[MS-ODASM]:
Open Data (OData) Server Management Protocol Extensions

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	2.0	Major	Significantly changed the technical content.
10/25/2012	3.0	Major	Significantly changed the technical content.
01/31/2013	4.0	Major	Significantly changed the technical content.

Contents

1 Introduction

This document specifies the Open Data (OData) Extensions for Server Management, which extend the Open Data (OData) Protocol defined in [MS-ODATA].

Open Data (OData) Extensions for Server Management extend the OData Protocol by defining specific resources that allow a client to send **pipelines** of **commands** to a server system over a network for execution by the server. OData Extensions for Server Management also specify limits on certain OData Protocol concepts that are more stringent than the limits specified by the OData Protocol itself.

A server supporting OData Extensions for Server Management may expose arbitrary OData Protocol resources, or the pipeline resources defined in this document, or both.

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]:

GUID
URI
URL
UTC (Coordinated Universal Time)
XML
XML namespace

The following terms are specific to this document:

command: An executable entity.

entity: An instance of an EntityType, as specified in [MC-CSDL].

invocation: A single attempt to execute a command or pipeline.

pipeline: An ordered collection of commands with the output of one command passed as the input to the next.

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.

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.

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

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MSFT-POWERSHELL] Microsoft Corporation, "Windows PowerShell Language Specification Version 2.0", http://www.microsoft.com/download/en/details.aspx?id=9706

[MS-NRTP] Microsoft Corporation, ".NET Remoting: Core Protocol".

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

[MS-PSRP] Microsoft Corporation, "PowerShell Remoting Protocol".

[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

[RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", STD 63, RFC 3629, November 2003, http://www.ietf.org/rfc/rfc3629.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC5023] Gregorio, J. Ed., and de hOra, B., Ed., "The Atom Publishing Protocol", RFC 5023, October 2007, http://www.ietf.org/rfc/rfc5023.txt

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

[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.S., 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., and Malhotra, A., Eds., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[MSDN-EventTracing] Microsoft Corporation, "Event Tracing", http://msdn.microsoft.com/en-us/library/bb968803(VS.85).aspx

[MSDN-PSWSSDG] Microsoft Corporation, "PowerShell Web Services Schema Designer and Guide", http://archive.msdn.microsoft.com/mgmtODataWebServ/Release/ProjectReleases.aspx?ReleaseId=5735

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

1.3 Overview

The OData Protocol is used to access REST-based data services over the HTTP transport.

In some configurations, a client's OData request may not be directly routable to the OData server. Instead, the request is set to a front-end server that forwards the request to the correct OData server. In this case, all entity references in the client request and the server response must be relative to a publically addressable OData service root rather than the innate service root of the OData server itself.

OData Extensions for Server Management defines a new public-server-uri HTTP header to facilitate this situation. When a front-end server forwards a client request, it specifies the public service root in the header. The OData server will interpret entity references in the request as being relative to the public service root. Similarly, entity references in the server response will be relative to the public service root.

OData Extensions for Server Management defines two OData Protocol resource sets related to pipeline execution: **CommandDescriptions** and **CommandInvocations**.

The **CommandDescriptions** resource set represents the collection of commands available on the server. By enumerating the resource set, a client can discover the commands that it is allowed to execute and their parameters. At a high level, if a client sends the following request:

GET http://servername/CommandDescriptions

Then the server might reply with the following information:

Entity 1: command "Get-Process"

Argument 1: "-Id" accepting an argument of type integer

Argument 2: "-Name" accepting an argument of type string

Entity 2: "Sort-Object"

This indicates that the client is allowed to execute the *Get-Process* and *Sort-Object* commands.

The **CommandInvocations** resource set represents the collection of commands or pipelines that have been invoked on the server. Each **entity** in the collection represents a single **invocation** of some pipeline. To invoke a pipeline, the client sends a POST request containing a new entity. The client specifies the pipeline itself (as a string), the desired output format (typically **XML** or JSON), and the length of time to wait synchronously for the command to complete. A pipeline string is a sequence of one or more commands, optionally with parameters and delimited by a vertical bar character. For example, if the server receives the pipeline string "Get-Process –Name svchost | Sort-Object", then it will execute the *Get-Process* command (with optional parameter Name set to "svchost"), send the output to the "Sort-Object" command, and finally format the output of *Sort-Object* in the syntax specified by client.

The server begins executing the pipeline when it receives the request. If the pipeline completes quickly (within the synchronous-wait time), then the server stores the output in the entity, marks the invocation status as "complete", and returns the completed entity to the client. The server does not keep a copy of the entity data.

Conversely, if the synchronous-wait time expires while the command is executing, then the server marks the entity as "still executing" and returns it to the client. In this case, the client must periodically request the updated entity from the server; once the retrieved entity's status is "complete", then the pipeline has completed and the client can inspect its output. The client should

then send an ODATA DeleteEntity request, allowing the server to delete resources associated with the pipeline.

1.4 Relationship to Other Protocols

The following figure illustrates the relationship of this protocol to other protocols.

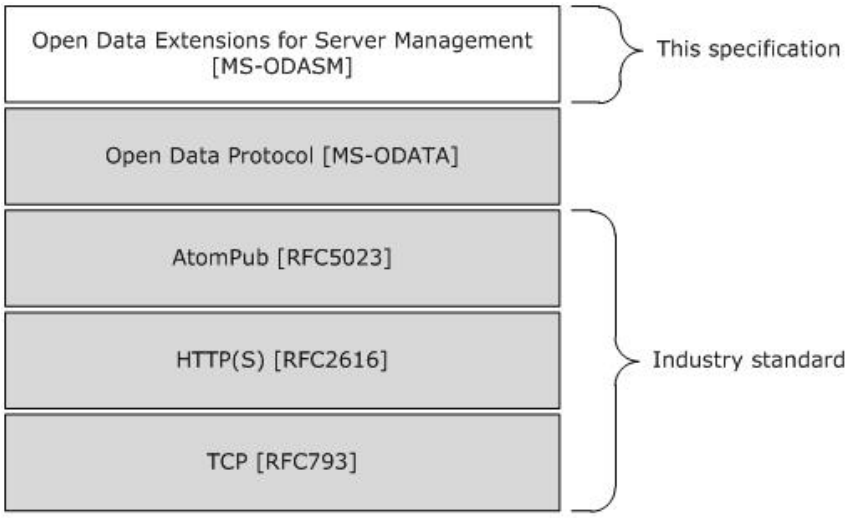

Figure 1: Protocols related to the OData Extensions for Server Management

This protocol uses TCP [RFC793] as its transport.

