[MS-SSSO]:
SQL Server System Overview

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **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 may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events 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 specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do 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 are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2011 Microsoft Corporation.

Release: Tuesday, November 8, 2011

Revision Summary

Date	Revision History	Revision Class	Comments
03/05/2010	0.1	Major	First release.
04/21/2010	0.1.1	Editorial	Revised and edited the technical content.
06/04/2010	0.1.2	Editorial	Revised and edited the technical content.
09/03/2010	0.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
02/09/2011	0.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
07/07/2011	0.1.2	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2011	0.1.2	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2011 Microsoft Corporation.

Contents

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

1 Introduction

This document is an overview of the client and server protocols used by Microsoft® SQL Server®. It 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 Master Data Services framework ensures that the internal functions of the product are better modularized to support both an API and a modular component development.

- 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 online analytical processing (OLAP) by enabling an implementer to design, create, and manage multidimensional structures that contain data that is aggregated from other data sources, such as relational databases. 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-SSDE].

- 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.

To deliver these functionalities, SQL Server uses seven major sets of protocols:

- Network connectivity and application development
- Master Data Services
- Reporting Services
- Analysis Services
- Database Engine
- Complex event processing engine
- Manageability

Copyright © 2011 Microsoft Corporation.

This document provides an overview of the protocols used by Microsoft® SQL Server® 2008 R2, Microsoft® SQL Server® 2008, and Microsoft® SQL Server® 2005. The specific release is specified when the information applies to a particular release.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory
connection
remote procedure call (RPC)
session
SOAP
stream
Transmission Control Protocol (TCP)
Web services

The following terms are defined in [MS-OFCGLOS]:

connection string
OLE DB
Online Analytical Processing (OLAP)

The following terms are specific to this document:

analysis server: A server that provides both **OLAP** and data mining services.

common language runtime user-defined type (CLR UDT): A data type that is created and defined by the user in SQL Server by using a .NET Framework CLR assembly.

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.

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.

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.

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. ODBC is an implementation of [TSQL-CLI-2003] and provides extensions to that standard.

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.

1.2 References

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

Copyright © 2011 Microsoft Corporation.

[MC-SQLR] Microsoft Corporation, "SQL Server Resolution Protocol Specification".

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

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

[MS-DSDG] Microsoft Corporation, "Dataset DiffGram Structure Specification".

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

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MS-ODBCSTR] Microsoft Corporation, "ODBC Connection String Structure Specification".

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

[MS-OFSGLOS] See [MS-OFCGLOS].

[MS-OLEDBSTR] Microsoft Corporation, "OLEDB Connection String Structure Specification".

[MS-RDL] Microsoft Corporation, "Report Definition Language File Format Structure Specification".

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

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

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

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

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

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

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

[MS-SSAS] Microsoft Corporation, "SQL Server Analysis Services Protocol Specification".

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

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

[MS-SSDPWP] Microsoft Corporation, "Database Publishing Wizard Protocol Specification".

[MS-SSMDSWS] Microsoft Corporation, "Master Data Services Web Service Specification".

[MS-SSNWS] Microsoft Corporation, "Native Web Services Protocol Specification".

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

Copyright © 2011 Microsoft Corporation.

[MS-TDS] Microsoft Corporation,"Tabular Data Stream Protocol Specification".

[MSDN-ASMD Microsoft Corporation, "SQL Server Analysis Services - Multidimensional Data", http://msdn.microsoft.com/en-us/library/bb522607.aspx

[MSDN-SSDE] Microsoft Corporation, "SQL Server Database Engine", http://msdn.microsoft.com/en-us/library/ms187875.aspx

[MSDN-SSIS] Microsoft Corporation, "SQL Server Integration Services", http://msdn.microsoft.com/en-us/library/ms141026.aspx

[MSDN-SSR] Microsoft Corporation, "SQL Server Replication", http://msdn.microsoft.com/en-us/library/ms151198.aspx

[MDSN-SSRS] Microsoft Corporation, "SQL Server Reporting Services", http://msdn.microsoft.com/en-us/library/ms159106.aspx

[MSDN-SSSB] Microsoft Corporation, "SQL Server Service Broker", http://msdn.microsoft.com/en-us/library/bb522893.aspx

[RFC2279] Yergeau, F., "UTF-8, A Transformation Format of ISO10646", RFC 2279, January 1998, http://www.ietf.org/rfc/rfc2279.txt

[TSQL-CLI-2003] International Organization for Standardization and International Electrotechnical Commission, "Information Technology – Database Languages – SQL – Part 3: Call-Level Interface", ISO/IEC 9075-3:2003, 2003, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34134

Copyright © 2011 Microsoft Corporation.

