

MS-WDVMODUU Test Suite Specification

**Abstract:** This document provides information about how to configure the test suite and how the MS-WDVMODUU test suite is designed to test MS-WDVMODUU Open Specification usability and accuracy. It describes test assumptions, scope and constraints of the test suite. It also specifies test scenarios, detail test cases, test suite architecture and adapter design.

Contents

[1 Configuring the test suite 4](#_Toc402372046)

[1.1 Configuring the test suite client 4](#_Toc402372047)

[1.1.1 Configuring the test suite client manually 4](#_Toc402372048)

[1.1.2 Configuring the test suite client by scripts 5](#_Toc402372049)

[1.2 Configuring the system under test (SUT) 5](#_Toc402372050)

[1.2.1 Configuring the SUT manually 5](#_Toc402372051)

[1.2.2 Configuring the SUT by scripts 5](#_Toc402372052)

[1.3 Configuring the SHOULD/MAY requirements 5](#_Toc402372053)

[2 Test suite design 6](#_Toc402372054)

[2.1 Assumptions, scope and constraints 6](#_Toc402372055)

[Assumptions 6](#_Toc402372056)

[Scope 6](#_Toc402372057)

[In scope 6](#_Toc402372058)

[Out of scope 6](#_Toc402372059)

[Constraints 6](#_Toc402372060)

[2.2 Test suite architecture 6](#_Toc402372061)

[2.3 Technical dependencies and considerations 8](#_Toc402372062)

[Technical dependencies 8](#_Toc402372063)

[Encryption consideration 8](#_Toc402372064)

[2.4 Adapter design 8](#_Toc402372065)

[2.4.1 Adapter overview 8](#_Toc402372066)

[Protocol adapter 8](#_Toc402372067)

[2.4.2 Technical feasibility of adapter approach 8](#_Toc402372068)

[Message generation 8](#_Toc402372069)

[Message consumption 8](#_Toc402372070)

[2.4.3 Adapter abstract layer 9](#_Toc402372071)

[Protocol adapters 9](#_Toc402372072)

[MS-WDVMODUU adapter interface 9](#_Toc402372073)

[2.4.4 Adapter details 9](#_Toc402372074)

[*2.4.4.1* Protocol adapter 9](#_Toc402372075)

[2.5 Test scenarios 10](#_Toc402372076)

[2.5.1 S01\_XVirusInfectedHeader 11](#_Toc402372077)

[Description 11](#_Toc402372078)

[Operations 11](#_Toc402372079)

[Prerequisites 11](#_Toc402372080)

[Cleanup 11](#_Toc402372081)

[2.5.2 S02\_IgnoredHeaders 11](#_Toc402372082)

[Description 11](#_Toc402372083)

[Operations 11](#_Toc402372084)

[Prerequisites 11](#_Toc402372085)

[Cleanup 11](#_Toc402372086)

[2.5.3 S03\_PropFindExtension 12](#_Toc402372087)

[Description 12](#_Toc402372088)

[Operations 12](#_Toc402372089)

[Prerequisites 12](#_Toc402372090)

[Cleanup 12](#_Toc402372091)

[2.6 Test case design 12](#_Toc402372092)

[2.6.1 Traditional test case design 12](#_Toc402372093)

[2.6.2 Test case description 13](#_Toc402372094)

# Configuring the test suite

## Configuring the test suite client

### Configuring the test suite client manually

Before you run the test suite, update the values in the MS-SITESS\_TestSuite.deployment.ptfconfig file. The MS-WDVMODUU \_TestSuite.deployment.ptfconfig file can also be configured by running the client setup script.

1. Open MS-WDVMODUU\TestSuite\MS-WDVMODUU\_TestSuite.deployment.ptfconfig.
2. Update the following value to specify the common configuration file.

Property name="CommonConfigurationFileName" value="SharePointCommonConfiguration.deployment.ptfconfig"

**Note**  This property can be removed or set to empty if the required properties are copied to the test suite specific configuration file. Any other changes to this property will cause all test cases in the test suite to fail during execution. The test suite first search through its specific configuration file and use the properties from there if they are defined, before looking for them from the common configuration file (if specified).

1. Update the following properties' values to match SUT settings and configuration.

* Property name="SiteCollectionName" value="MSWDVMODUU\_SiteCollection"
* Property name="Server\_DefaultDocumentLibName" value="MSWDVMODUU\_DocumentLibrary1"
* Property name="Server\_DefaultDocLibUri" value="http://[SutComputerName]/sites/[SiteCollectionName]/[Server\_DefaultDocumentLibName]/"
* Property name="Server\_NewFile001Uri" value="[Server\_DefaultDocLibUri]MSWDVMODUU\_TestData1.txt"
* Property name="Server\_NewFile002Uri" value="[Server\_DefaultDocLibUri]MSWDVMODUU\_TestData2.txt"
* Property name="Server\_TestTxtFileUri\_Put" value="[Server\_DefaultDocLibUri][Client\_TestTxtFileName]"
* Property name="Server\_SubFolderUri" value="[Server\_DefaultDocLibUri]MSWDVMODUU\_TestFolder/"
* Property name="Server\_NewFile003Uri" value="[Server\_SubFolderUri]MSWDVMODUU\_TestData3.txt"
* Property name="Server\_NewFile004Uri" value="[Server\_SubFolderUri][Client\_NewFile004Name]"
* Property name="Server\_TestDocumentLibName" value="MSWDVMODUU\_DocumentLibrary2"
* Property name="Server\_TestDocLibUri" value="http://[SutComputerName]/sites/[SiteCollectionName]/[Server\_TestDocumentLibName]/"
* Property name="Server\_FakeVirusInfectedFileUri\_Get" value="[Server\_TestDocLibUri]FakeVirusInfectedFile\_Get.txt"
* Property name="Server\_FakeVirusInfectedFileUri\_Put" value="[Server\_TestDocLibUri]FakeVirusInfectedFile\_Put.txt"

1. The following properties are not associated with SUT settings and can normally retain with default values.

* Property name="Client\_TestTxtFileName" value="Test.txt"
* Property name="Client\_NewFile004Name" value="New\_File004.txt"
* Property name="Client\_FakeVirusInfectedFileName" value="FakeVirusInfectedFile.txt" (The property value is the fake virus file name, and the file must be copied to the local output path of the client, such as: <Solution Directory>\TestSuite\Resources\.)