1.5 Prerequisites/Preconditions

The protocol defined in this document does not provide a mechanism for a client to discover the existence and location of arbitrary data services (of the server). It is a prerequisite that the client obtain a **URI** to the server before the protocol can be used.

Neither the protocol defined in this document nor its base protocols define an authentication or authorization scheme. Implementers of this protocol should review the recommended security prerequisites in Security Considerations for Implementers (section 5.1) of this document and in [RFC5023] section 15.

1.6 Applicability Statement

This protocol defines two categories of functionality. First, it defines a profile of the OData Protocol, which is appropriate for exposing a wide variety of server data as structured RESTful resources. This profile is applicable to both Internet and intranet client-server scenarios.

Second, the protocol defines specific resource types that can be used to execute a pipeline of chained commands on the server in a generic and asynchronous way. This usage is appropriate in both Internet and intranet scenarios. It does not provide a reverse channel server to client, making it inappropriate for commands requiring actions such as prompting and confirmation.

1.7 Versioning and Capability Negotiation

This protocol does not provide any mechanism for capability negotiation beyond that specified in [MS-ODATA] section 1.7.

1.8 Vendor-Extensible Fields

This protocol does not provide any vendor-extensible fields beyond those specified in [MS-ODATA] section 1.8.

1.9 Standards Assignments

This protocol has not been assigned any standard parameters.

2 Messages

2.1 Transport

OData Extensions for Server Management MUST be encoded by the Open Data Protocol as described in [MS-ODATA]. OData Extensions for Server Management does not define a specific port, and provides no mechanism for a client to discover which transport(s) and port number(s) are supported by a server.

2.2 Message Syntax

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 with each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and is not significant for interoperability.

Prefix	Namespace URI	Reference
Odasm	http://schemas.microsoft.com/powershell-web-services/2010/09	

2.2.2 Custom HTTP Headers

The messages exchanged in this protocol use the following HTTP headers in addition to the existing set of standard HTTP headers.

Header	Description
client-request-id	A GUID. The contents SHOULD uniquely identify a particular client request in order to allow correlation of the message with implementation-specific activity that occurs on the server.
request-id	A GUID. The contents SHOULD uniquely identify a particular server response in order to allow correlation of the message with implementation-specific activity that occurs on the client.
public-server-uri	A URL that specifies the desired OData service root.

2.2.2.1 client-request-id

Any client request may include the **client-request-id** header to aid in correlating a client activity with a server activity.

The format for the **client-request-id** header is a CurlyBraceGuidString as defined in section [MS-DTYP] section 2.3.4.3.

2.2.2.2 request-id

Any server response MAY include the **request-id** header to aid in correlating a client activity with a server activity.

The format for the **request-id** header is a CurlyBraceGuidString as defined in [MS-DTYP] section 2.3.4.3.

2.2.2.3 public-server-uri

Any request message MAY include the **public-server-uri** header. A response message MUST NOT include this header.

The format for the **public-server-uri** header is a URL. The protocol MUST be either "http" or "https".

2.2.3 Complex Types

The following table summarizes the set of complex type definitions included in this specification. Types are defined in the CSDL notation used by the OData protocol. See section 3.1.5 for the resource sets that use these types.

Complex type	Description
CommandDescription	An entity representing a server command that can be executed.
CommandInvocation	An entity representing a single invocation of a pipeline of one or more server commands.
CommandParameter	A description of a single parameter that can be passed to a server command.
ErrorCategoryInfo	Describes the general category and the source of an error.
ErrorDetails	Describes the details of an error and the recommended response.
ErrorRecord	Provides full details of an error that occurred during invocation of a pipeline.

2.2.3.1 CommandDescription

The **CommandDescription** resource type represents a server command that can be executed, either by itself or as part of a pipeline.

Schema namespace:

```
<EntityType Name="CommandDescription">
  <Key>
    <PropertyRef Name="Name"/>
  </Key>
  <Property Name="Name" Type="Edm.String" Nullable="false"/>
  <Property Name="HelpUrl" Type="Edm.String" Nullable="true"/>
  <Property Name="AliasedCommand" Type="Edm.String" Nullable="true"/>
  <Property Name="Parameters" Type="MultiValue" Nullable="false">
    <TypeRef Type="PowerShell.CommandParameter" Nullable="false" />
  </Property>
</EntityType>
```

Name: The name of the command.

HelpUrl: A link to a **URL** with a human-readable description of the command, or NULL if no such URL is provided.

AliasedCommand: If this command is an alias of some other command, then the name of the underlying command; otherwise NULL.

Parameters: A list of parameters that can be passed to the command.

2.2.3.2 CommandInvocation

The **CommandInvocation** resource type represents a single invocation of a command or pipeline.

```
<EntityType Name="CommandInvocation">
  <Key>
     <PropertyRef Name="ID"/>
  </Key>
  <Property Name="ID" Type="Edm.GUID" Nullable="false"/>
  <Property Name="Command" Type="Edm.String" Nullable="true" />
  <Property Name="Status" Type=" Edm.String" Nullable="true"/>
  <Property Name="OutputFormat" Type="Edm.String" Nullable="true" />
  <Property Name="Output" Type="Edm.String" Nullable="true"/>
  <Property Name="Errors" Type="MultiValue" Nullable="false">
     <TypeRef Type="PowerShell.ErrorRecord" Nullable="false" />
  </Property>
  <Property Name="ExpirationTime" Type="Edm.DateTime" Nullable="false"/>
  <Property Name="WaitMsec" Type="Edm.Int32" Nullable="false"/>
</EntityType>
```

ID: A **GUID** that serves as an identifier for the specific invocation. It MUST be unique within the scope of the server endpoint.

Command: One or more commands to execute in a pipeline, expressed as a *pipeline* statement as defined in section 8.2 of [MSFT-POWERSHELL].

OutputFormat: The requested output format. The server SHOULD support the following values:

- "xml": represents the MIME type "application/xml"
- "json": represents the MIME type "application/json"

Status: A string that represents whether the command has completed. It MUST be one of the following values:

- "EXECUTING": The command is in progress.
- "SUCCESS": The command has completed with no errors.
- "ERROR": The command has completed with errors.