2 Functional Architecture

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

Microsoft® 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.

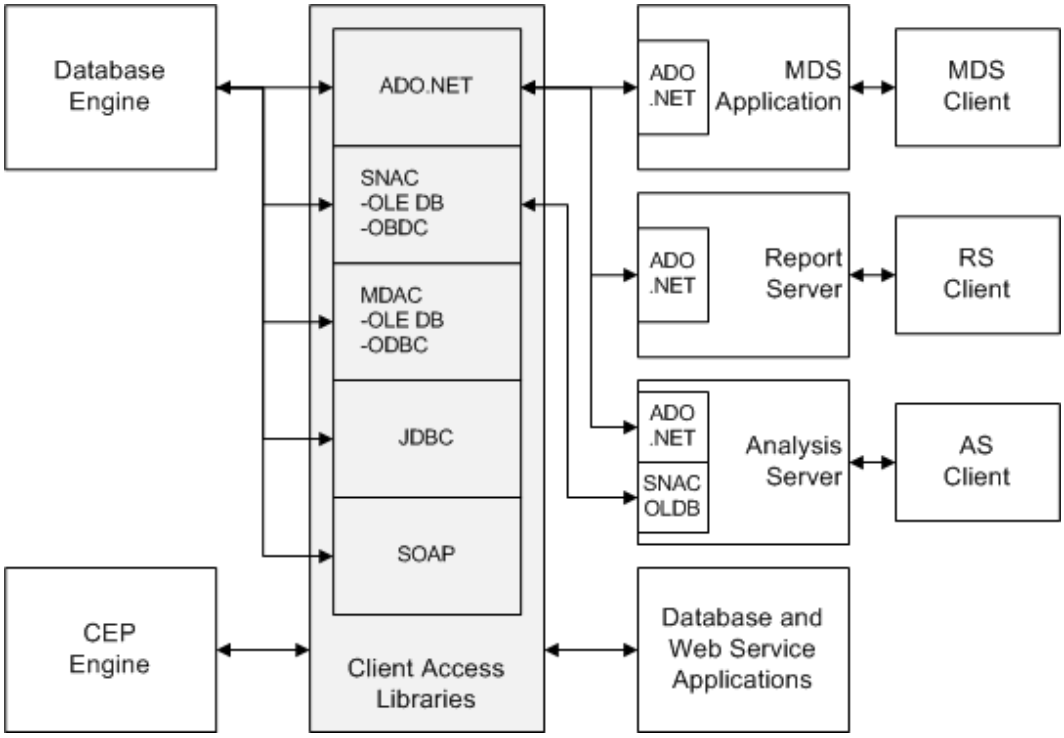


Figure 1: SQL Server architecture overview

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).

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 the 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

Copyright © 2011 Microsoft Corporation.

specified in [MS-TDS] and [MS-SSTDS] and is used to transfer Transact-SQL requests and responses between clients and database products.

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 **DataSetDiffGram** 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 **DataSetDiffGram** 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**.

The Binary XML structure [MS-BINXML] and the OLEDB connection string structure [MS-OLEDBSTR] are used by SQL Server Analysis Services [MS-SSAS] and the TDS protocol. The **Open Database Connectivity (ODBC)** connection string structure [MS-ODBCSTR] is used by the TDS protocol.

