

[MS-SSSO]:

SQL Server System Overview

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](#).
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
3/5/2010	0.1	Major	First release.
4/21/2010	0.1.1	Editorial	Changed language and formatting in the technical content.
6/4/2010	0.1.2	Editorial	Changed language and formatting in the technical content.
9/3/2010	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
2/9/2011	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
7/7/2011	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
11/3/2011	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
1/19/2012	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
2/23/2012	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
3/27/2012	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
5/24/2012	0.1.2	None	No changes to the meaning, language, or formatting of the technical content.
6/29/2012	1.0	Major	Updated and revised the technical content.
7/16/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.0	Major	Updated and revised the technical content.
10/23/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
3/26/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
6/11/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
12/5/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2014	3.0	Major	Updated and revised the technical content.
5/20/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
5/10/2016	4.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
8/16/2017	5.0	Major	Significantly changed the technical content.
3/16/2018	5.1	Minor	Clarified the meaning of the technical content.
3/5/2020	6.0	Major	Significantly changed the technical content.

Table of Contents

1	Introduction	6
1.1	Glossary	7
1.2	References	9
2	Functional Architecture	11
2.1	Overview	11
2.1.1	Network Connectivity and Application Development	12
2.1.2	Master Data Services	13
2.1.3	Reporting Services.....	14
2.1.4	Analysis Services.....	15
2.1.5	Database Engine	16
2.1.6	Complex Event Processing Engine.....	16
2.1.7	Manageability.....	17
2.1.8	Big Data Clusters.....	17
2.2	Protocol Summary.....	18
2.2.1	Network Connectivity and Application Development	18
2.2.2	Master Data Services	19
2.2.3	Reporting Services.....	20
2.2.4	Analysis Services.....	21
2.2.5	Database Engine	21
2.2.6	CEP Engine	22
2.2.7	Manageability.....	22
2.2.8	Big Data Clusters.....	22
2.3	Environment.....	22
2.3.1	Dependencies on This System	22
2.3.2	Dependencies on Other Systems or Components	23
2.3.3	Communications within the System.....	23
2.3.4	Assumptions and Preconditions	23
2.4	Use Cases	23
2.4.1	Network Connectivity and Application Development Use Cases	23
2.4.1.1	SQL Server Instance Discovery Use Case	23
2.4.1.2	Named SQL Server Instance Resolution/Enumeration	24
2.4.1.3	Client Connection (TDS, SSTDS, or SSNWS).....	25
2.4.2	MDS Integration Use Cases.....	26
2.4.2.1	Use the MDS UI to Query	26
2.4.2.2	Query a List from the MDS Store.....	27
2.4.3	Reporting Services Use Cases.....	27
2.4.3.1	Report Authoring, Management, and Viewing with Native Report Portal	27
2.4.3.2	Mobile Report Authoring, Management, and Viewing with Native Report Portal.....	28
2.4.3.3	Report Authoring, Management, and Viewing with External Report Portal.....	28
2.4.4	Analysis Services Use Cases.....	29
2.4.4.1	Authentication with the Analysis Server	29
2.4.4.2	Information Discovery	29
2.4.4.3	Sending an MDX Query	29
2.4.4.4	Sending a DAX Query	29
2.4.4.5	Usage Reporting	29
2.4.5	Database Engine Use Cases	30
2.4.5.1	Authentication with the Database Engine	30
2.4.5.2	Information Discovery	30
2.4.5.3	Sending a Query	30
2.4.6	CEP Engine Use Case	30
2.4.7	Manageability Use Case.....	31
2.4.8	Big Data Cluster Protocols Use Cases	31
2.4.8.1	Control Plane REST API Use Cases	31
2.4.8.1.1	Provision a Big Data Cluster.....	31

2.4.8.1.2	Retrieve Big Data Cluster Endpoints.....	32
2.4.8.2	Hive Metastore HTTP Use Cases.....	32
2.5	Versioning, Capability Negotiation, and Extensibility	32
2.6	Error Handling	32
2.7	Coherency Requirements	32
2.8	Security	32
3	Examples.....	33
3.1	Configuring and Administering Multiple Servers.....	33
3.1.1	Analysis Services Authoring and Management	34
3.1.2	Reporting Services Authoring and Management	34
3.1.3	MDS Management.....	34
3.1.4	Database Engine Management	35
3.2	Obtaining Data	35
3.2.1	Obtaining Data via Analysis Services	35
3.2.2	Obtaining Data via Reporting Services.....	35
3.2.3	Obtaining Data via MDS.....	36
4	Microsoft Implementations	37
4.1	Product Behavior.....	37
5	Change Tracking.....	38
6	Index.....	39

1 Introduction

The SQL Server System Overview document provides an overview of the client and server protocols that are used by Microsoft SQL Server. This document covers protocols that are commonly shared by SQL Server components and those protocols that are used only by specific components. Where appropriate, this document describes the relationships between protocols and provides example scenarios to show how they are used.

SQL Server is a data platform that includes several data management and analysis technologies. This document covers those elements of the platform that require protocols that interoperate.

- [Master Data Services](#): The Master Data Services (MDS) service and API provide a service-oriented design architecture (SOA) that encapsulates and modularizes the internal workings of SQL Server, in addition to a standard API to interact and integrate with SQL Server Master Data Services. The SQL Server MDS framework ensures that the internal functions of the product are better modularized to support both an API and a modular component development. For more information, see [\[MSDN-MDS\]](#).
- [Reporting Services](#): Reporting Services delivers enterprise, web-enabled reporting functionality for creating reports that draw content from a variety of data sources, for publishing reports in various formats, and for centrally managing security and subscriptions. For more information, see [\[MSDN-SSRS\]](#).
- [Analysis Services](#): Analysis Services supports high performance analytical applications by enabling an implementer to design, create, manage, and query Multidimensional and Tabular data models. For more information, see [\[MSDN-ASMD\]](#).
- [Database Engine](#): The Database Engine is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data-consuming applications within an enterprise. The Database Engine also provides rich support for sustaining high availability. For more information, see [\[MSDN-SSDBEng\]](#).
- [Complex event processing](#): **Complex event processing (CEP)** is the continuous and incremental processing of event (data) **streams** from multiple sources based on declarative query and pattern specifications with near-zero latency. The goal is to identify meaningful patterns, relationships, and data abstractions from among seemingly unrelated events and to trigger immediate response actions. Typical event stream sources include data from manufacturing applications, financial trading applications, web analytics, and operational analytics. The CEP Engine provides a dedicated web service to handle requests from client applications for managing the system.
- [Big data clusters](#): Big data clusters provide the ability to deploy scalable clusters of SQL Server containers that can combine and analyze high-value relational data with high-volume big data. This flexibility of interaction with big data lets the user query external data sources, store big data in external file systems managed by SQL Server, or query data from multiple external data sources through the cluster. For more information, see [\[MSDOCS-SSBDC\]](#).

To deliver these functionalities, SQL Server uses eight major sets of technologies:

- Network connectivity and application development
- Master Data Services
- Reporting Services
- Analysis Services
- Database Engine

- Complex event processing engine
- Manageability
- Big data clusters

This document provides an overview of the protocols that can be used by one or more of the SQL Server products that are listed in [Microsoft Implementations \(section 4\)](#). Specific release information for each protocol is indicated in the individual technical specifications only, unless otherwise indicated in the summary information provided in section [2.2](#).

1.1 Glossary

This document uses the following terms:

Active Directory: The Windows implementation of a general-purpose directory service, which uses LDAP as its primary access protocol. Active Directory stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by Kerberos [\[MS-KILE\]](#). Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [\[MS-ADOD\]](#): Active Directory Protocols Overview.

analysis server: A server that supports high performance and complex analytics for business intelligence applications.

Apache Knox: A gateway system that provides secure access to data and processing resources in an Apache Hadoop cluster.

big data cluster: A grouping of high-value relational data with high-volume big data that provides the computational power of a cluster to increase scalability and performance of applications.

common language runtime (CLR): The core runtime engine in the Microsoft .NET Framework for executing applications. The common language runtime supplies managed code with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support.

complex event processing (CEP): The continuous and incremental processing of event streams from multiple sources, based on declarative query and pattern specifications with near-zero latency.

connection string: A series of arguments, delimited by a semicolon, that defines the location of a database and how to connect to it.

control plane: A logical plane that provides management and security for a Kubernetes cluster. It contains the controller, management proxy, and other services that are used to monitor and maintain the cluster.

control plane service: The service that is deployed and hosted in the same Kubernetes namespace in which the user wants to build out a **big data cluster**. The service provides the core functionality for deploying and managing all interactions within a Kubernetes cluster.

Hive Metastore: A database that is external to the Apache Hive that stores Hive metadata. The Hive Metastore is responsible for storing table statistics, including table storage location, column names, and table index information.

JavaScript Object Notation (JSON): A text-based, data interchange format that is used to transmit structured data, typically in Asynchronous JavaScript + XML (AJAX) web applications, as described in [\[RFC7159\]](#). The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

ODBC application: An application that uses **Open Database Connectivity (ODBC)** to access data sources.

OLE DB: A set of interfaces that are based on the Component Object Model (COM) programming model and expose data from a variety of sources. These interfaces support the amount of Database Management System (DBMS) functionality that is appropriate for a data store and they enable a data store to share data.

OLE DB consumer: A software component that requests information through a set of OLE DB interfaces.

OLE DB provider: A software component that returns information to an OLE DB consumer through a set of OLE DB interfaces. Each provider exposes data access to a particular type of data source.

Online Analytical Processing (OLAP): A technology that uses multidimensional structures to provide access to data for analysis. The source data for OLAP is stored in data warehouses in a relational database. See also cube.

Open Database Connectivity (ODBC): A standard software API method for accessing data that is stored in a variety of proprietary personal computer, minicomputer, and mainframe databases. It is an implementation of [ISO/IEC9075-3:2008](#) and provides extensions to that standard.

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 [\[C706\]](#).

report server: A location on the network to which clients can connect by using SOAP over HTTP or SOAP over HTTPS to publish, manage, and execute reports.

session: A collection of state information on a directory server. An implementation of the SOAP session extensions (SSE) is free to choose the state information to store in a session.

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [\[SOAP1.2-1/2003\]](#).

stream: A sequence of bytes written to a file on the target file system. Every file stored on a volume that uses the file system contains at least one stream, which is normally used to store the primary contents of the file. Additional streams within the file can be used to store file attributes, application parameters, or other information specific to that file. Every file has a default data stream, which is unnamed by default. That data stream, and any other data stream associated with a file, can optionally be named.

Transact-SQL: The Microsoft proprietary version of SQL, the structured query language.

Transmission Control Protocol (TCP): A protocol used with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

user-defined type (UDT): User-defined types can extend the scalar type system of the protocol server database, enabling storage of **common language runtime** objects in a protocol server database. UDTs can contain multiple elements, and they can have behaviors to differentiate

them from the traditional alias data types that consist of a single protocol server system data type.

web service: A software system designed to support interoperable machine-to-machine interaction over a network, using XML-based standards and open transport protocols.

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](#).

[ApacheKnox] Apache Software Foundation, "Apache Knox", <https://knox.apache.org/>

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

[MC-SQLR] Microsoft Corporation, "[SQL Server Resolution Protocol](#)".

[MS-ASUR] Microsoft Corporation, "[Analysis Services Usage Reporting Protocol](#)".

