[MS-CSOMREST]:

SharePoint Client Query OData Protocol

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. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **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.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional

development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
1/20/2012	0.1	New	Released new document.
4/11/2012	0.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	0.1	No Change	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	0.1	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	1.0	Major	Significantly changed the technical content.
2/11/2013	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	1.0	No Change	No changes to the meaning, language, or formatting of the technical content.
8/24/2015	2.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The SharePoint Client Query OData Protocol allows a protocol client to use common web technologies to send Open Data protocol (OData) request to access data on a protocol server when the protocol server implements SharePoint Client Query Protocol. The OData requests to be executed are sent by a protocol client, and the OData responses are returned by a protocol server.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

CSOM array: An ordered collection of values that can be used in an **XML** request or **JSON** response text. The values are identified by their position and their position is determined by a zero-based integer index.

CSOM binary: An array of 8-bit, unsigned integers that can be used in an **XML** request or as a string in **JSON** response text.

CSOM Boolean: A Boolean value that can be used in an **XML** request or **JSON** response text. A CSOM Boolean value is either "true" or "false".

CSOM Byte: An 8-bit, unsigned integer value that represents the BYTE type, as described in [MS-DTYP]. The range of CSOM Byte values is 0-255 and it has different representations, depending on whether it is used in an **XML** request or **JSON** response text.

CSOM Char: A Unicode character value that can be used in an **XML** request or as a string in **JSON** response text.

CSOM DateTime: An Int64 value that represents the number of 100-nanosecond time intervals that have elapsed since 12:00:00, January 1, 0001. It can be used in an **XML** request or as a string in **JSON** response text. The value can represent time intervals through 23:59:59.9999999, December 31, 9999. It can also specify whether a local, UTC, or no time zone applies.

CSOM dictionary: An object that contains an unordered collection of key/value pairs that can be used in an **XML** request or **JSON** response text. Each key in a CSOM dictionary has a unique name.

CSOM Double: A 64-bit, double-precision, floating-point value, which is the DOUBLE type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM Double values is from "-1.79769313486232e308" to "1.79769313486232e308".

CSOM GUID: A GUID, as described in [MS-DTYP], that can be used in an **XML** request or as a string in **JSON** response text.

CSOM Int16: A 16-bit, signed integer value, which is the INT16 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM Int16 values is from "-32768" to "32767".

CSOM Int32: A 32-bit, signed integer value, which is the INT32 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM Int32 values is from "-2147483648" to "2147483647".

CSOM Int64: A 64-bit, signed integer value, which is the INT64 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM Int64 values is from "-9223372036854775808" to "9223372036854775807".

CSOM method: A procedure that is executed by a protocol server for a **CSOM Object**.

CSOM Object: An object that contains a set of members, which are named values and methods. It has a Unicode string value, which is referred to as a CSOM type name, that identifies its type.

CSOM Object type: A reference to a standard definition of methods, properties, and behavior for a logical object in the SharePoint Client-Side Object Model.

CSOM property: A representation of a field of data that is stored for a type of **CSOM Object**.

CSOM SByte: An 8-bit, signed integer value, which is the INT8 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM SByte values is from "-128" to "127".

CSOM Single: A 32-bit, single-precision, floating-point value, which is the FLOAT type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM Single values is from "-3.402823e38" to "3.402823e38".

CSOM String: A representation of text as a series of Unicode characters. It can be used in an **XML** request or **JSON** response text.

CSOM type name: A Unicode string that identifies the type of a **CSOM Object**.

CSOM UInt16: A 16-bit, unsigned integer value, which is the UINT16 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM UInt16 values is from "0" to "65535".

CSOM UInt32: A 32-bit, unsigned integer value, which is the UINT32 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM UInt32 values is from "0" to "4294967295".

CSOM UInt64: A 64-bit, unsigned integer value, which is the UINT64 type described in [MS-DTYP], that can be used in an **XML** request or as a number in **JSON** response text. The range of CSOM UInt64 values is from "0" to "18446744073709551615".

CSOM value object: An object that contains a set of named values, which are referred to as members. It has a Unicode string value, referred to as a CSOM type name, that identifies its type.

CSOM value object type: A CSOM type that contains a set of named values, which are referred to as members. It has type information, which is identified by a Unicode string, and is associated with a specific identifier, which is a **CSOM GUID**.

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

Hypertext Transfer Protocol Secure (HTTPS): An extension of HTTP that securely encrypts and decrypts web page requests. In some older protocols, "Hypertext Transfer Protocol over Secure Sockets Layer" is still used (Secure Sockets Layer has been deprecated). For more information, see [SSL3] and [RFC5246].

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 [RFC4627]. The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

Open Data Protocol (OData): A web protocol for querying and updating data specified in the OData protocol.

Request-URI: A URI in an **HTTP** request message, as described in [RFC2616].

website: A group of related pages and data within a SharePoint site collection. The structure and content of a site is based on a site definition. Also referred to as SharePoint site and site.

XML: The Extensible Markup Language, as described in [XML1.0].

XML schema: A description of a type of XML document that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by **XML** itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

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

1.2 References

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

1.2.1 Normative References

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

[MC-CSDL] Microsoft Corporation, "Conceptual Schema Definition File Format".

[MS-CSOM] Microsoft Corporation, "SharePoint Client Query Protocol".

[MS-ODATA] Microsoft Corporation, "Open Data Protocol (OData)".

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006, http://www.ietf.org/rfc/rfc4627.txt

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[MS-CSOMSPT] Microsoft Corporation, "SharePoint Client-Side Object Model Protocol".

1.3 Overview

This protocol enables a protocol client to send an **OData** request to a protocol server using common web technologies when the protocol server implements SharePoint Client Query Protocol, [MS-CSOM].

This protocol defines a system for locating instances of types, calling methods that are defined by those types, and performing read/write operations for properties of those types. This protocol defines two roles: protocol client and protocol server. A protocol client initiates communication by generating an OData request. The protocol client then sends that OData request to the protocol server for processing. The protocol server locates the instances of types, performs action and then returns the OData response.