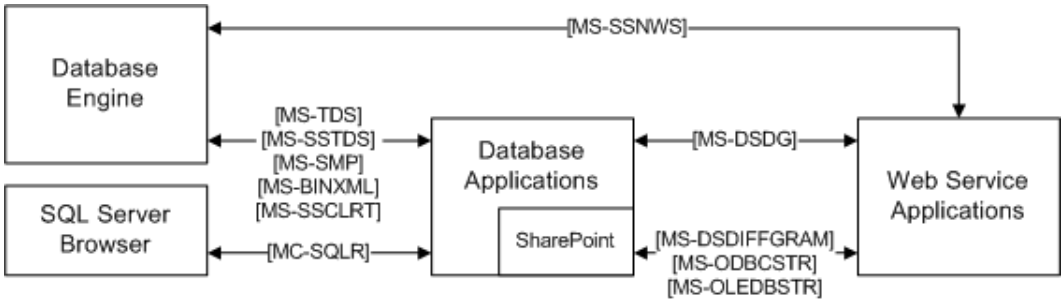


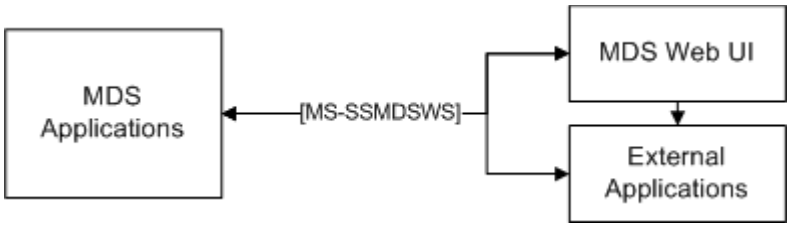
Figure 2: Network connectivity and application development architecture

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, transactions
- **Stewardship Portal** for model management, documentation, workflow, integration

MDS includes protocols that communicate between the MDS application and the MDS Web user interface [MS-SSMDSWS], and the MDS application and external applications (see the following figure). The TDS protocol is used to communicate with the Database Engine.



Copyright © 2011 Microsoft Corporation.

Figure 3: MDS architecture

2.1.3 Reporting Services

SQL Server Reporting Services (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 (Report Manager), or it can be integrated with an external portal. See the following figures for a view of each architecture.

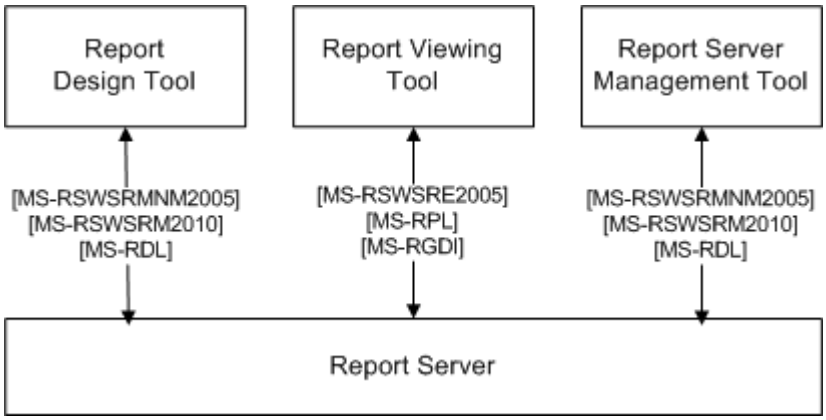


Figure 4: Reporting Services architecture with native management tools

In Native mode, the Report Designer, Report Manager, and any other report server management tools communicate with the report server by using the protocols and formats that are specified in [MS-RSWSRMNM2005], [MS-RSWSRM2010], and [MS-RDL]. The Report Viewer control uses the protocols specified in [MS-RPL] or [MS-RGDI] and [MS-RSWSRE2005] to communicate with the report server.

Copyright © 2011 Microsoft Corporation.

