[MS-SQMCS2]:
Software Quality Metrics (SQM) Client-to-Service Version 2 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, email 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.

Revision Summary

Date	Revision History	Revision Class	Comments
12/16/2011	1.0	New	Released new document.
03/30/2012	1.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	1.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	1.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	1.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	2.0	Major	Significantly changed the technical content.

Contents

1 Introduction

This specification describes the Software Quality Metrics (SQM) Client-to-Service Version 2 Protocol, which is used to send software instrumentation metrics to the SQM service and for the client to download client-specific control data. The protocol extends the concepts of the Software Quality Metrics (SQM) Client-to-Service Protocol, as specified in [MS-SQMCS]. Implementers should be familiar with the SQM Client-to-Service Protocol and with the message structure used by the protocol as specified in [MS-TPXS].

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 RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

BLOB
GUID
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
SSL
UTC (Coordinated Universal Time)

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-SQMCS] Microsoft Corporation, "Software Quality Metrics (SQM) Client-to-Service Version 1 Protocol".

[MS-TPXS] Microsoft Corporation, "Telemetry Protocol XML Schema".

[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.ietf.org/rfc/rfc2616.txt

1.2.2 Informative References

[MSDN-CAB] Microsoft Corporation, "Microsoft Cabinet Format", http://msdn.microsoft.com/en-us/library/bb417343.aspx

[MSDN-CryptgrpFunts] Microsoft Corporation, "Cryptography Functions", http://msdn.microsoft.com/en-us/library/aa380252(VS.85).aspx

[MSDN-PwrMgmtFunts] Microsoft Corporation, "Power Management Functions", http://msdn.microsoft.com/en-us/library/aa373163(VS.85).aspx

[MSDN-RegQueryValueEx] Microsoft Corporation, "RegQueryValueEx function", http://msdn.microsoft.com/en-us/library/ms724911(VS.85).aspx

[MSDN-RPCF] Microsoft Corporation, "RPC Functions", http://msdn.microsoft.com/en-us/library/aa378623(VS.85).aspx

[MSDN-SysInfoFuncts] Microsoft Corporation, "System Information Functions", http://msdn.microsoft.com/en-us/library/ms724953(VS.85).aspx

[MSDN-WinSysAssmntTool] Microsoft Corporation, "Windows System Assessment Tool", http://msdn.microsoft.com/en-us/library/cc948912(VS.85).aspx

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

1.3 Overview

The Software Quality Metrics (SQM) Client-to-Service Version 2 Protocol defines how a SQM-enabled client sends instrumentation data to the SQM service. The protocol specifies the data transfer method, which includes an instrumentation namespace identifier and binary structured instrumentation data.

The SQM instrumentation data provided by SQM-enabled clients allows application developers to understand product usage and failure information in order to improve their products. Each SQM-enabled client belongs to a SQM namespace known as a SQM partner. All SQM data is associated with a SQM partner namespace in the SQM service.

The structure and method of transferring the data from the SQM-enabled client to the SQM service is defined by the SQM Client-to-Service Version 2 Protocol. The method of creating the SQM instrumentation data definition is SQM service implementation-specific.

The SQM Client-to-Service Version 2 Protocol also defines a method for a SQM-enabled client to download SQM partner-specific information. Typically this information is used by the SQM-enabled client to control what instrumentation data is uploaded. This functionality is known as Adaptive Software Quality Metrics (A-SQM). A-SQM data is created at the SQM service by the SQM-enabled client application owner if the SQM partner wants to download and use this functionality. The method of creating the A-SQM data is SQM service implementation-specific.

The SQM Client-to-Service Version 2 Protocol provides the following communication:

- Uploading instrumentation data from the client to the SQM service.

- Uploading instrumentation data through a proxy (relay) to the SQM service.
- Downloading A-SQM data created at the SQM service.

1.4 Relationship to Other Protocols

This protocol depends on the **Hypertext Transfer Protocol (HTTP)** and **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** for transport, as specified in [RFC2616]. It extends Version 1 of the SQM Client-to-Service Protocol, as specified in [MS-SQMCS] and uses the schema defined in the Telemetry Protocol XML Schema as specified in [MS-TPXS].

1.5 Prerequisites/Preconditions

To use the SQM service, a client is required be SQM-enabled and registered as an SQM partner with the SQM service.

1.6 Applicability Statement

This protocol is applicable to SQM-enabled clients that are required to collect telemetry data by using the SQM service.

1.7 Versioning and Capability Negotiation

The SQM Client-to-Service Version 2 Protocol does not perform version or capability negotiation. The protocol uses HTTP/HTTPS as the transport. The client communicates with a SQM service that supports the SQM Client-to-Service Version 2 Protocol.

1.8 Vendor-Extensible Fields

None. Any vendor extensions (such as adding a key value argument to arg elements) are not interpreted unless they are used by the vendor or identified by specific contract between the client and the service. See [MS-TPXS] section 1.7.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol is implemented on top of HTTP/HTTPS<1>. The proxy MAY impose additional requirements as part of the transfer. There is no authentication between the SQM client and SQM service, or between the SQM proxy and the SQM service.

2.2 Message Syntax

This protocol is an extension of the Software Quality Metrics Client-to-Service Protocol as specified in [MS-SQMCS]. The Software Quality Metrics Client-to-Service Version 2 Protocol adds a set of client-to-server request messages and server-to-client response messages.

The request message is initiated by the client and sent to the server or proxy. The form is a binary header followed by an XML message as specified in [MS-TPXS] section 2.1.1 and illustrated in Figure 1. The XML message is constructed of XML elements that describe the client and the work the client is requesting of the server. The request message includes the SQM session data payload attached in the HTTP POST body, as illustrated in Figure 2, if the XML message contains a **dataupload** request as specified in section 2.2.2.16.2.

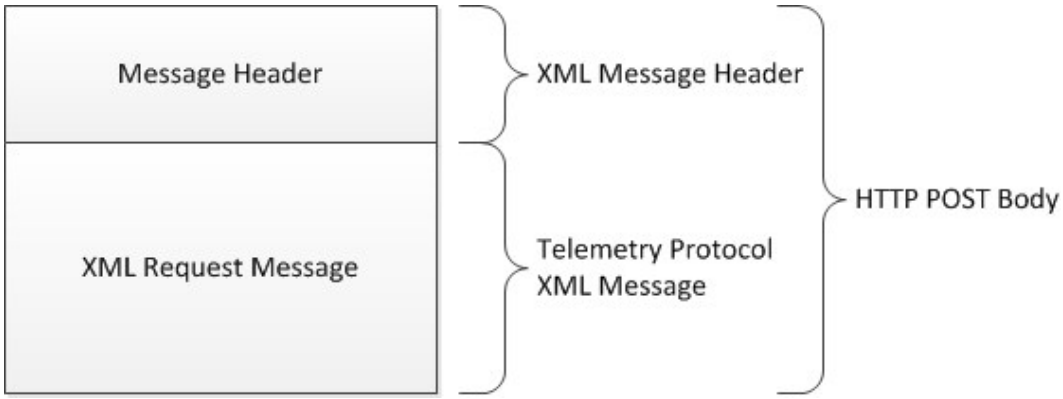


Figure 1: SQM request message in the HTTP POST body

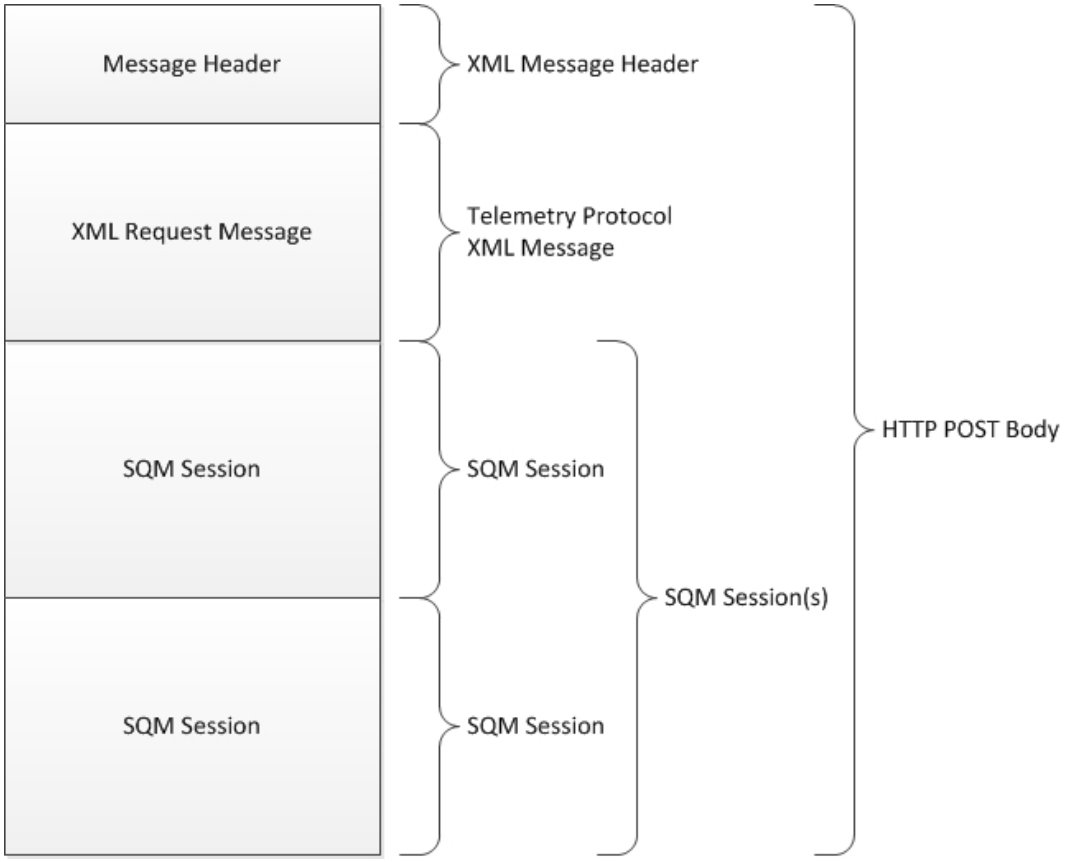


Figure 2: SQM request upload data message with SQM session data payload in the HTTP POST body

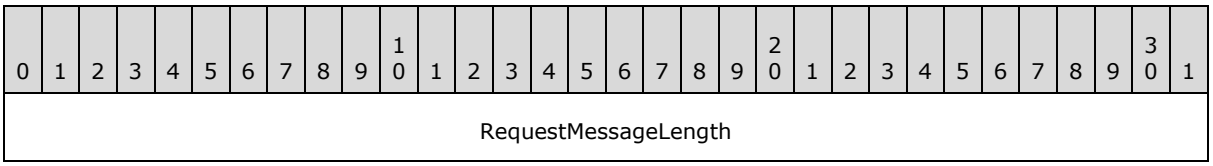
2.2.1 Namespaces

2.2.2 Request Messages

A request message is an XML document as specified in [MS-TPXS] section 2.1. The variable elements and attributes are specified in the following sections.