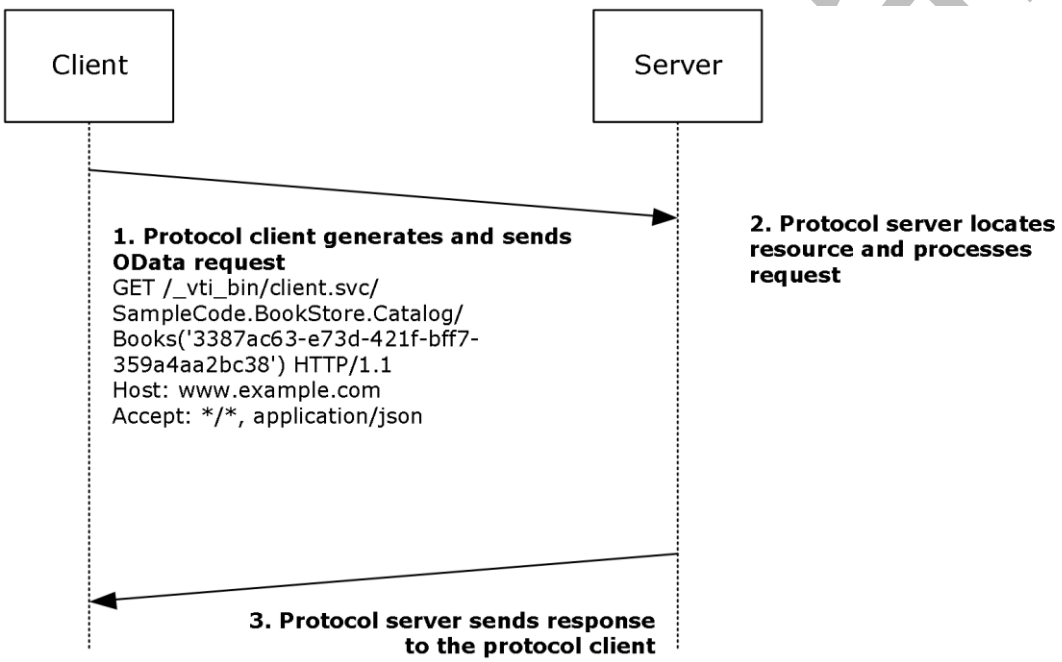

Figure 1: Overview of a request/response sequence

1.4 Relationship to Other Protocols

This protocol enables a protocol client to send a request that calls methods and accesses data on a protocol server, and then receive a corresponding response from the protocol server. This protocol depends on other structures and protocols to transport messages. Additional protocols, such as the SharePoint Client-Side Object Model Protocol described in [MS-CSOMSPT], define the properties and methods that are exposed to protocol clients through this protocol. Applications are layered on top of this protocol, in combination with related protocols that define sets of logical types and related methods and properties.

The messages that are sent from the protocol client to the protocol server are formatted as OData format as described in [MS-ODATA], either as **XML** or **JSON**. It transmits those messages by using **HTTP**, as described in [RFC2616], or **HTTPS**, as described in [RFC2818]. Responses from the protocol server are formatted as OData format as described in [MS-ODATA], either as XML or JSON.

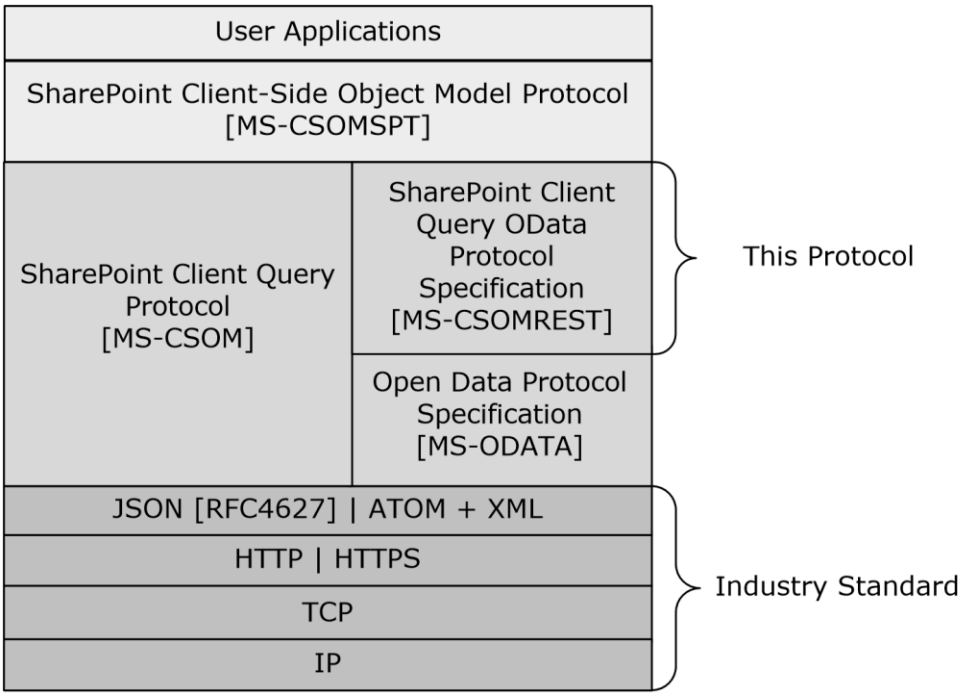

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol operates against a protocol server that is configured to listen for HTTP or HTTPS requests and a protocol client that knows the **Request-URI** of the protocol server.

This protocol assumes that the protocol server implements SharePoint Client Query Protocol as described in [MS-CSOM].

1.6 Applicability Statement

This protocol can be used as a protocol for a protocol client to perform create, retrieve, update and delete operations using common web technologies against a protocol server that implements SharePoint Client Query protocol as described in [MS-CSOM]. It is not suitable for a protocol client to perform batch operations against the protocol server.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol can use HTTP or HTTPS as a transport. For more information, see Transport (section 2.1).
- **Protocol Versions:** The protocol client specifies the OData protocol version by using **DataServiceVersion** HTTP header as described in [MS-ODATA] section 2.2.5.3 and **MaxDataServiceVersion** HTTP header as describe in [MS-ODATA] section 2.2.5.7.

- **Capability Negotiation:** Capability negotiation is performed as described in [MS-ODATA] section 1.7.

1.8 Vendor-Extensible Fields

This protocol supports custom HTTP headers, as specified in [RFC2616] section 4.2.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Protocol servers MUST support HTTP, as specified in [RFC2616]. Protocol servers SHOULD additionally support HTTPS, as specified in [RFC2818], to help secure communications with protocol clients. Messages that a protocol client sends to a protocol server MUST be formatted as **ProcessRestQueryIn** messages, as specified in section 3.1.5.1.1.1. Protocol clients MUST use the DELETE, GET, MERGE, PATCH, POST, or PUT method to send messages to protocol servers. Messages that a protocol server sends to a protocol client MUST be formatted as **ProcessRestQueryOut** messages, as specified in section 3.1.5.1.1.2.