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

[MS-CEPM] Microsoft Corporation, "[Microsoft Complex Event Processing Engine Manageability Protocol](#)".

[MS-CPREST] Microsoft Corporation, "[Control Plane REST API](#)".

[MS-DSDG] Microsoft Corporation, "[DataSet DiffGram Structure](#)".

[MS-DSDIFFGRAM] Microsoft Corporation, "[SharePoint Web Services: DataSet DiffGram Structure](#)".

[MS-DTCO] Microsoft Corporation, "[MSDTC Connection Manager: OleTx Transaction Protocol](#)".

[MS-HMSHTTP] Microsoft Corporation, "[Hive Metastore HTTP Protocol](#)".

[MS-LETSF] Microsoft Corporation, "[LINQ Expression Tree Serialization Format](#)".

[MS-ODBCSTR] Microsoft Corporation, "[ODBC Connection String Structure](#)".

[MS-OLEDBSTR] Microsoft Corporation, "[OLEDB Connection String Structure](#)".

[MS-RDL] Microsoft Corporation, "[Report Definition Language File Format](#)".

[MS-RGDI] Microsoft Corporation, "[Remote GDI+ \(RGDI\) Binary Stream Format](#)".

[MS-RPL] Microsoft Corporation, "[Report Page Layout \(RPL\) Binary Stream Format](#)".

[MS-RSREST] Microsoft Corporation, "[Reporting Services REST API](#)".

[MS-RSWSRE2005] Microsoft Corporation, "[Report Server Web Service for Report Execution: ReportExecution2005](#)".

[MS-RSWSRM2010] Microsoft Corporation, "[Report Server Web Service for Report Management: ReportService2010](#)".

[MS-RSWSRMNM2005] Microsoft Corporation, "[Report Server Web Service for Report Management for Native Mode: ReportService2005](#)".

[MS-RSWSRMSM2006] Microsoft Corporation, "[Report Server Web Service for Report Management for SharePoint Mode: ReportService2006](#)".

[MS-RSWSSFA] Microsoft Corporation, "[Report Server Web Service for SharePoint Forms Authentication: ReportServiceAuthentication](#)".

[MS-SSAS-T] Microsoft Corporation, "[SQL Server Analysis Services Tabular Protocol](#)".

[MS-SSAS8] Microsoft Corporation, "[SQL Server Analysis Services Version 8.0 Protocol](#)".

[MS-SSAS] Microsoft Corporation, "[SQL Server Analysis Services Protocol](#)".

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

[MS-SSDPWP] Microsoft Corporation, "[Database Publishing Wizard Protocol](#)".

[MS-SSMDSWS-15] Microsoft Corporation, "[Master Data Services Web Service 15](#)".

[MS-SSMDSWS] Microsoft Corporation, "[Master Data Services Web Service](#)".

[MS-SSNWS] Microsoft Corporation, "[Native Web Services Protocol](#)".

[MS-SSTDS] Microsoft Corporation, "[Tabular Data Stream Protocol Version 4.2](#)".

[MS-TDS] Microsoft Corporation, "[Tabular Data Stream Protocol](#)".

[MSDN-ASMD] Microsoft Corporation, "What is Analysis Services?", <https://learn.microsoft.com/en-us/analysis-services/analysis-services-overview>

[MSDN-MDS] Microsoft Corporation, "Master Data Services Overview (MDS)", <https://learn.microsoft.com/en-us/sql/master-data-services/master-data-services-overview-mds>

[MSDN-SSDBEng] Microsoft Corporation, "SQL Server Database Engine", <https://learn.microsoft.com/en-us/sql/database-engine/sql-server-database-engine-overview?view=sql-server-2014>

[MSDN-SSRS] Microsoft Corporation, "What is SQL Server Reporting Services (SSRS)?", <https://learn.microsoft.com/en-us/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports>

[MSDOCS-SSBDC] Microsoft Corporation, "Introducing SQL Server Big Data Clusters", <https://learn.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview?view=sqlallproducts-allversions>

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998, <http://www.rfc-editor.org/rfc/rfc2279>

2 Functional Architecture

Microsoft SQL Server delivers a rich set of integrated services that enable a user to query, search, synchronize, report, and analyze data. This section describes the architecture to deliver and support this framework.

2.1 Overview

SQL Server is a database management and analysis system for e-commerce, line-of-business, and data warehousing solutions, providing storage and query retrieval.

The following figure shows a high-level architectural view of SQL Server elements that work together to achieve interoperability. Note that Microsoft delivers the implementation of the protocols described as client access libraries, which are used by Microsoft and third-party applications.

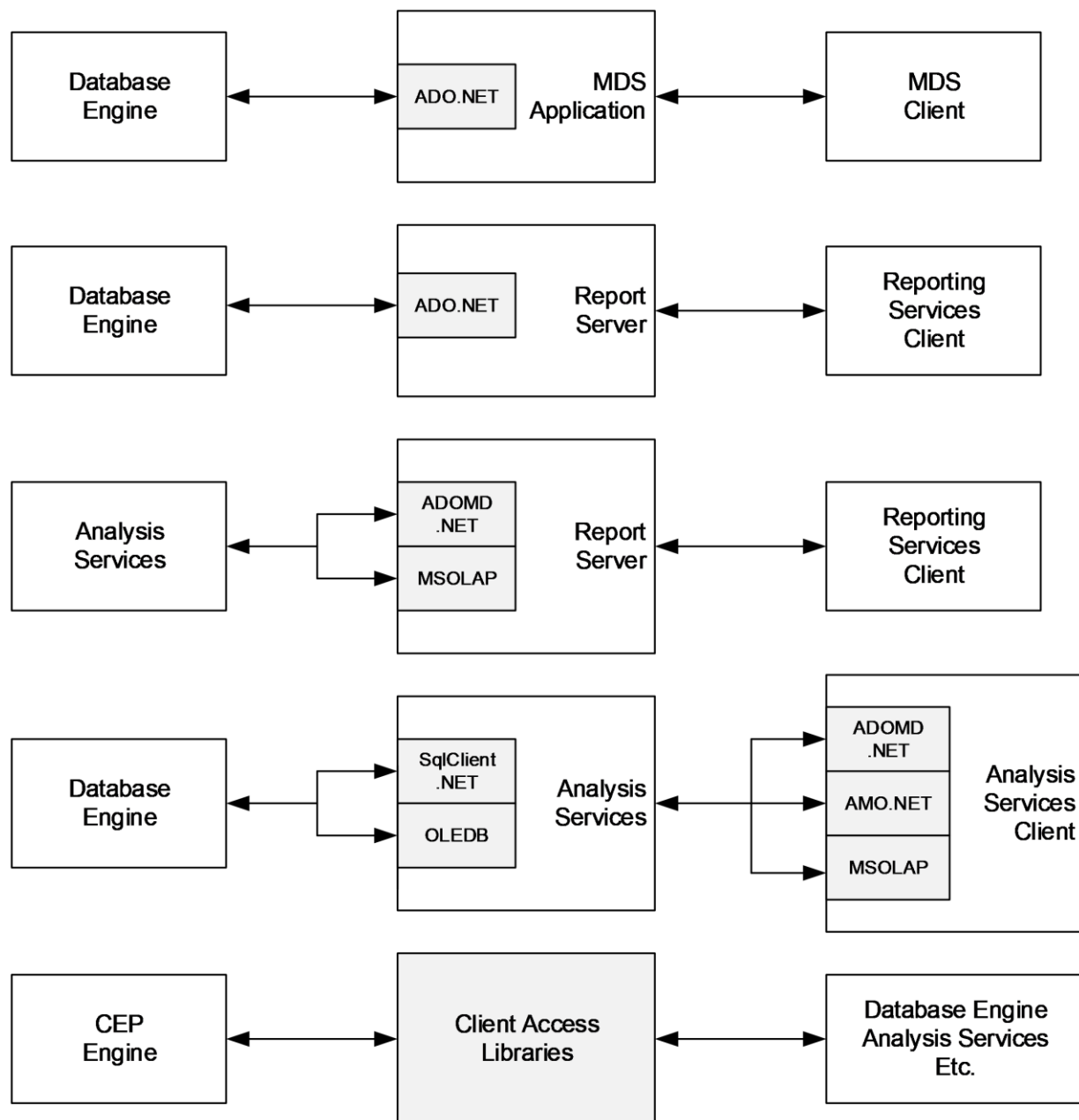


Figure 1: SQL Server architecture overview

2.1.1 Network Connectivity and Application Development

Network connectivity and application development include protocols and formats that are used for communication between the SQL Server Browser, the application, and the Database Engine, as shown in the following figure. For more information about the Database Engine, see section [2.1.5](#).

The Native Web Services protocol [\[MS-SSNWS\]](#) is used to transfer Transact-SQL requests and responses between **web service** client applications and the Database Engine.

The protocols that are used between database applications and the Database Engine are the Session Multiplex Protocol [\[MC-SMP\]](#), which is used to multiplex database communication **sessions** over a

single reliable transport connection, and the Tabular Data Stream (TDS) protocol, which is specified in [\[MS-TDS\]](#) and [\[MS-SSTDS\]](#) and is used to transfer Transact-SQL requests and responses between clients and database products.

The Tabular Data Stream (TDS) protocol also uses the binary XML structure [\[MS-BINXML\]](#), the OLE DB **connection string** structure [\[MS-OLEDBSTR\]](#), the **Open Database Connectivity (ODBC)** connection string structure [\[MS-ODBCSTR\]](#), and the system-provided and user-defined SQL Server CLR types [\[MS-SSCLRT\]](#).

The SQL Server Browser uses the SQL Server Resolution Protocol [\[MC-SQLR\]](#) to resolve the name of a named database server instance and to enumerate available database server instances.

The Microsoft ADO.NET **DataSet DiffGram** structure specification [\[MS-DSDG\]](#) describes how a **DataSet**, a component in the .NET Framework, serializes schema and data for transmission over a network. The Microsoft SharePoint **DataSet DiffGram** structure [\[MS-DSDIFFGRAM\]](#) is used to represent the results of a SharePoint Search service web service call; the **DiffGram** structure is useful for serializing schema and data for transmission over a network or for storage on disk. Note that the **DiffGram** specification that is used by the SharePoint Search service is a subset of the full **DiffGram** structure that is used by the ADO.NET **DataSet**.

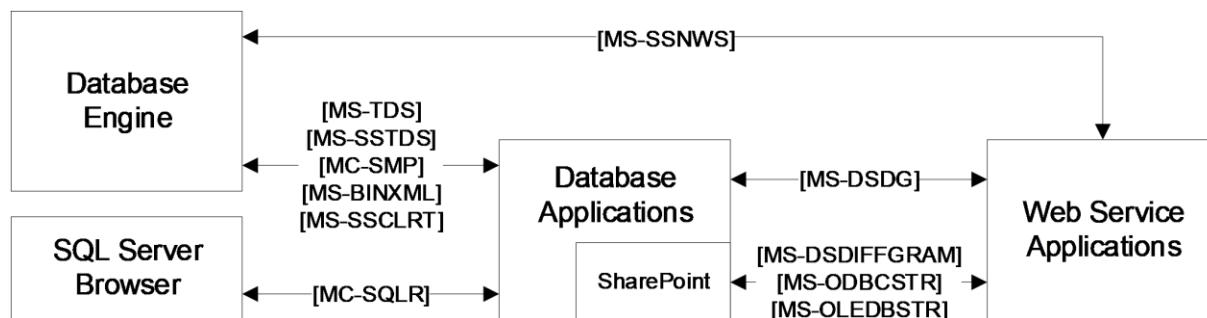


Figure 2: Network connectivity and application development architecture

2.1.2 Master Data Services

Master Data Services (MDS) provides master data management (MDM) capabilities that help provide customers with a single authoritative data source to ensure the integrity of the data they use to make decisions. MDS includes an any-domain hub, a set of services, and an interface that enables organizations to manage important data assets for both line-of-business and analytic applications. MDS is a SQL Server database application, a Windows Communication Foundation (WCF) Services application, and an ASPX application that includes the following:

- **Master Data Hub** for central storage, authoritative source, versioning, rules, and transactions.
- **Stewardship Portal** for model management, documentation, workflow, and integration.