2.2.2.1 Request Message Header

Every SQM request message MUST begin with a 4-byte message header.



RequestMessageLength (4 bytes): A 32-bit unsigned integer that specifies the length of the XML request message, in bytes. This field is encoded using little-endian format.

2.2.2.2 req Element

The Telemetry request (**req**) element is required. The schema is specified in [MS-TPXS] section 2.1.1.

2.2.2.3 tlm Element

The telemetry (**tlm**) element is required. The schema is specified in [MS-TPXS] section 2.1.1.1.

2.2.2.4 src Element

The client source (**src**) element is required. The schema is specified in [MS-TPXS] section 2.1.1.1.1.

2.2.2.5 desc Element

The client description (**desc**) element is required. It is a child of the **src** element specified in [MS-TPXS] section 2.1.1.1.1.

2.2.2.6 mach Element

The client machine (**mach**) element is required. It is a child of the **src** element specified in [MS-TPXS] section 2.1.1.1.1.

2.2.2.7 os Element

The operating system (**os**) element schema is specified in [MS-TPXS] section 2.1.1.1.1.1. The **os** element is the parent element of a set of child **arg** elements describing the operating system. The **os** element is required and MUST include the following **arg** elements.

os arg element:<2>

- **nm** attribute: **vermaj**
- **val** attribute: A 32-bit decimal number specifying the operating system major version number.

os arg element:

- **nm** attribute: **vermin**
- **val** attribute: A 32-bit decimal number specifying the operating system minor version number.

os arg element:

- **nm** attribute: **verbld**
- **val** attribute: A 32-bit decimal number specifying the operating system build number.

os arg element:

- **nm** attribute: **versp**
- **val** attribute: A 32-bit decimal number specifying the operating system service pack number.

os arg element:

- **nm** attribute: **csdbld**<3>

- **val** attribute: A 32-bit decimal number specifying the operating system revision number.

os arg element:

- **nm** attribute: **sku**<4>
- **val** attribute: A 32-bit decimal number specifying the operating system stock keeping unit (SKU) value.

The **os** element MAY include the following **arg** elements.

os arg element:

- **nm** attribute: **arch**<5>
- **val** attribute: A value specifying the operating system processor architecture.

os arg element:

- **nm** attribute: **ntprodtype**<6>
- **val** attribute: A 32-bit decimal number value specifying the operating system product type.

os arg element:

- **nm** attribute: **platid**<7>
- **val** attribute: A 32-bit decimal number value specifying the operating system platform identifier.
- **val** attribute: A binary value (0 or 1) specifying if the operating system is portable.

os arg element:

- **nm** attribute: **prodsuite**<8>
- **val** attribute: A 32-bit decimal number value specifying the operating system product suite bitmap.

Additional **arg** elements MAY be specified and are dependent upon the client and server implementation. Unrecognized **arg** key-value pairs are ignored by the server.

2.2.2.8 hw Element

The hardware (**hw**) element schema is specified in [MS-TPXS] section 2.1.1.1.1.2. The **hw** element is the parent element of a set of child **arg** elements that describe the hardware platform. The **hw** element is required and MAY include the following **arg** elements. Child **arg** elements are not required.

hw arg element:

- **nm** attribute: **arch**<9>
- **val** attribute: A string value describing the processor architecture.

hw arg element:

- **nm** attribute: **bv**<10>
- **val** attribute: A 32-bit decimal number specifying the hardware BIOS version (bv) number.

hw arg element:

- **nm** attribute: **sysmfg**
- **val** attribute: A string value specifying the system manufacturer name.

hw arg element:

- **nm** attribute: **syspro**<11>
- **val** attribute: A string value specifying the product model name.

hw arg element:

- **nm** attribute: **form**<12>
- **val** attribute: A 32-bit decimal number specifying the hardware form factor.

hw arg element:

- **nm** attribute: **aoac**<13>
- **val** attribute: A binary value (0 or 1) specifying if the client is always connected.

hw arg element:

- **nm** attribute: **wscpudn**
- **val** attribute: Windows System Assessment Tool (WinSAT)<14> CPU Description Name

hw arg element:

- **nm** attribute: **wscpusc**
- **val** attribute: Windows System Assessment Tool (WinSAT) CPU Score

hw arg element:

- **nm** attribute: **wsdgsc**
- **val** attribute: Windows System Assessment Tool (WinSAT) Desktop Graphics Score

hw arg element:

- **nm** attribute: **wsdsksc**
- **val** attribute: Windows System Assessment Tool (WinSAT) Disk Score

Additional **arg** elements MAY be specified and are dependent upon the client and server implementation. Unrecognized **arg** key-value pairs are ignored by the server.

2.2.2.9 ctrl Element

The control (**ctrl**) element schema is specified in [MS-TPXS] section 2.1.1.1.1.3. The **ctrl** element is the parent element of a set of child **arg** elements describing a set of client control values. The **ctrl** element is required and MUST include the following **arg** elements.

ctrl arg element:

- **nm** attribute: **mid**<15>
- **val** attribute: A one-time randomly generated globally unique identifier (GUID) value uniquely identifying the client machine.

ctrl arg element:

- **nm** attribute: **sample**<16>
- **val** attribute: A 64-bit decimal number specifying a client one-time randomly generated value in the range [1, 108].

ctrl arg element:

- **nm** attribute: **tm**
- **val** attribute: The client 64-bit decimal FILETIME value specifying the message creation time.

Additional **arg** elements MAY be specified and are dependent upon the client and server implementation. Unrecognized **arg** key-value pairs are ignored by the server.

2.2.2.10 reqs Element

The requests (**reqs**) element schema is specified in [MS-TPXS] section 2.1.1.1.2. The **reqs** element is the parent element to the specific SQM client-to-server requests. The **reqs** element is required.

2.2.2.11 payload Element

The **payload** element specifies any SQM session data payload in the POST body following the XML message (see Figure 2). A payload element without child **arg** elements MAY be included when there is no SQM binary data in the POST body.

The **payload** element schema is specified in [MS-TPXS] section 2.1.1.1.2.1. The **payload** element is the parent element of a set of child **arg** elements describing the payload length and compression (if any). The **payload** element is required and MUST include the following **arg** elements if the client request includes the request command **dataupload** as specified in section 2.2.2.16.2.

payload arg element:

- **nm** attribute: size
- **val** attribute: A 64-bit decimal number specifying the length of the binary SQM payload in bytes.

The **payload** element is required and MUST include the following **arg** elements if the SQM session binary data is compressed. If the SQM session binary data is not compressed, then the client MUST NOT include the following arguments.

payload arg element:

- **nm** attribute: comp
- **val** attribute: A value specifying compression type.<17>

payload arg element:

- **nm** attribute: **precompsize**

- **val** attribute: A 64-bit decimal number specifying the length of the binary data before compression.

Additional **arg** elements MAY be specified and are dependent upon the client and server implementation. Unrecognized **arg** key-value pairs are ignored by the server.

2.2.2.12 req inner Element

The request (**req**) element specifies the request from the client to the server. The **req** element schema is specified in [MS-TPXS] section 2.1.1.1.2.2. The **req** element is required and MUST specify an attribute name-value pair. The attribute name MUST be specified as **key** with a messagewide unique value.

2.2.2.13 namespace Element

The **namespace** element specifies the request namespace from the client to the server. The **namespace** element schema is specified in [MS-TPXS] section 2.2.1.1.1.1.1. The **namespace** element and all attributes (**svc**, **ptr**, **gp**, **app**) are required.

arg child elements MAY be specified and are dependent upon the client and server implementation. Unrecognized **arg** key-value pairs are ignored by the server.

2.2.2.13.1 svc Attribute

Service (**svc**) is a required attribute of the **namespace** element as specified in [MS-TPXS] section 2.1.1.1.2.2.1. The **svc** attribute MUST specify the string value **sqm**.

2.2.2.13.2 ptr Attribute

Partner (**ptr**) is a required attribute of the **namespace** element as specified in [MS-TPXS] section 2.1.1.1.2.2.1. The **ptr** attribute MUST specify a predefined SQM partner name. The SQM partner name MUST be defined at the SQM server service. The SQM partner name is an abstract entity within the SQM service that logically groups instrumentation information.

2.2.2.13.3 gp Attribute

Group (**gp**) is a required attribute of the **namespace** element as specified in [MS-TPXS] section 2.1.1.1.2.2.1. The **gp** attribute is SQM client-defined.

2.2.2.13.4 app Attribute

Application (**app**) is a required attribute of the **namespace** element as specified in [MS-TPXS] section 2.1.1.1.2.2.1. The **app** attribute is SQM client-defined.

2.2.2.14 ctrl inner Element

The control (**ctrl**) element schema is specified in [MS-TPXS] section 2.1.1.1.2.2.2. The **ctrl** element is the parent element of a set of child **arg** elements describing a set of client request control values. The **ctrl arg** elements are required as specified in the following sections.

ctrl arg element: This argument is optional for **requpload** and **dataupload** requests.