2.2 Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses the XML Schema as specified in [XMLSCHEMA1] and [XMLSCHEMA2], Web Services Description Language as specified in [WSDL], and JavaScript Object Notation (JSON) as specified in [RFC4627].

2.2.1 Namespaces

This specification defines and references various XML namespaces using the mechanisms specified in [XMLNS]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespace URI	Reference
mstns	http://schemas.microsoft.com/sharepoint/clientquery/2009	
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	
wsaw	http://www.w3.org/2006/05/addressing/wsdl	
tns	http://schemas.microsoft.com/sharepoint/soap/	
xsd	http://www.w3.org/2001/XMLSchema	
http	http://schemas.microsoft.com/ws/06/2004/policy/http	
wsdl	http://schemas.xmlsoap.org/wsdl/	
(none)	http://schemas.microsoft.com/sharepoint/clientquery/2009	

In addition, this protocol supports the namespaces as described in [MS-ODATA] section 2.2.6.

2.2.2 Custom HTTP Methods

This protocol supports all HTTP Methods as defined in [MS-ODATA] section 2.2.4.

2.2.3 Custom HTTP Headers

This protocol supports all HTTP Headers as defined in [MS-ODATA] section 2.2.5.

2.2.4 Elements

This specification does not define any common **XML schema** element definitions.

2.2.5 Complex Types

The following table summarizes the set of complex type definitions defined by this specification. XML schema complex type definitions that are specific to a particular operation are described with the operation.

Complex type	Description
CSOM array	A CSOM array complex type as specified in [MS-CSOM] section 2.2.4.1.
CSOM dictionary	A CSOM dictionary complex type as specified in [MS-CSOM] section 2.2.4.2.
CSOM null	A CSOM null complex type as specified in [MS-CSOM] section 2.2.4.3.
CSOM Object	A CSOM Object complex type as specified in [MS-CSOM] section 2.2.4.4.
CSOM value object	A CSOM value object complex type as specified in [MS-CSOM] section 2.2.4.5.
CSOM Stream	A CSOM Stream complex type as specified in [MS-CSOM] section 2.2.4.6.
CSOM Error	A CSOM error as specified in [MS-CSOM] section 3.1.4.1.8.1.2.

The following table summarizes the mapping between the complex types as specified in [MS-CSOM] and the types as specified in [MS-ODATA].

CSOM complex type	OData type
CSOM array	**Collection of Complex Type Instances** as specified in [MS-ODATA] section 2.2.6.3.5 and [MS-ODATA] section 2.2.6.5.2. **Collection of EDMSimpleType Values** as specified in [MS-ODATA] section 2.2.6.3.7 and [MS-ODATA] section 2.2.6.5.4. **Entity Set** as specified in [MS-ODATA] section 2.2.6.2.1 and [MS-ODATA] section 2.2.6.3.2.
CSOM dictionary	**Collection of Complex Type Instances** as specified in [MS-ODATA] section 2.2.6.3.5 and [MS-ODATA] section 2.2.6.5.2.
CSOM null	**null** as specified in [MS-ODATA] section 2.2.2.

CSOM complex type	OData type
CSOM Object	**Entity Type** as specified in [MS-ODATA] section 2.2.6.2.2 and [MS-ODATA] section 2.2.6.3.3. **Entity Set** as specified in [MS-ODATA] section 2.2.6.2.1 and [MS-ODATA] section 2.2.6.3.2.
CSOM value object	**Complex Type** as specified in [MS-ODATA] section 2.2.6.2.3 and [MS-ODATA] section 2.2.6.3.4.
CSOM Stream	**Edm.Stream** as specified in [MS-ODATA] section 2.2.2.
CSOM error	**Error Response** as specified in [MS-ODATA] section 2.2.8.1.

2.2.5.1 CSOM Array

The CSOM array type, which conforms to the **CSOM Array** complex type as specified in [MS-CSOM] section 2.2.4.1, is used in OData formats as Collection of Complex Type Instances, Collection of EDMSimple Type Values or Entity Set as specified in [MS-ODATA].

If the elements in the CSOM array are **EDM complex types**, the CSOM array formats as Collection of Complex Type Instances as specified in [MS-ODATA] section 2.2.6.3.5 and [MS-ODATA] section 2.2.6.5.2.

If the elements in the CSOM array are **EDM simple types**, the CSOM array formats as Collection of EDMSimpleType Values as specified in [MS-ODATA] section 2.2.6.3.7 and [MS-ODATA] section 2.2.6.5.4.

If the elements in the CSOM array are **EDM entity types**, the CSOM array formats as Entity Set as specified in [MS-ODATA] section 2.2.6.2.1 and [MS-ODATA] section 2.2.6.3.2.

2.2.5.2 CSOM Dictionary

The CSOM dictionary complex type, which conforms to the **CSOM dictionary** complex type as specified in [MS-CSOM] section 2.2.4.2, is a collection of **EDM complex types**. When it is used in OData, it formats as Collection of Complex Type Instances as specified in [MS-ODATA] section 2.2.6.3.5 and [MS-ODATA] section 2.2.6.5.2. The elements in the collection of **EDM complex types** are of type **SP.KeyValue** as specified in section 3.1.5.1.3.1, where the **SP.KeyValue** represents the key value pair in the CSOM dictionary. The value of the property key in **SP.KeyValue** type is the value of the key in the key value pair. The value of the property value in the **SP.KeyValue** type is the string representation of the value in the key value pair. The value of property **ValueType** in the **SP.KeyValue** type is the type name of the value in the key value pair.

2.2.5.3 CSOM Null

The **CSOM Null** type, which conforms to the **CSOM Null** type as specified in [MS-CSOM] section 2.2.4.3, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This type MUST also conform to the **null** type, as specified in [MS-ODATA] section 2.2.2

2.2.5.4 CSOM Object

The CSOM Object type, which conforms to the **CSOM Object** complex type as specified in [MS-CSOM] section 2.2.4.4, is used in Open Data Protocol (OData) formats as Entity Type or Entity Set as specified in [MS-ODATA].