MDS includes protocols that communicate between the MDS application and the MDS web user interface and between the MDS application and external applications [\[MS-SSMDSWS\]](#) [\[MS-SSMDSWS-15\]](#), as shown in the following figure.

MDS uses the Tabular Data Stream (TDS) protocol [\[MS-TDS\]](#) to communicate with the Database Engine.

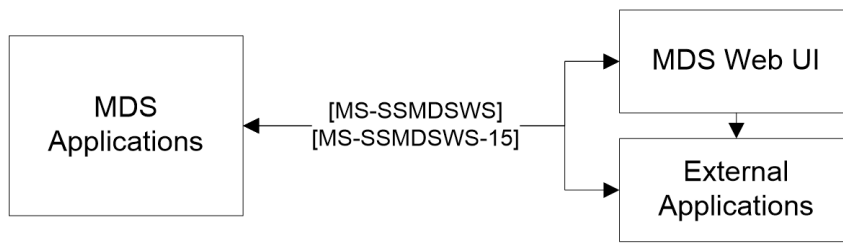


Figure 3: MDS architecture

2.1.3 Reporting Services

SQL Server Reporting Services (SSRS) [\[MSDN-SSRS\]](#) is a server-based reporting platform that provides comprehensive reporting functionality for a variety of data sources. It includes a complete set of tools for creating, managing, and delivering reports and APIs that enable developers to integrate or extend data and report processing in custom applications. Reporting Services tools work within the Visual Studio environment and are fully integrated with SQL Server tools and components.

The **report server** is available by default with a native portal, called the Report Portal, or it can be integrated with an external portal. See the following figures for a view of each architecture.

In Native mode, the report designers, report portals, and any management tools communicate with the report server by using the protocols and formats that are specified in [\[MS-RDL\]](#), [\[MS-RSREST\]](#), [\[MS-RSWSRM2010\]](#), and [\[MS-RSWSRMNM2005\]](#). The Report Viewer tools use the protocols specified in [\[MS-RSREST\]](#), [\[MS-RWSRE2005\]](#), and [\[MS-RGDI\]](#) or [\[MS-RPL\]](#) to communicate with the report server.

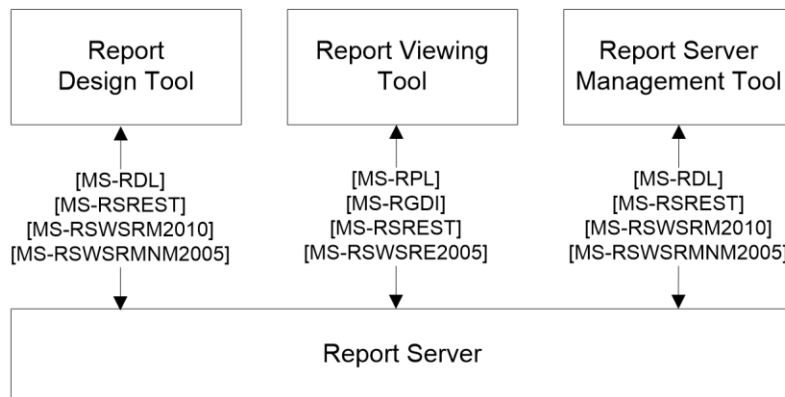


Figure 4: Reporting Services architecture with native management tools

When integrated with an external portal, report designers and any report server management tools communicate with the report server portal via protocols and formats specified in [\[MS-RDL\]](#), [\[MS-RSWSRM2010\]](#), [\[MS-RSWSRMNM2006\]](#), and [\[MS-RSWSSFA\]](#). The Report Viewer Web Part communicates with the report server by using the protocols and formats that are specified in [\[MS-RWSRE2005\]](#) and [\[MS-RGDI\]](#) or [\[MS-RPL\]](#). The portal redirects all requests to the report server.

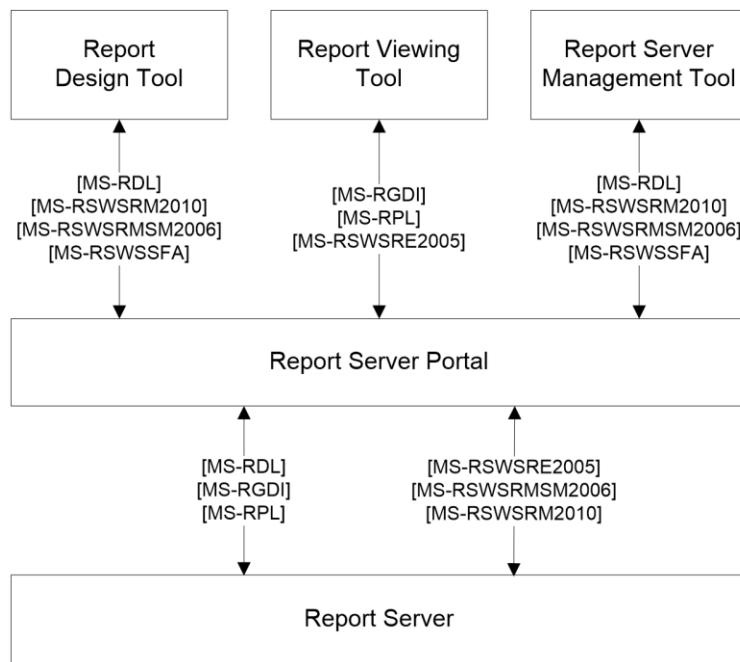


Figure 5: Reporting Services architecture when integrated with an external portal

The report server uses the Tabular Data Stream (TDS) protocol [\[MS-TDS\]](#) to communicate with the Database Engine.

The Report Definition Language [\[MS-RDL\]](#) is used as a payload when setting the definition of a report for execution in the ReportExecution2005 web service [\[MS-RSWSRE2005\]](#). It is also used as a payload when setting or retrieving the definition of a report for the ReportService2010 web service [\[MS-RSWSRM2010\]](#), the ReportService2005 web service [\[MS-RSWSRMNM2005\]](#), or the ReportService2006 web service [\[MS-RSWSRMSM2006\]](#).

Remote GDI+ [\[MS-RGDI\]](#) and Report Page Layout [\[MS-RPL\]](#) are binary **stream** formats that are used during the communication between the Report Viewer and the report server.

2.1.4 Analysis Services

SQL Server Analysis Services (SSAS) enables business intelligence (BI) applications to perform complex and high performance analytics on Multidimensional and Tabular data models. The data models can be enriched with business logic by using Multidimensional Expressions (MDX) or Data Analysis Expressions (DAX) calculations. Additionally, they can expose a unified conceptual model that BI applications can query by using the MDX and DAX languages. Data mining APIs enable the creation, management, and exploration of data mining models. BI applications can run prediction queries on these models by using the Data Mining Extensions (DMX) language.

The SQL Server Analysis Services [\[MS-SSAS\]](#) and SQL Server Analysis Services Tabular [\[MS-SSAS-T\]](#) protocols provide methods for a client to communicate with and perform operations on an analysis server, as shown in the following figure. These protocols are based on **SOAP** and XML for Analysis (XMLA) and use binary XML [\[MS-BINXML\]](#) and OLE DB connection string [\[MS-OLEDBSTR\]](#) formats. The SSAS protocols support TCP/IP as an underlying transport mechanism in addition to HTTP/HTTPS.

The SSAS protocols define the **Authenticate**, **Discover**, and **Execute** operations:

- **Authenticate** is used by the client and server to exchange UTF-8 [\[RFC2279\]](#) encoded security token data blocks as part of the authentication process.

- **Discover** is used to obtain information from an analysis server, such as a list of catalogs on a server. Properties are used to control what data is obtained. This generic interface and the use of properties allow for extensibility without the need to rewrite existing functions.
- **Execute** is used to execute commands against a particular analysis server, optionally returning a result set in either tabular or multidimensional form.

Additionally, by using the Analysis Services Usage Reporting protocol [\[MS-ASUR\]](#), external applications can load Analysis Services models from host servers and then load them onto servers that are running Analysis Services. The host server can then receive information from Analysis Services about how the models are used and what machine resources these models use. This information helps administrators of the host servers to optimize these models.

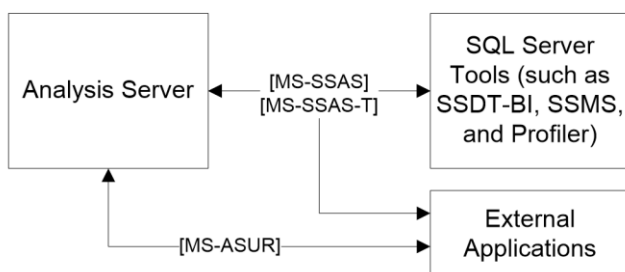


Figure 6: Analysis Services architecture

2.1.5 Database Engine

The Database Engine is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of data-consuming applications within an enterprise.

The Database Engine is used to create relational databases for online transaction processing (OLTP) or online analytical processing data. This includes creating tables for storing data and database objects such as indexes, views, and stored procedures for viewing, managing, and securing data. SQL Server Management Studio can be used to manage the database objects, and SQL Server Profiler can be used for capturing server events.

The Database Engine communicates with its client access libraries (such as **Open Database Connectivity (ODBC)**, **OLE DB**, or ADO.NET) via the Tabular Data Stream (TDS) protocol [\[MS-TDS\]](#), which transports binary XML [\[MS-BINXML\]](#) and the system-provided and user-defined SQL Server CLR types, as specified in [\[MS-SSCLRT\]](#). The client access libraries are used by the database applications to communicate with the Database Engine.

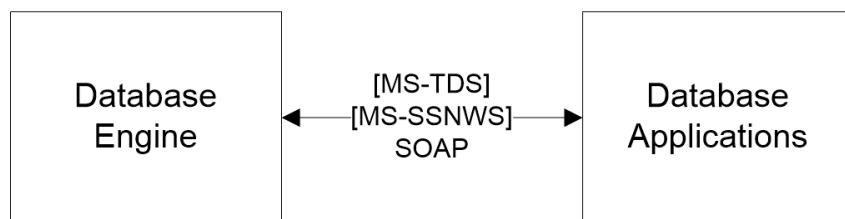


Figure 7: Database Engine architecture

2.1.6 Complex Event Processing Engine

The SQL Server Platform includes a **Complex Event Processing (CEP)** Engine. This is a separate engine with no dependencies on SQL Server and its other components, such as Reporting Services,

Analysis Services, and the Database Engine. Client applications can communicate with the CEP Engine via **web services** for manageability, as shown in the following figure. These client applications can also include database applications that are built on SQL Server.