- **nm** attribute: **sid**

- **val** attribute: A SQM partner-defined 32-bit decimal Study Identifier (sid) value. This value MAY be used to enforce different sample rates for incoming client to server requests.

ctrl arg element: This argument is optional for **requpload** and **dataupload** requests.

- **nm** attribute: **uid**
- **val** attribute: A SQM partner-defined **GUID** identifying a user.

ctrl arg element: This argument is reqired for a **dataupload** request.

- **nm** attribute: **startutc**
- **val** attribute: The SQM session start time in 64-bit decimal FILETIME format.

ctrl arg element: This argument is required for a **dataupload** request.

- **nm** attribute: **endutc**
- **val** attribute: The SQM session end time in 64-bit decimal FILETIME format.

2.2.2.15 contents Element

The **contents** element schema is specified in [MS-TPXS] section 2.1.1.1.2.2.3. The **contents** element is the parent element of a set of child **arg** elements describing the SQM session data. The **contents** element is optional and MAY include SQM partner-defined name-value pair **arg** elements describing the SQM session data.

contents arg element:

- **nm** attribute: A SQM partner-defined nm attribute
- **val** attribute: A SQM partner-defined **val** attribute

Unrecognized **arg** name-value pairs are ignored by the server.

2.2.2.16 cmd Element

The command (**cmd**) element schema is specified in [MS-TPXS] section 2.1.1.1.2.2.4. The **cmd** element is the parent of a set of child **arg** elements with key-value attribute pairs specifying the command arguments (if any). The **cmd** element has one attribute, **nm**, specifying the command verb value. The command verb specifies the action the SQM client is requesting from the SQM service.

There are three defined commands: request to upload SQM session data (**requpload**), SQM session data upload (**dataupload**), and query A-SQM resource (**qryrsrc**).

2.2.2.16.1 requpload Command

The request to upload data (**requpload**) command specifies that the client is requesting permission to upload SQM session data as described in the **req** element section 2.2.2.12, section 2.2.2.13, and section 2.2.2.14. The SQM server is required to approve or deny the request. There are no child **arg** elements.

2.2.2.16.2 dataupload Command

The data upload (**dataupload**) command specifies that the client is uploading SQM session data. The SQM session data is appended to the XML message in the HTTP POST or PUT body either compressed, (as shown in Figure 6), or uncompressed, (as shown in Figure 7). Each SQM session in the binary data payload MUST be referenced by a separate request element (see section 2.2.2.12). The **dataupload** command references one SQM session only. The **dataupload** command MUST include the following **arg** elements.

cmd arg element:

- **nm** attribute: **tm**
- **val** attribute: The client 64-bit decimal FILETIME value specifying the upload time.

cmd arg element:

- **nm** attribute: **size**
- **val** attribute: A 64-bit decimal value specifying the length, in bytes, of the SQM session (uncompressed) in the data **BLOB** of the appended SQM session binary data stream.

cmd arg element:

- **nm** attribute: **offset**
- **val** attribute: A 64-bit decimal value specifying the offset, in bytes, of the SQM session (uncompressed) that this command references in the SQM session data payload (see Figure 2).

cmd arg element:

- **nm** attribute: **token**
- **va**l attribute: An opaque string value specifying the token provided to the SQM client by the SQM server in the **approved** response message from the client **requpload** command as specified in section 2.2.3.6.2.

Unrecognized **arg** name-value pairs are ignored by the server.

2.2.2.16.3 qryrsrc Command

The query resource (**qryrsrc**) command specifies that the client is requesting A-SQM version and path information. The A-SQM manifest is specific to the namespace specified in section 2.2.2.13. The **qryrsrc** command MUST include the following **arg** element.

cmd arg element:

- **nm** attribute: **name**
- **val** attribute: The string value **manifest** MUST be specified.

Unrecognized **arg** name-value pairs are ignored by the server.

2.2.3 Response Messages

A SQM response message is an XML document as specified in [MS-TPXS] section 2.2. The elements and attributes are specified in the following sections. A SQM response message is returned to the SQM client for each valid SQM request message. There is a one-to-one relationship of SQM request

message to SQM response message. The SQM response message is returned to the SQM client in the HTTP server response stream.

2.2.3.1 resp Element

The telemetry response (**resp**) element is required. The schema is specified in [MS-TPXS] section 2.2.1.

2.2.3.2 tlm Element

The telemetry (**tlm**) element is required. The schema is specified in [MS-TPXS] section 2.2.1.1.

2.2.3.3 resps Element

The responses (**resps**) element schema is specified in [MS-TPXS] section 2.2.1.1.1. The **resps** element is the parent element to the specific SQM server to client response. The **resps** element is required.

2.2.3.4 resp inner Element

The responses (**resp**) element specifies the request from the server to the client. The **resp** element schema is specified in [MS-TPXS] section 2.2.1.1.1.1. The **resp** element is required. There is one **resp** element for each **req** element (section 2.2.2.12). The **resp** element MUST specify an attribute name-value pair. The attribute name MUST be specified as **key**. The attribute value MUST match the key value in the corresponding **req** element (see section 2.2.2.12).

2.2.3.5 namespace Element

The **namespace** element specifies the request namespace from the client to the server. The **namespace** element schema is specified in [MS-TPXS] section 2.2.1.1.1.1.1. The **namespace** element and all attributes (**svc**, **ptr**, **gp**, **app**) are required.

The **namespace** element MUST be the identical namespace as defined in the corresponding **req** element child **namespace** element (see section 2.2.2.13).

2.2.3.6 cmd Element

The command (**cmd**) element schema is specified in [MS-TPXS] section 2.2.1.1.1.1.2. The **cmd** element is the parent of a set of child **arg** elements with key-value attribute pairs specifying the command arguments (if any). The **cmd** element has one attribute, **nm**, specifying the command name value.

The response **cmd** element is the server response to the client request **cmd** (see section 2.2.2.16). One or more **cmd** elements are required. The response commands are specified in the following sections.

2.2.3.6.1 receipt Command

The **receipt** command specifies that the server is acknowledging a successful data upload command (see section 2.2.2.16.2). This command is valid only for client data upload request commands as specified in section 2.2.2.16.2. The **receipt** command MUST include the following **arg** elements.

cmd arg element:

- **nm** attribute: **tm**

- **val** attribute: Server FILETIME value specifying the received time of the client data upload. The form MUST be a 64-bit decimal number.

2.2.3.6.2 approved Command

The **approved** command specifies that the server approves the SQM client request for the upload command (see section 2.2.2.16.1). This command is valid for client request for upload commands as specified in section 2.2.2.16.1. The **approved** command MUST include the following **arg** elements.

cmd arg element:

- **nm** attribute: **token**
- **val** attribute: An opaque token string that MUST be included as an command argument in the client data upload command as specified in section 2.2.2.16.2

cmd arg element:

- **nm** attribute: **tm**
- **val** attribute: A 64-bit decimal FILETIME value specifying the expiration time of the token.

2.2.3.6.3 rsrc Command

The **rsrc** command specifies the A-SQM resource version and path for the SQM client query resource command (see section 2.2.2.16.3). This command is valid only for client query resource commands as specified in section 2.2.2.16.3. The **rsrc** command MUST include the following **arg** elements.

cmd arg element:

- **nm** attribute: **ver**
- **val** attribute: A 32-bit decimal value specifying the SQM server version of the A-SQM requested resource.

cmd arg element:

- **nm** attribute: **path**
- **val** attribute: A path fragment specifying the A-SQM manifest path. The fragment MUST be appended to the SQM host path designated by the SQM client in order to form an HTTP(S) GET URL.

2.2.3.6.4 error Command

The **error** command specifies that the server failed to process the SQM client request. This command is valid for all client request commands as specified in section 2.2.2.16. The **error** command MUST include the following **arg** element.

cmd arg element:

- **nm** attribute: **retry**
- **val** attribute: An integer specifying the value 0 or 1. If the value is 0 (false), the client does not retry the request. If the value is 1 (TRUE) the client MAY retry the request.

The **error** command MAY include the following **arg** elements.

cmd arg element:

- **nm** attribute: **code**
- **val** attribute: A string value specifying the failure code. The failure code is implementation-specific.

cmd arg element:

- **nm** attribute: **message**
- **val** attribute: A string value specifying the failure message. The failure message is implementation-specific.

2.2.3.6.5 throttle Command

The **throttle** command specifies that the server rejects the request to upload data and requires the SQM client to halt requests for the period of days specified in the argument. This command is valid only for client request for upload commands as specified in section 2.2.2.16.1. The **throttle** command MUST include the following **arg** elements.

cmd arg element:

- **nm** attribute: **period**
- **val** attribute: A 32-bit decimal integer specifying the number of days the server requests the client to halt sending **requpload** messages.

cmd arg element:

- **nm** attribute: **namespace**
- **val** attribute: A string value describing the namespace hierarchy to enforce the throttle response. The value MUST be specified from the following table:

Value	Meaning
root	Stop all SQM communication.
svc	Throttle requests for the period at the level of the svc namespace attribute defined in the **namespace** element. Filter these requests based on the path: root.svc.
ptr	Throttle requests for the period at the level of the ptr namespace attribute defined in the **namespace** element. Filter these requests based on the path: root.svc.ptr.
gp	Throttle requests for the period at the level of the gp namespace attribute defined in the **namespace** element. Filter these requests based on the path: root.svc.ptr.gp.
app	Throttle requests for the period at the level of the app namespace attribute defined in the **namespace** element. Filter these requests based on the path: root.svc.ptr.gp.app.
all	Throttle requests for the period at the level of the complete **namespace** element including any arguments. Filter these requests based on the path: root.svc.ptr.gp.app * all arguments.

2.2.3.6.6 none Command

The **none** command specifies that the server acknowledges the request message and that there is no information to send to the client or any action requested from the client. This command is valid only for client query resource request commands as specified in section 2.2.2.16.3. There are no command **arg** elements.