Output: A reference to a media entry that contains the output of the command, in the format listed in OutputFormat. This link MUST be NULL if Status is EXECUTING.

Errors: A sequence of ErrorRecord objects describing errors generated by the command. This sequence MUST be empty if Status is not ERROR.

ExpirationTime: **UTC** time when the command will be deleted by the server.

WaitMsec: Number of milliseconds for the server to wait synchronously for command completion. See section 3.1.5.1.2 for details.

2.2.3.3 CommandParameter

The **CommandParameter** resource type represents a single parameter to a command.

```
<ComplexType Name="CommandParameter">
  <Key>
    <PropertyRef Name="Name"/>
  </Key>
  <Property Name="Name" Type="Edm.String" Nullable="true"/>
  <Property Name="ParameterType" Type="Edm.String" Nullable="true"/>
</ComplexType>
```

Name: The name of the parameter.

ParameterType: The type of argument that the parameter accepts, or NULL if none. The format of the type is defined by the **TypeName** production in [MS-NRTP] section 2.2.5. Note that all parameter arguments of an invocation request are received as strings.

2.2.3.4 ErrorCategoryInfo

The **ErrorCategoryInfo** type describes the general category and the source of an error.

```
<ComplexType Name="ErrorCategoryInfo">
  <Property Name="Activity" Type="String" />
  <Property Name="Category" Type="String" />
  <Property Name="Reason" Type="String" />
  <Property Name="TargetName" Type="String" />
  <Property Name="TargetType" Type="String" />
</ComplexType>
```

Activity: A string that describes the activity being performed when the error occurred, which SHOULD be the name of the specific command that was being executed.

Category: A string that describes the category of error that was encountered, which SHOULD be one of the string values from [MS-PSRP] section 2.2.3.9.

Reason: An optional string that describes the cause of the error.

TargetName: An optional string describing the object upon which the activity has operated.

TargetType: An optional string describing the type of the object upon which the activity has operated.

2.2.3.5 ErrorDetails

The **ErrorDetails** type contains details of an error that occurred during command invocation, and guidance for how the client should respond.

```
<ComplexType Name="ErrorDetails">
  <Property Name="Message" Type="String" />
  <Property Name="RecommendedAction" Type="String"/>
</ComplexType>
```

Message: An optional string that SHOULD explain the meaning of the related error.

RecommendedAction: An optional string that SHOULD describe the recommended action to take as a result of the related error.

2.2.3.6 ErrorRecord

The **ErrorRecord** type represents an error that was generated by invocation of a command or pipeline.

```
<ComplexType Name="ErrorRecord">
  <Property Name="FullyQualifiedErrorId" type="String" />
  <Property Name="CategoryInfo" Type="PowerShell.ErrorCategoryInfo" />
  <Property Name="Details" Type="PowerShell.ErrorDetails" />
  <Property Name="Exception" Type="String" />
</ComplexType>
```

CategoryInfo: An ErrorCategoryInfo describing the location and category of the error.

Details: An ErrorDetails that gives additional guidance about the error.

Exception: An optional string that describes the error that occurred.

FullyQualifiedErrorId: A string that represents the specific type of error that was generated. It MUST conform to the following ABNF:

ErrorType = ErrorId ["," CommandTypeName]

ErrorId = IDENTIFIER

CommandTypeName = IDENTIFIER *("." IDENTIFIER)

IDENTIFIER = ALPHA *(ALPHA / DIGIT / "_")

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization 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 this document.

OData Extensions for Server Management incorporate the OData Extensions for Server Management data model described in [MS-ODATA] section 2.2.1. It extends the model with the following elements.

CommandsEnabled: A Boolean that specifies whether the **CommandInvocations** and **CommandDescriptions** EntitySets (as specified in [MC-CSDL] section 2.1.18) are exposed (see section 3.1.5).

MaxWaitMsec: Maximum allowed value for the **CommandInvocation.WaitMsec** property.

DefaultWaitMsec: Inferred value for the **CommandInvocation.WaitMsec** property when it is omitted from a client request.

Command Description Table: A table of **CommandDescription** entities, with one entry for each command available on the server.

Command Invocation Table: A table of active and completed command invocations, where each entry contains all properties of the **CommandInvocation** entity.

3.1.2 Timers

Invocation Expiration Timer: A periodic timer that checks for stale invocations and deletes them. The default interval is 10 seconds; the interval may be any nonzero value.

Async Response Timer : A timer that governs whether the server returns the results of a pipeline invocation synchronously or asynchronously. The interval may be any non-negative value; the default is zero.

3.1.3 Initialization

The server MUST initialize the ODATA server role.

If the **CommandEnabled** flag is true, then the server MUST:

- Initialize the Command Descriptions Table in an implementation-dependent way.<1>
- Set the Command Invocation Table to an empty table.
- Start the Invocation Sweeper timer.
- Add the CommandInvocations and CommandDescriptions EntitySets (as specified in [MC-CSDL] section 2.1.18) to the ODATA server role.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Pipeline Finishes Executing

When a command or pipeline finishes executing, the higher layer provides the output data (in the client's requested format) and/or error records generated by the pipeline.

The server MUST locate the entry of the **Command Invocation Table** that is associated with the command by implementation-specific means.

If the command generated any output data, the server MUST store it in the entry's **OutputData** element.

If the command generated any error records, the server MUST store them in the entry's **ErrorRecords** element.

If **ErrorRecords** is non-null, then the server MUST set **Status** to "ERROR"; otherwise, the server MUST set **Status** to "COMPLETED".

3.1.5 Message Processing Events and Sequencing Rules

Resource	Description
http://server/CommandInvocations	Collection of **CommandInvocation** entities read from the **Command Invocation Table**, each representing a single invocation of a command or pipeline of commands. The available commands and parameters can be obtained from http://server/CommandDescriptions.
http://server/CommandDescriptions	Collection of **CommandDescription** entities, each representing a command that can be executed via http://server/CommandInvocations.

The server SHOULD include the `client-request-id` header in all responses.<2>

The server SHOULD support OData Protocol batch requests as specified in [MS-ODATA] section 3.2.5.8.

