[MS-DSCPM]:

Desired State Configuration Pull Model Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
8/8/2013	1.0	New	Released new document.
11/14/2013	2.0	Major	Updated and revised the technical content.
2/13/2014	2.1	Minor	Clarified the meaning of the technical content.
5/15/2014	2.1	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	3.0	Major	Significantly changed the technical content.
10/16/2015	4.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Desired State Configuration Pull Model Protocol is based on the **Hypertext Transfer Protocol (HTTP)** (as specified in [RFC2616]). It is used for getting a client's **configuration** and **modules** from the server and for reporting back the client's status to the server.

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

1.1 Glossary

The following terms are specific to this document:

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

binary large object (BLOB): A collection of binary data stored as a single entity in a database.

checksum: A value that is the summation of a byte stream. By comparing the checksums computed from a data item at two different times, one can quickly assess whether the data items are identical.

configuration: Represents a **binary large object (BLOB)**. The protocol does not process the content of the **BLOB** and it is passed as-is to the higher layer.

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

module: A BLOB in the Desired State Configuration Pull Model Protocol [MS-DSCPM]. The protocol does not process the content of the BLOB, and it is passed as it is to the higher layer.

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

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

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

1.2 References

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

1.2.1 Normative References

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

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

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

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.rfc-editor.org/rfc/rfc4234.txt

[RFC4634] Eastlake III, D. and Hansen, T., "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006, http://www.ietf.org/rfc/rfc4634.txt

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.rfc-editor.org/rfc/rfc4648.txt

1.2.2 Informative References

None.

1.3 Overview

Note: Some of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it in the Product Behavior appendix.

The Desired State Configuration Pull Model Protocol is used to register a client, to get the configuration and the module from the server, and to report back some elements to the server.

The protocol depends on HTTP for the transfer of all protocol messages, including the transfer of the binary data. In this specification, the entity that initiates the HTTP connection is referred to as the client, and the entity that responds to the HTTP connection is referred to as the server. With the Desired State Configuration Pull Model Protocol, binary data flows from the server to the client.

1.4 Relationship to Other Protocols

This protocol depends on HTTP as specified in [RFC2616]. HTTP version 1.1 is used with this protocol.

1.5 Prerequisites/Preconditions

This protocol does not provide a mechanism for a client to discover the **Uniform Resource Locator (URL)** of the server. Thus, it is a prerequisite that the client obtain the URL of the server before this protocol can be used.

1.6 Applicability Statement

The Desired State Configuration Pull Model Protocol is capable of downloading the configuration and modules from the server.

This document covers versioning issues in the following areas:

Supported Transports: This protocol can be implemented on top of HTTP, as specified in section 2.1.

Security and Authentication Methods: This protocol supports HTTP access authentication, as specified in [RFC2616] section 11.

Localization: This specification does not specify any localization-dependent protocol behavior.

1.7 Vendor-Extensible Fields

None.

1.8 Standards Assignments

None.

2 Messages

2.1 Transport

The Desired State Configuration Pull Model Protocol uses HTTP 1.1, as specified in [RFC2616], as the transport layer.

A **Transmission Control Protocol (TCP)** port has not been reserved for this protocol. TCP port 80 is commonly used because many HTTP proxy servers forward only HTTP traffic that uses port 80.

The protocol uses the access authentication functionality of the HTTP layer as specified in [RFC2616] section 11.

2.2 Common Data Types

None.

2.2.1 Namespaces

None.

2.2.2 HTTP Headers

The Desired State Configuration Pull Model Protocol uses existing headers as specified in [RFC2616].

Unless specified otherwise, the headers defined in this specification are used in both request and response messages.

If a client or server receives an HTTP header that is not defined in this section, or if the header is not defined in the current context (for example, receiving a request-only header in a response), the header MUST be interpreted as specified in [RFC2616].

This section defines the syntax of the HTTP headers that use the **Augmented Backus-Naur Form (ABNF)** syntax, as specified in [RFC4234].

The following table summarizes the HTTP headers defined by this specification.

Header	Description
Content-Type	Section 2.2.2.1
Checksum	Section 2.2.2.2

2.2.2.1 Content-Type

The Content-Type header specifies the type of data that is included in the body of the GET or POST request.

The syntax of the Content-Type header is defined as follows.

```
Ctype          = "application/octet-stream" /
                 "application/json"
```

```
Content-Type  = "Content-Type: " "application/octet-stream" CRLF /
                  "Content-Type: " "application/json" [";charset=UTF-8"] CRLF
```

Example: Content-Type: application/octetstring

Content-Type: application/json;charset=UTF-8

2.2.2.1.1 Application/octet-stream

This Content-Type is defined only for use in a request sent to the server and used in a GET request to get the module or configuration from a server.

2.2.2.1.2 Application/json

This Content-Type is defined only for use in a request sent to the server and used in POST request.

2.2.2.2 Checksum

The Checksum header field is defined only for use in a response message sent to a client as part of a GET request for the module and configuration.

```
Checksum = "Checksum" : DQUOTE Check-sumvalue DQUOTE CRLF
Check-sumvalue = BASE16 ; specified in [RFC4648]
  / 0x00 (Null Character)
```

Example: "Checksum":"8eDMbsSDig15Xx+B3msvRrDa5N1njaf5smVujQjhOeI="

"Checksum":""

2.2.2.3 ChecksumAlgorithm

The ChecksumAlgorithm header field specifies the checksum algorithm used to generate the checksum.

```
ChecksumAlgorithm = "ChecksumAlgorithm" : DQUOTE Check-sumAlgorithmvalue DQUOTE CRLF
Check-sumAlgorithmvalue = "SHA-256" ; specified in [RFC4634]
```

Example: "ChecksumAlgorithm":"SHA-256"

2.2.2.4 ConfigurationName

The ConfigurationName header field<1> is used in the request message sent to the server as part of a GET request for the configuration.

ConfigurationName: "ConfigurationName" : DQUOTE Configuration-Namevalue DQUOTE CRLF

Configuration-Namevalue: Element *(Element)

Element: DIGIT / ALPHA

Example: "ConfigurationName":"SubPart1"

2.2.2.5 ProtocolVersion

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The ProtocolVersion header field<2> is used in the request message sent to the server as part of every request. The current version is hardcoded to 2.0.