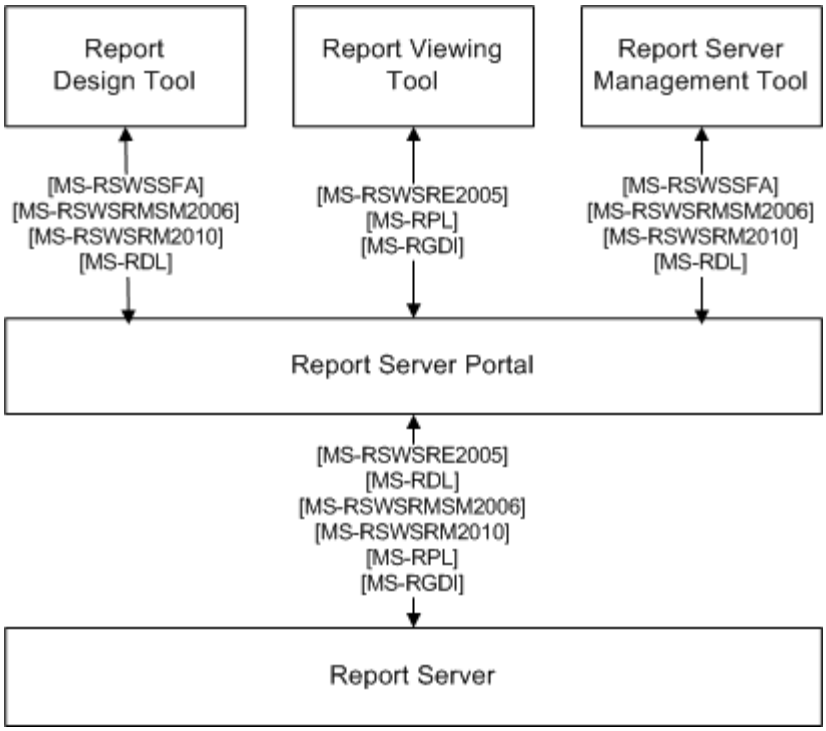


Figure 5: Reporting Services architecture when integrated with an external portal

When integrated with an external portal, the Report Designer and any report server management tools communicate with the report server portal via protocols and formats specified in [MS-RDL], [MS-RSWSRMSM2006], [MS-RSWSRM2010], and [MS-RSWSSFA]. The Report Viewer Web Part communicates with the report server by using the protocols and formats that are specified in [MS-RSWSRE2005] and [MS-RPL] or [MS-RGDI]. The portal simply redirects all requests to the report server.

The report server uses the TDS protocol 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

Analysis Services enables business intelligence (BI) applications to analyze data via online analytical processing (OLAP) and data mining techniques. OLAP enables the design, creation, and management of cubes that contain detail and aggregated data from multiple data sources such as relational databases. Cubes can be enriched with business logic by using Multidimensional Expressions (MDX) calculations, and they expose a unified dimensional model that BI applications can query using the MDX language. Data mining enables 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.

Copyright © 2011 Microsoft Corporation.

SQL Server Analysis Services Protocol [MS-SSAS] provides methods for a client to communicate with, and perform operations on, an **analysis server** (see the following figure). This protocol is based on **SOAP** and XML for Analysis (XMLA). This protocol supports TCP/IP as an underlying transport mechanism in addition to HTTP/HTTPS.

The SQL Server Analysis Services Protocol defines 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.

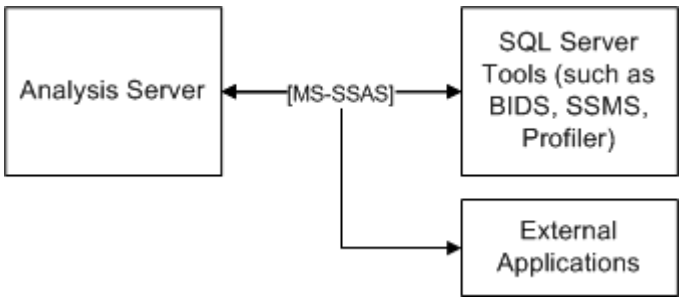


Figure 6: Analysis Services architecture

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 SQL Native Client (SNAC), **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.

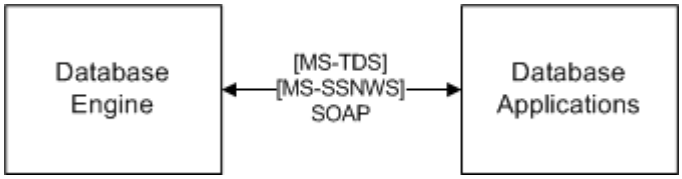


Figure 7: Database Engine architecture

Copyright © 2011 Microsoft Corporation.