The server SHOULD limit the complexity of "$expand" expressions, as defined in [MS-ODATA] section 2.2.3.6.1.3.<3>

If a request includes the **public-server-uri** header, the server MUST verify that the header value is a well-formed HTTP URL prior to processing the body of the request. If the header value is not a well-formed HTTP URL, the server MUST ignore the header and continue processing.

If the header is a well-formed HTTP URL, the server MUST construct an alternate OData service root by replacing the scheme, host, and port components of its own service root with the corresponding values from the **public-server-uri** header. The server MUST interpret all resource paths in the request using the alternate service root, not the server's innate service root. Similarly, all resource paths in the OData response MUST be relative to the alternate service root.

The responses to all the operations can result in the following status codes in addition to the codes defined in [MS-ODATA] section 3.2.8.

Status code	Description
403	The operation was forbidden because the client does not have the necessary permissions.

3.1.5.1 http://server/CommandInvocations

This URI is a collection of **CommandInvocation** entities, each representing a single invocation of a command or command pipeline. Section 3.1.5.2 defines the format of a single instance of the collection.

The following operations are allowed to be performed on this resource.

Operation	Description
GET	Enumerates the set of running and completed pipeline instances.
POST	Creates a new pipeline instance.

3.1.5.1.1 GET

The GET operation is an OData Protocol RetrieveEntitySet operation as defined in [MS-ODATA] section 2.2.7.2.1.

To construct the relevant set of response entities, the server MUST begin with a list of all rows of the **Command Invocation Table**. If the server implements access controls, it MUST remove inaccessible entries from the list before applying any OData Protocol query options.<4>

The properties of each response entity MUST be initialized as specified in section 3.1.5.2.1.

3.1.5.1.2 POST

The POST operation is an OData Protocol InsertEntity operation as defined in [MS-ODATA] section 2.2.7.1.1.

The server SHOULD validate that the **Command** property conforms to the syntax in section 2.2.3.2. If not, the server MUST return an OData Protocol "top-level" error response as defined in [MS-ODATA] section 2.2.8.1.

The server MUST validate that **OutputFormat** is a format supported by local commands, returning an OData Protocol "top-level" error response if not.

The server MUST then create a row in the **Command Invocation Table**, setting its properties as follows:

Property	Initial value
ID	A newly generated GUID.
Command	The request entity's **Command** property.
WaitMsec	The request entity's **WaitMsec** property, if present; otherwise, **DefaultWaitMsec**. If the value is greater than **MaxWaitMsec**, replace it with **MaxWaitMsec**.
Status	"EXECUTING".

Property	Initial value
OutputFormat	The request entity's **OutputFormat** property.
Output	Null.
Errors	Null.
ExpirationTime	Current UTC time plus **MaxCommandDuration.**

The server MUST then start the command or pipeline specified in **Command**, specifying the requested output format in an implementation-dependent way.<5> The server MUST create a new **Async Response Timer** with an interval of **WaitMsec** milliseconds, start the timer, and wait for either the timer or the pipeline to complete:

- If the command fails to start, the server MUST set Status to "ERROR" and set Errors to one or more error records describing the reason.
- If the pipeline completed prior to the timer expiration, the server MUST set Output to the command's output and then examine the command's error records. If the command generated one or more error records, the server MUST set Errors to the list of errors and set Status to "ERROR"; otherwise, it MUST set Status to "COMPLETED".
- If the timer expired before the pipeline completed, the command continues to execute and no properties are modified.

Finally, the server MUST populate the OData Protocol response entity in accordance with the OData Protocol request's query options, with the values of the requested entity properties taken from the row properties.

3.1.5.2 http://server/CommandInvocations(<invocationId>)

This URI refers to a single **CommandInvocation** entity.

The URI parameters are defined by the following ABNF:

`invocationId = CurlyBraceGuidString` as defined in [MS-DTYP] section 2.3.4.3.

invocationId: The unique identifier of the invocation.

The following operations are allowed to be performed on this resource.

Operation	Description
GET	Retrieves the invocation instance.
DELETE	Deletes the invocation instance.

3.1.5.2.1 GET

This operation is an OData Protocol RetrieveEntity request as defined in [MS-ODATA] section 2.2.7.2.2.

The server MUST locate a row in **Command Invocation Table** whose **ID** field matches ***invocationId.*** The server MUST use the data in this row to populate all OData Protocol entity properties returned to the client.

If no row matches, or access controls prevent the client from accessing the row, then the request URI does not identify a valid resource; the server MUST respond as described in [MS-ODATA] section 3.2.5.4 and terminate processing of the request.

3.1.5.2.2 DELETE

This operation is an ODATA DeleteEntity request as defined in section [MS-ODATA] section 2.2.7.4.1.

The server MUST locate a row in **Command Invocation Table** whose **ID** field matches ***invocationId.***

If no row matches, or access controls prevent the client from accessing the row, then the request URI does not identify a valid resource; the server MUST respond as described in [MS-ODATA] section 3.2.5.4 and terminate processing of the request.

If a row matches, the server MUST interrupt the command to which **Handle** refers and then delete the row.

3.1.5.3 http://server/CommandDescriptions

This URI is a collection of **CommandDescription** entities, each representing a single executable command and the parameters it accepts. Such commands can be used in the **Command** field of a **CommandInvocation** instance. Section 3.1.5.4 describes the format of a single instance.

The following operations are allowed to be performed on this resource.

Operation	Description
GET	Enumerates the set of instances.

3.1.5.3.1 GET

The GET operation is an OData Protocol RetrieveEntitySet operation as defined in [MS-ODATA] section 2.2.7.2.1.

To construct the relevant set of response entities, the server MUST determine the set of commands that are accessible to the client, in an implementation-dependent way.<6>

3.1.5.4 http://server/CommandDescriptions(<commandName>)

This URI refers to a single **CommandDescription** entity.

The URI parameters are defined by the following ABNF:

`commandName` = A percent-encoded UTF-8 character sequence (see [RFC3986] Section 2.1 for the definition of percent encoding, and [RFC3629] for the definition of the UTF-8 encoding).

commandName: The name of the command.

The following operations are allowed to be performed on this resource.

Operation	Description
GET	Retrieves the instance.

3.1.5.4.1 GET

This operation is an OData Protocol RetrieveEntity request as defined in [MS-ODATA] section 2.2.7.2.2.