2.3 Directory Service Schema Elements

None.

3 Protocol Details

3.1 Client Details

The client role in the SQM Client-to-Service Version 2 Protocol is an extension of the client role described in [MS-SQMCS] section 3.1. The SQM Client-to-Service Version 2 Protocol extends the base protocol by adding an XML message describing the service request and the client response.

3.1.1 Abstract Data Model

None.

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

3.1.5.1 Message Construction

The client constructs an XML message as specified in section 2.2.2.<18> The SQM response XML message is returned in the HTTP response stream.

An HTTP 200 status code specifies that the SQM server received and processed the XML request message. A malformed request message will have an empty response stream. Any other HTTP status code specifies an HTTP error, and the SQM client does not attempt to process a response XML message.

3.1.5.2 Request to Upload Data Message

The SQM client creates a request to upload (**requpload**) message to request permission from the SQM server, as shown in Figure 3, for the SQM client to send SQM session binary data.

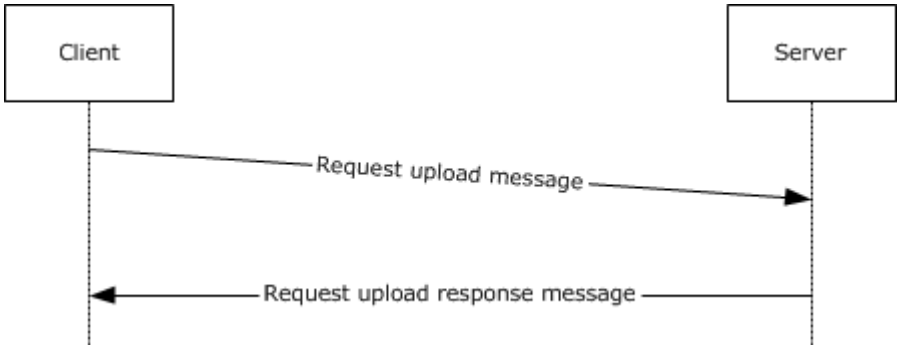


Figure 3: SQM Client to SQM server request upload message and response

Prior to constructing a SQM **requpload** request, the SQM client determines if any Throttle response directive is still in effect and, if so, does not execute a **requpload** request to the SQM server (see section 3.1.5.2.2).

The SQM client creates a SQM XML message as specified in section 2.2.2 by using the command **requpload** as specified in 2.2.2.16.1. The XML message MAY contain more than one **requpload** request.

The SQM client sends the request upload message to the SQM server or SQM proxy by using HTTP(S) POST. The SQM client processes the response message for each request as specified in section 2.2.3 and this process is detailed in the following sections.

3.1.5.2.1 Approved Response Command

This command specifies that the SQM server has approved the SQM client to upload data as specified in the request upload message. The SQM client sends the data upload before the expiration time specified in the response message **approved** command **tm** argument as specified in section 2.2.3.6.2. The **approved** command **token** argument MUST be included in the data upload message as specified in section 2.2.2.16.1.

3.1.5.2.2 Throttle Response Command

This command specifies that the SQM server rejects the SQM client request to upload data as specified in section 2.2.3.6.5. The SQM client does not send the data upload associated with the request upload message to the SQM server.

The client does not send a request upload message where the request upload namespace matches the **throttle namespace** argument for the time period specified in the **throttle period** argument as specified in section 2.2.3.6.5 and detailed in the following sections. The client maintains a catalog of throttle responses in order to query future requests to determine whether to send a future **requpload** message to the SQM server or to discard it.

period: argument: The SQM client is required to stop any request upload or data upload messages to the SQM server for all messages, for the period of days from the client current system time, where the request namespace (see section 2.2.2.13) matches the response namespace (see section 2.2.3.5) by using the throttle namespace comparison as detailed in the following sections.

namespace: argument: During the throttle period, the SQM client compares the request message namespace and discard all requests where the request namespace matches the response namespace by using the throttle comparison namespace argument as specified in section 2.2.3.6.5 and detailed in the following sections.

root: The SQM client stops all requests for the throttle **period**.

svc: The SQM client stops all requests where the **namespace svc** attribute equals **sqm** (see section 2.2.3.5).

ptr: The SQM client stops all requests where the **namespace svc** attribute equals **sqm** and the **ptr** attribute equals the throttle response message namespace **ptr** attribute value (see section 2.2.3.5).

gp: The SQM client stops all requests where the **namespace svc** attribute equals **sqm**, the **ptr** attribute equals the throttle response message namespace **ptr** attribute value, and the **gp**

attribute equals the throttle response message namespace **gp** attribute value (see section 2.2.3.5).

app: The SQM client stops all requests where the **namespace svc** attribute equals **sqm**, the **ptr** attribute equals the throttle response message namespace **ptr** attribute value, the **gp** attribute equals the throttle response message namespace **gp** attribute value, and the **app** attribute equals the throttle response message namespace **app** attribute value (see section 2.2.3.5).

all: The SQM client stops all requests where the **namespace svc** attribute equals **sqm**, the **ptr** attribute equals the throttle response message namespace **ptr** attribute value, the **gp** attribute equals the throttle response message namespace **gp** attribute value, the **app** attribute equals the throttle response message namespace **app** attribute value, and all namespace arguments (if any) equal the throttle response message namespace arguments (see section 2.2.3.5).

3.1.5.2.3 Error Response Command

The Error Response command specifies that the SQM server cannot process the message. The error command MAY contain an error code and error message that the SQM client MAY log for reference. The SQM client captures the retry value as specified in section 2.2.3.6.4 The SQM client retries the operation after a random time period between 8 and 24 hours if the **retry** value is TRUE. The SQM client does not retry the request if the **retry** value is FALSE.

3.1.5.3 Data Upload Message

The SQM client creates a data upload (**dataupload**) message to request that the SQM server accept the uploaded SQM session data as shown in figure 4.

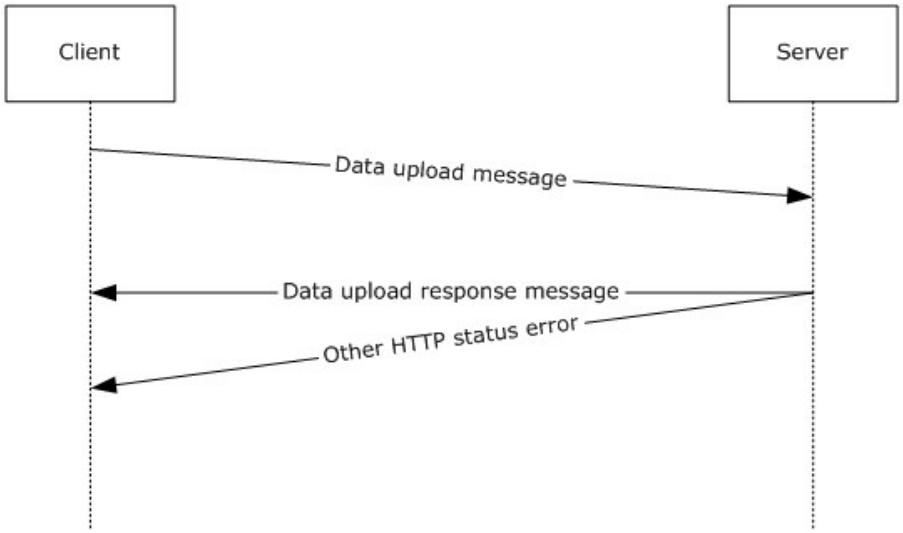


Figure 4: SQM Client to SQM Server data upload message

The SQM client uploads SQM session data by creating a data upload message and appending the SQM session data to the XML message in the HTTP upload. The complete XML message MAY contain more than one data upload request. There is a one-to-one mapping of data upload command request (**req**) elements to each SQM session binary data package as specified in section 2.2.2.12.

The data upload message MUST be preceded by a request to upload a message in order to obtain a valid data upload token as specified in section 2.2.2.16.2. The client checks the token expiration time provided in the approved response (section 2.2.3.6.2). The client does not send the data upload if the client system time has exceeded the token expiration time. The server rejects the upload if the token is malformed or expired.

The SQM client sends the data upload message to the SQM server or SQM proxy by using HTTP(S) PUT or POST.

3.1.5.3.1 Data Upload – SQM Session Data Construction

The SQM client produces a SQM session binary data package as specified in [MS-SQMCS] section 2.2.3. The SQM client MUST NOT compress the session binary data as specified in [MS-SQMCS] section 3.1.5.1.2.1. The SQM client creates one SQM session binary data package per approved **requpload** request.<19>

The SQM client MAY concatenate the SQM session binary data packages (if there are more than one) into a single contiguous binary data BLOB. The client is required to record the total length of the BLOB, the length of each SQM session, and the offset of each SQM session within the BLOB as illustrated in Figure 5.