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 may also include database applications that are built on SQL Server.

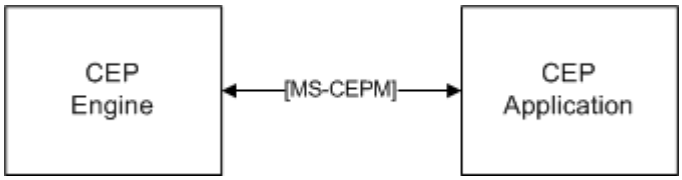


Figure 8: CEP Engine architecture

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.

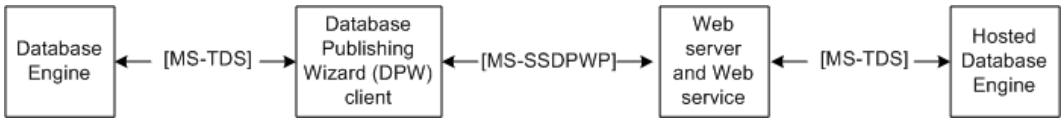


Figure 9: Manageability architecture

2.2 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) Protocol 4.2	This protocol is an application-level protocol that is used to transfer Transact-SQL requests and responses between clients and database products. The TDS protocol provides for: ▪ Authentication and channel encryption negotiation.	[MS-SSTDS]

Copyright © 2011 Microsoft Corporation.

Protocol name	Description	Reference
	▪ Specification of requests in SQL (including bulk insert). ▪ **Remote procedure calls (RPCs)**. ▪ Return of data. ▪ Transaction manager requests.	
Session Multiplex Protocol (SMUX)	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 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 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 must be established.	[MS-TDS]

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 **DataSet DiffGram** 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	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**	[MS-OLEDBSTR]

Copyright © 2011 Microsoft Corporation.

Format	Description	Reference
String Structure	**provider** and that specifies the information that is needed to establish a connection to a data source.	

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]

2.2.3 Reporting Services

The following protocols and formats are used by the Reporting Services.

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 must 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 Specification for Report	This protocol is used for managing objects and settings on all report servers, starting with	[MS-RSWSRM2010]

Copyright © 2011 Microsoft Corporation.

Format	Description	Reference
Management: ReportService2010	Microsoft® SQL Server® 2008 R2.	
Report Server Web Service Specification 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-RSWSRMSM2006]
Report Server Web Service Specification 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-RSWSRE2005]
Report Server Web Service Specification 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-RSWSRMNM2005]
Reporting Services SharePoint Forms Authentication SOAP Specification: 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]

2.2.4 Analysis Services

The following protocols are used by Analysis Services.

Protocol	Description	Reference
SQL Server Analysis Services Protocol	This protocol provides a mechanism for a client to communicate with, and perform operations on, an analysis server. SQL Server Analysis Services Protocol is the protocol for Microsoft® SQL Server® 2005, Microsoft® SQL Server® 2008, and Microsoft® SQL Server® 2008 R2.	[MS-SSAS]
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 version of SQL Server Analysis Services Protocol is the protocol 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 protocol is used with the Database Engine.

Format	Description	Reference
SQL Server Common Language Runtime (CLR) Types Structure	This binary format consists of the **GEOGRAPHY**, **GEOMETRY**, **HIERARCHYID**, and **CLR user-defined type (CLR UDT)** structures managed by SQL Server. The structures used to transmit **geography** and **geometry** data types over the wire implement the OpenGIS Consortium's (OGC) Simple Feature 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. The **hierarchyid** data type enables SQL Server applications to store,	[MS-SSCLRT]

Copyright © 2011 Microsoft Corporation.

Format	Description	Reference
	retrieve, and manipulate hierarchical data.	

2.2.6 CEP Engine

The following protocol is used with the complex event processing (CEP) engine.

Protocol	Description	Reference
Microsoft 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]

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]

2.3 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.

Copyright © 2011 Microsoft Corporation.

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.