The server MUST locate a row in **Command Description Table** whose **Name** field matches ***commandName.*** The server MUST use the data in this row to populate all ODATA entity properties returned to the client.

If no row matches, or the client's network identity is not allowed to execute the command, then the request URI does not identify a valid resource; the server MUST respond as described in [MS-ODATA] section 3.2.5.4 and terminate processing of the request.

3.1.6 Timer Events

3.1.6.1 Invocation Expiration Timer

When the timer expires, the server MUST examine each row of the **Command Invocation Table.** For each entry whose **ExpirationTime** is earlier than the current time, the server MUST delete the entry and SHOULD cancel the command in progress. Then the server MUST start the timer again.

3.1.6.2 Async Response Timer

When the timer expires, the server must allow the algorithm in section 3.1.5.1.2 to continue.

3.1.7 Other Local Events

The server role defines no additional events.

4 Protocol Examples

4.1 Invocation, Finishing Synchronously, with Output

In this example, the client sends the command,"Get-Process -Name svchost | select-object -property ID,Handles" by using JSON encoding to the server hosted at http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc, which is protected by Basic authentication.

The client sends a POST request with the JSON-encoded entity:

```
- Http: Request, POST
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations,
    Query:$format=json, Using Basic Authorization
    Command: POST
  + URI: /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations?$format=json
    ProtocolVersion: HTTP/1.1
    UserAgent:  Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US)
    WindowsPowerShell/3.0
  + ContentType:  application/json
  + Authorization: Basic
    Host:  jroberts26v:8000
    ContentLength:  140
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json
     HTTPPayloadLine: {
     HTTPPayloadLine:     "OutputFormat":  "json",
     HTTPPayloadLine:     "Command":  "Get-Process -Name svchost |
     select-object -property ID,Handles",
     HTTPPayloadLine:     "WaitMsec":  7000
     HTTPPayloadLine: }
```

The server executes the command and returns the updated entity in JSON format:

```
- Http: Response, HTTP/1.1, Status: Created, URL:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    ProtocolVersion: HTTP/1.1
    StatusCode: 201, Created
    Reason: Created
    Cache-Control:  no-cache
    ContentLength:  1385
  + ContentType:  application/json;charset=utf-8
    Location:  http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc/
    CommandInvocations(guid'3e7ddd11-1b62-4472-a968-d79e81ccbae1')
    Server:  Microsoft-IIS/8.0
    X-Content-Type-Options:  nosniff
    request-id:  f29fce11-7fb0-0000-d319-a3f2b07fcc01
    DataServiceVersion:  3.0;
    XAspNetVersion:  4.0.30319
    XPoweredBy:  ASP.NET
    Date:  Mon, 10 Oct 2011 20:50:03 GMT
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json;charset=utf-8
{"d":{"__metadata":{"id":"http://jroberts26v:8000/SampleApp/Microsoft.Management.
    Odata.svc/CommandInvocations(guid'
    3e7ddd11-1b62-4472-a968-d79e81ccbae1')","uri":"http://jroberts26v:8000
```

```
/SampleApp/Microsoft.Management.Odata.svc/CommandInvocations(
guid'3e7ddd11-1b62-4472-a968-d79e81ccbae1')","type":"PowerShell.
CommandInvocation"},"ID":"3e7ddd11-1b62-4472-a968-d79e81ccbae1",
"Command":"Get-Process -Name svchost | select-object -property
ID,Handles","Status":"Completed","OutputFormat":"json","Output":"{\r\n
\"Id\":  484,\r\n    \"Handles\":  1026\r\n}{\r\n    \"Id\":  704,\r\n
\"Handles\":  923\r\n}{\r\n    \"Id\":  820,\r\n    \"Handles\":  378\r\n}
{\r\n    \"Id\":  880,\r\n    \"Handles\":  370\r\n}{\r\n    \"Id\":  948,
\r\n    \"Handles\":  960\r\n}{\r\n    \"Id\":  1120,\r\n    \"Handles\":
516\r\n}{\r\n    \"Id\":  1380,\r\n    \"Handles\":  150\r\n}{\r\n    \"Id\":
1436,\r\n    \"Handles\":  126\r\n}{\r\n  \"Id\":  1472,\r\n    \"Handles\":
72\r\n}{\r\n    \"Id\":  1572,\r\n    \"Handles\":  347\r\n}{\r\n    \"Id\":
1596,\r\n    \"Handles\":  631\r\n}{\r\n    \"Id\":  2328,\r\n    \"Handles\":
178\r\n}{\r\n    \"Id\":  3000,\r\n    \"Handles\":  1395\r\n}{\r\n    \"Id\":
3600,\r\n    \"Handles\":
241\r\n}","Errors":{"__metadata":{"type":"MultiValue(PowerShell.
ErrorRecord)"},"results":[]},"ExpirationTime":"\/Date(1318280409333)\/",
"WaitMsec":5000}}
```

Since the Status is Completed, the command has completed without errors.

4.2 Invocation, Finishing Synchronously, with Errors

In this example, the client sends the command, "Get-Process -Name svchost | incorrect-object -property ID,Handles" by using JSON encoding to the server hosted at http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc, which is protected by Basic authentication. The server does not implement the command "incorrect-object".

The client sends a POST request with the JSON-encoded entity:

```
- Http: Request, POST /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations,
    Query:$format=json, Using Basic Authorization
    Command: POST
  + URI: /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations?$format=json
    ProtocolVersion: HTTP/1.1
    UserAgent:  Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US)
    WindowsPowerShell/3.0
  + ContentType:  application/json
  + Authorization: Basic
    Host:  jroberts26v:8000
    ContentLength:  143
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json
     HTTPPayloadLine: {
     HTTPPayloadLine:     "OutputFormat":  "json",
     HTTPPayloadLine:     "Command":  "Get-Process -Name svchost |
     incorrect-object -property ID,Handles",
     HTTPPayloadLine:     "WaitMsec":  7000
     HTTPPayloadLine: }
```