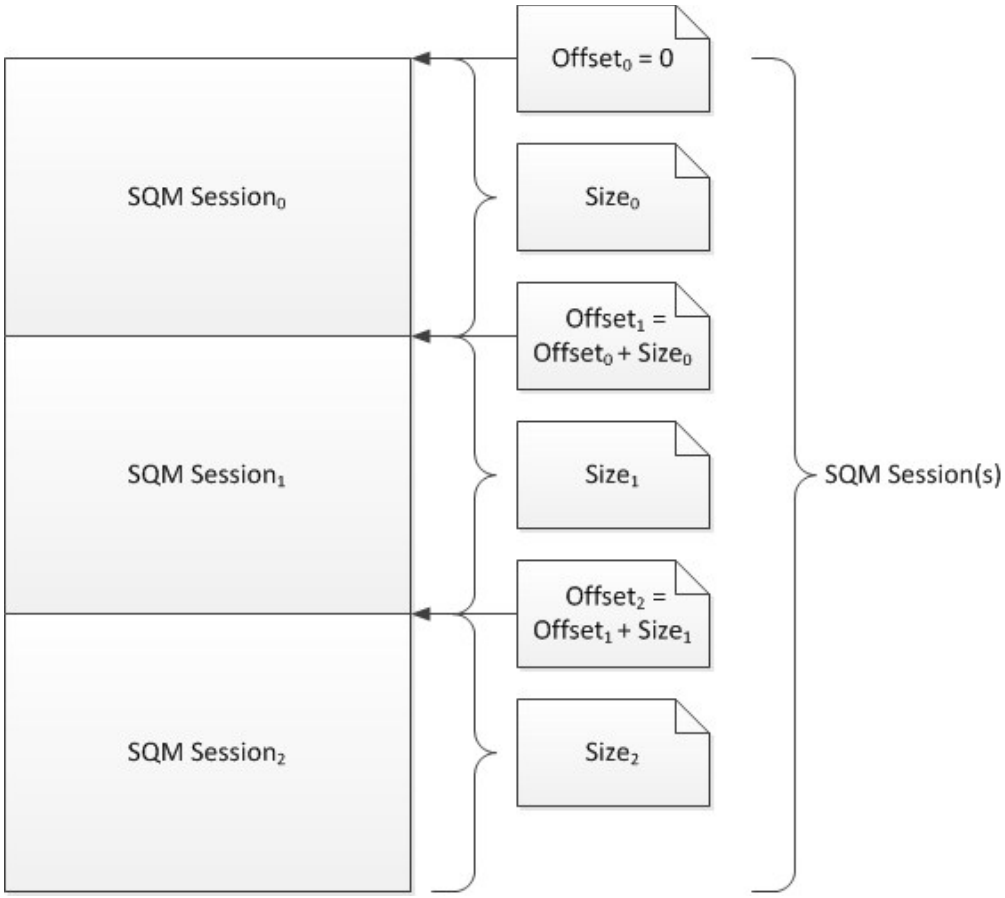


Figure 5: Concatenated SQM sessions

The client MAY compress the entire BLOB recording the compressed data length and compression method as illustrated in Figure 6. The compression method MUST be supported by the SQM server.

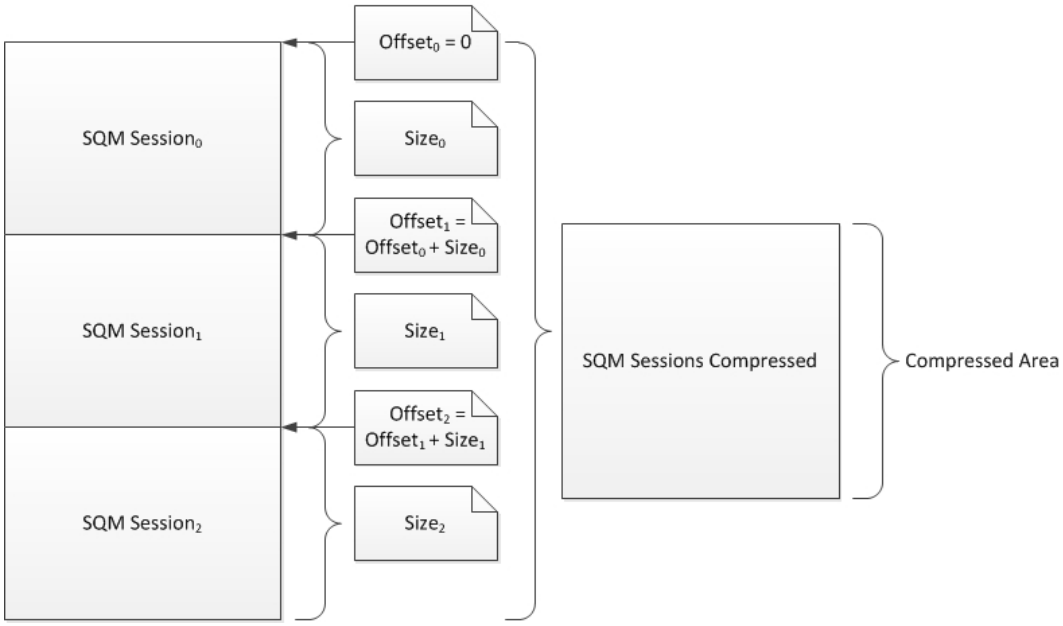


Figure 6: Concatenated SQM sessions compressed

3.1.5.3.2 Data Upload – XML Message Construction

The SQM client creates a SQM XML message as specified in section 2.2.2 by using the command **dataupload** as specified in section 2.2.2.16.2. The **dataupload** request MUST map to the SQM session binary data.

- The client constructs the global metadata elements; specifically, the **os** element (section 2.2.2.7), the **hw** element (section 2.2.2.8), and the **ctrl** element (section 2.2.2.9).
- The client constructs the **reqs** element as specified in section 2.2.2.10.
- The client constructs a **payload** element as a child element of **reqs**. The payload arguments are set to the values computed in the SQM session creation as specified in section 2.2.2.11 and section 3.1.5.3.1.
- The client constructs a **req** element, as a child element of **reqs,** for each SQM session in the appended SQM session's data BLOB. A unique **key** attribute value is created for each **req** element as specified in section 2.2.2.12.
- The client constructs a **namespace** element as a child element of **req** as specified in section 2.2.2.13. Each of the namespace attributes MUST be set to a valid value.
  - **svc**: The client sets this value to **sqm**.
  - **ptr**: The client sets this value to a well-known partner name value. The partner name MUST be known by the SQM server as described in section 2.2.2.13.2.
  - **gp**: The client sets this value to a SQM client-defined value as specified in section 2.2.2.13.3.

- **app**: The client sets this value to a SQM client-defined value as specified in section 2.2.2.13.4.

- The client constructs a **ctrl** element, as a child element of **req**, as specified in section 2.2.2.14.
- The client constructs a **cmd** element, as a child element of **req**, as specified in section 2.2.2.16 with the command **dataupload** as specified in section 2.2.2.16.2.
  - The client creates a command argument **tm**, setting the value to the current system **UTC** time as specified in 2.2.2.16.2.
  - The client creates a command argument **token**, setting the value to the token string as specified in 2.2.2.16.2.
  - The client creates a command argument **size**, setting the value to the SQM session length as specified in 2.2.2.16.2 and specified in section 3.1.5.3.1.
  - The client creates a command argument **offset**, setting the value to the SQM session offset as specified in 2.2.2.16.2 and described in 3.1.5.3.1.

The SQM client completes the XML message and computes the length of the XML message, in bytes.

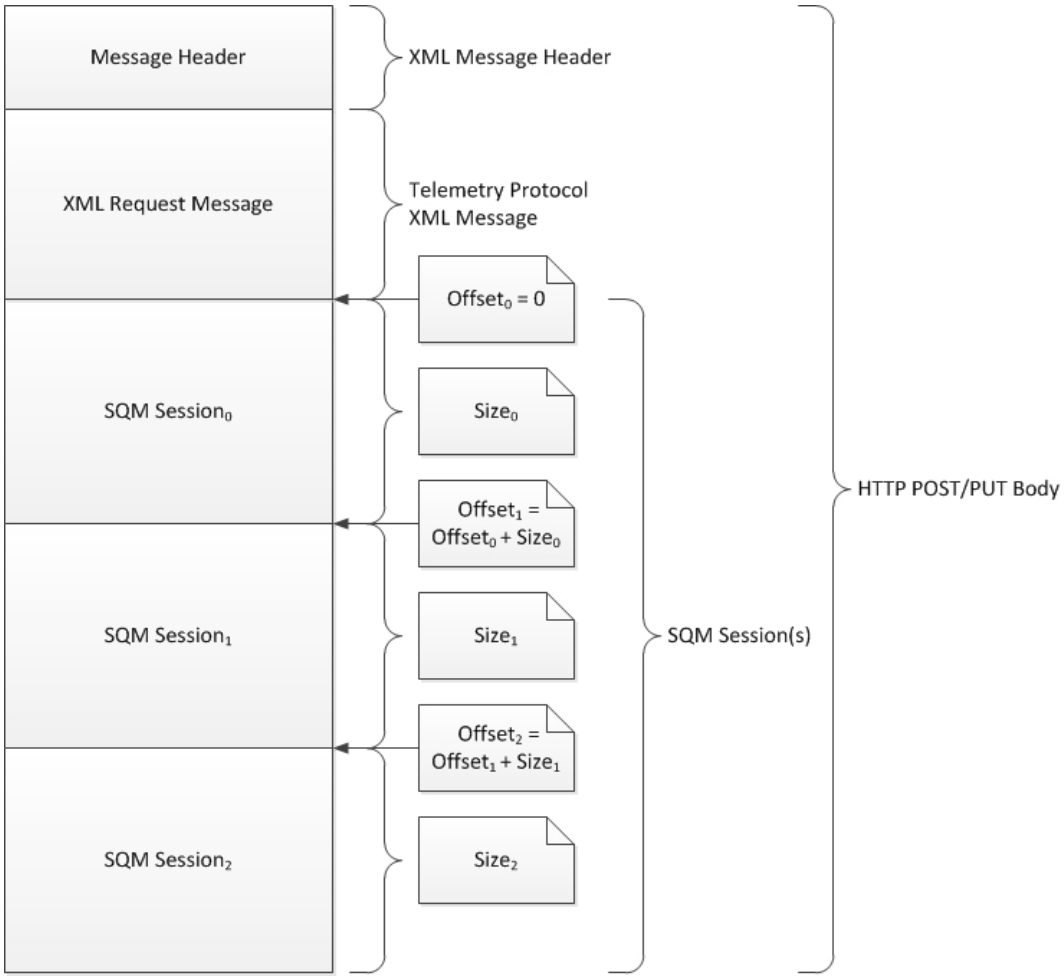


Figure 7: SQM session data upload (uncompressed)