If the CSOM Object is a collection and its child elements are **EDM entity types**, the CSOM Object is an **EDM entity set** and it formats as Entity Set as specified in [MS-ODATA] section 2.2.6.2.1 and [MS-ODATA] section 2.2.6.3.2.

Otherwise, the CSOM Object is an **EDM entity type** and it formats as Entity Type as specified in [MS-ODATA] section 2.2.6.2.2 and [MS-ODATA] section 2.2.6.3.3. The name of the **EDM entity type** for the CSOM Object is the **CSOM type name** of the CSOM Object. Each property in the CSOM Object is a property in the **EDM entity type**. In addition, if the CSOM Object is a collection and its child elements are **EDM complex types**, or its child elements are **EDM simple types**, the **EDM entity type** of this CSOM Object has a property named "Items". When the child elements are **EDM complex types**, the type of the property "Items" is a collection of **EDM complex types**. When the child elements of this CSOM Object are **EDM simple types**, the type of property "Items" is a collection of **EDM simple types**.

2.2.5.5 CSOM Value Object

The CSOM value object type, which conforms to the **CSOM value object** complex type as specified in [MS-CSOM] section 2.2.4.5, is an **EDM complex type**. When it is used in OData, it formats as a complex type as specified in [MS-ODATA] section 2.2.6.2.3 and [MS-ODATA] section 2.2.6.3.4. The name of the **EDM complex type** for the CSOM value object is the CSOM type name of the CSOM value object. Each property in the CSOM value object is a property in the **EDM complex type**.

2.2.5.6 CSOM Stream

The **CSOM Stream** type, which conforms to the **CSOM Stream** complex type as specified in [MS-CSOM] section 2.2.4.6, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This **CSOM Stream** type MUST also conform to the **Edm.Stream** type, as specified in [MS-ODATA] section 2.2.2.

2.2.5.7 CSOM Error

The CSOM error, which conforms to CSOM Error JSON value as specified in [MS-CSOM] section 3.1.4.1.8.1.2, is sent to the protocol client by the protocol server when there is an error while the protocol server processes the OData request. It formats as an OData error response as specified in [MS-ODATA] section 2.2.8.1. The following is the map between the properties in the OData error response and the properties of CSOM Error JSON value.

Error Code: The error code property in the OData error response is the value of CSOM error code concatenated with CSOM error type name with separator comma (,). For example, if the CSOM error's error code is -2147024846 and the CSOM error type name is Microsoft.SharePoint.Client. ApiBlockedException, the error code in the OData error response is:

```
-2147024846,Microsoft.SharePoint.Client. ApiBlockedException
```

Error Message: The error message property in the OData error response is the value of CSOM error message.

Inner Error: The inner error property in the OData error response is the value of CSOM error call stack.

2.2.6 Simple Types

The following table summarizes the set of common XML schema simple type definitions defined by this specification. XML schema simple type definitions that are specific to a particular operation are described with the operation.

Simple type	Description
CSOM binary	A **CSOM binary** simple type as specified in [MS-CSOM] section 2.2.5.1.
CSOM Boolean	A **CSOM Boolean** simple type as specified in [MS-CSOM] section 2.2.5.2.
CSOM Byte	A **CSOM Byte** simple type as specified in [MS-CSOM] section 2.2.5.3.
CSOM Char	A **CSOM Char** simple type as specified in [MS-CSOM] section 2.2.5.4.
CSOM DateTime	A **CSOM DateTime** simple type as specified in [MS-CSOM] section 2.2.5.5.
CSOM Decimal	A **CSOM Decimal** simple type as specified in [MS-CSOM] section 2.2.5.18.
CSOM Double	A **CSOM Double** simple type as specified in [MS-CSOM] section 2.2.5.6.
CSOM Enum	A **CSOM Enum** simple type as specified in [MS-CSOM] section 2.2.5.7.
CSOM GUID	A **CSOM GUID** simple type as specified in [MS-CSOM] section 2.2.5.8.
CSOM Int16	A **CSOM Int16** simple type as specified in [MS-CSOM] section 2.2.5.9.
CSOM Int32	A **CSOM Int32** simple type as specified in [MS-CSOM] section 2.2.5.10.
CSOM Int64	A **CSOM Int64** simple type as specified in [MS-CSOM] section 2.2.5.11.
CSOM SByte	A **CSOM SByte** simple type as specified in [MS-CSOM] section 2.2.5.12.
CSOM Single	A **CSOM Single** simple type as specified in [MS-CSOM] section 2.2.5.13.
CSOM String	A **CSOM String** simple type as specified in [MS-CSOM] section 2.2.5.14.
CSOM TimeSpan	A **CSOM TimeSpan** simple type as specified in [MS-CSOM] section 2.2.5.19.

Simple type	Description
CSOM UInt16	A **CSOM UInt16** simple type as specified in [MS-CSOM] section 2.2.5.15.
CSOM UInt32	A **CSOM UInt32** simple type as specified in [MS-CSOM] section 2.2.5.16.
CSOM UInt64	A **CSOM UInt64** simple type as specified in [MS-CSOM] section 2.2.5.17.

The following table summarizes the mapping between the simple types as specified in [MS-CSOM] and the **Edm** types as specified in [MS-ODATA] section 2.2.2.

CSOM simple type	Edm type
CSOM binary	**Edm.Binary**
CSOM Boolean	**Edm.Boolean**
CSOM Byte	**Edm.Byte**
CSOM Char	**Edm.String**
CSOM DateTime	**Edm.DateTime**
CSOM Decimal	**Edm.Decimal**
CSOM Double	**Edm.Double**
CSOM Enum	**Edm.Int16, Edm.Int32, or Edm.Int64**
CSOM GUID	**Edm.Guid**
CSOM Int16	**Edm.Int16**
CSOM Int32	**Edm.Int32**
CSOM Int64	**Edm.Int64**
CSOM SByte	**Edm.SByte**

CSOM simple type	Edm type
CSOM Single	**Edm.Single**
CSOM String	**Edm.String**
CSOM TimeSpan	**Edm.Time**
CSOM UInt16	**Edm.Int32**
CSOM UInt32	**Edm.Int64**
CSOM UInt64	**Edm.Int64**

2.2.6.1 CSOM binary

The CSOM binary simple type, which conforms to the **CSOM binary** simple type as specified in [MS-CSOM] section 2.2.5.1, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Binary** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.2 CSOM Boolean

The CSOM Boolean simple type, which conforms to the **CSOM Boolean** simple type as specified in [MS-CSOM] section 2.2.5.2, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Boolean** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.3 CSOM Byte

The CSOM Byte simple type, which conforms to the **CSOM Byte** simple type as specified in [MS-CSOM] section 2.2.5.3, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Byte** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.4 CSOM Char

The CSOM Char simple type, which conforms to the **CSOM Char** simple type as specified in [MS-CSOM] section 2.2.5.4, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.String** type, as specified in [MS-ODATA] section 2.2.2. The length of the **Edm.String** value MUST be 1.