The server generates an error because of the unimplemented command, and returns the updated entity in JSON format:

```
- Http: Response, HTTP/1.1, Status: Created, URL:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
```

```
    ProtocolVersion: HTTP/1.1
    StatusCode: 201, Created
    Reason: Created
    Cache-Control:  no-cache
    ContentLength:  1317
  + ContentType:  application/json;charset=utf-8
    Location:
    http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.
    svc/CommandInvocations(guid'4dea8d5a-9544-41d2-a5f9-84a6073a0ccf')
    Server:  Microsoft-IIS/8.0
    X-Content-Type-Options:  nosniff
    request-id:  f29fce11-7fb0-0000-e019-a3f2b07fcc01
    DataServiceVersion:  3.0;
    XAspNetVersion:  4.0.30319
    XPoweredBy:  ASP.NET
    Date:  Mon, 10 Oct 2011 21:04:24 GMT
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json;charset=utf-8
{"d":{"__metadata":{"id":"http://jroberts26v:8000/SampleApp/Microsoft.Management.
    Odata.svc/CommandInvocations(guid'4dea8d5a-9544-41d2-a5f9-
    84a6073a0ccf')","uri":"http://jroberts26v:8000/SampleApp/
    Microsoft.Management.Odata.svc/CommandInvocations
    (guid'4dea8d5a-9544-41d2-a5f9-84a6073a0ccf')","type":"PowerShell.
    CommandInvocation"},"ID":"4dea8d5a-9544-41d2-a5f9-84a6073a0ccf",
    "Command":"Get-Process -Name svchost | incorrect-object -property
    ID,Handles","Status":"Error","OutputFormat":"json","Output":null,
    "Errors":{"__metadata":{"type":"MultiValue(PowerShell.
    ErrorRecord)"},"results":[{"FullyQualifiedErrorId":
    "CommandNotFoundException","CategoryInfo":{"__metadata":
    {"type":"PowerShell.ErrorCategoryInfo"},"Activity":"",
    "Category":"ObjectNotFound","Reason":
    "ParentContainsErrorRecordException","TargetName":
    "incorrect-object","TargetType":"String"},"ErrorDetails":{
    "__metadata":{"type":"PowerShell.ErrorDetails"},
    "Message":null,"RecommendedAction":null},"Exception":
    "System.Management.Automation.ParentContainsErrorRecordException:
    The term 'incorrect-object' is not recognized as the name of a
    cmdlet, function, script file, or operable program. Check the spelling
    of the name, or if a path was included, verify that the path is correct
    and try again."}]},"ExpirationTime":"\/Date(1318281269683)\/","WaitMsec":
    5000}}
```

Since the Status is Error, the command has completed with errors. The Errors array contains a single error:

```
"Errors" :{
"__metadata":{"type":"MultiValue(PowerShell.ErrorRecord)"},
"results":
[
{
    "FullyQualifiedErrorId":"CommandNotFoundException",
    "CategoryInfo":
    {
        "__metadata":{"type":"PowerShell.ErrorCategoryInfo"},
        "Activity":"",
        "Category":"ObjectNotFound",
        "Reason":"ParentContainsErrorRecordException",
```

```
        "TargetName":"incorrect-object",
        "TargetType":"String"
    },
    "ErrorDetails":
    {
    "__metadata":{"type":"PowerShell.ErrorDetails"},
    "Message":null,
    "RecommendedAction":null
    },
    "Exception":"System.Management.Automation.ParentContainsErrorRecordException:
        The term 'incorrect-object' is not recognized as the name of a cmdlet,
        function, script file, or operable program. Check the spelling of the name,
         or if a path was included, verify that the path is correct and try again."
}
]
```

4.3 Invocation, Finishing Async; Client Polls Until Complete

In this example, the client sends the command, "Start-Sleep 20" to the server hosted at http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc, which is protected by Basic authentication.

The client sends a POST with the JSON-encoded entity:

```
- Http: Request, POST
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations,
    Query:$format=json, Using Basic Authorization
    Command: POST
  + URI: /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations?$format=json
    ProtocolVersion: HTTP/1.1
    UserAgent:  Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US)
    WindowsPowerShell/3.0
  + ContentType:  application/json
  + Authorization: Basic
    Host:  jroberts26v:8000
    ContentLength:  92
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json
     HTTPPayloadLine: {
     HTTPPayloadLine:     "OutputFormat":  "json",
     HTTPPayloadLine:     "Command":  "Start-Sleep 20",
     HTTPPayloadLine:     "WaitMsec":  7000
     HTTPPayloadLine: }
```

The "Start-Sleep 20" command will take 20 seconds to complete. The server enforces a limit of 5000 milliseconds on the **WaitMsec** property, and the command is still executing at that time, so the server returns the entity after that time with **WaitMsec** set to the actual value:

```
- Http: Response, HTTP/1.1, Status: Created, URL:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    ProtocolVersion: HTTP/1.1
    StatusCode: 201, Created
    Reason: Created
    Cache-Control:  no-cache
    ContentLength:  601
```

```
  - ContentType:  application/json;charset=utf-8
   - MediaType:  application/json;charset=utf-8
      MainType:  application/json
      charset: utf-8

    Location:  http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc/
    CommandInvocations(guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')
    Server:  Microsoft-IIS/8.0
    X-Content-Type-Options:  nosniff
    request-id:  f29fce11-7fb0-0000-951a-a3f2b07fcc01
    DataServiceVersion:  3.0;
    XAspNetVersion:  4.0.30319
    XPoweredBy:  ASP.NET
    Date:  Mon, 10 Oct 2011 22:17:58 GMT
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json;charset=utf-8
{"d":{"__metadata":{"id":"http://jroberts26v:8000/SampleApp/
    Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')","uri":
    "http://jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc/
    CommandInvocations(guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')",
    "type":"PowerShell.CommandInvocation"},"ID":
    "a1489a59-bed0-412e-b66b-9a0e5a142c46","Command":"Start-Sleep 20",
    "Status":"Executing","OutputFormat":"json","Output":null,"Errors":
    {"__metadata":{"type":"MultiValue(PowerShell.ErrorRecord)"},"results":[]},
    "ExpirationTime":"\/Date(1318285678197)\/","WaitMsec":5000}}
```

The **Status** property is "Executing" because the command is still executing.