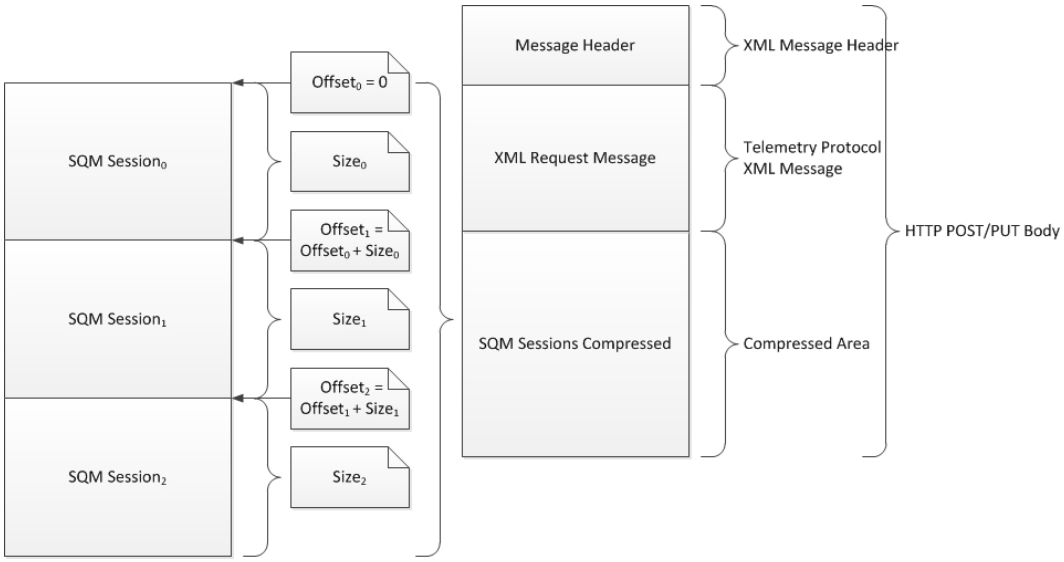


Figure 8: SQM session data upload (compressed)

3.1.5.3.3 Data Upload – Send

The SQM client creates a message header specifying the XML document length as specified in section 2.2.2.1. The message header, XML message, and SQM session binary data BLOB (compressed or uncompressed) are concatenated together into one stream as shown in Figure 7 and Figure 8.

The SQM client sends the entire message stream to the SQM server by using HTTP(S) POST or PUT. The SQM client processes the data upload response message as specified in section 2.2.3 and shown in the following section.

3.1.5.3.4 Data Upload – Receipt Response

The SQM client receives a receipt response for each **dataupload** accepted by the SQM server as specified in section 2.2.3.6.1. The receipt specifies that the SQM server received and accepted the data upload without error. The SQM client MAY log the receipt information.

3.1.5.3.5 Data Upload – Error Response

The SQM server failed to process the **dataupload** message. The error command MAY contain an error code and error message that the SQM client MAY log for reference. The SQM client captures the retry value as specified in 2.2.3.6.4. The SQM client MAY retry the operation if the **retry** value is TRUE. The SQM client does not retry the request if the **retry** value is FALSE.

3.1.5.4 Query Resource Message

The SQM client creates a query resource (**qryrsrc**) message to request from the SQM server the current version and path of the specified named resource as shown in Figure 9.

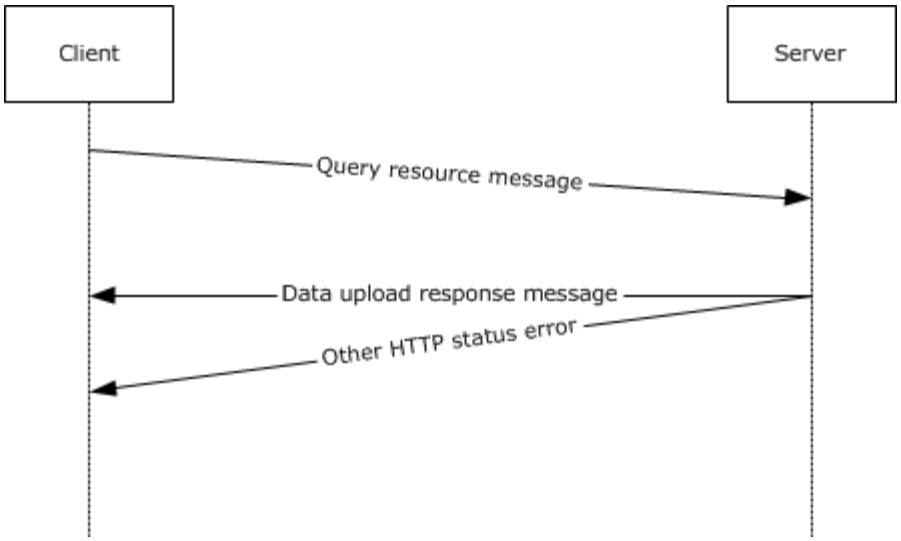


Figure 9: SQM Client to SQM Server query resource message

The SQM client creates a SQM XML message as specified in section 2.2.2 by using the **qryrsrc** command as specified in section 2.2.2.16.3. The SQM client processes the response message for the request as specified in section 2.2.3 and detailed in the following sections.

3.1.5.4.1 Resource Response Command

The client receives a resource response as specified in section 2.2.3.6.3. The SQM client compares the version (**ver**) number in the **rsrc** command against the current client version number. If the version numbers are not equal, the SQM client downloads the A-SQM manifest described in the response.

The SQM client forms an HTTP(S) GET path by concatenating the **path** argument (as specified in section 2.2.3.6.3) from the command response message. The form is shown in the following example, where **%PATH%** is replaced with the value specified in the **path** argument.

```
https://sqm.telemetry.microsoft.com/%PATH%
```

The client downloads the resource file by using HTTP(S) GET. The resource is described in [MS-SQMCS] section 2.2.6.

3.1.5.4.2 None Response Command

The SQM client receives a **none** response as specified in section 2.2.3.6.6. The response indicates that no resource is available. The SQM client takes no action.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

This section describes the server role in the SQM Client-to-Service Version 2 Protocol.

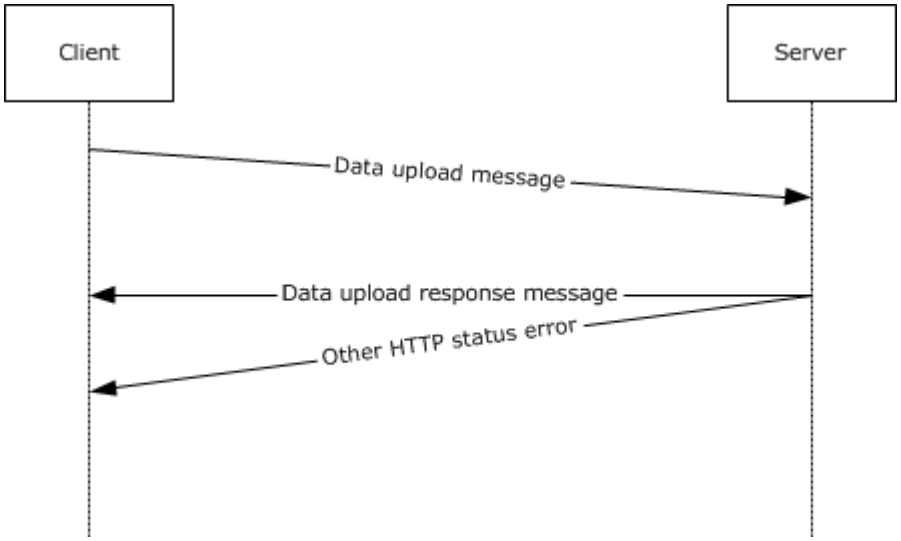


Figure 10: Server role in the SQM Client-to-Service Version 2 Protocol

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The SQM client request message is processed during the client-initiated HTTP connection. The server MUST capture the HTTP POST body. The POST body contains the message and, if it is a data upload, the binary SQM session data.

The server reads and processes the client XML request message in the POST body as follows:

1. The server reads the XML message length in the XML message header as specified in section 2.2.2.1.

2. The server parses the XML message per the length specified in the XML message header and validates the XML schema as specified in [MS-TPXS] section 5.1.

3. The server validates the message content as specified in section 2.2.2,

4. The server decompresses any compressed data as specified in section 2.2.2.11.

5. The server processes each client request in the XML message as specified in section 2.2.2.12. The request (req) contains the specific client request metadata and command.

6. The server creates a response message as specified in section 2.2.3.

7. The server creates a response command for each request command as specified in section 2.2.3.4.

3.2.5.1 Processing Client Request Upload Request

The server validates the **requpload** command as specified in section 2.2.2.16.1.<20> The server creates a valid response command and returns the command to the client.

The server creates an approved response, as specified in section 2.2.3.6.2, if the server accepts the request to send data (that is, the server is prepared to accept the data upload).

The server creates a throttle response, as specified in section 2.2.3.6.5, if the server rejects the client request to upload data. The reject decision is implementation-specific.

The server creates an error response, as specified in section 2.2.3.6.4, if the server is not prepared to accept the data but needs the client to resend the request or continue sending requests in the future. Typically this is the response if the server cannot accept requested data (for example, a partial service outage).

3.2.5.2 Processing Client Data Upload Request

The server validates the **dataupload** command as specified in section 2.2.2.16.2. The server creates a valid response command as specified in section 2.2.3.6.1 and section 2.2.3.6.4.

The server processes the SQM session data by performing the following:

1. Verifies the SQM session payload per the **size** and **offset** values as specified in section 2.2.2.16.2.

2. Processes the SQM session payload as specified in [MS-SQMCS] section 3.2.5.1, Processing a Client Message and [MS-SQMCS] section 3.2.5.3, Processing SQM Section Data – Option 2 – Uncompressed.

3.2.5.3 Processing Client Query Resource Request

The server validates the **qryrsrc** command as specified in section 2.2.2.16.3. The server creates a response command as specified in section 2.2.3.6.3 and section 2.2.3.6.6.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Proxy Details

This section specifies the proxy role in the SQM Client-to-Service Version 2 Protocol.