ProtocolVersion: "ProtocolVersion" : DQUOTE ProtocolVersion-Namevalue DQUOTE CRLF

ProtocolVersion-Namevalue: Element . *(Element)

Element: DIGIT

Example: "ProtocolVersion":"2.0"

2.2.2.6 AgentId

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The AgentId header field<3> is used in the request message sent to the server as part of every request. The AgentId is a unique GUID value generated once and stored in the Registry.

AgentId: "AgentId" : DQUOTE AgentId-Namevalue DQUOTE CRLF

AgentId: Element . *(Element)

Element: DIGIT / ALPHA

Example: "AgentId":"{34C8104D-F7BA-4672-8226-0809B0A3BEC3}"

2.2.2.7 Authorization

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The Authorization header field<4> is used in the request message sent to the server as part of every request. The Shared authorization key is generated during the registration and serves to verify the payload sent to the server.

Authorization: "Authorization": Shared DQUOTE Authorization-Namevalue DQUOTE CRLF

Authorization-Namevalue: A signature generated from an HMAC hash of the request body.

Element: DIGIT / ALPHA

Example: "Authorization": "Shared GKtPgocze1AM/pgc3LQzyGpDCRs15KoKx/2eXxlL8+w="

2.2.3 Common URI Parameters

The following table summarizes the set of common **URI** parameters defined by this protocol.

URI parameter	Section
ConfigurationId	Section 2.2.3.1
ModuleName	Section 2.2.3.2
ModuleVersion	Section 2.2.3.3

2.2.3.1 ConfigurationId

The *ConfigurationId* parameter is a **universally unique identifier (UUID)** as specified in [RFC4122] section 3.

2.2.3.2 ModuleName

The *ModuleName* parameter is a string that is used by the server to identify a specific module.

MODULENAME = Element *(Element)

Element = DIGIT / ALPHA / 0x5f

2.2.3.3 ModuleVersion

The *ModuleVersion* parameter identifies the version of a module. It can be either an empty string or a string containing two to four groups of digits where the groups are separated by a period.

```
MODULEVERSION = MULTIDIGIT 0x2E MULTIDIGIT
  / MULTIDIGIT 0x2E MULTIDIGIT 0x2E MULTIDIGIT
  / MULTIDIGIT 0x2E MULTIDIGIT 0x2E MULTIDIGIT 0x2E MULTIDIGIT
  / 0x00 (NULL character)
MULTIDIGIT = DIGIT *[DIGIT]
```

3 Protocol Details

3.1 GetConfiguration Details

The purpose of the GetConfiguration request is to get the configuration from the server. The GetConfiguration request maps to an HTTP GET request in which the Content-Type header is an application/octet-stream.

3.1.1 Abstract Data Model

The server MUST maintain a **ConfigurationTable** where each entry contains:

ServerConfigurationId: A unique combination of a ConfigurationId (section 2.2.3.1) and a ConfigurationName (section 2.2.2.4). The ConfigurationId MUST NOT be empty.

ServerConfigurationData: A **binary large object (BLOB)**. This data is not interpreted as part of this protocol and is passed on to higher layers as is.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The server MUST match the ConfigurationId from the URL and, if specified, the ConfigurationName from the headers with the **ServerConfigurationId** in the **ConfigurationTable**. The server MUST use case-insensitive ordinal comparison to match the ConfigurationId (section 2.2.3.1) and ConfigurationName (section 2.2.2.4). If a match is found, the server MUST return **ConfigurationData** to the client with status code 200. If a match is not found, the server MUST return status code 404.

Resource	Description
Action(ConfigurationId={ConfigurationId})/ConfigurationContent	Gets the configuration from the server.

Request header	Usage	Value
ConfigurationName	Optional	As specified in section 2.2.2.4.

The responses to all the methods can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Returned when the request is completed.
400	BAD REQUEST	The request could not be understood by the server due to malformed syntax.

Status code	Reason phrase	Description
404	NOT FOUND	Returned when the resource is not found.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
checksum	Required	As specified in section 2.2.2.2.
checksumalgorithm	Required	As specified in section 2.2.2.3.

3.1.5.1 Action(ConfigurationId={ConfigurationId})/ConfigurationContent

The following HTTP method is allowed to be performed on this resource.

HTTP method	Description
GET	Gets the configuration from the server.

3.1.5.2 GET

The URL specified by the client in the HTTP request line of the GET request identifies a "configuration point" targeted for the client. The server can have multiple configuration points and different configuration points can have different access permissions associated with them. For example, some configuration points require HTTP access authentication (as specified in [RFC2616] section 11). As another example, a configuration point could also allow only clients that connect from a specific IP address.

The syntax of the GetConfiguration request is defined as follows.

```
DSC-GetConfiguration-Request    =  DSC-GetConfiguration-Req-Line
                            DSC-GetConfigurationSetReq-Headers

DSC-GetConfiguration-Req-Line   = "GET" SP Request-URI SP HTTP-Version CRLF

Request-URI = Request-URI-Start DSC-GetConfigurationRequest-URI-End

DSC-GetConfigurationSetReq-Headers = *( DSC-GetConfigurationSetReq-Header-REQ
     / DSC-GetConfigurationSetReq-Header-OPT )

DSC-GetConfigurationSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]

DSC-GetConfigurationSetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
                        / ConfigurationName ; section 2.2.2.4
DSC-GetConfigurationRequest-URI-End = "Action(ConfigurationId=" SQUOTE CONFIGURATIONID SQUOTE

RBRACKET FSLASH "ConfigurationContent"
SQUOTE = %x27  ;  ' (Single Quote)
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
CONFIGURATIONID = UUID ; as specified in [RFC4122]
```

The syntax of the GetConfiguration response is defined as follows:

```
DSC-GetConfiguration-Response  = Status-Line
DSC-GetConfigurationResp-Headers
DSC-GetConfigurationResp-Body

DSC-GetConfigurationResp-Headers = *( DSC-GetConfigurationResp-Header-REQ
                          / DSC-GetConfigurationResp-Header-OPT
                          / DSC-GetConfigurationResp-Body)

DSC-GetConfigurationResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
        / Content-Type ; section 2.2.2.1.1
        / Checksum ; section 2.2.2.2
        / ChecksumAlgorithm ; section 2.2.2.3

DSC-GetConfigurationResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]

DSC-GetConfigurationResp-Body = Configuration ; section 3.1.5.1.1.2
```

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
checksum	Required	As specified in section 2.2.2.2.
checksumalgorithm	Required	As specified in section 2.2.2.3.