Figure 8: CEP Engine architecture

The client-side CEP library communicates to the server by using the Complex Event Processing Engine Manageability (CEPM) protocol [\[MS-CEPM\]](#). The CEPM protocol uses the LINQ Expression Tree serialization format [\[MS-LETSF\]](#) to communicate expressions from client to server.

2.1.7 Manageability

Manageability is part of the Database Engine and includes the Database Publishing Wizard Protocol [\[MS-SSDPWP\]](#). This protocol enables a user to publish an existing database to a remote server via a web service. This enables database deployment in hosted scenarios without requiring direct access to the database server, as shown in the following figure.

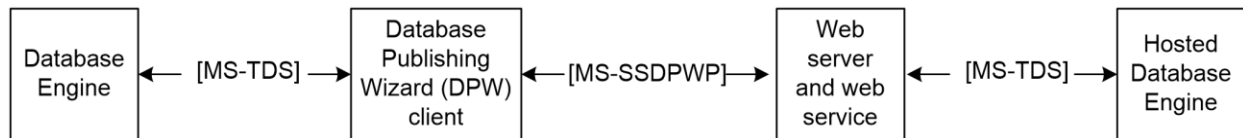


Figure 9: Manageability architecture

2.1.8 Big Data Clusters

SQL Server Big Data Clusters allow the deployment of scalable cluster containers that enable users to read, write, and process big data from **Transact-SQL**. In this way, users can easily combine and analyze high-value relational data with high-volume big data in a managed cluster environment.

The Control Plane REST API [\[MS-CPREST\]](#) provides methods to create a **big data cluster** in which the user manages the lifecycles of resources deployed in a cluster. The Hive Metastore HTTP protocol [\[MS-HMSHTTP\]](#) uses a light-weight interface that defines data services to store and read metadata from **Hive Metastore** databases that are inside a big data cluster and are exposed out of the big data cluster through the **Apache Knox** [\[ApacheKnox\]](#) access point.

The communication between these client applications and SQL Server Big Data Clusters is shown in the following figure.

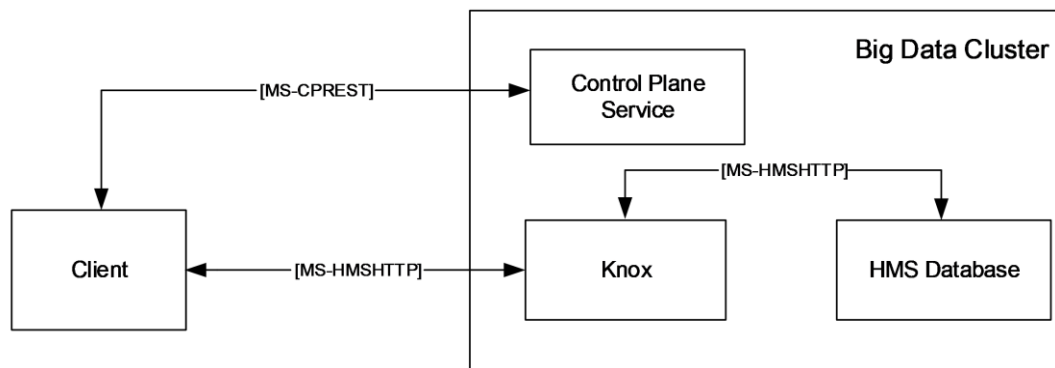


Figure 10: SQL Server Big Data Cluster architecture

2.2 Protocol Summary

The tables in the following sections provide a comprehensive list of the member protocols of the SQL Server system. Member protocols are grouped according to their primary purpose.

2.2.1 Network Connectivity and Application Development

The following protocols enable communication between the browser, an application, and the Database Engine.

Protocol name	Description	Reference
Native Web Services protocol	This protocol is an application layer request/response protocol that facilitates interaction with a database server; it leverages the standard SOAP 1.1 and SOAP 1.2 protocols to provide specific communication logic and a message format that enables ad hoc Transact-SQL query requests and subsequent query result responses.	[MS-SSNWS]
Tabular Data Stream (TDS) version 4.2 protocol	This protocol is an application-level protocol that is used to transfer Transact-SQL requests and responses between clients and database products. The TDS 4.2 protocol provides for: <ul style="list-style-type: none"> Authentication and channel encryption negotiation. Specification of requests in SQL (including bulk insert). Remote procedure calls (RPCs). Return of data. Transaction manager requests. 	[MS-SSTDs]
Session Multiplex protocol (SMP)	This protocol is used between clients and database servers to multiplex database communication sessions over a single reliable transport connection.	[MC-SMP]
SQL Server Resolution Protocol (SSRP)	This protocol is used by clients to locate a named database server instance and to enumerate available database server instances.	[MC-SQLR]
Tabular Data Stream (TDS) protocol	This protocol is an application-level protocol that is used to transfer requests and responses between clients and database server systems. After the connection is established by 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 needs	[MS-TDS]

Protocol name	Description	Reference
	to be established.	

The following formats are used in application development.

Format	Description	Reference
Dataset DiffGram structure	This structure is used to represent the results of a dataset web service call. The DiffGram structure is useful for serializing schema and data for transmission over a network; it can be used to encapsulate both the schema and the data of the dataset.	[MS-DSDG]
SharePoint Web Services: DataSet DiffGram structure	The DataSetDiffGram web service is the subset of the DiffGram structure that is used by the ADO.NET dataset (the ADO.NET dataset is used by the SharePoint Search service to represent the results of a SharePoint Search service web service call). This protocol supports the wire format of the dataset as returned by Enterprise Search in Microsoft Office SharePoint Server 2007.	[MS-DSDIFFGRAM]
SQL Server binary XML structure	Binary XML is a binary format that is used to encode the text form of an XML document into an equivalent binary form, which can be parsed and generated more efficiently. The format provides full fidelity to the original XML document.	[MS-BINXML]
ODBC connection string structure	The reference document describes the format of connection strings that are used by Open Database Connectivity (ODBC) applications. A connection string is a string that specifies information about a data source and the means of connecting to it. The ODBC application determines how to read the connection string to initiate a connection to a data source.	[MS-ODBCSTR]
OleDb connection string structure	The reference document describes the format of connection strings that are used by OLE DB consumers . A connection string is a string that is sent from an OLE DB consumer to an OLE DB provider and that specifies the information that is needed to establish a connection to a data source.	[MS-OLEDBSTR]

2.2.2 Master Data Services

The following protocols are used by Master Data Services (MDS).

Protocol	Description	Reference
Master Data Services web service	This protocol is used when managing or editing any data or containers for data within the Master Data Services (MDS) system. It encapsulates all of the MDS functionality that is exposed to the user. The framework ensures that the internal functions of the product are better modularized in support of both an API and modular component development. "Services" means a modular architecture based on the Windows Communication Foundation (WCF).	[MS-SSMDSWS]
Master Data Services web service 15	This protocol updates the web services implementation of MDS [MS-SSMDSWS] for use with SQL Server 2016.	[MS-SSMDSWS-15]

2.2.3 Reporting Services

The following protocols and formats are used by the Reporting Services [\[MSDN-SSRS\]](#).

Format	Description	Reference
Report Definition Language file format	SQL Server Report Definition Language (RDL) is a file type that is used to represent the metadata for defining a report.	[MS-RDL]
Report Page Layout (RPL) binary stream format	RPL is a binary format that is used by SQL Server Reporting Services when it communicates with viewer controls to offload some of the rendering work from the server to the client viewer control. By using RPL, implementers can create custom report designers that will create RPL, as well as custom report renderers that process and display RPL to display reports.	[MS-RPL]
Remote GDI+ (RGDI) binary stream format	RGDI is a binary format that is produced by SQL Server Reporting Services when it communicates with viewer controls to offload some of the rendering work from the server to the client viewer control. By using RGDI, the implementer can create custom report designers that will generate RGDI, and can create custom report renderers that process and display RGDI to display reports. A client control that consumes RGDI is required to translate RGDI content to GDI+ objects and calls. RGDI works with only one page of report data at a time; each page is represented by an RGDI stream.	[MS-RGDI]
Report Server Web Service for Report Management: ReportService2010	This protocol is used for managing objects and settings on all report servers , starting with Microsoft SQL Server 2008 R2.	[MS-RWSRM2010]
Report Server Web Service for Report Management for SharePoint Mode: ReportService2006	This protocol is used for managing objects and settings on a report server that is configured for SharePoint integrated mode.	[MS-RWSRMSM2006]
Report Server Web Service for Report Execution: ReportExecution2005	This API protocol is provided by the ReportService2005 web service for managing objects and settings on a report server that is configured for SharePoint integrated mode.	[MS-RWSRE2005]
Report Server Web Service for Report Management Native Mode: ReportService2005	This protocol is provided by the ReportService2005 web service for managing objects and settings on a report server that is configured for native integrated mode.	[MS-RWSRMNM2005]
Report Server Web Service for SharePoint Forms Authentication ReportServiceAuthentication	This API protocol is provided by the ReportService2006 web service for managing objects and settings on a report server that is configured for SharePoint integrated mode.	[MS-RSWSSFA]
Reporting Services REST API protocol	This RESTful API specifies an HTTP-based web service API to manage objects and settings on a report server that is configured for native integrated mode.	[MS-RSREST]

2.2.4 Analysis Services

The following protocols are used by Analysis Services.

Protocol	Description	Reference
Analysis Services Usage Reporting protocol	This protocol specifies a method by which a client application, that gathers Analysis Services models from a host server and then loads them onto other servers that are running Analysis Services, can report back to that host server with the details about how those models are being used and the resources those models consume on the other servers.	[MS-ASUR]
SQL Server Analysis Services protocol	This protocol provides a mechanism for a client to communicate with and perform operations on an analysis server.	[MS-SSAS]
SQL Server Analysis Services Tabular protocol	This protocol extends the SQL Server Analysis Services protocol [MS-SSAS] by specifying the methods for a client to communicate with and perform operations on an analysis server that uses Tabular databases that are at compatibility level 1200 or higher.	[MS-SSAS-T]
SQL Server Analysis Services 8.0 protocol	This protocol provides a mechanism for a client to communicate with and perform operations on an Online Analytical Processing (OLAP) server. This protocol is for Microsoft SQL Server 7.0 and Microsoft SQL Server 2000. This legacy protocol is superseded by [MS-SSAS] .	[MS-SSAS8]

2.2.5 Database Engine

The following format is used with the Database Engine.

Format	Description	Reference
SQL Server Common Language Runtime (CLR) Types structure	<p>This binary format consists of the GEOGRAPHY, GEOMETRY, HIERARCHYID, and common language runtime (CLR) user-defined type (UDT) structures managed by SQL Server.</p> <p>The structures used to transmit geography and geometry data types over the wire implement the Open Geospatial Consortium's (OGC) OpenGIS Simple Features Specification (SFS). SQL Server, therefore, provides data to a client program as text or binary representations of points, lines, polygons, and collections that conform to the specification.</p> <p>The hierarchyid data type enables SQL Server applications to store, retrieve, and manipulate hierarchical data.</p>	[MS-SSCLRT]

The following protocol is used with the Database Engine.

Protocol	Description	Reference
MSDTC Connection Manager: OleTx Transaction Protocol	<p>This protocol enables the creation, initiation, and distributed propagation of transactions, and the participation in transactions.</p> <p>SQL Server running on either a Windows or Linux operating system can provide two-phase commit functionality for distributed transactions by using the OleTx Transaction Protocol to enable an MSDTC service to communicate with other MSDTC service instances.<1></p> <p>However, when SQL Server is running on Linux, the MSDTC communication with other MSDTC instances occurs over the OleTx Transaction Protocol by using SQL Server binary. This implementation of SQL Server on Linux does not extend or modify the OleTx Transaction Protocol.</p>	[MS-DTCO]

2.2.6 CEP Engine

The following protocol is used with the Complex Event Processing (CEP) Engine.

