microsoft NTLM", https://learn.microsoft.com/en-us/windows/desktop/SecAuthN/microsoft-ntlm

[MSDN-QUERYNOTE] Microsoft Corporation, "Using Query Notifications", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms175110(v=sql.105)

[MSDN-SNAC] Microsoft Corporation, "Microsoft SQL Server Native Client and Microsoft SQL Server 2008 Native Client", https://learn.microsoft.com/en-us/archive/blogs/sqlnativeclient/microsoft-sql-server-native-client-and-microsoft-sql-server-2008-native-client

[MSDN-SQLCollation] Microsoft Corporation, "Selecting a SQL Server Collation", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms144250(v=sql.105)

[MSDN-TDSENDPT] Microsoft Corporation, "Network Protocols and TDS Endpoints", https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191220(v=sql.105)

[MSDN-UPDATETEXT] Microsoft Corporation, "UPDATETEXT (Transact-SQL)", https://learn.microsoft.com/en-us/sql/t-sql/queries/updatetext-transact-sql

[MSDN-WRITETEXT] Microsoft Corporation, "WRITETEXT (Transact-SQL)", https://learn.microsoft.com/en-us/sql/t-sql/queries/writetext-transact-sql

[MSDOCS-DBMirror] Microsoft Corporation, "Database Mirroring in SQL Server", https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/sql/database-mirroring-in-sql-server

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, https://www.rfc-editor.org/rfc/rfc4120

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and Ingersoll, W., "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, October 2005, https://www.rfc-editor.org/rfc/rfc4178.txt

[SSPI] Microsoft Corporation, "SSPI", https://learn.microsoft.com/en-us/windows/desktop/SecAuthN/sspi

1.3 Overview

The Tabular Data Stream (TDS) protocol versions 7 and 8, hereinafter referred to as "TDS", is an application-level protocol used for the transfer of requests and responses between clients and database server systems. In such systems, the client will typically establish a long-lived connection with the server. Once the connection is established using a transport-level protocol, TDS messages are used to communicate between the client and the server. A database server can also act as the client if needed, in which case a separate TDS connection has to be established. Note that the **TDS session** is directly tied to the transport-level session, meaning that a TDS session is established when the transport-level connection is established and the server receives a request to establish a TDS connection. It persists until the transport-level connection is terminated (for example, when a TCP socket is closed). In addition, TDS does not make any assumption about the transport protocol used, but it does assume the transport protocol supports reliable, in-order delivery of the data.

TDS includes facilities for authentication and identification, channel encryption negotiation, issuing of **SQL batches**, **stored procedure** calls, returning data, and **transaction manager** requests. Returned data is self-describing and record-oriented. The **data streams** describe the names, types, and optional descriptions of the rows being returned.

The difference between the TDS 7.x version family and the TDS 8.0 version centers on where and how network channel encryption is initiated:

- In the TDS 7.x version family, encryption is optional and is negotiated and handled in the TDS layer.
- The TDS 8.0 version introduces mandatory encryption that is handled in the lower layer before TDS begins functioning.

The following diagram depicts a (simplified) typical flow of communication in the TDS protocol.

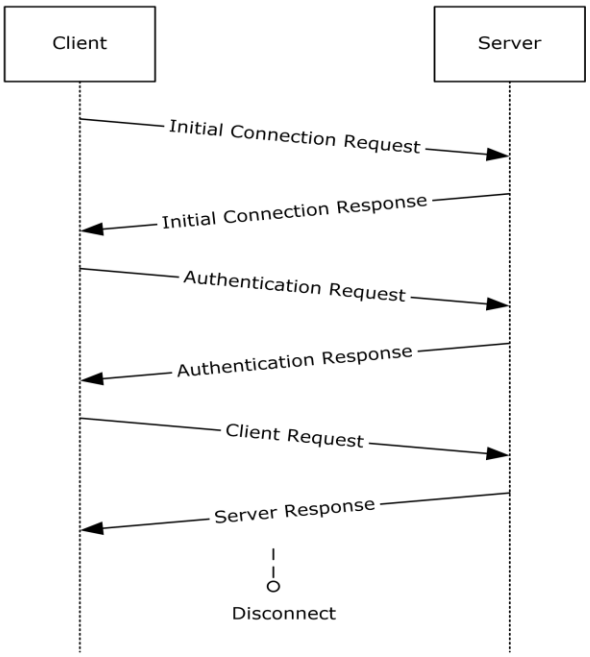

Figure 1: Communication flow in the TDS protocol

The following example is a high-level description of the messages exchanged between the client and the server to execute a simple client request such as the execution of a **SQL statement**. It is assumed that the client and the server have already established a connection and authentication has succeeded.

```
Client:SQL statement
```

The server executes the SQL statement and then sends back the results to the client. The data columns being returned are first described by the server (represented as column metadata or COLMETADATA [section 2.2.7.4) and then the rows follow. A completion message is sent after all the row data has been transferred.

```
Server:COLMETADATAdata stream
ROWdata stream
.
.
ROWdata stream
DONEdata stream
```

For more information about the correlation between data stream and TDS packet, see section 2.2.4.<1>

Additional details about which SQL Server version corresponds to which TDS version number are defined in LOGINACK (section 2.2.7.14).

1.4 Relationship to Other Protocols

The Tabular Data Stream (TDS) protocol depends upon a network transport connection being established prior to a TDS conversation occurring (the choice of transport protocol is not important to TDS).

TDS depends on Transport Layer Security (TLS)/Secure Socket Layer (SSL) for network channel encryption. In the TDS 7.x version family, TLS/SSL is optional and the negotiation of the encryption setting between the client and server and the initial TLS/SSL handshake are handled in the TDS layer.

