[MS-DSCPM]:
Desired State Configuration Pull Model Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
08/08/2013	1.0	New	Released new document.
11/14/2013	2.0	Major	Significantly changed the technical content.
02/13/2014	2.1	Minor	Clarified the meaning of the technical content.
05/15/2014	2.1	No change	No changes to the meaning, language, or formatting of the technical content.

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 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]:

Augmented Backus-Naur Form (ABNF)
binary large object (BLOB)
checksum
Hypertext Transfer Protocol (HTTP)
Managed Object Format (MOF)
Transmission Control Protocol (TCP)
Uniform Resource Locator (URL)
universally unique identifier (UUID)
URI

The following terms are specific to this document:

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.

module: Represents a **BLOB**. The protocol does not process the content of the **BLOB** and it is passed as it is to the higher layer.

action: Represents a string that is returned as part of a GetAction response.

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.

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

[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.ietf.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.ietf.org/rfc/rfc4648.txt

[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

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, http://www.unicode.org/

1.2.2 Informative References

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

[DMTF-DSP0004] Distributed Management Task Force, "Common Information Model (CIM) Infrastructure Specification", version 2.3, October 2005, http://www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf

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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

1.3 Overview

The Desired State Configuration Pull Model Protocol is used 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"

2.2.3 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 table **ConfigurationTable** where each entry contains:

ServerConfigurationId: A unique ConfigurationId (section 2.2.3.1).

ServerConfigurationData: A **BLOB** of data. 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 with the **ServerConfigurationId** in the **ConfigurationTable**. The server MUST use case-insensitive ordinal comparison to match ConfigurationId. 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.

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.
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 might have multiple configuration points and different configuration points may 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]

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 binary 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.

Status code	Reason phrase	Description
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 might have multiple action points and different action points may have different access permissions associated with them. For example, some action points may require HTTP access authentication (as specified in [RFC2616] section 11). As another example, action points may 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.

3.3.5.1.1.2 Response Body

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

Value: MUST be either GetConfiguration 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.

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.

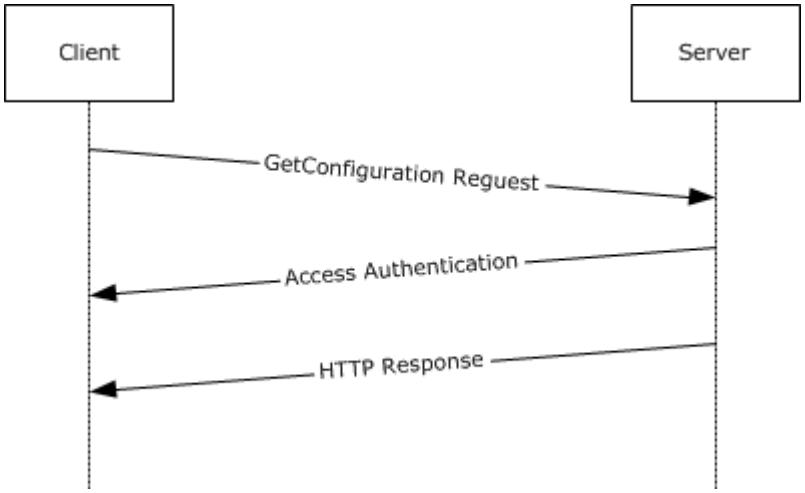

Figure 1: Message sequence for a single GetConfiguration request

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.

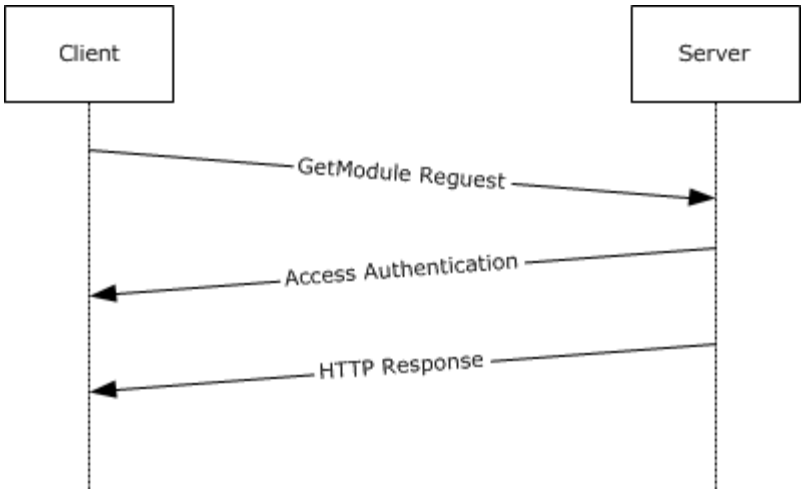

Figure 2: Message sequence for a single GetModule request

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.

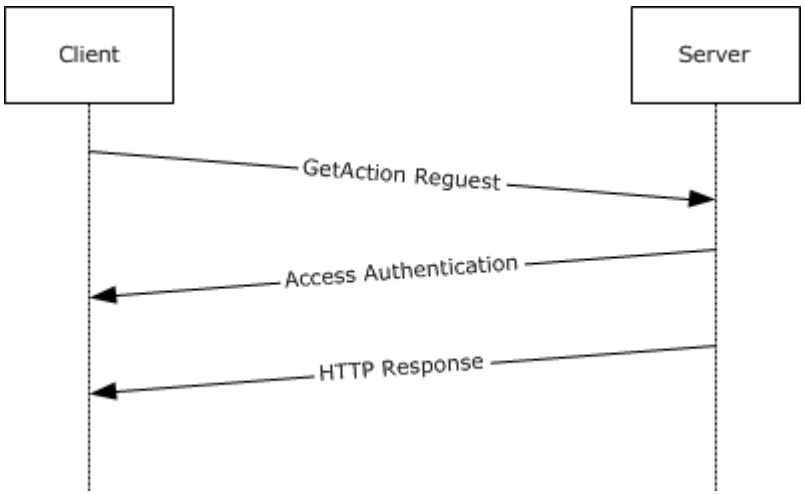

Figure 3: Message sequence with a single GetAction request

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) 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* and/or *ModuleName*, *ModuleVersion* value. To mitigate the attack, *ConfigurationId* 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"]
            },
        "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"],
        "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:

- 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

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

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

8 Change Tracking

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

9 Index