2.2.6.5 CSOM DateTime

A **CSOM DateTime** value specified by [MS-CSOM] section 2.2.5.5 used in OData formats specified by [MS-ODATA] section 2.2.6 MUST conform to the **Edm.DateTime** type as specified in [MS-ODATA] section 2.2.2.

2.2.6.6 CSOM Decimal

A **CSOM Decimal** value specified by [MS-CSOM] section 2.2.5.18 used in OData formats specified by [MS-ODATA] section 2.2.6 MUST conform to the **Edm.Decimal** type as specified in [MS-ODATA] section 2.2.2.

2.2.6.7 CSOM Double

The CSOM Double simple type, which conforms to the **CSOM Double** simple type as specified in [MS-CSOM] section 2.2.5.6, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Double** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.8 CSOM Enum

The **CSOM Enum** simple type, which conforms to the **CSOM Enum** simple type as specified in [MS-CSOM] section 2.2.5.7, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int16**; **Edm.Int32**; or **Edm.Int64** type, as specified in [MS-ODATA] section 2.2.2. When the integer value of the **CSOM Enum** can range from –32768 through 32767, the **CSOM Enum** MUST be represented as an **Edm.Int16**. When the integer value can range from –2147483648 through 2147483647, the **CSOM Enum** MUST be represented as an **Edm.Int32**. When the integer value can range from –9223372036854775808 through 9223372036854775807, the **CSOM Enum** MUST be represented as an **Edm.Int64**. The integer value of the **CSOM Enum** will correspond to the value of the **Edm.Int16**, **Edm.Int32**, or **Edm.Int64**, respectively.

2.2.6.9 CSOM GUID

The CSOM GUID simple type, which conforms to the **CSOM GUID** simple type as specified in [MS-CSOM] section 2.2.5.8, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Guid** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.10 CSOM Int16

The CSOM Int16 simple type, which conforms to the **CSOM Int16** simple type as specified in [MS-CSOM] section 2.2.5.9, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int16** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.11 CSOM Int32

The CSOM Int32 simple type, which conforms to the **CSOM Int32** simple type as specified in [MS-CSOM] section 2.2.5.10, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int32** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.12 CSOM Int64

The CSOM Int64 simple type, which conforms to the **CSOM Int64** simple type as specified in [MS-CSOM] section 2.2.5.11, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int64** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.13 CSOM SByte

The CSOM SByte simple type, which conforms to the **CSOM SByte** simple type as specified in [MS-CSOM] section 2.2.5.12, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.SByte** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.14 CSOM Single

The CSOM Single simple type, which conforms to the **CSOM Single** simple type as specified in [MS-CSOM] section 2.2.5.13, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Single** type, as specified in [MS-ODATA] section 2.2.2. The value of the corresponding **Edm.Single** MUST exist in the range of values for **CSOM Single** as specified in [MS-CSOM] section 2.2.5.13.

2.2.6.15 CSOM String

The CSOM String simple type, which conforms to the **CSOM String** simple type as specified in [MS-CSOM] section 2.2.5.14, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.String** type, as specified in [MS-ODATA] section 2.2.2.

2.2.6.16 CSOM TimeSpan

A **CSOM TimeSpan** value specified by [MS-CSOM] section 2.2.5.19 used in OData formats specified by [MS-ODATA] section 2.2.6 MUST conform to the **Edm.Time** type as specified in [MS-ODATA] section 2.2.2.

2.2.6.17 CSOM UInt16

The CSOM UInt16 simple type, which conforms to the **CSOM UInt16** simple type as specified in [MS-CSOM] section 2.2.5.15, is used in OData formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int32** type, as specified in [MS-ODATA] section 2.2.2. The value of the corresponding **Edm.Int32** MUST exist in the range for **CSOM UInt16** as specified in [MS-CSOM] section 2.2.5.15.

2.2.6.18 CSOM UInt32

The CSOM UInt32 simple type, which conforms to the **CSOM UInt32** simple type as specified in [MS-CSOM] section 2.2.5.16, is used in Open Data Protocol (OData) formats as specified in [MS-ODATA] section 2.2.6. This simple type MUST also conform to the **Edm.Int64** type, as specified in [MS-ODATA] section 2.2.2. The value of the corresponding **Edm.Int64** MUST exist in the range for **CSOM UInt32** as specified in [MS-CSOM] section 2.2.5.16.

2.2.6.19 CSOM UInt64

A **CSOM UInt64** value specified by [MS-CSOM] section 2.2.5.17 used in OData formats specified by [MS-ODATA] section 2.2.6 MUST conform to the **Edm.Int64** type as specified in [MS-ODATA] section

2.2.2. When the **CSOM UInt64** value is greater than 9223372036854775807, the protocol server MUST return error.

2.2.7 Attributes

This specification does not define any common XML schema attribute definitions.

2.2.8 Groups

This specification does not define any common XML schema group definitions.

2.2.9 Attribute Groups

This specification does not define any common XML schema attribute group definitions.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Server Details

This protocol allows protocol servers to perform implementation specific authorization checks and to notify protocol clients of authorizations faults by using either HTTP status codes or OData Error Response as specified in [MS-ODATA] section 2.2.8.1. Except where specified otherwise, protocol clients SHOULD interpret HTTP status codes as specified in [RFC2616] section 10 and OData Error Response as specified in [MS-ODATA] section 2.2.8.1.

3.1.1 Abstract Data Model

This section describes the relationship between the abstract data model specified in [MS-CSOM] section 3.1.1 and the abstract data model specified in [MS-ODATA] section 2.2.1. The abstract data model specified in [MS-ODATA] section 2.2.1 uses the Entity Data Model as its data modeling vocabulary, which is specified in [MC-CSDL] section 2. The described relationship is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in [MS-CSOM] and [MS-ODATA].