At the 13-second mark, the client polls for completion:

```
- Http: Request, GET /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c, Query:$format=json,
    Using Basic Authorization
    Command: GET
  + URI: /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')?$format=json
    ProtocolVersion: HTTP/1.1
    UserAgent:  Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US)
    WindowsPowerShell/3.0
  + ContentType:  application/json
  + Authorization: Basic
    Host:  jroberts26v:8000
    HeaderEnd: CRLF
```

The command is still executing, so the **Status** property is unchanged:

```
- Http: Response, HTTP/1.1, Status: Ok, URL:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c
    ProtocolVersion: HTTP/1.1
    StatusCode: 200, Ok
    Reason: OK
    Cache-Control:  no-cache
    ContentLength:  601
  + ContentType:  application/json;charset=utf-8
```

```
    Server:  Microsoft-IIS/8.0
    X-Content-Type-Options:  nosniff
    request-id:  f29fce11-7fb0-0000-981a-a3f2b07fcc01
    DataServiceVersion:  3.0;
    XAspNetVersion:  4.0.30319
    XPoweredBy:  ASP.NET
    Date:  Mon, 10 Oct 2011 22:18:06 GMT
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json;charset=utf-8
{"d":{"__metadata":{"id":"http://jroberts26v:8000/SampleApp/
    Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')","uri":"http:
    //jroberts26v:8000/SampleApp/Microsoft.Management.Odata.svc/
    CommandInvocations(guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')",
    "type":"PowerShell.CommandInvocation"},"ID":
    "a1489a59-bed0-412e-b66b-9a0e5a142c46","Command":
    "Start-Sleep 20","Status":"Executing","OutputFormat":"json",
    "Output":null,"Errors":{"__metadata":{"type":"MultiValue
    (PowerShell.ErrorRecord)"},"results":[]},"ExpirationTime":"
    \/Date(1318285678197)\/","WaitMsec":5000}}
```

At the 25-second mark, the client polls for completion again:

```
- Http: Request, GET
    /SampleApp/Microsoft.Management.Odata.svc/
    CommandInvocations(guid'a1489a59-bed0-412e-b66b-9a0e5a142c,
    Query:$format=json, Using Basic Authorization
    Command: GET
  + URI:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')?$format=json
    ProtocolVersion: HTTP/1.1
    UserAgent:  Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US)
    WindowsPowerShell/3.0
  + ContentType:  application/json
  + Authorization: Basic
    Host:  jroberts26v:8000
    HeaderEnd: CRLF
```

The command has completed, so the **Status** property is changed to "Completed":

```
- Http: Response, HTTP/1.1, Status: Ok, URL:
    /SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c
    ProtocolVersion: HTTP/1.1
    StatusCode: 200, Ok
    Reason: OK
    Cache-Control:  no-cache
    ContentLength:  601
  + ContentType:  application/json;charset=utf-8
    Server:  Microsoft-IIS/8.0
    X-Content-Type-Options:  nosniff
    request-id:  f29fce11-7fb0-0000-991a-a3f2b07fcc01
    DataServiceVersion:  3.0;
    XAspNetVersion:  4.0.30319
    XPoweredBy:  ASP.NET
```

```
    Date:  Mon, 10 Oct 2011 22:18:18 GMT
    HeaderEnd: CRLF
  - payload: HttpContentType =  application/json;charset=utf-8
{"d":{"__metadata":{"id":"http://jroberts26v:8000/SampleApp/Microsoft.Management.
    Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')","uri":"http://jroberts26v:8000/
    SampleApp/Microsoft.Management.Odata.svc/CommandInvocations
    (guid'a1489a59-bed0-412e-b66b-9a0e5a142c46')","type":
    "PowerShell.CommandInvocation"},"ID":
    "a1489a59-bed0-412e-b66b-9a0e5a142c46","Command":"Start-Sleep
    20","Status":"Completed","OutputFormat":"json","Output":null,
    "Errors":{"__metadata":{"type":"MultiValue(PowerShell.
    ErrorRecord)"},"results":[]},"ExpirationTime":"\/
    Date(1318285678197)\/","WaitMsec":5000}}
```

4.4 Retrieve a Command Description

In this example, the base server URL is http://jroberts26v:8000/SampleApp/Cmd.svc .

To fetch the syntax of the "Select-Object" command in JSON format, the client sends an HTTP GET request to http://jroberts26v:8000/SampleApp/Cmd.svc/CommandDescriptions('Select-Object')?$format=json .

The server returns an HTTP 200 OK reply with the following body data:

```
{"d":
{
"__metadata":{
    "id":"http://jroberts26v:8000/SampleApp/Microsoft.
    Management.Odata.svc/CommandDescriptions('Select-Object')",
    "uri":"http://jroberts26v:8000/SampleApp/Microsoft.
    Management.Odata.svc/CommandDescriptions('Select-Object')",
    "type":"PowerShell.CommandDescription"
    },
"Name":"Select-Object",
"HelpUrl":null,
"AliasedCommand":null,
"Parameters":
{
    "__metadata":
    {
        "type":"MultiValue(PowerShell.CommandParameter)"
    },
    "results":
    [
        {"Name":"InputObject","ParameterType":"System.Management.
        Automation.PSObject"},
        {"Name":"Property","ParameterType":"System.Object[]"},
        {"Name":"ExcludeProperty","ParameterType":"System.String[]"},
        {"Name":"ExpandProperty","ParameterType":"System.String"},
        {"Name":"Unique","ParameterType":"System.Management.Automation.
        SwitchParameter"},
        {"Name":"Last","ParameterType":"System.Int32"},
        {"Name":"First","ParameterType":"System.Int32"},
        {"Name":"Skip","ParameterType":"System.Int32"},
        {"Name":"Wait","ParameterType":"System.Management.Automation.
```

```
            SwitchParameter"},
            {"Name":"Index","ParameterType":"System.Int32[]"},
            {"Name":"Verbose","ParameterType":"System.Management.Automation.
            SwitchParameter"},
            {"Name":"Debug","ParameterType":"System.Management.Automation.
            SwitchParameter"},
            {"Name":"ErrorAction","ParameterType":"System.Management.Automation.
            ActionPreference"},
            {"Name":"WarningAction","ParameterType":"System.Management.Automation.
            ActionPreference"},
            {"Name":"ErrorVariable","ParameterType":"System.String"},
            {"Name":"WarningVariable","ParameterType":"System.String"},
            {"Name":"OutVariable","ParameterType":"System.String"},
            {"Name":"OutBuffer","ParameterType":"System.Int32"}
        ]
    }
    }
    }
```

5 Security

5.1 Security Considerations for Implementers

The OData Protocol may use either HTTP or HTTPS as a transport. HTTP provides no protection against eavesdropping, and should not be used to transport confidential information.