Protocol	Description	Reference
Complex Event Processing Engine Manageability protocol	Complex event processing (CEP) is the continuous and incremental processing of event (data) streams from multiple sources based on declarative query and pattern specifications with near-zero latency. The CEP Engine provides a dedicated web service to handle requests for managing the system.	[MS-CEPM]
LINQ Expression Tree serialization format	An XML serialization format for arbitrary LINQ expressions. This is used by the CEP Engine to communicate logic between client and server.	[MS-LETSF]

2.2.7 Manageability

The following protocol is used for manageability functionality.

Protocol	Description	Reference
Database Publishing Wizard Protocol	With this protocol, a publishing session can be initiated, data can be published, and scripts can be executed against an instance of SQL Server.	[MS-SSDPWP]

2.2.8 Big Data Clusters

The following protocols are used with **big data clusters**.

Protocol	Description	Reference
Control Plane REST API Protocol	The SQL Control REST API is used to deploy and manage data services and applications into a managed cluster environment. The cluster environment can exist on premise, in the cloud, or in hybrid cloud configurations.	[MS-CPREST]
Hive Metastore HTTP Protocol	The Hive Metastore HTTP protocol specifies a web service API that provides a lightweight interface for clients to read catalog metadata from a Hive Metastore database that has been deployed as a data service inside a managed cluster environment.	[MS-HMSHTTP]

2.3 Environment

The following sections identify the context in which the system exists. This includes the systems that use the interfaces provided by this system of protocols, other systems that depend on this system, and as appropriate, the method by which components of the system communicate.

2.3.1 Dependencies on This System

None.

2.3.2 Dependencies on Other Systems or Components

None.

2.3.3 Communications within the System

Communications within the system refers to protocols that are externally visible. All of these are documented in the previous sections. The servers that make up the system—Analysis Services, Reporting Services, MDS, the Database Engine, and the Complex Event Processing (CEP) Engine—use these protocols for communication.

2.3.4 Assumptions and Preconditions

This section briefly documents the assumptions and preconditions that are required by the system. The scope of this discussion is intended to be implementation-independent and is limited to the system level.

The servers running SQL Server are reachable by external clients via an established IP address or IP addresses.

The functional components of the servers running SQL Server are started collectively and the servers accept client requests.

The SQL Server servers—Reporting Services, Analysis Services, MDS, and the Database Engine—are matching versions or are within an acceptable range of versions. For more information about versioning, see section [2.5](#).

When **Active Directory** is used to provide authentication, the directory service is accessible to the server running SQL Server. Any intermediate firewalls, routers, or connection points between components of the system have all required ports and gateways open for communication between them.

2.4 Use Cases

2.4.1 Network Connectivity and Application Development Use Cases

2.4.1.1 SQL Server Instance Discovery Use Case

This use case describes how an instance of SQL Server is discovered in the local network. The actors are the application and the SQL Server Browser.

Actions

1. A database application broadcasts an SSRP [\[MC-SQLR\]](#) message (UDP packet).
2. Each SQL Server Browser service that receives the message responds with the instances of SQL Server (another UDP packet) on its computer. The database application now has a list of instances of SQL Server to which it can connect.

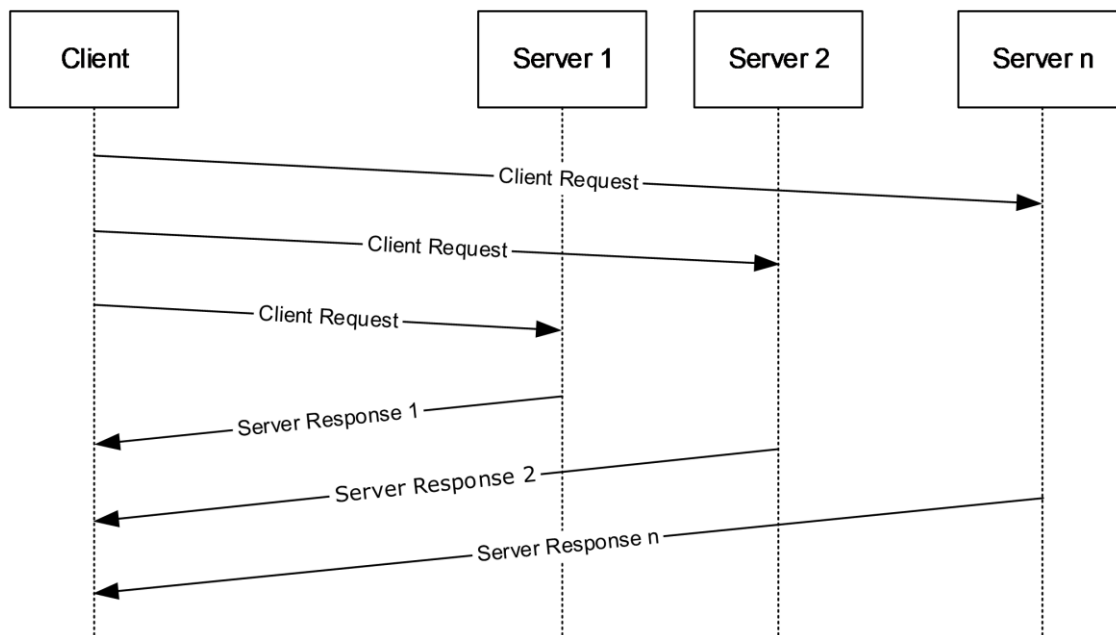


Figure 11: SQL Server instance discovery use case

2.4.1.2 Named SQL Server Instance Resolution/Enumeration

This use case describes how the connectivity protocol details of a named instance of SQL Server are discovered. The actors are the application and the SQL Server Browser.

Actions

1. A database application sends an SSRP [\[MC-SQLR\]](#) message to the SQL Server Browser service with the name of the desired instance of SQL Server.
2. The SQL Server Browser responds either with protocol details that specify where the instance of SQL Server is listening or with an error indicating that there is no such named instance of SQL Server.
3. The database application has a list of instances of SQL Server to which it can connect.

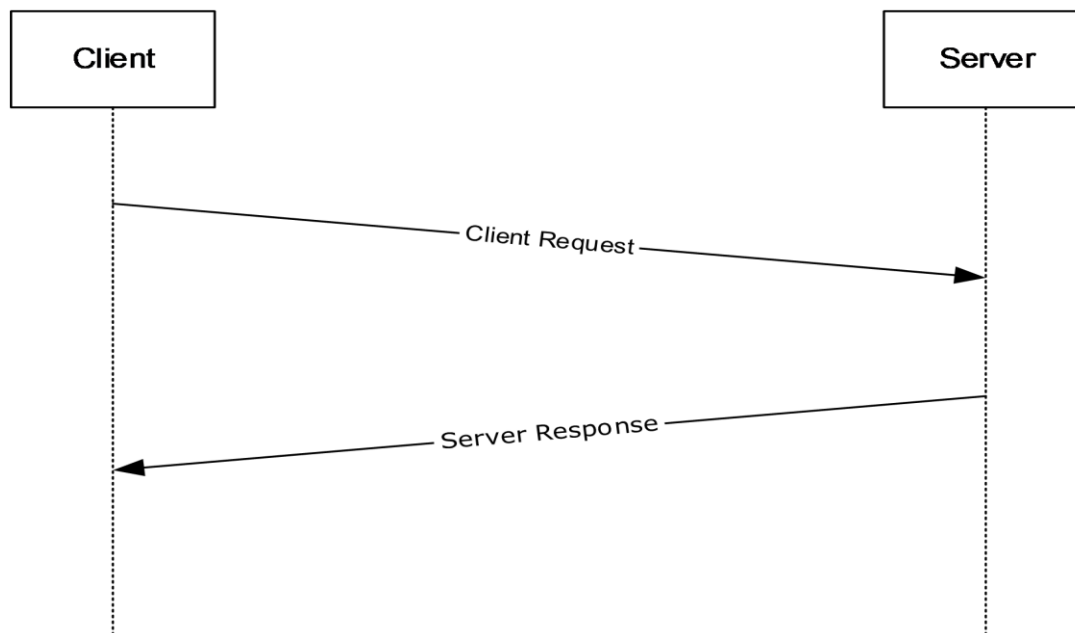


Figure 12: Named SQL Server instance resolution use case

2.4.1.3 Client Connection (TDS, SSTDS, or SSNWS)

This use case describes how to establish a connection and to execute commands to exchange data or to produce side effects. The actors are the database application and SQL Server.

Actions

1. A database application engages in an authentication handshake with a desired authentication scheme (SQL Server or Windows).
2. SQL Server verifies the database application's credentials and either confirms that the connection was established or terminates the connection if authentication fails.
3. The database application sends a command.
4. SQL Server responds with the execution status and, eventually, data if the command produces a result. (The last two steps might be repeated.)
5. The database application terminates the connection.
6. The database application might have caused side effects by executing commands and/or might have exchanged data with SQL Server.

For Tabular Data Stream (TDS) only: If the database application requires the use of Multiple Active Result Sets (MARS), the Session Multiplex Protocol (SMP) is used underneath for virtual connection multiplexing.