Data model as specified in [MS-CSOM] map to Entity Data Model terms as shown in the following table:

CSOM data model	Entity Data Model
CSOM value object type	**EDM complex type**
CSOM Object type	**EDM entity type**
CSOM array	A collection of **EDM simple type** values, or a collection of **EDM complex type** values, or **EDM entity set**
CSOM dictionary	A collection of **EDM complex type** values
CSOM null	The **EDM simple type null**
CSOM property	**EDM property** or **EDM navigation property**
CSOM method	**EDM function import**

CSOM Value Object Type

A CSOM value object type is mapped to an **EDM complex type**. The name of the **EDM complex type** is the CSOM type name of the CSOM value object type. For each CSOM property in the CSOM value

object type, if the type of the CSOM property could be mapped to **EDM complex type**, **EDM simple type**, collection of **EDM complex type** or collection of **EDM simple type**, the CSOM property is mapped to an **EDM property** in the corresponding **EDM complex type**, otherwise, the CSOM property is not mapped to any **EDM property** in the corresponding **EDM complex type**.

CSOM Object Type

A CSOM Object type is mapped to an **EDM entity type**. The name of the **EDM entity type** is the CSOM type name of the CSOM Object type. For each CSOM property in the CSOM Object type, if the type of the CSOM property could be mapped to **EDM complex type**, **EDM simple type**, collection of **EDM complex type** or collection of **EDM simple type**, the CSOM property is mapped to an **EDM property** in the corresponding **EDM entity type**. If the type of the CSOM property is mapped to **EDM entity type**, the CSOM property is mapped to an **EDM navigation property** in the corresponding **EDM entity type**. Otherwise, the CSOM property is not mapped to any **EDM properties** or **EDM navigation properties** in the corresponding **EDM entity type**.

CSOM Array

A CSOM array is mapped to a collection of **EDM simple type** values, or a collection of **EDM complex type** values or **EDM entity set**.

Type of Element in CSOM Array	Type of the CSOM Array
Edm simple type	Collection of **EDM simple type** values
Edm complex type	Collection of **EDM complex type** values
Edm entity type	**EDM entity set**

CSOM Dictionary

A CSOM dictionary is mapped to a collection of **EDM complex type** values. Each key value pair in the CSOM dictionary is mapped to an element in the collection. The type of the element in the collection is **SP.KeyValue** as specified in section 3.1.5.1.3.1.

CSOM Null

A **CSOM null** is mapped to the **EDM simple type null**.

CSOM Property

A CSOM property is mapped to an **EDM property** or **EDM navigation property**.

CSOM Method

A CSOM method is mapped to an **EDM function import**.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The following table summarizes the operations of this protocol.

Operation	Description
ProcessRestQuery	Process the **oData** request message that contains resource path, **oData** query options and **oData** payload, and respond with **oData** response message.

3.1.5.1 ProcessRestQuery

During this operation, the protocol server receives an OData request that contains the resource path, OData query options (as specified in [MS-ODATA] section 2.2.3.6), and OData payload (as specified in [MS-ODATA] section 2.2.6); processes the request; and then responds with an OData response.

The protocol server endpoint for this operation is formed by appending "/_vti_bin/client.svc" or "/_api" to the URL of the **Web site**, for example "http://www.example.com/Repository/_vti_bin/client.svc" or "http://www.example.com/Repository/_api".

The resource path and OData query options are appended to the endpoint; for example, "http://www.example.com/Repository/_vti_bin/client.svc/resourcePath?ODataQueryOptions" or "http://www.example.com/Repository/_api/resourcePath?ODataQueryOptions".

The resource path contained in the request message MUST conform to the following format:

```
resourcePath            = resourcePathAlias "/" resourceMembers /
                          resourceRoot "/" resourceMembers
resourcePathAliasValue  = resourceRoot /
                          resourceRoot "/" resourceMembers
resourceRoot            = staticMethodInvoke /
                          staticPropertyGet /
                          constructorInvoke
staticMethodInvoke      = typeName "." staticMethodName [ "(" methodParameters ")" ]
staticPropertyGet       = typeName "." staticPropertyName
constructorInvoke       = typeName [ "(" methodParameters ")" ]
methodParameters        = parameterValue /
                          namedParameterValueList
parameterNameValuePair  = parameterName "=" parameterValue
namedParameterValueList = parameterNameValuePair /
                          parameterNameValuePair "," namedParameterValueList
resourceMembers         = resourceMember /
                          resourceMember "/" resourceMembers
resourceMember          = instancePropertyGet /
                          instanceMethodInvoke
insancePropertyGet      = propertyName
instanceMethodInvoke    = methodName [ "(" methodParameters ")" ]
typeName                = *PCHAR ; CSOM type name
methodName              = *PCHAR ; CSOM method name
propertyName            = *PCHAR ; CSOM property name
parameterName           = *PCHAR ; CSOM parameter name
staticMethodName        = *PCHAR ; CSOM static method name
staticPropertyName      = *PCHAR ; CSOM static property name
resourcePathAlias       = *PCHAR ; resource path alias
parameterValue          = EdmPrimitiveLiteral ; MS-ODATA section 2.2.2
```

where the **resourcePathAlias** is a string whose value is **resourcePathAliasValue**.

The protocol client sends a **ProcessRestQueryIn** message, and the protocol server responds with a **ProcessRestQueryOut** message, as follows:

1. The protocol client sends a **ProcessRestQueryIn** request message that contains resource path, OData query options, and OData payload.

2. The protocol server receives the **ProcessRestQueryIn** request message, processes the resource path to locate the CSOM object, performs actions against the CSOM object, and responds to the protocol client with **ProcessRestQueryOut** response message. If an error occurs while the protocol server processes the request, the response contains an error message that provides details about the error.

The protocol server processes the resource path according to the following steps:

1. If the resource path starts with a resource path alias defined by the protocol server, the resource path alias is replaced with the value of the resource path alias and the protocol server gets the complete resource path.

2. The protocol server processes the complete resource path. From the **resourceRoot**, the protocol gets the first CSOM object by invoking a static CSOM method on a CSOM object type, getting the static property on a CSOM object type, or invoking the constructor of a CSOM object type.

3. The protocol server checks the next resource member. If there is a resource member, the protocol server invokes the CSOM method or CSOM property on the CSOM object from the previous step and repeats step 3. If there is no resource member, the protocol server continues with the following step.

4. The protocol server performs the action against the CSOM object from the previous step.

3.1.5.1.1 Message

The following message definitions are specific to this operation.