The response message for this method can result in the following status codes.

Status code	Description
200	Request completed.
400	Bad request.
404	The resource was not found.

3.1.5.2.1 Request Body

None.

3.1.5.2.2 Response Body

In the response body, configuration represents a BLOB.

3.1.5.2.3 Processing Details

The client gets the configuration from the server as content-type application/octet-stream in the response body for the GetConfiguration request. The server MUST send the **checksum** in the response headers as specified in section 2.2.2.2. The server MUST send the ChecksumAlgorithm in the response headers as specified in section 2.2.2.3. The server MUST generate the checksum using the following algorithm:

- Use the algorithm specified in section 2.2.2.3 to compute the hash of the response body as specified in [RFC4634] section 4.1.
- Perform base16 encoding of the computed hash as specified in [RFC4648] section 8.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 GetModule Details

The purpose of the GetModule request is to get the module from the server. GetModule request maps to HTTP GET request with content-type as application/octet-stream as part of the request.

3.2.1 Abstract Data Model

The server MUST maintain a **ModuleTable** in which each entry contains the following:

ServerConfigurationId: The ConfigurationId (section 2.2.3.1).

ServerModuleName: The ModuleName (section 2.2.3.2).

ServerModuleVersion: The ModuleVersion (section 2.2.3.3).

ServerModuleData: A BLOB of data. This data is not interpreted as part of this protocol and is passed on to higher layers as is.

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 server MUST match ConfigurationId from the URL with **ServerConfigurationId**, ModuleName with **ServerModuleName**, and ModuleVersion with **ServerModuleVersion** of the **ModuleTable**. The server MUST use case-insensitive ordinal comparison to match ConfigurationId, ModuleName, and ModuleVersion. For a **ServerConfigurationId**, the server MUST have a unique combination of **ServerModuleName** and **ServerModuleVersion**. **ServerModuleName** in the **ModuleTable** MUST NOT be empty. If a match is found, the server MUST return **ModuleData** to the client with status code 200. If a match is not found, the server MUST return status code 404.

Resource	Description
Module(ConfigurationId={ConfigurationId},ModuleName={moduleName},ModuleVersion={moduleVersion})/ModuleContent	Get the module from the server.

The responses to all the methods can result in the following status codes.

Status code	Reason phrase	Description
200	OK	Request completed.
400	BAD REQUEST	The request could not be understood by the server due to malformed syntax.
404	NOT FOUND	Used in cases where the resource is not found.

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
checksum	Required	As specified in section 2.2.2.2.
checksumalgorithm	Required	As specified in section 2.2.2.3.

3.2.5.1 Module(ConfigurationId={ConfigurationId},ModuleName={moduleName},ModuleVersion={moduleVersion})/ModuleContent

The following HTTP method is allowed to be performed on this resource.

HTTP method	Description
GET	Gets the module from the server.

3.2.5.1.1 GET

The URL specified by the client in the HTTP request line of the GET request identifies a "module point" targeted for the client. The server might have multiple module points and different modules points can have different access permissions associated with them. For example, some module points require HTTP access authentication (as specified in [RFC2616] section 11). Other module points allow only clients that connect from a specific IP address.

The syntax of the GetModule request is defined as follows.

```
DSC-GetModule-Request    =  DSC-GetModule-Req-Line
                            DSC-GetModuleSetReq-Headers

DSC-GetModule-Req-Line   = "GET" SP Request-URI SP HTTP-Version CRLF
Request-URI = Request-URI-Start DSC-GetModuleRequest-URI-End
DSC-GetModuleSetReq-Headers = *( DSC-GetModuleSetReq-Header-REQ
     / DSC-GetModuleSetReq-Header-OPT )

DSC-GetModuleSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]

DSC-GetModuleSetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
DSC-GetModuleRequest-URI-End = "Module(ConfigurationId=" SQUOTE CONFIGURATIONID SQUOTE
                               ",ModuleName=" SQUOTE MODULENAME SQUOTE
                               ",ModuleVersion=" SQUOTE MODULEVERSION SQUOTE
                               RBRACKET FSLASH "ModuleContent"

SQUOTE = %x27  ;  ' (Single Quote)
MODULENAME = ModuleName; as specified in section 2.2.3.2
MODULEVERSION = *(DIGIT)[ 0x2E *(DIGIT)]
```

```
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
CONFIGURATIONID = UUID ; as specified in [RFC4122]
```

The syntax of the GetModule response is defined as follows:

```
DSC-GetModule-Response  = Status-Line
                                DSC-GetModuleResp-Headers
                DSC-GetModuleResp-Body

DSC-GetModuleResp-Headers = *( DSC-GetModuleResp-Header-REQ
                            / DSC-GetModuleResp-Header-OPT)

DSC-GetModuleResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
       / Content-Type ; section 2.2.2.1.1
       / Checksum ; section 2.2.2.2
       / ChecksumAlgorithm ; section 2.2.2.3

DSC-GetModuleResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]
DSC-GetModuleResp-Body = ModuleData ; section 3.2.5.1.1.2
```

The response message for this operation contains the following HTTP headers.

Response header	Usage	Value
checksum	Required	As specified in section 2.2.2.2.
checksumalgorithm	Required	As specified in section 2.2.2.3.

The response message for this method can result in the following status codes.

Status code	Description
200	Request completed.
400	Bad request.
404	The resource is not found.

3.2.5.1.1.1 Request Body

None.

3.2.5.1.1.2 Response Body

ModuleData represents a binary large object (BLOB) .

3.2.5.1.1.3 Processing Details

The client gets the module from the server as content-type application/octet-stream in the response body for GetModule request. The server MUST send the checksum in response headers as specified in section 2.2.2.2. The server MUST send the ChecksumAlgorithm in the response headers as specified in section 2.2.2.3. The server generates the checksum using the following algorithm:

- Use the algorithm specified in ChecksumAlgorithm to compute the hash of the response body as specified in [RFC4634] section 4.1.