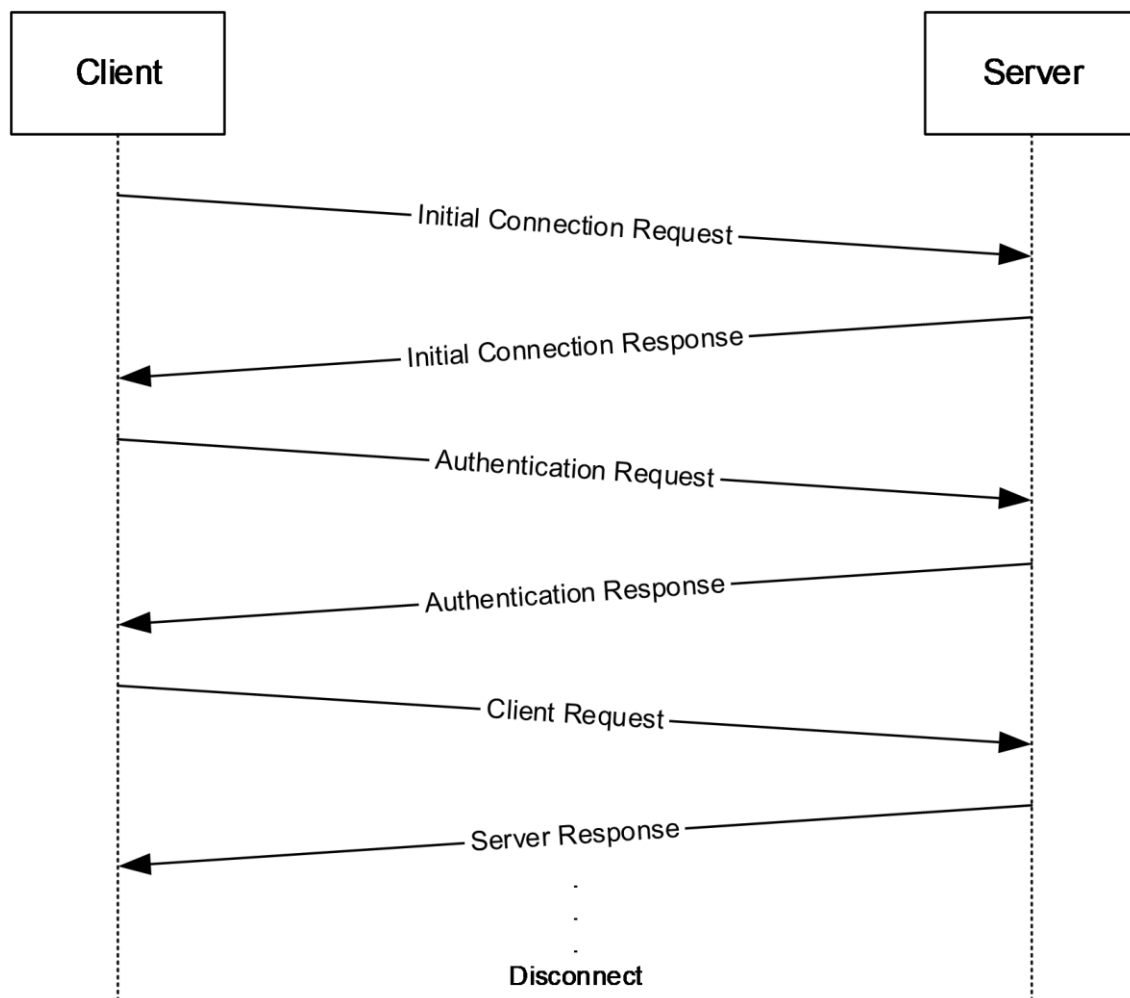


Figure 13: Client connection use case

Payloads

Payloads and **connection strings** are carried on the protocols, as follows:

- Binary XML type values, as specified in [\[MS-BINXML\]](#) are transported by the Tabular Data Stream (TDS) protocol, as specified in [\[MS-TDS\]](#).
- SQL Server CLR types specified in [\[MS-SSCLRT\]](#) are transported by the TDS protocol.
- The **DataSet DiffGram** structure types specified in [\[MS-DSDG\]](#) and [\[MS-DSDIFFGRAM\]](#) are carried by a generic (or standard) SOAP protocol.
- Open Database Connectivity (ODBC) and OLE DB connection string types specified in [\[MS-ODBCSTR\]](#) and [\[MS-OLEDBSTR\]](#) are carried by a generic SOAP.

2.4.2 MDS Integration Use Cases

2.4.2.1 Use the MDS UI to Query

This use case describes how to perform the **Create**, **Read Update**, and **Delete Object** operations within the MDS store. The actors are the MDS web user interface and the external application. Note

that security authorization to the target object with the Master Data Services web service protocol [\[MS-SSMDSWS\]](#) [\[MS-SSMDSWS-15\]](#) is required.

Actions

1. The MDS web UI provides a Master Data Services web service protocol request message that specifies the object or objects to modify or query.
2. The application modifies or returns these objects in a response message.

2.4.2.2 Query a List from the MDS Store

Actions

1. Using the **EntityMembersGet** operation, the user forms a request that describes the context of the data desired and the criteria to retrieve it.
2. The user submits the request message.
3. The requesting client's security permissions are evaluated, and if authorization is successful, the data is returned in the response message.
4. If the user is unauthorized, the user is returned an empty result set and an error message detailing the security required for access.
5. The valid result set is formatted according to the criteria provided in the request message.

2.4.3 Reporting Services Use Cases

2.4.3.1 Report Authoring, Management, and Viewing with Native Report Portal

Actions

1. Within a report authoring tool, a report author designs a report and saves it in RDL file format, as specified in [\[MS-RDL\]](#).
2. The authoring tool publishes the saved report to a **report server** via methods of the ReportService2010 API, as specified in [\[MS-RWSRM2010\]](#).
3. Within the report server management tool, the report server administrator manages the reports, resources, schedules, subscriptions, and users within the report catalog.
4. Each management operation performed by the administrator is implemented as an API call from the management tool to the ReportService2010 API.
5. Administrators can publish reports in RDL format via these APIs.
6. Within a report viewing tool, a report consumer specifies a report to execute from within the report server catalog.
7. The report viewer passes this instruction to the report server via the ReportExecution2005 API, as specified in [\[MS-RWSRE2005\]](#).
8. The report server returns the results of the report execution to the report viewer as either Report Page Layout, as specified in [\[MS-RPL\]](#) or a Remote GDI+, as specified in [\[MS-RGDI\]](#), depending on which was requested by the report viewer. Note that other standard formats such as HTML, XML, and PDF can be requested and returned instead.

2.4.3.2 Mobile Report Authoring, Management, and Viewing with Native Report Portal

Actions

1. Within a report authoring tool, a report author designs a mobile report and publishes it to a server by using the Reporting Services RESTful API, as specified in [\[MS-RSREST\]](#).
2. By using a report server management tool, the report server administrator manages the reports, schedules, and users within the report catalog.
3. Each management operation performed by the administrator is implemented as an API call from the management tool to the ReportService2010 API, as specified in [\[MS-RSWSRM2010\]](#).
4. By using a report viewing tool, a report consumer specifies a report to execute from within the report server catalog.
5. The report viewing tool passes this instruction to the report server via the Reporting Services RESTful API.
6. The report server returns the results of the report execution to the report viewer via the Reporting Services RESTful API.

2.4.3.3 Report Authoring, Management, and Viewing with External Report Portal

Actions

1. Within a report authoring tool, a report author designs a report and saves it in RDL file format, as specified in [\[MS-RDL\]](#).
2. The authoring tool authenticates the user against the **report server** portal via the ReportServiceAuthentication API, as specified in [\[MS-RSWSSFA\]](#), and publishes the saved report to a report server portal via the methods of the ReportService2010 API, as specified in [\[MS-RSWSRM2010\]](#).
3. Calls to ReportService2010 are either individually redirected to the report server by the report server portal or are proxied to the report server via the ReportServiceAuthentication API.
4. Within the report server management tool, the report server administrator manages the reports, resources, schedules, subscriptions, and users within the report catalog. The management tool authenticates the administrator against the report server portal via the ReportServiceAuthentication API.
5. Each management operation performed by the administrator is implemented as an API call from the management tool to the ReportService2010 API.
6. Calls to both ReportServiceAuthentication and ReportService2010 are either individually redirected to the report server by the report server portal or are proxied to the report server via ReportServiceAuthentication. Reports in the RDL format can be published by the administrator via this API.
7. Within a report viewing tool, a report consumer specifies a report to execute from within the report server catalog. The report viewer passes this instruction to the report server portal via the ReportExecution2005 API, as specified in [\[MS-RSWSRE2005\]](#), and it is either individually redirected to the report server by the report server portal or is proxied to the report server via the ReportServiceAuthentication API.
8. The report server returns the results of the report execution to the report viewer as either Report Page Layout [\[MS-RPL\]](#) or Remote GDI+ [\[MS-RGDI\]](#), depending on which was requested by the report viewer. Note that other standard formats such as HTML, XML, and PDF can be requested and returned instead.

2.4.4 Analysis Services Use Cases

The following use cases show how to authenticate, discover, and execute a request.

2.4.4.1 Authentication with the Analysis Server

This use case describes how a client connects and authenticates itself with an **analysis server**.

Actions

1. The client establishes a **TCP** connection to the server.
2. The client sends an **Authenticate** request that contains an SSPI security token.
3. The server returns an **AuthenticateResponse** response containing an SSPI security token.
4. Steps 2 and 3 are repeated as necessary to complete the authentication as required by the SSPI provider.

2.4.4.2 Information Discovery

This use case describes how a client discovers information about databases on an **analysis server**.

Actions

1. The client sends a **Discover** request that has the DBSCHEMA_CATALOGS request type to retrieve the list of databases.
2. The server responds with a rowset that contains the list of databases and some properties associated with each database.

2.4.4.3 Sending an MDX Query

This use case describes how a client sends an MDX query to a data model on an **analysis server**.

Actions

1. The client sends an **Execute** request containing an MDX query to a cube on the analysis server.
2. The server responds with a multidimensional dataset that contains the query result.

2.4.4.4 Sending a DAX Query

This use case describes how a client sends an DAX query to a data model on an analysis server.

Actions

1. The client sends an **Execute** request containing a DAX query to a data model on the analysis server.
2. The server responds with a tabular dataset that contains the query result.

2.4.4.5 Usage Reporting

This use case describes how a client receives information about model usage from an analysis server.

Actions

1. The client sends a SOAP request to the Analysis Services server that includes a WSDL operation that is defined by usage reporting.
2. The server responds with information about the model. The values in the response are models or handles that are used by other WSDL operations.

2.4.5 Database Engine Use Cases

2.4.5.1 Authentication with the Database Engine

This use case describes how a client connects and authenticates itself with the Database Engine.

Actions

1. The client establishes a connection to SQL Server by using one of the protocols in the client access library.
2. The client sends an **Authenticate** request that contains an SSPI security token.
3. The server returns an **AuthenticateResponse** response that contains an SSPI security token.
4. Steps 2 and 3 are repeated as necessary to complete the authentication as required by the SSPI provider.

2.4.5.2 Information Discovery

This use case describes how a client discovers information on the Database Engine by using one of the protocols in the client access library.

Actions

1. The client sends a **Discover** request to retrieve the list of databases.
2. The server responds with a rowset that contains the list of databases and some properties that are associated with each database.

2.4.5.3 Sending a Query

This use case describes how a client sends an SQL query to the Database Engine by using one of the protocols in the client access library.

Actions

1. The client sends an **Execute** request that contains an SQL query to the Database Engine.
2. The server responds with a rowset that contains the results of the SQL query.

Payloads

- CLR-type data, as specified in [\[MS-SSCLRT\]](#) are transported by the Tabular Data Stream (TDS) protocol, as specified in [\[MS-TDS\]](#).

2.4.6 CEP Engine Use Case

The Complex Event Processing Engine Manageability (CEPM) protocol, as specified in [\[MS-CEPM\]](#), is a web service protocol that defines the communication protocol between a client application and a complex event processing (CEP) server. Using this protocol, a client application can create metadata objects on a CEP server, start and stop queries, and query about the CEP system state. The CEPM

protocol is stateless. All communication is initiated by the client. The server sends responses only in response to messages received.

The client and server use the LINQ Expression Tree serialization format, as specified in [\[MS-LETSF\]](#), to transfer expression logic between each other.

Actions

1. A protocol message to start the query is sent.
2. The message causes the CEP Engine to tap into the streaming data.
3. The engine calculates and sends output data.
4. Another message stops the engine from computing data.

2.4.7 Manageability Use Case

The following is an example procedure in which a client uses the Database Publishing Wizard Protocol to create a simple database and then populate it with data. Note that an appropriate server connection and user credentials are required.

Actions

1. The client invokes the **BeginPublish** operation together with the appropriate server connection and user credentials.
2. The client invokes the **PublishScript** operation with a **Transact-SQL** script, which creates the new database and the database objects.
3. The client invokes the **EndPublish** operation to finish the publishing session and to release server resources.

2.4.8 Big Data Cluster Protocols Use Cases

The use cases in this section show how to use the [\[MS-CPREST\]](#) protocol to provision a **big data cluster** and retrieve endpoint information, and also show how to use the [\[MS-HMSHTTP\]](#) protocol to retrieve information from the **Hive Metastore**.