3.1.5.1.1.1 Request Body

The **ProcessRestQueryIn** message is the request message for the **ProcessRestQuery** Web service method. The message contains resource path, OData query options and OData payload.

3.1.5.1.1.2 Response Body

The **ProcessRestQueryOut** message is the OData response message for the **ProcessRestQuery** Web service method.

3.1.5.1.2 Elements

None.

3.1.5.1.3 Complex Types

3.1.5.1.3.1 SP.KeyValue

SP.KeyValue complex type represents a key value pair. It has the following properties:

Property	Property Type	Description
Key	Edm.String	The value of the key in the key value pair.

Property	Property Type	Description
Value	Edm.String	The string representation of the value in the key value pair.
ValueType	Edm.String	The EDM type name of the value in the key value pair.

3.1.5.1.3.2 SP.SimpleDataRow

SP.SimpleDataRow complex type represents a row in a data table. It has the following property:

Property	Property Type	Description
Cells	Collection of SP.KeyValue	The cells in the data table row.

3.1.5.1.3.3 SP.SimpleDataTable

SP.SimpleDataTable complex type represents a data table. It has the following property:

Property	Property Type	Description
Rows	Collection of SP.SimpleDataRow	The rows in the data table.

3.1.5.1.4 Simple Types

None.

3.1.5.1.5 Attributes

None.

3.1.5.1.6 Attribute Groups

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

The following examples illustrate how a protocol client retrieves and updates data on a protocol server by using this protocol. The example scenario is a book store that has a catalog of books. Each book within that catalog has an identifier, title, author, and status.

In this scenario, the following CSOM types are defined on the protocol server:

CSOM type	CSOM type name (identifier)	Description	Properties	Methods
BookStore	SampleCode.BookStore (acc57e47-24b0-4400-b1c7-aa1cf3c9542d)	A CSOM Object type that defines the book store, which contains a catalog.	**Catalog** – A **Catalog** type that contains a static property specifying a catalog of books.	None.
Catalog	SampleCode.Catalog (04a81e1b-a5b9-445d-97c0-681fe3f61189)	A CSOM Object type that defines the catalog, which contains a collection of books.	**Books** – A list of books.	None.
BookCollection	SampleCode.BookCollection (4c456811-3967-4021-8d9f-237ebd9c1170)	A CSOM Object type that defines a collection of books within a catalog.	None.	**Add** – Returns a **Book** type and adds a book to the collection. The only parameter is a **BookCreationInformation** object, which contains the title, author, publish date, and status of a book. **GetById** – Returns a **Book** type that represents the book with the specified book identifier. If the book does not exist, the protocol server returns **CSOM null**. The only parameter is a CSOM GUID that contains the book identifier of the book. The **BookCollection** type supports **OData InsertEntity** request as specified in [MS-ODATA] section 2.2.7.1.1 to add a book to the collection.
Book	SampleCode.Book (030f9ac0-5f2b-4422-9e32-bcdfc1a0c93a)	A CSOM Object type that defines a specific book in a collection of books.	**Author** – A **CSOM String** that contains the name of the	**Update** – Updates information about the book by saving the property values that are currently set for the

CSOM type	CSOM type name (identifier)	Description	Properties	Methods
			author of the book. **Id** – A **CSOM String** value that contains the identification number for the book. **PublishDate** – A **CSOM DateTime** value that contains the date when the book was published. **Status** – A **Status** type that contains the status of the book. **Title** – A **CSOM String** that contains the title of the book.	book. This method does not define any parameters. **CheckOut** – Checks out the book for a user. This method has only one parameter, which is the user's name. The return value of the **CheckOut** method is the date that the book is due. The **Book** type supports **OData UpdateEntity Request** as specified in [MS-ODATA] section 2.2.7.3.1 to update information about the book.
BookCreationInformation	SampleCode.BookCreationInformation (dda98aeb-f87d-490f-9a61-be08644ad461)	A CSOM value object type that contains information about a book to be added to the catalog for the book store.	**Author** – A **CSOM String** that contains the name of the author of the book. **PublishDate** – A **CSOM DateTime** value that contains the date when the book was published. **Status** – A **Status** type that contains the status of the book.	None.

CSOM type	CSOM type name (identifier)	Description	Properties	Methods
			Title – A **CSOM String** that contains the title of the book.	
Status		A CSOM enumeration type that specifies whether a book is in stock. If a book is in stock, this value is "0" (zero). If a book is out of stock, this value is "1". If a book is on back order, this value is "2".	None.	None.

In the examples, the catalog contains the following books.

Identifier	Title	Author	Status	Publish date
3387ac63-e73d-421f-bff7-359a4aa2bc38	How to Cook Chinese Food	Soha Kamal	0	March 1, 2008
f6a265ab-86e5-4fbf-937c-49923604b91d	How to Cook Japanese Food	Soha Kamal	1	May 5, 2007
2e80eb25-b64a-4506-b87b-2fff6ddb3f57	Best Recipes	Lisa Andrews	0	January 3, 2009
704655a3-c136-469c-a578-f79652a93f9b	Family Recipes	Patrick Hines	0	December 1, 2005

4.1 Retrieve Book Information

In this example, a protocol client requests information about a book by using the book identifier "3387ac63-e73d-421f-bff7-359a4aa2bc38".

The protocol client sends the following message including some HTTP headers:

```
GET /_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac63-e73d-421f-bff7-
359a4aa2bc38') HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Type: application/json
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
  "d":{
```

```
    "__metadata":{
      "id":"56af5519-0604-4cb5-9ffc-d753bb314dbc",

"uri":"http://www.example.com/_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac6
3-e73d-421f-bff7-359a4aa2bc38')",
      "type":"SampleCode.Book"
    },
    "Author":"Soha Kamal",
    "Id":"3387ac63-e73d-421f-bff7-359a4aa2bc38",
    "PublishDate":"\/Date(1204329600000)\/",
    "Status":0,
    "Title":"How to Cook Chinese Food"
  }
}
```

4.2 Retrieve Books by a Specific Author

In this example, a protocol client requests information about all of the books that are associated with the author named Soha Kamal.