- Perform base16 encoding of the computed hash as specified in [RFC4648] in section 8.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 GetAction Details

The purpose of the GetAction request is to get the action, as specified in section 3.3.5.1.1.2, from the server. GetAction request maps to HTTP POST request with content-type as application/json, as specified in Appendix A: Full JSON Schema (section 6), as part of the request.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

Resource	Description
Action(ConfigurationId={ConfigurationId})/GetAction	Get the action from the server.

The responses to all the methods can result in the following status codes.

Status code	Reason phrase	Description
200	OK	The Request completed.
400	BAD REQUEST	The request could not be understood by the server due to malformed syntax.
404	NOT FOUND	The resource was not found.

3.3.5.1 Action(ConfigurationId={ConfigurationId})/GetAction

The following HTTP method is allowed to be performed on this resource.

HTTP method	Description
POST	POSTs the data to the server and gets action from the server.

3.3.5.1.1 POST

The URL specified by the client in the HTTP request line of the POST request identifies the action point targeted for the client. The server can have multiple action points and different action points can have different access permissions associated with them. For example, some action points require HTTP access authentication (as specified in [RFC2616] section 11). Or, action points can also allow only clients that connect from a specific IP address.

The syntax of the GetAction request is defined as follows.

```
DSC-GetAction-Request  =  DSC-GetAction-Req-Line
                          DSC-GetActionSetReq-Headers
                          DSC-GetActionReq-Body

DSC-GetAction-Req-Line   = "POST" SP Request-URI SP HTTP-Version CRLF
Request-URI = Request-URI-Start DSC-GetActionRequest-URI-End

DSC-GetActionRequest-URI-End = "Action(ConfigurationId=" SQUOTE CONFIGURATIONID SQUOTE

RBRACKET FSLASH "GetAction"
SQUOTE = %x27  ;  ' (Single Quote)
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
CONFIGURATIONID = UUID ; as specified in [RFC4122]


DSC-GetActionSetReq-Headers = *( DSC-GetActionSetReq-Header-REQ
     / DSC-GetActionSetReq-Header-OPT )

DSC-GetActionSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]
          / Accept ; section 14.1 of [RFC2616]
          / ContentType ' section 2.2.2.1.2
          / Content-Length ; section 14.13 of [RFC2616]

DSC-GetActionSetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
           / Expect ; section 14.20 of [RFC2616]

DSC-GetActionReq-Body = ActionRequest ; section 3.3.5.1.1.1
```

The syntax of the GetAction response is defined as follows:

```
DSC-GetAction-Response  = Status-Line
DSC-GetActionResp-Headers
DSC-GetActionResp-Body

DSC-GetActionResp-Headers = *( DSC-GetActionResp-Header-REQ
                           / DSC-GetActionResp-Header-OPT)

DSC-GetActionResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
       / Content-Type ; section 2.2.2.1.2

DSC-GetActionResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]
DSC-GetActionResp-Body = ActionContent ; section 3.3.5.1.1.2
```

The response message for this method can result in the following status codes.

Status code	Description
200	Request completed.
400	Bad request.
404	The resource is not found.

3.3.5.1.1.1 Request Body

The ActionRequest packet is used by the client to transfer the following data fields:

Checksum: A checksum as specified in section 2.2.2.2.

ChecksumAlgorithm: The algorithm for the Checksum as specified in section 2.2.2.3.

NodeCompliant: A value indicating TRUE or FALSE.

StatusCode: A value indicating a number between –2147483648 and 2147483647.

ConfigurationName: An opaque name as specified in section 2.2.2.4.

3.3.5.1.1.2 Response Body

The ActionContent packet is used by the server to transfer the following data field:

Value: MUST be either GetConfiguration, Retry<5>, or OK.

3.3.5.1.1.3 Processing Details

The client sends the GetAction request with content-type as application/json to the server with a request body as specified in section 3.3.5.1.1.1. The client MUST include **Checksum** and **ChecksumAlgorithm** in the request body. The client SHOULD include **StatusCode** in the request body. The server responds back with content-type as application/json with response body as specified in section 3.3.5.1.1.2.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 SendStatusReport Details

The SendStatusReport request<6> sends the status report, as specified in section 3.4.5.1.1.1, to the server. The SendStatusReport request maps to the HTTP POST request with content-type as application/json, as specified in Appendix A: Full JSON Schema (section 6), as part of the request.

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

None.

3.4.4 Higher-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rules

Resource	Description
Nodes(ConfigurationID={ConfigurationId})/SendStatusReport	Send the status report to the server.

The response to all the methods results in one of the following status codes.

Status code	Reason phrase	Description
200	OK	The request completed.
400	BAD REQUEST	The server could not read the request due to malformed syntax.
404	NOT FOUND	The resource was not found.

3.4.5.1 Nodes(ConfigurationID={ConfigurationId})/SendStatusReport

The following HTTP method can be performed on this resource.

HTTP method	Description
POST	Posts the data to the server and gets status from the server.

3.4.5.1.1 POST

The URL specified by the client in the HTTP request line of the POST request identifies the status point targeted for the client. The server can have multiple status points that have different access permissions associated with them. For example, some status points require HTTP access authentication (as specified in [RFC2616] section 11). As another example, status points allow only clients that connect from a specific IP address.

The syntax of the SendStatusReport request is defined as follows.

```
DSC-SendStatusReport-Request  =  DSC-SendStatusReport-Req-Line
                          DSC-SendStatusReportSetReq-Headers
                          DSC-SendStatusReportReq-Body

DSC-SendStatusReport-Req-Line   = "POST" SP Request-URI SP HTTP-Version CRLF
```

```
Request-URI = Request-URI-Start DSC-SendStatusReportRequest-URI-End

DSC-SendStatusReportRequest-URI-End = "Nodes(ConfigurationId=" SQUOTE CONFIGURATIONID SQUOTE

RBRACKET FSLASH "SendStatusReport"
SQUOTE = %x27  ;  ' (Single Quote)
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
ConfigurationID = UUID ; as specified in [RFC4122]

DSC-SendStatusReportSetReq-Headers = *( DSC-SendStatusReportSetReq-Header-REQ
     / DSC-SendStatusReportSetReq-Header-OPT )

DSC-SendStatusReportSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]
          / Accept ; section 14.1 of [RFC2616]
          / ContentType ' section 2.2.2.1.2
          / Content-Length ; section 14.13 of [RFC2616]

DSC-SendStatusReportSetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
           / Expect ; section 14.20 of [RFC2616]

DSC-SendStatusReportReq-Body = StatusReportRequest ; section 3.4.5.1.1.1
```