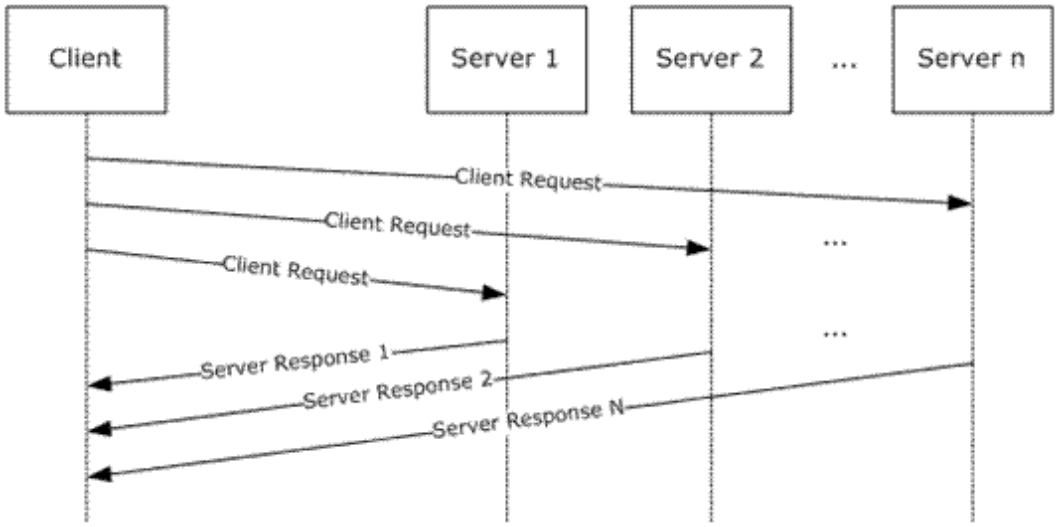


Figure 10: SQL Server instance discovery use case

Copyright © 2011 Microsoft Corporation.

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.

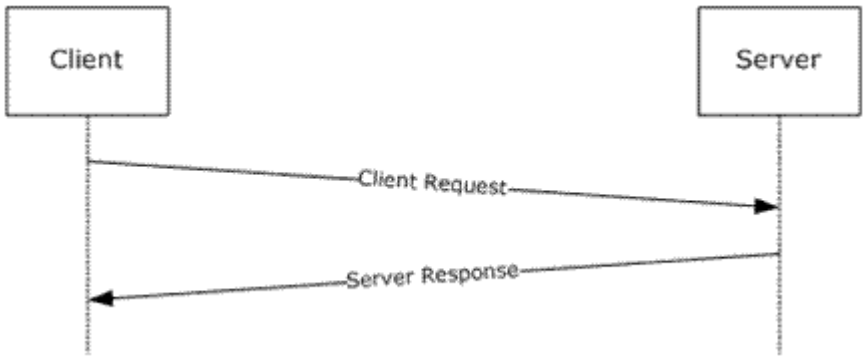


Figure 11: Named SQL Server instance resolution use case

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 may be repeated.)

5. The database application terminates the connection.

6. The database application may have caused side effects by executing commands and/or may have exchanged data with SQL Server.

For 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.

Copyright © 2011 Microsoft Corporation.

For SOAP only: This use case has been deprecated in Microsoft® SQL Server® 2008.