The protocol client sends the following message including some HTTP headers:

```
GET
/_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books?$filter=Author%20eq%20'Soha%20Kamal'
HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Type: application/json
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
  "d":{
    "results":[
      {
        "__metadata":{
          "id":"a2ac7976-b585-4217-850c-c18a273da35c",

"uri":"http://www.example.com/_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac6
3-e73d-421f-bff7-359a4aa2bc38')",
          "type":"SampleCode.Book"
        },
        "Author":"Soha Kamal",
        "Id":"3387ac63-e73d-421f-bff7-359a4aa2bc38",
        "PublishDate":"\/Date(1204329600000)\/",
        "Status":0,
        "Title":"How to Cook Chinese Food"
      },{
        "  metadata":{
          "id":"1449675e-0d53-435f-9953-c23c36a15c4e",

"uri":"http://www.example.com/_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('f6a265a
b-86e5-4fbf-937c-49923604b91d')",
          "type":"SampleCode.Book"
        },
        "Author":"Soha Kamal",
        "Id":"f6a265ab-86e5-4fbf-937c-49923604b91d",
        "PublishDate":"\/Date(1178236800000)\/",
        "Status":1,
```

```
          "Title":"How to Cook Japanese Food"
        }
      ]
    }
  }
```

4.3 Update Book Information

In this example, a protocol client sends a request to update author and status information for the book that is associated with the identifier "2e80eb25-b64a-4506-b87b-2fff6ddb3f57"

The protocol client sends the following message including some HTTP headers:

```
POST / vti bin/client.svc/SampleCode.BookStore.Catalog/Books('2e80eb25-b64a-4506-b87b-
2fff6ddb3f57') HTTP/1.1
Host: www.example.com
Accept: */*, application/json
x-http-method: PATCH
Content-Type: application/json

{'__metadata': {'type': 'SampleCode.Book' }, 'Author': 'Lisa Andrews', 'Status': 1 }
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 204 No Content
```

4.4 Add a Book to a Catalog

In this example, the protocol client submits a request to add a book to the catalog.

The protocol client sends the following message**:**

```
POST / vti bin/client.svc/SampleCode.BookStore.Catalog/Books HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Type: application/json

{
  '__metadata': {'type': 'SampleCode.Book'},
  'Title': 'Simple Cookbook',
  'Author': 'Neil Black',
  'Status': 2,
  'PublishDate': '2009-08-01T00:00:00.0000000'
}
```

The protocol server responds with the following message:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
```

```
{
  "d":{
    "__metadata":{
      "id":"fdda44df-a566-4df6-9b63-c3f8ac236377",

"uri":"http://www.example.com/_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('63a687c
6-921b-f898-4204-22f09533f28e')",
      "type":"SampleCode.Book"
    },
    "Author":"Neil Black",
    "Id":"63a687c6-921b-f898-4204-22f09533f28e",
    "PublishDate":"\/Date(1249084800000)\/",
    "Status":2,
    "Title":"Simple Cookbook"
  }
}
```

4.5 Unsuccessfully Add a Book to a Catalog

In this example, the protocol client submits a request to add a book to the catalog. The protocol server returns error because the book already exists in the catalog.

The protocol client sends the following message including some HTTP headers:

```
POST /_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Type: application/json

{
  '  metadata': {'type': 'SampleCode.Book'},
  'Title': 'Best Recipe',
  'Author': 'Lisa Andrews',
  'Status': 0,
  'PublishDate': '2006-07-20T00:00:00.0000000'
}
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=utf-8

{
  "d":{
    "error":{
      "code":"-2147024809, System.ArgumentException","message":{
        "lang":"en-US","value":"The book with title 'Best Recipe' already exits in the
book store."
      }
    }
  }
}
```

4.6 Retrieve Book Sample Content

In this example, the protocol client submits a request to get the media resource of the book that is associated with the identifier 3387ac63-e73d-421f-bff7-359a4aa2bc38.

The protocol client sends the following message including some HTTP headers:

```
GET /_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac63-e73d-421f-bff7-
359a4aa2bc38')/$value HTTP/1.1
Host: www.example.com
Accept: */*, application/json
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 200 OK
Content-Type: application
Content-Length: 48

Sample Content of book How to Cook Chinese Food.
```

4.7 Update Book Sample Content

In this example, the protocol client submits a request to update the media resource of the book that is associated with the identifier 3387ac63-e73d-421f-bff7-359a4aa2bc38.

The protocol client sends the following message including some HTTP headers:

```
POST /_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac63-e73d-421f-bff7-
359a4aa2bc38')/$value HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Length: 26

New Sample Content of book
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 204 No Content
```

4.8 Check out a Book

In this example, the protocol client submits a request to check out the book that is associated with the identifier 3387ac63-e73d-421f-bff7-359a4aa2bc38.

The protocol client sends the following message including some HTTP headers:

```
POST /_vti_bin/client.svc/SampleCode.BookStore.Catalog/Books('3387ac63-e73d-421f-bff7-
359a4aa2bc38')/Checkout(user='Sam%20Bruce') HTTP/1.1
Host: www.example.com
Accept: */*, application/json
Content-Length: 0
```

The protocol server responds with the following message including some HTTP headers:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
```

```
{
  "d":{
    "CheckOut":"\/Date(1322778642704)\/"
  }
}
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full CSDL

For ease of implementation, the following is the full CSDL schema for this protocol.

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="3.0" xmlns:edmx="http://schemas.microsoft.com/ado/2009/11/edmx">
  <edmx:DataServices m:DataServiceVersion="3.0"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    <Schema Namespace="SP" xmlns="http://schemas.microsoft.com/ado/2009/11/edm">
      <ComplexType Name="KeyValue">
        <Property Name="Key" Type="Edm.String" />
        <Property Name="Value" Type="Edm.String" />
        <Property Name="ValueType" Type="Edm.String" />
      </ComplexType>
      <ComplexType Name="SimpleDataRow">
        <Property Name="Cells" Type="MultiValue(SP.KeyValue)" />
      </ComplexType>
      <ComplexType Name="SimpleDataTable">
        <Property Name="Rows" Type="MultiValue(SP.SimpleDataRow)" />
      </ComplexType>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

7 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 SharePoint Foundation 2013
- Microsoft SharePoint Server 2016 Preview

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.

8 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

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 or functionality.
- The removal of a document from the documentation set.

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 **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
7 Appendix B: Product Behavior	Updated list of supported products.	Y	Content updated due to protocol revision.
8 Change Tracking	Replaced instances of [MS-ODATA] reference (retired) with [OData-Protocol] reference (archived) where it appears throughout document.	N	Content update.

9 Index