The syntax of the SendStatusReport response is defined as follows:

```
DSC-SendStatusReport-Response  = Status-Line
DSC-SendStatusReportResp-Headers
DSC-SendStatusReportResp-Body

DSC-SendStatusReportResp-Headers = *( DSC-SendStatusReportResp-Header-REQ
                           / DSC-SendStatusReportResp-Header-OPT)

DSC-SendStatusReportResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
       / Content-Type ; section 2.2.2.1.2

DSC-SendStatusReportResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]
DSC-SendStatusReportResp-Body = StatusReportContent ; section 3.4.5.1.1.2
```

The response message for this method can result in the following status codes.

Status code	Description
200	Request completed.
400	Bad request.
404	The resource was not found.

3.4.5.1.1.1 Request Body

The StatusReportRequest packet is used by the client to transfer the following data fields:

JobId: The JobId parameter is a universally unique identifier (UUID) as specified in [RFC4122] section 3.

NodeName: A value that is used to identify the name of the client.

OperationType: A value that identifies the type for the operation.

LCMVersion: A value that identifies the report generator on the client.

ReportFormatVersion: A value that finds the identifier for the report.

ConfigurationVersion: .A value that contains a string of two to four groups of digits where the groups are separated by a period.

IpAddress: A value that identifies the IP addresses of the client separated by a semicolon (;).

StartTime: A value that identifies the start time of an operation on the client.

EndTime: A value that identifies the end time of an operation on the client.

Errors: A value that represents the errors for an operation on the client.

StatusData: A value that represents the status of an operation on the client.

3.4.5.1.1.2 Response Body

The StatusReportContent packet does not contain any data.

3.4.5.1.1.3 Processing Details

The client sends the SendStatusReport request with content-type as application/json to the server with a request body as specified in section 3.3.5.1.1.1. The client MUST include JobId in the request body. The server responds back with status codes as specified in section 3.4.5.1.1.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 GetStatusReport Details

The GetStatusReport<7> request gets the status report from the server, as specified in section 3.5.5.1.1. The GetStatusReport request maps to the HTTP GET request with content-type as application/json, as specified in Appendix A: Full JSON Schema (section 6), as part of the request.

3.5.1 Abstract Data Model

None.

3.5.2 Timers

None.

3.5.3 Initialization

None.

3.5.4 Higher-Layer Triggered Events

None.

3.5.5 Message Processing Events and Sequencing Rules

Resource	Description
Nodes(ConfigurationId={ConfigurationId})/Reports(JobId={JobId}))	Get the status report from the server.

The responses to all the methods can result in the following status codes.

Status Code	Reason Phrase	Description
200	OK	The request completed.
400	BAD REQUEST	The request could not be understood by the server due to malformed syntax.
404	NOT FOUND	The resource was not found.

3.5.5.1 Nodes(ConfigurationId={ConfigurationId})/Reports

The following HTTP method can be performed on this resource.

HTTP Method	Description
GET	Gets the status report from the server.

3.5.5.1.1 GET

The URL specified by the client in the HTTP request line of the GET request identifies the "status point" targeted for the client. The server might have multiple status points and different status points might have different access permissions associated with them. For example, some status points require HTTP access authentication (as specified in [RFC2616] section 11). Other status points allow only clients that connect from a specific IP address.

The syntax of the GetStatusReport request is defined as follows.

```
DSC-GetStatusReport-Request  =  DSC-GetStatusReport-Req-Line
                                DSC-GetStatusReportSetReq-Headers
```

```
DSC-GetStatusReport-Req-Line   = "GET" SP Request-URI SP HTTP-Version CRLF
Request-URI = Request-URI-Start DSC-GetStatusReportRequest-URI-End

DSC-GetStatusReportRequest-URI-End = "Nodes(ConfigurationId=" SQUOTE CONFIGURATIONID SQUOTE

RBRACKET FSLASH "Reports(JobId=" SQUOTE JOBID SQUOTE
RBRACKET
SQUOTE = %x27  ;  ' (Single Quote)
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
ConfigurationID = UUID ; as specified in [RFC4122]
JOBID = UUID; as specified in [RFC4122]

DSC-GetStatusReportSetReq-Headers = *( DSC-GetStatusReportSetReq-Header-REQ
     / DSC-GetStatusReportSetReq-Header-OPT )

DSC-GetStatusReportSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]
          / Accept ; section 14.1 of [RFC2616]
          / ContentType ' section 2.2.2.1.2
          / Content-Length ; section 14.13 of [RFC2616]

DSC-GetStatusReportSetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
           / Expect ; section 14.20 of [RFC2616]
```

The syntax of the GetStatusReport response is defined as follows:

```
DSC-GetStatusReport-Response  = Status-Line
DSC-GetStatusReportResp-Headers
DSC-GetStatusReportResp-Body

DSC-GetStatusReportResp-Headers = *( DSC-GetStatusReportResp-Header-REQ
                          / DSC-GetStatusReportResp-Header-OPT)

DSC-GetStatusReportResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
       / Content-Type ; section 2.2.2.1.2

DSC-GetStatusReportResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]
DSC-GetStatusReportResp-Body = StatusReportContent ; section 3.4.5.1.1.2
```

The response message for this method can result in the following status codes.

Status code	Description
200	Request completed.
400	Bad request.
404	The resource was not found.

3.5.5.1.1.1 Request Body

The StatusReportRequest packet is used by the client to transfer the following data fields:

JobId: A universally unique identifier (UUID) as specified in [RFC4122] section 3.

NodeName: A value that identifies the name of the client.

OperationType: A value that identifies the type for the operation.

LCMVersion: A value that identifies the report generator on the client.

ReportFormatVersion: A value that finds the identifier for the report.

ConfigurationVersion: .A value that contains a string of two to four groups of digits where the groups are separated by a period.