2.4.8.1 Control Plane REST API Use Cases

2.4.8.1.1 Provision a Big Data Cluster

This use case shows how a client connects to the **control plane** and then provisions a **big data cluster**.

Actions

1. The client initiates a **TCP** connection to the control plane.
2. The client sends a **Create Big Data Cluster** request to initiate the creation of a big data cluster.
3. The control plane accepts the request and begins the creation of the big data cluster.
4. The client sends a **Get Big Data Cluster Status** request, which returns the deployment status of the big data cluster creation.
5. Step 4 is repeated as necessary until the big data cluster is ready.

2.4.8.1.2 Retrieve Big Data Cluster Endpoints

This use case shows how a client retrieves information about the endpoints within a **big data cluster**.

Actions

1. The client initiates a **TCP** connection to the **control plane service** endpoint.
2. The client sends a **Get All BDC Endpoints** request.
3. The **control plane** returns a **JSON** object that contains information about all endpoints that are exposed by the big data cluster.

2.4.8.2 Hive Metastore HTTP Use Cases

This use case shows how to query a **Hive Metastore** for metadata information.

Actions

1. The client initiates a **TCP** connection to an **Apache Knox** [\[ApacheKnox\]](#) endpoint.
2. The client uses an HTTP POST operation to send a query requesting database, table, or partition information.
3. The Hive Metastore HTTP service responds to the POST operation with the requested information.

2.5 Versioning, Capability Negotiation, and Extensibility

There are no versioning, capability negotiation, or extensibility considerations.

2.6 Error Handling

There are no system-level error-handling behaviors. In general, for errors returned as part of a protocol in SQL Server, the protocol documents describe what the error means for the system when they are defined. How they are handled, based on the protocol description, is an issue for the implementer.

2.7 Coherency Requirements

SQL Server has no system-level coherency requirements beyond specific details that are covered in the protocol documents.

2.8 Security

Security can be handled differently by the different protocols. For detailed information about protocol security, see the protocols that are listed in section [2.2](#).

3 Examples

The examples that follow map to the use cases that are described in section [2.4](#).

3.1 Configuring and Administering Multiple Servers

This example shows a user setting up an enterprise system by using authoring and management tools so that all servers can be administered from a central location.

Content for these servers—the **analysis server**, the **report server**, and the MDS server—is defined by using authoring tools.

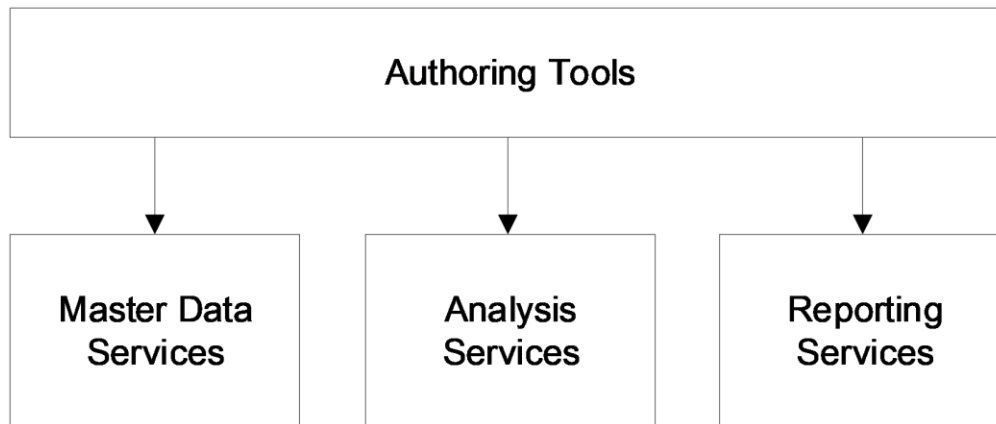


Figure 14: Defining content for the servers

Management tools are then used to set up the enterprise system.

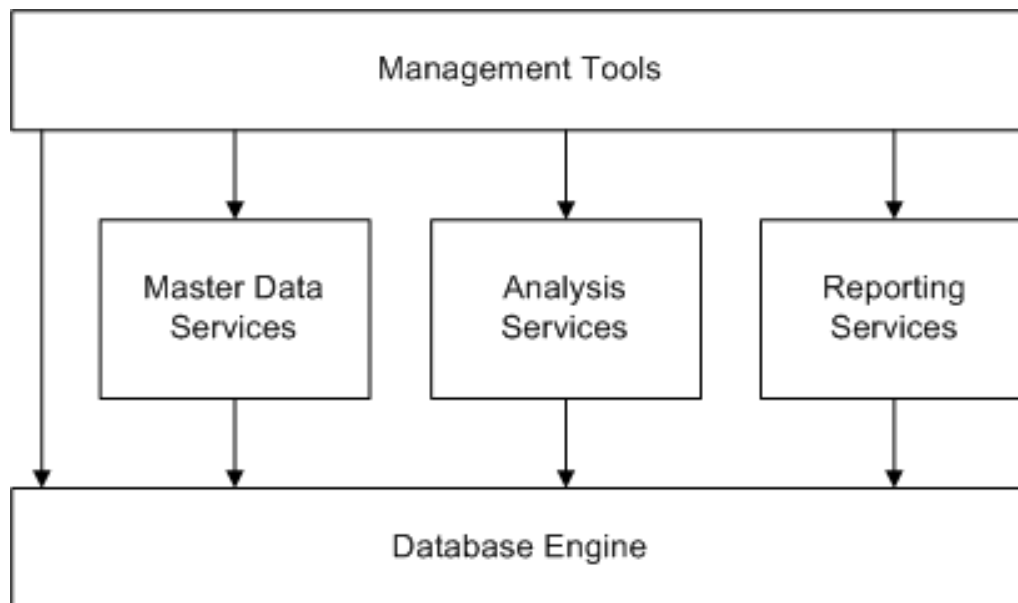


Figure 15: Using management tools to set up the system

3.1.1 Analysis Services Authoring and Management

This example maps to the use cases for Analysis Services in section [2.2.4](#) and to the figures in section [3.1](#).

1. The IT administrator defines a data model via a data model authoring tool. The data model can be either a Multidimensional cube or a Tabular model. Both model types can provide a description of which information to retrieve and which transformations to perform.
2. The IT administrator publishes the data model to the Analysis Services server.
3. With a management tool, the IT administrator directs Analysis Services to process the data.
4. The Analysis Services server issues a sequence of queries against the Database Engine.
5. The Database Engine returns the data for each query.
6. Analysis Services processes and stores the results.
7. Analysis Services returns the status to the management tool.

3.1.2 Reporting Services Authoring and Management

This example maps to the use cases for Reporting Services in section [2.4.3](#) and to the figures in section [3.1](#).

1. Using a report authoring tool, the report author defines a report and publishes it to the Reporting Services server.
2. The Reporting Services server compiles the report and submits it to the Database Engine for storage.
3. The Database Engine returns the status.
4. The Reporting Services server then returns the status to the author.
5. The Reporting Services server uses Management Studio and other management tools to request a list of reports from the Reporting Services server.
6. The Reporting Services server issues a query to get reports.
7. The Database Engine returns the results.
8. The Reporting Services server returns the results.

3.1.3 MDS Management

This example maps to the use cases for Master Data Services (MDS) in section [2.4.2](#) and to the figures in section [3.1](#).

1. Using an MDS authoring tool, the data steward defines an entity and publishes it to the MDS server.
2. The MDS server compiles the entity and submits it to the Database Engine for storage.
3. The Database Engine returns the status.
4. The MDS server returns the status to the data steward.
5. The MDS server uses the MDS portal to request a list of models and entities from the MDS server.

6. The MDS server issues a query to get the models and entities.
7. The Database Engine returns the results.
8. The MDS server returns the results.

3.1.4 Database Engine Management

This example maps to the use cases for network connectivity and application development in section [2.4.1](#) and Database Engine in section [2.4.5](#) and to the figures in section [3.1](#).

1. Using management tools, such as Azure Data Studio or SQL Server Management Studio, an administrator attempts to get a list of database instances on the network.
2. The management tools broadcast the request to all computers on the network.
3. The SQL Server instances on the network respond.
4. The administrator selects a database instance and requests a list of databases on that instance.
5. The management tools issue a query to the Database Engine requesting the data that the administrator wants.
6. The administrator selects a database instance and requests a list of tables on that instance.
7. The management tools issue a query to the Database Engine, requesting the data that the administrator wants.
8. A list of tables is returned to the management tools and is displayed to the administrator.

3.2 Obtaining Data

This example shows a user consuming data in several forms.

3.2.1 Obtaining Data via Analysis Services

The steps for this example are similar to those in section [3.2.2](#), but a pivot table is published by Microsoft Excel at the Analysis Services server, that is, a pivot table is manipulated and data is retrieved from the Analysis Services server.



Figure 16: User obtaining data via Analysis Services

3.2.2 Obtaining Data via Reporting Services

Using Windows Internet Explorer, the user navigates to the SharePoint portal and clicks a report to execute.

1. The SharePoint portal, acting as a Reporting Services client, requests that the report be executed by Reporting Services.

2. Reporting Services requests the underlying data for the reports from the Database Engine or the Analysis Services Engine.
3. The engine returns the data that was processed by Reporting Services.
4. Reporting Services returns a representation of the report to SharePoint.
5. SharePoint returns the resulting web page to Internet Explorer.

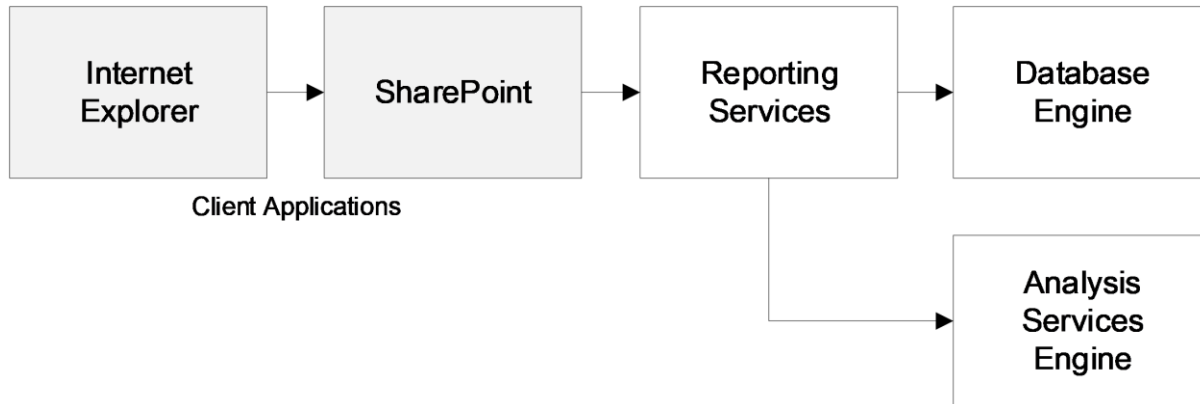


Figure 17: User obtaining data via Reporting Services

3.2.3 Obtaining Data via MDS

1. Using Internet Explorer, the user navigates to the MDS portal and clicks an entity to manage.
2. MDS requests the underlying data for the reports from the Database Engine.
3. The engine returns the data that was processed by MDS.
4. MDS returns the data to the MDS portal.
5. The MDS portal renders the data in Internet Explorer.