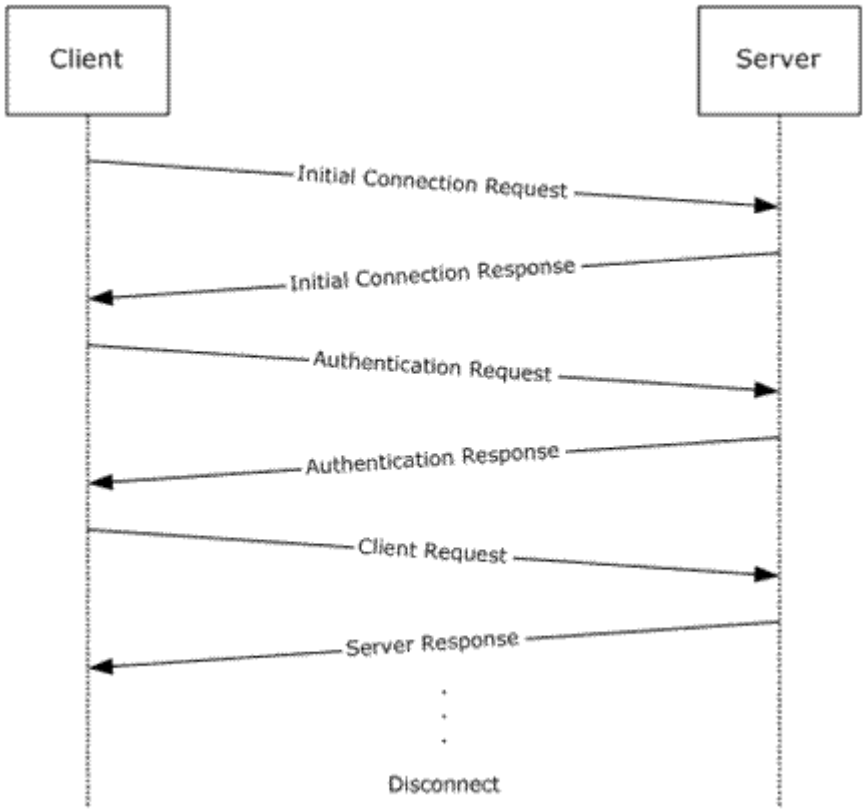


Figure 12: Client connection use case

Payloads

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

- Binary XML type values specified in [MS-BINXML] are transported by the Tabular Data Stream (TDS) protocol 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 protocol.

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] is required.

Copyright © 2011 Microsoft Corporation.

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-RSWSRM2010].
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-RSWSRE2005].
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 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 may be requested and returned instead.

Copyright © 2011 Microsoft Corporation.

2.4.3.2 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 may 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.

Copyright © 2011 Microsoft Corporation.

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 MDSCHEMA_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 a Query

This use case describes how a client sends an MDX query to a cube 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.

2.4.5 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.

Copyright © 2011 Microsoft Corporation.

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 TDS protocol.

2.4.6 CEP Engine Use Case

The complex event processing (CEP) Engine protocol as specified in [MS-CEPM] is a Web service protocol that defines the communication protocol between a client application and a 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 CEP Engine protocol is stateless. All communication is initiated by the client. The server sends responses only in response to messages received.

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.

2.5 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.

Copyright © 2011 Microsoft Corporation.

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 may be handled differently by the different protocols. For detailed information about protocol security, see the protocols that are listed in section 2.2.

Copyright © 2011 Microsoft Corporation.

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.