IpAddress: A value that identifies the IP addresses of the client separated by a semicolon (;).

StartTime: A value that identifies the start time of an operation on the client.

EndTime: A value that identifies the end time of an operation on the client.

Errors: A value that represents the errors for an operation on the client.

StatusData: A value that represents the status of an operation on the client.

3.5.5.1.1.2 Response Body

StatusReportContent represents a BLOB.

3.5.5.1.1.3 Processing Details

The client sends the GetStatusReport request with the content-type as application/json to the server with a request body as specified in section 3.5.5.1.1.1. The server responds back with status codes as specified in section 3.5.5.1.1.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 RegisterDscAgent Details

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The RegisterDscAgent<8> request registers a client with a server, as specified in section 3.5.5.1.1. The RegisterDscAgent request maps to the HTTP PUT request with content-type as application/json, as specified in Appendix A: Full JSON Schema (section 6), as part of the request.

3.6.1 Abstract Data Model

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

3.6.2 Timers

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

3.6.3 Initialization

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

3.6.4 Higher-Layer Triggered Events

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

3.6.5 Message Processing Events and Sequencing Rules

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

Resource	Description
Nodes(AgentId={AgentId})	Register a client.

The responses to all the methods can result in the following status codes:

Status Code	Reason Phrase	Description
200	OK	The request completed.
400	BAD REQUEST	The request could not be understood by the server due to malformed syntax.
404	NOT FOUND	The resource was not found.

3.6.5.1 Nodes(AgentId={AgentId})

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior

notes that pertain to the preliminary product version contain specific references to it as an aid to the reader. The following HTTP method can be performed on this resource:

HTTP Method	Description
PUT	Sends the registration to the server.

3.6.5.1.1 PUT

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The URL specified by the client in the HTTP request line of the PUT request identifies the "registration point" targeted for the client. The server might have multiple registration points and different registration points might have different access permissions associated with them. For example, some registration points require HTTP access authentication (as specified in [RFC2616] section 11). Other registration points allow only clients that connect from a specific IP address.

The syntax of the RegisterDscAgent request is defined as follows:

```
DSC-RegisterDscAgent-Request  =  DSC-RegisterDscAgent-Req-Line
                          DSC-RegisterDscAgentReq-Headers

DSC-RegisterDscAgent-Req-Line   = "PUT" SP Request-URI SP HTTP-Version CRLF
Request-URI = Request-URI-Start DSC-RegisterDscAgent-URI-End

DSC- RegisterDscAgentRequest-URI-End = "Nodes(Agentid=" SQUOTE AGENTID SQUOTE

RBRACKET FSLASH "Reports(JobId=" SQUOTE JOBID SQUOTE
RBRACKET
SQUOTE = %x27  ;  ' (Single Quote)
RBRACKET = %29 ; ) (Closing Bracket)
FSLASH = %2F ; / (Forward Slash)
AgentID = UUID ; as specified in [RFC4122]

DSC-RegisterDscAgentReq-Headers = *( DSC-RegisterDscAgentSetReq-Header-REQ
     / DSC-RegisterDscAgentSetReq-Header-OPT )

DSC- RegisterDscAgentSetReq-Header-REQ    = Host    ; section 14.23 of [RFC2616]
          / Accept ; section 14.1 of [RFC2616]
          / ContentType ' section 2.2.2.1.2
          / Content-Length ; section 14.13 of [RFC2616]

DSC- RegisterDscAgent SetReq-Header-OPT    = Connection  ; section 14.10 of [RFC2616]
           / Expect ; section 14.20 of [RFC2616]
```

The syntax of the RegisterDscAgent response is defined as follows:

```
DSC-RegisterDscAgent-Response  = Status-Line
DSC-RegisterDscAgentResp-Headers
DSC-RegisterDscAgentResp-Body

DSC-RegisterDscAgentResp-Headers = *( DSC-RegisterDscAgentResp-Header-REQ
                          / DSC-RegisterDscAgentResp-Header-OPT)

DSC-RegisterDscAgentResp-Header-REQ  =  Content-Length ; section 14.13 of [RFC2616]
       / Content-Type ; section 2.2.2.1.2

DSC-RegisterDscAgentResp-Header-OPT  =  Server     ; section 14.38 of [RFC2616]
```

```
DSC-RegisterDscAgentResp-Body = StatusReportContent ; section 3.4.5.1.1.2
```

The response message for this method can result in the following status codes:

Status code	Description
200	Request completed.
400	Bad request.
404	The resource was not found.

3.6.5.1.1.1 Request Body

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The RegisterDscAgentRequest packet is used by the client to transfer the following data fields:

JobId: A universally unique identifier (UUID) as specified in [RFC2616] section 3.

NodeName: A value that identifies the name of the client.

LCMVersion: A value that identifies the report generator on the client.

ConfigurationNames: A set of values that define the configurations that MAY be targeted to this client.

IpAddress: A value that identifies the IP addresses of the client separated by a semicolon (;).

Certificate: A set of values describing the certificate generated on the client used for client validation, including Certificate FriendlyName, Certificate Issuer, Certificate NotAfter, Certificate NotBefore, Certificate Subject, Certificate PublicKey, Certificate Thumbprint, and Certificate Version.

3.6.5.1.1.2 Response Body

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

RegisterDscAgentContent represents a BLOB.

3.6.5.1.1.3 Processing Details

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The client sends the RegisterDscAgent request with the content-type as application/json to the server with a request body as specified in section 3.6.5.1.1.1. The server responds back with status codes as specified in section 3.6.5.1.1.

3.6.6 Timer Events

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

3.6.7 Other Local Events

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

None.

4 Protocol Examples

4.1 GetConfiguration Sequence

The following sequence occurs between a client and a server during a GetConfiguration request.

1. The client sends a GetConfiguration request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a "200 OK" HTTP response.
4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single GetConfiguration request.