Figure 18: User obtaining data via MDS

4 Microsoft Implementations

The information in this specification is applicable to one or more of the following product versions:

- Microsoft SQL Server 2005
- Microsoft SQL Server 2008
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2012
- Microsoft SQL Server 2014
- Microsoft SQL Server 2016
- Microsoft SQL Server 2017
- Microsoft SQL Server 2019

Exceptions, if any, are noted in the following section.

4.1 Product Behavior

[<1> Section 2.2.5](#): The use of the MSDTC service and functionality in SQL Server on Linux is not supported by SQL Server 2005, SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, SQL Server 2014, SQL Server 2016, and SQL Server 2017.

5 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
1 Introduction	Added information about big data clusters.	Major
2.1 Overview	Updated figure and added ADOMD.NET and AMO to client access libraries.	Major
2.1.8 Big Data Clusters	Added section.	Major
2.2.5 Database Engine	Added information about the capability of SQL Server on Windows or Linux to provide two-phase commit functionality for distributed transactions by using an MSDTC service.	Major
2.2.8 Big Data Clusters	Added section.	Major
2.4.8 Big Data Cluster Protocols Use Cases	Added section.	Major
2.4.8.1 Control Plane REST API Use Cases	Added section.	Major
2.4.8.1.1 Provision a Big Data Cluster	Added section.	Major
2.4.8.1.2 Retrieve Big Data Cluster Endpoints	Added section.	Major
2.4.8.2 Hive Metastore HTTP Use Cases	Added section.	Major
3.1.4 Database Engine Management	Added Azure Data Studio as an available management tool.	Major
4 Microsoft Implementations	Added SQL Server 2019 to the list of applicable products and accompanying text to call attention to product behavior exceptions.	Major

6 Index

A

- [administering servers](#) 33
- [ADO.NET DataSet Diffgram structure](#) 12
- Analysis Services ([section 2.1.1](#) 12, [section 2.1.4](#) 15)
 - [architecture](#) 15
 - [authentication](#) 29
 - [DAX queries](#) 29
 - [information discovery](#) 29
 - [management example](#) 34
 - [MDX queries](#) 29
 - [network connectivity](#) 12
 - [obtaining data](#) 35
 - [protocols](#) 21
 - [use cases](#) 29
- Analysis Services Usage Reporting Protocol
 - [about](#) 15
- Analysis services use cases
 - [overview](#) 29
- [Applicable protocols](#) 18
- application development
 - [architecture](#) 12
 - [example](#) 35
 - [protocols](#) 18
 - [use cases](#) 23
- [Architecture](#) 11
- [Authenticate operation](#) 15
- authentication
 - [Analysis Services](#) 29
 - [Database Engine](#) 30
- authoring reports
 - [external report portal](#) 28
 - native report portal ([section 2.4.3.1](#) 27, [section 2.4.3.2](#) 28)

B

- Big data cluster protocols use cases
 - [overview](#) 31
- binary XML
 - [application development](#) 12
 - [Database Engine](#) 16
 - [network connectivity](#) 12
 - [protocol summary](#) 18

C

- [Capability negotiation](#) 32
- CEP Engine
 - [about](#) 16
 - [protocols](#) 22
 - [use case](#) 30
- Cep engine use case
 - [overview](#) 30
- [Change tracking](#) 38
- [client connection use case](#) 25
- CLR types
 - [Database Engine](#) 16
 - [reference document](#) 21
- [Coherency requirements](#) 32
- [Communications](#) 22

- [within the system](#) 22
- [Complex Event Processing Engine Manageability Protocol](#) 22
 - [about](#) 16
- [Concepts](#) 11
- [configuring servers](#) 33
- Considerations
 - [security](#) 32

D

- [Data Mining Extensions](#) 15
- [Database Engine](#) 12
 - [authentication](#) 30
 - [information discovery](#) 30
 - [management example](#) 35
- protocols ([section 2.1.1](#) 12, [section 2.1.5](#) 16, [section 2.2.5](#) 21)
 - [sending query to](#) 30
 - [use cases](#) 30
- Database Publishing Wizard Protocol
 - [about](#) 17
- Dataset DiffGram structure
 - [protocol summary](#) 18
- [DAX queries](#) 29
- Dependencies
 - [within the system](#) 22
- Design intent
 - [analysis services use cases](#) 29
 - [big data cluster protocols use cases](#) 31
 - [cep engine use case](#) 30
 - [manageability use case](#) 31
- [Discover operation](#) 15
- [DMX](#) 15

E

- [enterprise systems](#) 33
- [Environment](#) 22
- [Error handling](#) 32
- examples
 - [obtaining data](#) 35
 - [setting up enterprise systems](#) 33
- [Execute operation](#) 15
- Extensibility
 - [Microsoft implementations](#) 37
 - [overview](#) 32
- [External dependencies](#) 22

F

- [Functional architecture](#) 11
- [Functional requirements - overview](#) 11

G

- [geography data types](#) 21
- [geometry data type](#) 21
- [Glossary](#) 7

H

[Handling requirements](#) 32
[hierarchyid data type](#) 21

I

[Implementations - Microsoft](#) 37
[Implementer - security considerations](#) 32
[Informative references](#) 9
[Introduction](#) 6

L

LINQ Expression Tree Serialization Format Protocol
[about](#) 16

M

manageability
[CEP Engine](#) 16
[web service](#) 17
Manageability use case
[overview](#) 31
[manageability;Database Engine](#) 17
[Master Data Hub](#) 13
Master Data Services web service
[protocol summary](#) 19
Master Data Services web service 15
[protocol summary](#) 19
MDS
[about](#) 13
[management example](#) 34
[obtaining data](#) 36
protocols ([section 2.1.2](#) 13, [section 2.2.2](#) 19)
[use cases](#) 26
[MDX queries](#) 29
[Microsoft implementations](#) 37
mobile report management
[native report portal](#) 28
mobile reports
[native report portal](#) 28

N

Native Web Services
[protocol summary](#) 18
[network connectivity](#) 18
[example](#) 35
protocols ([section 2.1.1](#) 12, [section 2.2.1](#) 18)
[use cases](#) 23

O

[obtaining data](#) 35
[ODBC connection string structure](#) 18
[network connectivity](#) 12
[protocol summary](#) 18
OLEDB connection string structure ([section 2.1.1](#) 12,
[section 2.2.1](#) 18)
[network connectivity](#) 12
[protocol summary](#) 18
Overview
[summary of protocols](#) 18
[synopsis](#) 11

P

protocols
Analysis Services ([section 2.1.4](#) 15, [section 2.2.4](#) 21)
application development ([section 2.1.1](#) 12, [section 2.2.1](#) 18)
[CEP Engine](#) 22
Database Engine ([section 2.1.5](#) 16, [section 2.2.5](#) 21)
[errors](#) 32
MDS ([section 2.1.2](#) 13, [section 2.2.2](#) 19)
network connectivity ([section 2.1.1](#) 12, [section 2.2.1](#) 18)
Reporting Services ([section 2.1.3](#) 14, [section 2.2.3](#) 20)
[used by SQL Server](#) 6

Q

queries
[CEP Engine Manageability Protocol](#) 30
[MDS](#) 26
[querying MDS Store](#) 27
[sending to analysis server](#) 29
[SQL queries](#) 30

R

[References](#) 9
Remote GDI+ format
[Report Viewer control](#) 14
[Reporting Services protocols](#) 20
use cases ([section 2.4.3.1](#) 27, [section 2.4.3.3](#) 28)
Report Definition Language (RDL)
[Report Server web service payload](#) 14
[Reporting Services protocols](#) 20
use cases ([section 2.4.3.1](#) 27, [section 2.4.3.3](#) 28)
Report Definition Language file format
[protocol summary](#) 20
Report Page Layout (RPL) format
[Report Viewer control](#) 14
[Reporting Services protocols](#) 20
use cases ([section 2.4.3.1](#) 27, [section 2.4.3.3](#) 28)
Report Server web services
[Reporting Services architecture](#) 14
[Reporting Services protocols](#) 20
[Report Viewer control](#) 14
ReportExecution2005 API
[Reporting Services architecture](#) 14
[Reporting Services protocols](#) 20
use cases ([section 2.4.3.1](#) 27, [section 2.4.3.3](#) 28)
Reporting Services
[architecture](#) 14
[management example](#) 34
[obtaining data](#) 35
[use cases](#) 27
Reporting Services REST API
[protocol summary](#) 20
[use cases](#) 28
[ReportService2005 API](#) 20
ReportService2010
[use cases](#) 28
ReportService2010 API

- [Reporting Services architecture](#) 14
- [Reporting Services protocols](#) 20
- use cases ([section 2.4.3.1](#) 27, [section 2.4.3.3](#) 28)
- ReportServiceAuthentication API
 - [Reporting Services architecture](#) 14
 - [Reporting Services protocols](#) 20
- Requirements
 - [coherency](#) 32
 - [error handling](#) 32
 - [overview](#) 11
- RESTful API format
 - [use cases](#) 28

S

- [Security considerations](#) 32
- [Session Multiplex Protocol \(SMP\)](#) 12
 - [protocol summary](#) 18
- [SharePoint DataSet Diffgram structure](#) 12
- SharePoint Web Services Dataset DiffGram structure
 - [protocol summary](#) 18
- [SQL Native Client](#) 16
- [SQL queries](#) 30
- [SQL Server](#) 21
 - [CLR types](#) 16
 - [CRL types](#) 21
 - [error handling](#) 32
 - [protocols](#) 6
 - [technologies](#) 6
- [SQL Server Analysis Services 8.0 Protocol](#) 21
- SQL Server Analysis Services Protocol
 - [about](#) 15
 - [Analysis Services protocols](#) 21
- SQL Server Analysis Services Tabular Protocol
 - [about](#) 15
- SQL Server binary XML structure
 - [protocol summary](#) 18
- SQL Server Browser
 - [network connectivity](#) 12
 - use cases ([section 2.4.1.1](#) 23, [section 2.4.1.2](#) 24)
- SQL Server Resolution Protocol
 - [network connectivity](#) 12
 - [use case](#) 24
- SQL Server Resolution Protocol (SSRP)
 - [protocol summary](#) 18
- [Stewardship Portal](#) 13
- [System architecture](#) 11
- [System dependencies](#) 22
 - [within the system](#) 22
- [System errors](#) 32
- [System protocols](#) 18
- [System requirements - overview](#) 11
- System use cases
 - [analysis services use cases](#) 29
 - [big data cluster protocols use cases](#) 31
 - [cep engine use case](#) 30
 - [manageability use case](#) 31

T

- [Table of protocols](#) 18
- [TDS protocol](#) 18
 - [application development](#) 12
 - [client connection use case](#) 25
- [Database Engine](#) 16

- [network connectivity](#) 12
- [protocol summary](#) 18
- [Reporting Services](#) 14
- TDS Version 4.2
 - [protocol summary](#) 18
- [Tracking changes](#) 38

U

- use cases
 - [Analysis Services](#) 29
 - [analysis services use cases](#) 29
 - [big data cluster protocols use cases](#) 31
 - [CEP Engine Manageability Protocol](#) 30
 - [cep engine use case](#) 30
 - [Database Engine](#) 30
 - [manageability use case](#) 31
 - [Reporting Services](#) 27

V

- Versioning
 - [Microsoft implementations](#) 37
 - [overview](#) 32
- viewing reports
 - [external report portal](#) 28
 - native report portal ([section 2.4.3.1](#) 27, [section 2.4.3.2](#) 28)