Servers typically will need to restrict the set of resources that any given client can view or modify. OData Extensions for Server Management do not mandate support for any specific set of authentication protocols or any specific authorization model. These will vary based on the environment of a particular server implementation.

Neither OData Protocol nor OData Extensions for Server Management define a mechanism to limit a client's use of server resources, such as CPU, network bandwidth, and memory.

5.2 Index of Security Parameters

Security parameter	Section
Supported authentication methods	1.5

6 Appendix A: Full ABNF Syntax

ABNF group name	Section
Fully qualified error ID	2.2.3.6
CommandInvocation instance ID	3.1.5.2
CommandDescription instance ID	3.1.5.4

6.1 Fully Qualified Error ID ABNF Rules

```
ErrorType = ErrorId [ "," CommandTypeName ]
ErrorId = IDENTIFIER
CommandTypeName = IDENTIFIER *( "." IDENTIFIER )
IDENTIFIER = ALPHA *(ALPHA / DIGIT / "_")
```

6.2 CommandInvocation Instance ID ABNF Rules

`invocationId = CurlyBraceGuidString` as defined in [MS-DTYP] section 2.3.4.3.

6.3 CommandDescription Instance ID ABNF Rules

`commandName =` A percent-encoded UTF-8 character sequence (see [RFC3986] section 2.1 for the definition of percent encoding, and see [RFC3629] for the definition of the UTF-8 encoding).

7 Appendix B: Full CSDL

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

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
  <edmx:Edmx Version="1.0"
    xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
  <edmx:DataServices
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/
    metadata" m:DataServiceVersion="3.0" m:MaxDataServiceVersion="3.0">
  <Schema Namespace="PowerShell"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns="http://schemas.microsoft.com/ado/2010/02/edm">
no
    <EntityType Name="CommandDescription">
      <Key>
        <PropertyRef Name="Name"/>
      </Key>
      <Property Name="Name" Type="Edm.String" Nullable="false"/>
      <Property Name="HelpUrl" Type="Edm.String" Nullable="true"/>
      <Property Name="AliasedCommand" Type="Edm.String" Nullable="true"/>
      <Property Name="Parameters" Type="MultiValue" Nullable="false">
        <TypeRef Type="PowerShell.CommandParameter" Nullable="false" />
      </Property>
    </EntityType>
    <EntityType Name="CommandInvocation">
      <Key>
        <PropertyRef Name="ID" />
      </Key>
      <Property Name="ID" Type="Edm.Guid" Nullable="false" />
      <Property Name="Command" Type="Edm.String" Nullable="true" />
      <Property Name="Status" Type="Edm.String" Nullable="true" />
      <Property Name="OutputFormat" Type="Edm.String" Nullable="true" />
      <Property Name="Output" Type="Edm.String" Nullable="true" />
      <Property Name="Errors" Type="Bag" Nullable="false">
        <TypeRef Type="PowerShell.ErrorRecord" Nullable="false" />
      </Property>
      <Property Name="ExpirationTime" Type="Edm.DateTime" Nullable=
        "true" />
      <Property Name="WaitMsec" Type="Edm.Int32" Nullable="true" />
    </EntityType>
    <ComplexType Name="CommandParameter">
      <Key>
        <PropertyRef Name="Name"/>
      </Key>
      <Property Name="Name" Type="Edm.String" Nullable="true"/>
      <Property Name="ParameterType" Type="Edm.String" Nullable="true"/>
    </ComplexType>
    <ComplexType Name="ErrorRecord">
      <Property Name="FullyQualifiedErrorId" Type="Edm.String"
        Nullable="true" />
      <Property Name="CategoryInfo" Type="PowerShell.ErrorCategoryInfo"
        Nullable="false" />
      <Property Name="ErrorDetails" Type="PowerShell.ErrorDetails"
        Nullable="false" />
      <Property Name="Exception" Type="Edm.String" Nullable="true" />
    </ComplexType>
   <ComplexType Name="ErrorCategoryInfo">
```

```
      <Property Name="Activity" Type="Edm.String" Nullable="true" />
      <Property Name="Category" Type="Edm.String" Nullable="true" />
      <Property Name="Reason" Type="Edm.String" Nullable="true" />
      <Property Name="TargetName" Type="Edm.String" Nullable="true" />
      <Property Name="TargetType" Type="Edm.String" Nullable="true" />
    </ComplexType>
    <ComplexType Name="ErrorDetails">
      <Property Name="Message" Type="Edm.String" Nullable="true" />
      <Property Name="RecommendedAction" Type="Edm.String" Nullable=
         "true" />
    </ComplexType>
   </Schema>
  <Schema Namespace="PswsTest"
     xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
     xmlns="http://schemas.microsoft.com/ado/2010/02/edm">
    <EntityContainer Name="PSWSEntityContainer" m:IsDefaultEntityContainer=
         "true">
      <EntitySet Name="CommandInvocations" EntityType="PowerShell.
         CommandInvocation" />
    </EntityContainer>
  </Schema>
  </edmx:DataServices>
  </edmx:Edmx>
```

8 Appendix C: 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 Server 2012 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 3.1.3: In Windows Server 2012, the range of commands is the set of PowerShell cmdlets that are installed on the server.

<2> Section 3.1.5: Windows Server 2012 includes the header; its contents are a GUID that specifies the ETW tracing ID of the request. For more information, see [MSDN-EventTracing].

<3> Section 3.1.5: Windows Server 2012 places no limits by default but can be configured to limit the number of expressions in $expand and the number of identifiers in each expression.

<4> Section 3.1.5.1.1: Windows Server 2012 allows a client access only to elements created by the same network identity.

<5> Section 3.1.5.1.2: Windows pipes the output of the command or pipeline to a conversion cmdlet such as "**ConvertTo-Json**".

<6> Section 3.1.5.3.1: In Windows Server 2012, the accessible commands are determined by a session-configuration object that is defined in the ManagementOdata\powershell\sessionConfiguration node of the server role's web.config file. For more information, see [MSDN-PSWSSDG].

9 Change Tracking

This section identifies changes that were made to the [MS-ODASM] protocol document between the October 2012 and January 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: Full CSDL	67412 Added CommandDescription and CommandParameter types.	Y	Content updated.

10 Index