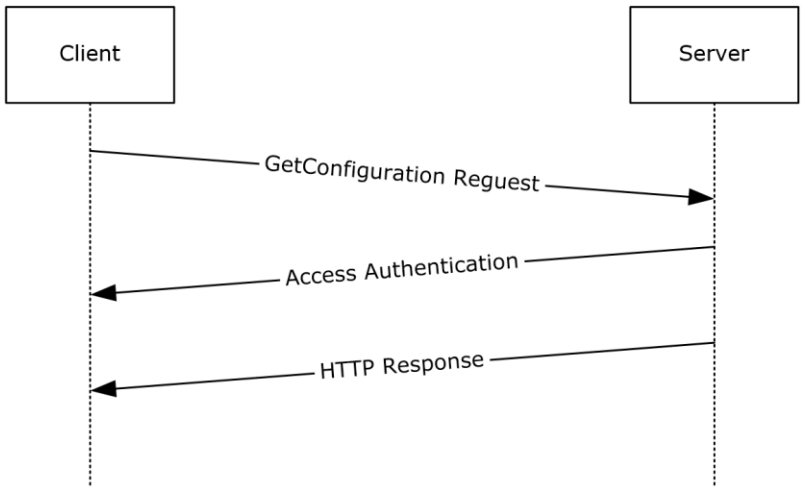

Figure 1: Message sequence for a single GetConfiguration request

4.2 GetModule Sequence

The following sequence occurs between a client and a server during a GetModule request.

1. The client sends a GetModule request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a "200 OK" HTTP response.
4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single GetModule request.

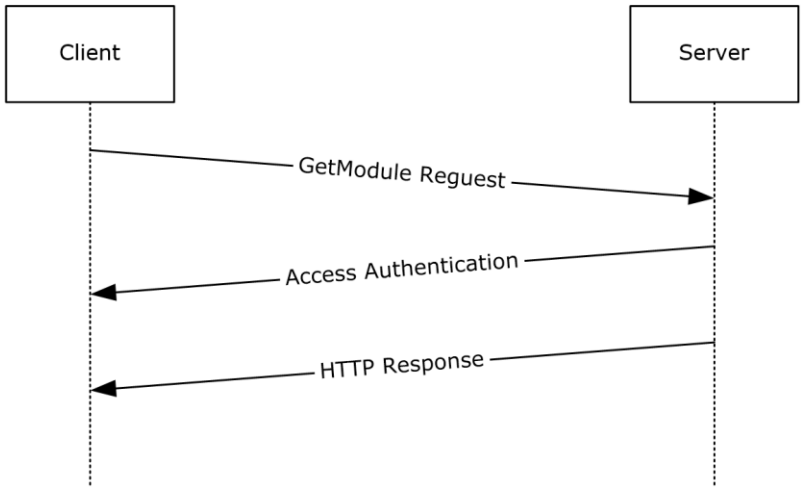

Figure 2: Message sequence for a single GetModule request

4.3 GetAction Sequence

The following sequence occurs between a client and a server during a GetAction request.

1. The client sends a GetAction request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a "200 OK" HTTP response.
4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single GetAction request.

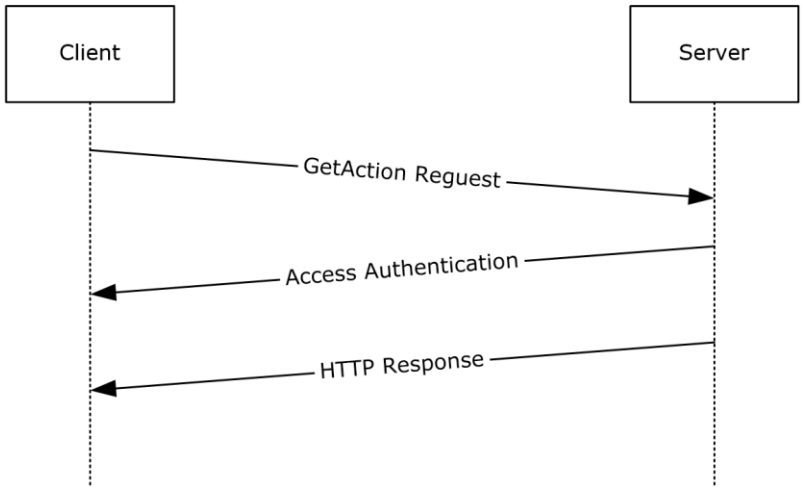

Figure 3: Message sequence with a single GetAction request

4.4 SendStatusReport Sequence

The following sequence occurs between a client and a server during a SendStatusReport request.

1. The client sends a SendStatusReport request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a HTTP 200 (OK) response.
4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single SendStatusReport request.

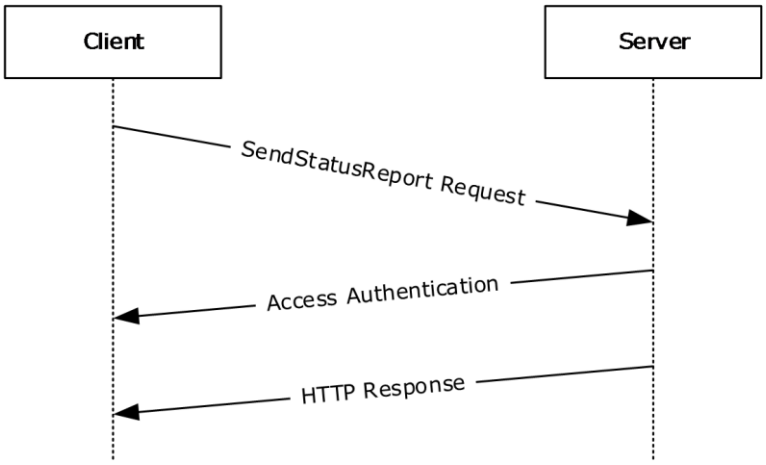

Figure 4: Message sequence with a single SendStatusReport

4.5 GetStatusReport Sequence

The following sequence occurs between a client and a server during a GetStatusReport request.

1. The client sends a GetStatusReport request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a HTTP 200 (OK) response.
4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single GetStatusReport request.