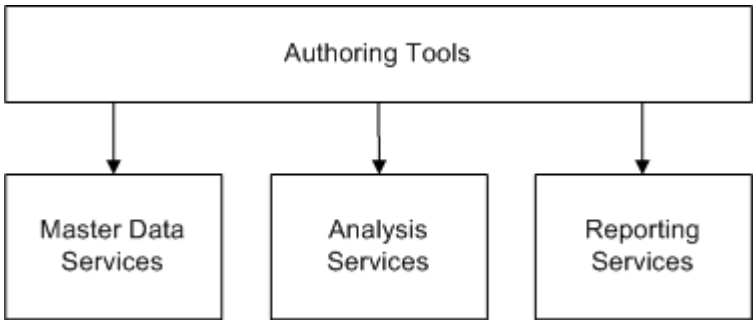


Figure 13: Defining content for the servers

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

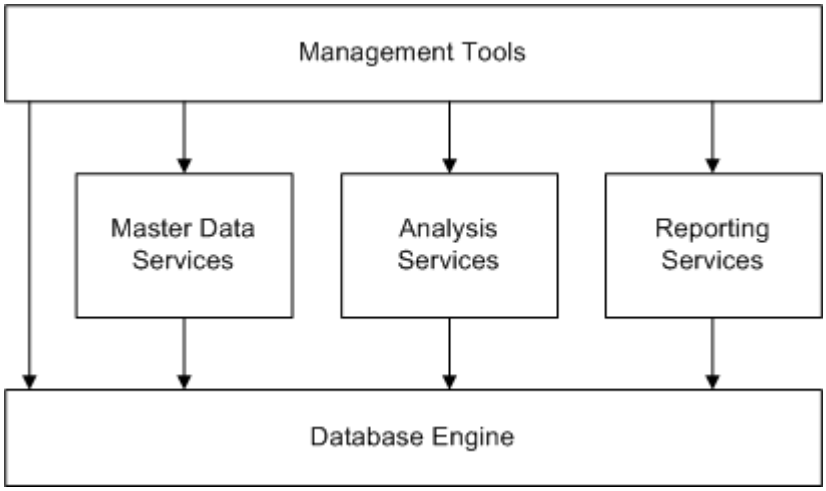


Figure 14: Using management tools to set up the system

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 UDM (such as a cube that provides a description of which information to retrieve and which transformations to be performed) via a UDM authoring tool.

Copyright © 2011 Microsoft Corporation.

2. The IT administrator publishes the UDM 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 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 Master Data Services 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 Master Data 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.

Copyright © 2011 Microsoft Corporation.

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 and SQL Server Management Studio, an administrator attempts to get a list of database instances on the network.
2. The management tools broadcasts 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 tool and 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 3.2.2, but a pivot table is published by Microsoft Excel at the Analysis Services server (a pivot table is manipulated and data is retrieved from the Analysis Services server).

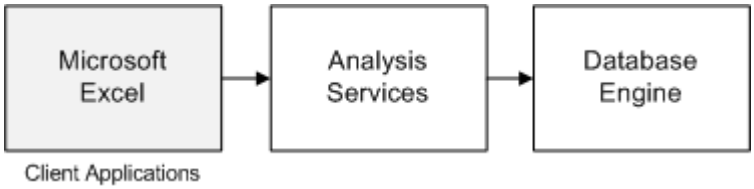


Figure 15: User obtaining data via Analysis Services

3.2.2 Obtaining Data via Reporting Services

1. Using Windows® Internet Explorer®, the user navigates to the SharePoint portal and clicks a report to execute.
2. The SharePoint portal, acting as a Reporting Services client, requests that the report be executed by Reporting Services.
3. Reporting Services requests the underlying data for the reports from the Database Engine or the Analysis Services engine.
4. The engine returns the data that was processed by Reporting Services.
5. Reporting Services returns a representation of the report to SharePoint.

Copyright © 2011 Microsoft Corporation.

6. SharePoint returns the resulting Web page to Internet Explorer.

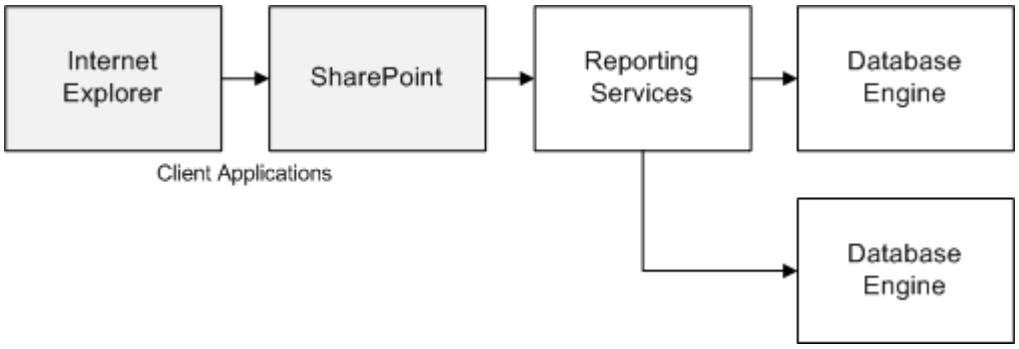


Figure 16: User obtaining data via Reporting Services

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 via XML.
5. The MDS portal renders the data in Internet Explorer.

Figure 17: User obtaining data via MDS

Copyright © 2011 Microsoft Corporation.

4 Microsoft Implementations

There are no variations in the behavior of SQL Server in different versions beyond those described in the specifications of the protocols supported by the system, as listed in the Protocol Summary in section 2.2.

The information in this specification is applicable to the following product versions:

- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

4.1 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

Copyright © 2011 Microsoft Corporation.

5 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

Copyright © 2011 Microsoft Corporation.

6 Index

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.

Copyright © 2011 Microsoft Corporation.