When a configured SQM client sends a message to the proxy that contains A SQM data, the proxy service opens the payload and adds a data point (see [MS-SQMCS] section 2.2.4.4.1) identifying the proxy. The payload is then repackaged and sent to the SQM service. All messages not containing payload information are sent by the proxy from the SQM client to the SQM server without modification.

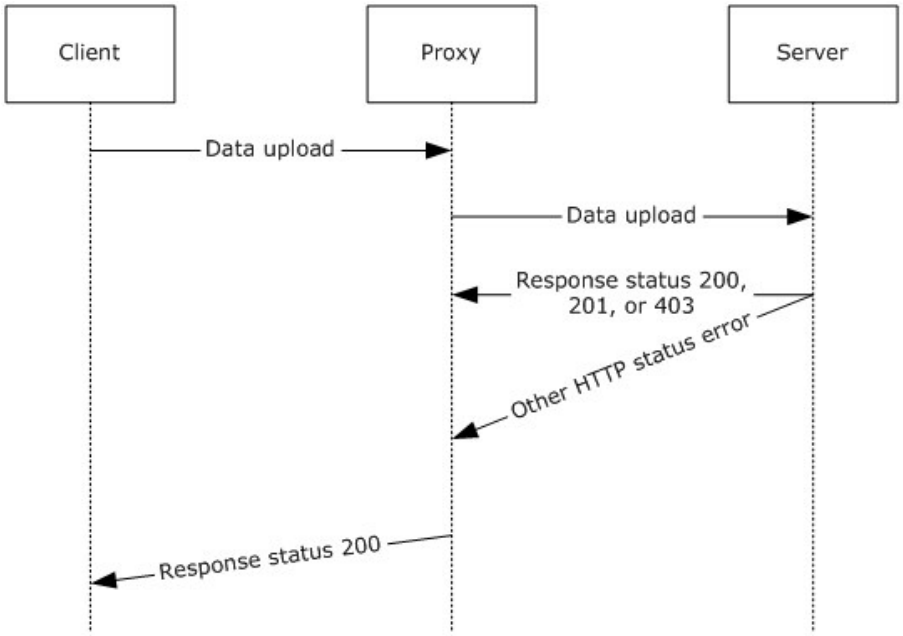


Figure 11: Client upload through a proxy

3.3.1 Abstract Data Model

The SQM protocol relay transmits protocol messages on behalf of a client in environments where the client cannot access the SQM service directly (primarily where the client is protected by the firewall). To enable the relay, a client MUST be configured to send data to the relay service by using the SQM CorporateURL registry keys (see section 3.3.3 for detailed description of registry keys).

When a configured client sends a message to the relay that contains a SQM payload, the relay service opens the payload and adds a data point identifying the relay. This data is added to the SQM data point section of the payload as described in section 2. The payload is then repackaged and sent to the SQM service. If the forwarder receives a message that does not fit the XML model for SQM, the message is forwarded directly, without modification. The ability to forward messages that do not match the XML model is necessary so that the relay can transmit valid SQM protocol messages such as A-SQM and SQM protocol headers

3.3.2 Timers

None.

3.3.3 Initialization

The proxy can be enabled by installing the Windows Feedback Forwarder. The installer for Windows Feedback Forwarder installs the SQM service binaries and configures the settings on the SQM service. Windows Feedback Forwarder contains two settings. The first setting configures the port number on which to receive SQM messages and the second setting configures proxy information so that the Windows Feedback Forwarder service can connect to the SQM service through a firewall.

The Windows Feedback Forwarder service will not relay any messages unless a client is configured to send SQM data to the relay. To enable a client to send data to the relay, the client MUST be SQM-enabled and have the Corporate URL registry keys configured. These registry keys are as follows:

The client is SQM-enabled by setting the following values:

- Registry Key: HKLM\Software\Policies\Microsoft\SQMClient\Windows\CEIPEnable
- Data type = RED_DWORD
- Value = 1 (enable)

The client is configured to send SQM data to Microsoft by setting the following values:

- Registry Key: HKLM\Software\Policies\Microsoft\SQMClient\CorporateSQMURL
- Data type = REG_SZ
- Value: http://<WindowsFeedbackForwarderServer_FQDN>:<WindowsFeedbackForwarderPortNumber>
- Example: http://MyServer.MyDomain.com:53533

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The relay receives a SQM messages by using a listening port configured as part of the setup for Windows Feedback Forwarder. The post is sent by using HTTP. If the POST contains a payload that adheres to the SQM format, the message payload is augmented with an additional data point that identifies the relay. This is an additive change only. The payload is then repackaged and sent to the SQM service using **SSL** over port 443. All other protocol messages are sent directly through the proxy without modification in a similar manner, where the first transmission from the client to the relay is communicated over HTTP and the second transmission is communicated over SSL using port 443.

To support A-SQM, if the proxy receives a message that does not adhere to a known XML format, the message is sent to the SQM service without augmenting the payload. In this scenario, the transport uses the same behavior, transmitting first using HTTP over the configured port and then using HTTPS over port 443.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

To view the full XML schema, see [MS-TPXS] section 5.

4.1 Example of a qryrsrc Request and rsrc Response

The following is an example of a **qryrsrc** request as specified in section 2.2.2.16.3

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<req ver="2">
 <tlm>
  <src>
   <desc>
    <mach>
     <os>
      <arg nm="vermaj" val="6" />
      <arg nm="vermin" val="2" />
      <arg nm="verbld" val="8061" />
      <arg nm="versp" val="0" />
      <arg nm="arch" val="0" />
      <arg nm="lcid" val="1033" />
      <arg nm="geoid" val="244" />
      <arg nm="sku" val="74" />
      <arg nm="csdbld" val="0" />
      <arg nm="prodsuite" val="256" />
      <arg nm="ntprodtype" val="1" />
      <arg nm="platid" val="2" />
      <arg nm="portos" val="0" />
     </os>
     <hw>
      <arg nm="form" val="2" />
      <arg nm="arch" val="9" />
      <arg nm="sysmfg" val="LENOVO" />
      <arg nm="syspro" val="6458A16" />
      <arg nm="bv" val="7LETB4WW (2.14 )" />
      <arg nm="ram" val="3070" />
      <arg nm="proccnt" val="2" />
      <arg nm="proclsp" val="2195" />
      <arg nm="wscpusc" val="0" />
      <arg nm="wsdsksc" val="0" />
      <arg nm="wscpudn" val="Intel(R) Core(TM)2 Duo CPU T7500 @ 2.20GHz" />
      <arg nm="wsdgsc" val="0" />
      <arg nm="aoac" val="0" />
     </hw>
     <ctrl>
      <arg nm="tm" val="129579264069372027" />
      <arg nm="mid" val="{fe166778-8e09-4bd8-b840-df6b79d40232}" />
      <arg nm="sample" val="46445977" />
     </ctrl>
    </mach>
   </desc>
  </src>
  <reqs>
   <req key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="default"></namespace>
    <cmd nm="qryrsrc">
     <arg nm="name" val="manifest" />
    </cmd>
```

```
    </req>
   </reqs>
  </tlm>
 </req>
```

The following is an example of an **rsrc** response as specified in section 2.2.3.6.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<resp ver="2">
 <tlm>
  <resps>
   <resp key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="default" />
    <cmd nm="rsrc">
     <arg nm="ver" val="10145" />
     <arg nm="path"
val="telemetry.manifests/sqm/windows/winsqm8.default.manifest/sqm10145.bin" />
    </cmd>
   </resp>
  </resps>
 </tlm>
</resp>
```

4.2 Example of a requpload Message and Approved Message Response

The following is an example of a **requpload** message as specified in section 2.2.2.16.1.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<req ver="2">
  <tlm>
    <src>
      <desc>
        <mach>
          <os>
            <arg nm="vermaj" val="6" />
            <arg nm="vermin" val="2" />
            <arg nm="verbld" val="8061" />
            <arg nm="versp" val="0" />
            <arg nm="arch" val="0" />
            <arg nm="lcid" val="1033" />
            <arg nm="geoid" val="244" />
            <arg nm="sku" val="74" />
            <arg nm="domain" val="1" />
            <arg nm="csdbld" val="0" />
            <arg nm="prodsuite" val="256" />
            <arg nm="ntprodtype" val="1" />
            <arg nm="platid" val="2" />
            <arg nm="tmsi" val="16290" />
            <arg nm="osinsty" val="3" />
            <arg nm="iever" val="9.0.8040.0-RTM" />
            <arg nm="portos" val="0" />
          </os>
          <hw>
            <arg nm="form" val="2" />
            <arg nm="arch" val="9" />
            <arg nm="sysmfg" val="LENOVO" />
```

```
            <arg nm="syspro" val="6458A16" />
            <arg nm="bv" val="7LETB4WW (2.14 )" />
            <arg nm="mrk" val="045E_FABRIKAM_OEM_RPM" />
            <arg nm="ram" val="3070" />
            <arg nm="proccnt" val="2" />
            <arg nm="proclsp" val="2195" />
            <arg nm="wscpusc" val="0" />
            <arg nm="wsdsksc" val="0" />
            <arg nm="wsdgsc" val="0" />
            <arg nm="aoac" val="0" />
          </hw>
          <ctrl>
            <arg nm="tm" val="129579283005426872" />
            <arg nm="mid" val="{FE166778-8E09-4BD8-B840-DF6B79D40232}" />
            <arg nm="sample" val="46445977" />
          </ctrl>
        </mach>
      </desc>
    </src>
    <reqs>
      <req key="1">
        <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6"></namespace>
        <ctrl>
          <arg nm="sid" val="4052" />
          <arg nm="uid" val="{2B2F5135-0075-4AB7-B3AD-6D9AE80891E4}" />
        </ctrl>
        <cmd nm="requpload"></cmd>
      </req>
      <req key="2">
        <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6"></namespace>
        <ctrl>
          <arg nm="sid" val="4052" />
          <arg nm="uid" val="{2B2F5135-0075-4AB7-B3AD-6D9AE80891E4}" />
        </ctrl>
        <cmd nm="requpload"></cmd>
      </req>
    </reqs>
  </tlm>
</req>
```

The following is an example of an **approved** message as specified in section 2.2.3.6.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<resp ver="2">
 <tlm>
  <resps>
   <resp key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="approved">
     <arg nm="token" val="3.737e6827d6765e60d6900768b36f8c84.01cc5ed08779ac68" />
     <arg nm="tokenexp" val="129582739006008424" />
    </cmd>
   </resp>
   <resp key="2">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="approved">
     <arg nm="token" val="3.737e6827d6765e60d6900768b36f8c84.01cc5ed08779ac68" />
     <arg nm="tokenexp" val="129582739006008424" />
```

```
    </cmd>
   </resp>
  </resps>
 </tlm>
</resp>
```