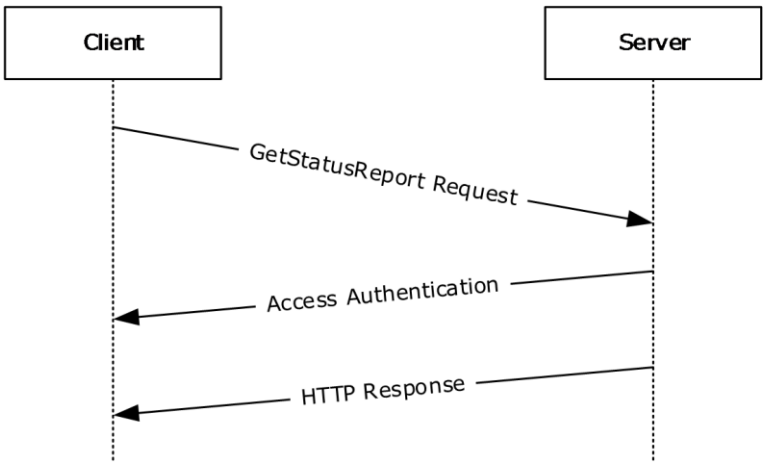

Figure 5: Message sequence with a single GetStatusReport

4.6 RegisterDscAgent Sequence

Note: All of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

The following sequence occurs between a client and a server during a RegisterDscAgent request.

1. The client sends a RegisterDscAgent request.
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [RFC2616] section 11.
3. If authentication is not required, or if authentication has succeeded, the server responds with a HTTP 200 (OK) response.

4. The client closes the TCP connection to the server.

The following figure shows a message sequence with a single RegisterDscAgent request.

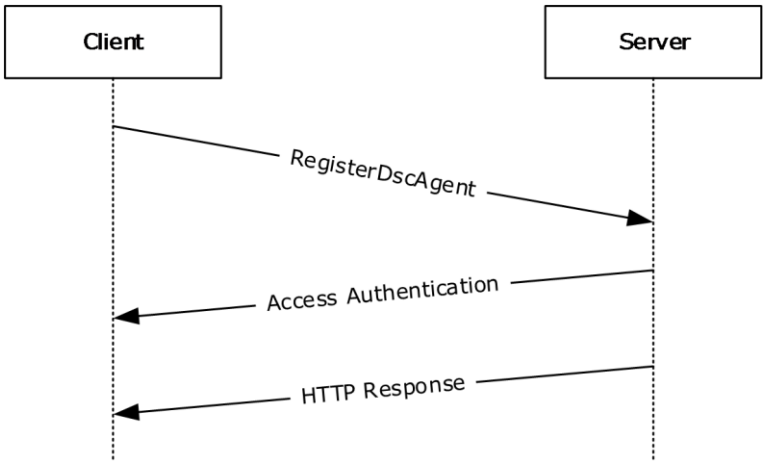

Figure 6: Message sequence with a single RegisterDscAgent

5 Security

5.1 Security Considerations for Implementers

The protocol is vulnerable to a hijacking attack in which the attacker guesses the value of the *ConfigurationId* (as specified in section 3.1.5.1), JobId, and/or *ModuleName*, *ModuleVersion* (as specified in section 3.2.5.1), and the TCP port number used by the client. This approach works because the attacker can establish its own TCP connection to the server and send a request by using the victim's *ConfigurationId*, JobId, and/or *ModuleName*, *ModuleVersion* value. To mitigate the attack, *ConfigurationId* and JobId should be a random value. Also, if HTTP access authentication is used, the server should authenticate access at least once on each new URL or TCP connection.

5.2 Index of Security Parameters

Security parameter	Section
HTTP access authentication	As specified in section 2.1.

6 Appendix A: Full JSON Schema

```
{
    "title": "GetAction request schema",
    "type" : "object",
    "properties": {
        "Checksum": {
            "type": ["string","null"]
            },
        "ConfigurationName": {
            "type": ["string","null"]
            },
        "NodeCompliant": {
            "type": "boolean"

            },
        "ChecksumAlgorithm": {
            "enum": ["SHA-256"],
            "description": "Checksum algorithm used to generate checksum"
            }
        },
    "required": ["Checksum", "NodeCompliant", "ChecksumAlgorithm"]
}
{
    "title": "GetAction response",
    "type": "object",
    "properties": {
        "value": {
        "enum": ["OK", "GetConfiguration", "Retry"],
        "required": "true"
        }
    }
}
```

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.

Note: Some of the information in this section is subject to change because it applies to a preliminary product version, and thus may differ from the final version of the software when released. All behavior notes that pertain to the preliminary product version contain specific references to it as an aid to the reader.

- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system
- Windows 10 operating system
- Windows Server 2016 Technical Preview 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 2.2.2.4: The ConfigurationName header field is available in Windows 10 and Windows Server 2016 Technical Preview only.

<2> Section 2.2.2.5: The ProtocolVersion header field is available in Windows 10 v1511 operating system and Windows Server 2016 Technical Preview only.

<3> Section 2.2.2.6: The AgentId header field is available in Windows 10 v1511 and Windows Server 2016 Technical Preview only.

<4> Section 2.2.2.7: The Authorization header field is available in Windows 10 v1511 and Windows Server 2016 Technical Preview only.

<5> Section 3.3.5.1.1.2: Retry is available in Windows 10 and Windows Server 2016 Technical Preview only.

<6> Section 3.4: The SendStatusReport request is available in Windows Server 2016 Technical Preview and Windows 10 only.

<7> Section 3.5: The GetStatusReport request is available in Windows 10 and Windows Server 2016 Technical Preview only.

<8> Section 3.6: The RegisterDscAgent request is available in Windows 10 v1511 and Windows Server 2016 Technical Preview only.

8 Change Tracking

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

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

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

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.3 Overview	Updated content for Windows 10 and Windows Server 2016 Technical Preview.	Y	Content update.
2.2.2.4 ConfigurationName	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
2.2.2.5 ProtocolVersion	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
2.2.2.6 AgentId	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
2.2.2.7 Authorization	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6 RegisterDscAgent Details	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.1 Abstract Data Model	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.2 Timers	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.3 Initialization	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.4 Higher-Layer Triggered Events	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.5 Message Processing Events and Sequencing Rules	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.5.1 Nodes(AgentId={AgentId})	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.5.1.1 PUT	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.6.5.1.1.1 Request Body	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.5.1.1.2 Response Body	Added section with content for WIndows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.5.1.1.3 Processing Details	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.6 Timer Events	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
3.6.7 Other Local Events	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.
4.6 RegisterDscAgent Sequence	Added section with content for Windows 10 and Windows Server 2016 Technical Preview.	Y	New content added.

9 Index

R

S

T

U

V