Introduced in the TDS 8.0 version, TLS is mandatory and is established in the lower layer before TDS begins functioning.

If the **Multiple Active Result Sets (MARS)** feature [MSDN-MARS] is enabled, the **Session Multiplex Protocol (SMP)** [MC-SMP] is required.

This relationship is illustrated in the following figure.

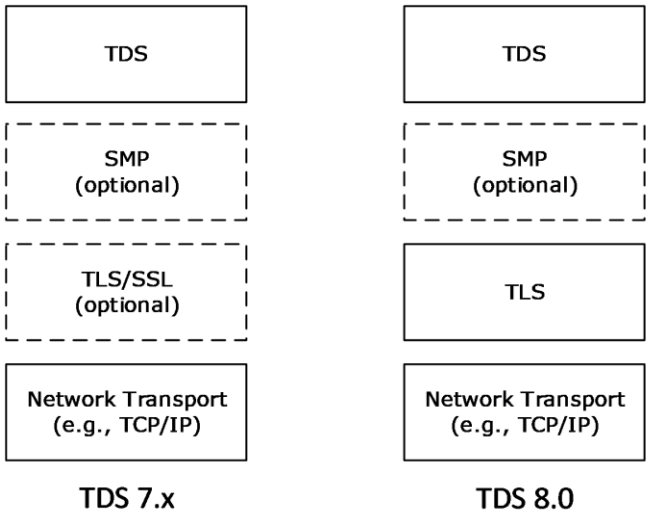

Figure 2: Protocol relationship

1.5 Prerequisites/Preconditions

This protocol can be used after the client has discovered the server and established a network transport connection for use with TDS.

In the TDS 7.x version family, no security association is assumed to have been established at the lower layer before TDS begins functioning. In the TDS 8.0 version, such a security association is assumed.

For **Security Support Provider Interface (SSPI)** [SSPI] authentication to be used, SSPI support needs to be available on both the client and server machines. For channel encryption to be used, TLS/SSL support needs to be present on both client and server machines, and a certificate suitable for encryption has to be deployed on the server machine.

For **federated authentication** to be used, a library that provides federated authentication support or an equivalent needs to be present on the server, and the client needs to be able to generate a token for federated authentication.

1.6 Applicability Statement

The TDS protocol is appropriate for use to facilitate request/response communications between an application and a database server in all scenarios where network or local connectivity is available.

1.7 Versioning and Capability Negotiation

This protocol includes versioning issues in the following areas.

- **Supported Transports:** This protocol can be implemented on top of any network transport protocol as discussed in section 2.1.
- **Protocol Versions:** The TDS protocol supports the TDS 7.x version family (which is composed of explicit versions TDS 7.0, TDS 7.1, TDS 7.2, TDS 7.3, and TDS 7.4) and the TDS 8.0 explicit version.

  In TDS 7.x, the explicit version is negotiated as part of the LOGIN7 message data stream, as described in section 2.2.6.4.

  In TDS 8.0, the explicit version must be identified from the TLS handshake by using an Application-Layer Protocol Negotiation (ALPN) TLS extension [RFC7301]. If ALPN is not present, the server must assume the TDS 8.0 version has been sent. Version information sent in the LOGIN7 message data that is later in the flow does not have to be specified and should be ignored by both the client and server.

  Aspects of later versions of the TDS protocol that do not apply to earlier versions are identified in the text.

  Note After a protocol feature is introduced, subsequent versions of the TDS protocol support that feature until that feature is removed.
- **Security and Authentication Methods:** The TDS protocol supports **SQL Server User Authentication (SQLAUTH)**. The TDS protocol also supports SSPI authentication and indirectly supports any authentication mechanism that SSPI supports. The use of SSPI in TDS is defined in sections 2.2.6.4 and 3.2.5.2. The TDS protocol also supports **federated authentication**. The use of federated authentication in TDS is defined in sections 2.2.6.4 and 3.2.5.
- **Localization:** Localization-dependent protocol behavior is specified in sections 2.2.5.1.2 and 2.2.5.6.
- **Capability Negotiation:** This protocol does explicit capability negotiation as specified in this section.

In general, the TDS protocol does not provide facilities for capability negotiation because the complete set of supported features is fixed for each version of the protocol. Certain features such as authentication type are not usually negotiated but rather are requested by the client. However, the protocol supports negotiation for the following two features:

- **Channel encryption:** In TDS 7.x, the encryption behavior that is used for the **TDS session** is negotiated in the initial messages exchanged by the client and the server. In TDS 8.0, encryption is mandatory and is established prior to initial messaging by the client and the server.
- **Authentication mechanism for integrated authentication identities:** The authentication mechanism that is used for the TDS session is negotiated in the initial messages exchanged by the client and the server.

For more details about encryption behavior in TDS 7.x and about how the client and server negotiate between SSPI authentication and federated authentication, see the PRELOGIN description in section 2.2.6.5.

Note that the cipher suite for TLS/SSL [RFC2246] [RFC5246] [RFC8446] [RFC6101], the authentication mechanism for SSPI [SSPI], and federated authentication are negotiated outside the influence of TDS.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

The TDS 7.x and TDS 8.0 protocols use the following assignment.

Parameter	TCP port value	Reference
Default SQL Server instance TCP port	1433	[IANAPORT]

TDS 8.0 also uses the following ALPN identification sequence to identify the TDS protocol.

Parameter	Identification Sequence	Reference
tds/8.0	0x74 0x64 0x73 0x2f 0x38 0x2e 0x30	[MS-TDS]

Note This identification sequence has been requested and is in the process of being registered with the Internet Assigned Numbers Authority (IANA). This note will be removed when registration is completed.