4.3 Example of an Upload Message and a Receipt Response

The following is an example of an upload message as specified in section 2.2.2.16.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<req ver="2">
  <tlm>
    <src>
      <desc>
        <mach>
          <os>
            <arg nm="vermaj" val="6" />
            <arg nm="vermin" val="2" />
            <arg nm="verbld" val="8061" />
            <arg nm="versp" val="0" />
            <arg nm="arch" val="0" />
            <arg nm="lcid" val="1033" />
            <arg nm="geoid" val="244" />
            <arg nm="sku" val="74" />
            <arg nm="domain" val="1" />
            <arg nm="csdbld" val="0" />
            <arg nm="prodsuite" val="256" />
            <arg nm="ntprodtype" val="1" />
            <arg nm="platid" val="2" />
            <arg nm="tmsi" val="16290" />
            <arg nm="osinsty" val="3" />
            <arg nm="iever" val="9.0.8040.0-RTM" />
            <arg nm="portos" val="0" />
          </os>
          <hw>
            <arg nm="form" val="2" />
            <arg nm="arch" val="9" />
            <arg nm="sysmfg" val="LENOVO" />
            <arg nm="syspro" val="6458A16" />
            <arg nm="bv" val="7LETB4WW (2.14 )" />
            <arg nm="mrk" val="045E_FABRIKAM_OEM_RPM" />
            <arg nm="ram" val="3070" />
            <arg nm="proccnt" val="2" />
            <arg nm="proclsp" val="2195" />
            <arg nm="wscpusc" val="0" />
            <arg nm="wsdsksc" val="0" />
            <arg nm="wscpudn" val="Intel(R) Core(TM)2 Duo CPU T7500  @ 2.20GHz" />
            <arg nm="wsdgsc" val="0" />
            <arg nm="aoac" val="0" />
          </hw>
          <ctrl>
            <arg nm="tm" val="129579283005582927" />
            <arg nm="mid" val="{FE166778-8E09-4BD8-B840-DF6B79D40232}" />
            <arg nm="sample" val="46445977" />
          </ctrl>
        </mach>
```

```
      </desc>
    </src>
    <reqs>
      <payload>
        <arg nm="size" val="2652" />
      </payload>
      <req key="1">
        <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6"></namespace>
        <ctrl>
          <arg nm="sid" val="4052" />
          <arg nm="uid" val="{2B2F5135-0075-4AB7-B3AD-6D9AE80891E4}" />
        </ctrl>
        <cmd nm="dataupload">
          <arg nm="tm" val="129579283005582927" />
          <arg nm="token" val="3.737e6827d6765e60d6900768b36f8c84.01cc5ed08779ac68" />
          <arg nm="size" val="1320" />
          <arg nm="offset" val="0" />
        </cmd>
      </req>
      <req key="2">
        <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6"></namespace>
        <ctrl>
          <arg nm="sid" val="4052" />
          <arg nm="uid" val="{2B2F5135-0075-4AB7-B3AD-6D9AE80891E4}" />
        </ctrl>
        <cmd nm="dataupload">
          <arg nm="tm" val="129579283005582927" />
          <arg nm="token" val="3.737e6827d6765e60d6900768b36f8c84.01cc5ed08779ac68" />
          <arg nm="size" val="1332" />
          <arg nm="offset" val="1320" />
        </cmd>
      </req>
    </reqs>
  </tlm>
</req>
```

For an example of a SQM session data upload as specified in section 2.2.2.16.2, see [MS-SQMCS] section 2.2.2.

The following is an example of a receipt message as specified in section 2.2.3.6.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<resp ver="2">
 <tlm>
  <resps>
   <resp key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="receipt">
     <arg nm="tm" val="129579283006476415" />
    </cmd>
   </resp>
   <resp key="2">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="receipt">
     <arg nm="tm" val="129579283006476415" />
    </cmd>
   </resp>
```

```
  </resps>
 </tlm>
</resp>
```

4.3.1 Example of an Error Response Message

The following is an example of an error response message as specified in section 2.2.3.6.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<resp ver="2">
 <tlm>
  <resps>
   <resp key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="error">
     <arg nm="retry" val="1" />
     <arg nm="code" val="4" />
    </cmd>
   </resp>
  </resps>
 </tlm>
</resp>
```

4.3.2 Example of a Throttle Response Message

The following is an example of a throttle response message as specified in section 2.2.3.6.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<resp ver="2">
 <tlm>
  <resps>
   <resp key="1">
    <namespace svc="sqm" ptr="windows" gp="winsqm8" app="6" />
    <cmd nm="throttle">
     <arg nm="period" val="30" />
     <arg nm="namespace" val="all" />
    </cmd>
   </resp>
  </resps>
 </tlm>
</resp>
```

5 Security

5.1 Security Considerations for Implementers

It is recommended that implementers use Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) as the transport for all client/server or proxy/server communication over the Internet.

5.2 Index of Security Parameters

None.

6 Appendix A: Full XML Schema

The XML schema is specified in [MS-TPXS] section 5.

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:

- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

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.

<1> Section 2.1: Microsoft SQM client and server implementations use HTTPS exclusively for client-server and proxy-server communication.

<2> Section 2.2.2.7: Windows SQM clients use the GetVersionEx function to query the operating system version values (**vermaj**, **vermin**, **verbld**, and **versp**). For information on system information functions, see [MSDN-SysInfoFuncts].

<3> Section 2.2.2.7: Windows SQM clients query the Windows registry value HKLM\System\CurrentControlSet\Control\Windows\CSDBuildNumber to specify the operating system revision value **csdbld** as described in [MSDN-RegQueryValueEx].

<4> Section 2.2.2.7: Windows SQM clients use the GetProductInfo function to query the operating system stock keeping unit (**sku**) value. See [MSDN-SysInfoFuncts].

<5> Section 2.2.2.7: Windows SQM clients use the GetSystemInfo function to query the operating system architecture (**arch**) value. See "[MSDN-SysInfoFuncts].

<6> Section 2.2.2.7: Windows SQM clients use the GetVersionEx function to query the operating system version type value (**ntprodtype**).

<7> Section 2.2.2.7: Windows SQM clients use the GetVersionEx function to query the operating system platform identifier value (**platid**).

<8> Section 2.2.2.7: Windows SQM clients use the GetVersionEx function to query the operating system product suite value (**prodsuite**).

<9> Section 2.2.2.8: Windows SQM clients uses the GetNativeSystemInfo to specify the hardware processor architecture (**arch**), as described in [MSDN-SysInfoFuncts].

<10> Section 2.2.2.8: Windows SQM clients query the Windows registry value HKLM\System\CurrentControlSet\Control\SystemInformation\BIOSVersion, using the RegQueryValueEx function, as described in [MSDN-RegQueryValueEx], to specify the BIOS version (**bv**) value.

<11> Section 2.2.2.8: Windows SQM clients query the Windows registry value HKLM\System\CurrentControlSet\Control\SystemInformation\SystemProductName, by using the RegQueryValueEx function, as described in [MSDN-RegQueryValueEx], to specify the system manufacturer (**syspro**) value.

<12> Section 2.2.2.8: Windows SQM clients use the PowerDeterminePlatformRole function, as described in [MSDN-PwrMgmtFunts], to specify the hardware form factor (**form**).

<13> Section 2.2.2.8: Windows SQM clients use the CallNtPowerInformation function, as described in [MSDN-PwrMgmtFunts], to specify the always connected value (**aoac**).

<14> Section 2.2.2.8: Windows SQM client queries the Windows registry value HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinSAT, by using the RegQueryValueEx function as described in [MSDN-WinSysAssmntTool] to specify the WinSAT values.

<15> Section 2.2.2.9: Windows SQM clients use the UuidCreate function as described [MSDN-RPCF] to create a one-time client machine identifier (**mid**) value. This value is stored on the client registry at the location of: HKLM\Software\Microsoft\SQMClient\MachineId, and is specified as the **mid** value in every XML message.

<16> Section 2.2.2.9: Windows SQM client uses the CryptGenRandom function, as described in [MSDN-CryptgrpFunts], to create a one-time random number (**sample**) value in the range [1, 10^8]. This value is stored on the client and specified as the **sample** value in every XML message.

<17> Section 2.2.2.11: Windows SQM clients use cabinet compression based on Microsoft Cabinet Format as described in [MSDN-CAB].

<18> Section 3.1.5.1: Windows SQM servers accept an XML message with a maximum length of 1 MB (2^{20}).

<19> Section 3.1.5.3.1: Windows SQM clients limit the length of the SQM session data to a maximum length of 20 MB.

<20> Section 3.2.5.1: The Windows SQM service determines whether to accept or throttle a request upload message by evaluating the data in the message and applying a set of rules to determine the response.

8 Change Tracking

This section identifies changes that were made to the [MS-SQMCS2] protocol document between the January 2013 and August 2013 releases. 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.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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 language and 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 or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- 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 protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
7 Appendix B: Product Behavior	Modified this section to add references to Windows 8.1 and Windows Server 2012 R2.	Y	Content updated.

9 Index

Q

R

S

T

U

V

X