### Configuring the test suite client by scripts

To configure the test suite client using scripts, see section 5.2.4 of the [SharePointTestSuiteDeploymentGuide.docx](../SharePointTestSuiteDeploymentGuide.docx).

After running the scripts successfully, a fake virus file that can be detected by the valid virus scanner software installed in the SUT should be copied to the local output path (such as: <Solution Directory>\TestSuite\Resources\) of the client. The file name should be same as the value of the property “Client\_FakeVirusInfectedFileName”. (See more details in section 1.1.1 step 4.)

## Configuring the system under test (SUT)

### Configuring the SUT manually

To manually configure the SUT, see section 5.1.3 of the [SharePointTestSuiteDeploymentGuide.docx](../SharePointTestSuiteDeploymentGuide.docx).

### Configuring the SUT by scripts

To configure the SUT using scripts, see section 5.1.2 of the [SharePointTestSuiteDeploymentGuide.docx](../SharePointTestSuiteDeploymentGuide.docx).

After running the scripts successfully, a fake virus file that can be detected by the valid virus scanner software installed in the SUT should be uploaded to the SUT under the URI, as specified by the PTF property "Server\_FakeVirusInfectedFileUri\_Get".(See more details in MS-WDVMODUU test suite SUT configuration step 5 in section 5.1.3 of the [SharePointTestSuiteDeploymentGuide.docx](../SharePointTestSuiteDeploymentGuide.docx).)

## Configuring the SHOULD/MAY requirements

Implementation of the SHOULD/MAY and endnote-related requirements are pre-configured in the format "<Property name="RXXXEnabled" value="XXXX"/>" for three product versions in the following config files:

* MS-WDVMODUU\_WindowsSharePointServices3\_SHOULDMAY.deployment.ptfconfig
* MS-WDVMODUU\_SharePointServer2007\_SHOULDMAY.deployment.ptfconfig
* MS-WDVMODUU\_SharePointFoundation2010\_SHOULDMAY.deployment.ptfconfig
* MS-WDVMODUU\_SharePointServer2010\_SHOULDMAY.deployment.ptfconfig
* MS-WDVMODUU\_SharePointFoundation2013\_SHOULDMAY.deployment.ptfconfig
* MS-WDVMODUU\_SharePointServer2013\_SHOULDMAY.deployment.ptfconfig

If RXXXEnabled is set to true, the requirement must be checked. If false, the requirement must not be checked. For Microsoft product versions, all values should not be changed. For third-party products, the closest Microsoft product version should be chosen, and the value of RXXXEnabled should be updated according to the real product behavior. For example, if choosing SharePoint Foundation 2010, open **MS-WDVMODUU\_SharePointFoundation2010\_SHOULDMAY.deployment.ptfconfig** and update the RXXXEnabled accordingly.

# Test suite design

## Assumptions, scope and constraints

Assumptions

* None

Scope

In scope

* MS-WDVMODUU test suite will verify the accuracy and integrity of the technical content in the Open Specification against the results returned from the protocol server by using MS-WDVMODUU defined extension headers and extension properties:
* X-Virus-Infected Header
* Repl: collblob XML Element
* Repl: repl XML element collection
* MS-WDVMODUU test suite will verify the requirements by running all the test cases on HTTP/1.1.

Out of scope

* MS-WDVMODUU test suite will not verify the internal implementations of its transport protocol stack.  
  MS-WDVMODUU test suite will not verify the server internal behaviors which are not observable on the wire.

Constraints

* None

## Test suite architecture

The MS-WDVMODUU test suite validates the requirements obtained from the Open Specification. The following figure shows the architecture of MS-WDVMODUU test suite.



The architecture of the test suite

The details of the MS-WDVMODUU test suite architecture

* SUT hosts the WebDAV Service which this test suite runs against.
* SUT is the protocol server implementation from third-party user’s point of view.
* SUT implements the Web DAV Service and the extension headers and extension properties that defined in [MS-WDVMODUU].
* The test suite is used to test the MS-WDVMODUU protocol implementation against the following SharePoint versions.
* Windows SharePoint Services 3.0 Service Pack 3 (SP3)
* Microsoft SharePoint Foundation 2010 Service Pack 2 (SP2)
* Microsoft SharePoint Foundation 2013 SP1
* Microsoft Office SharePoint Server 2007 Service Pack 3 (SP3)
* Microsoft SharePoint Server 2010 Service Pack 2 (SP2)
* Microsoft SharePoint Server 2013 SP1
* The test suite acts as the client to send request based on HTTP/1.1 to the SUT and verify the responses from the SUT.
* MS-WDVMODUU Adapter Interface declares the interface of the MS-WDVMODUU Adapter, which is used in the test cases.
* The MS-WDVMODUU Adapter contains managed data structures of MS-WDVMODUU and directly send request to the SUT by HTTP/1.1.
* The test cases can use the MS-WDVMODUU Adapter to call some HTTP methods with extension headers and extension properties, and get the HTTP response from the SUT.

## Technical dependencies and considerations

Technical dependencies

* This test suite depends on HTTP/1.1 protocol to transmit the messages.
* This test suite depends on the Protocol Test Framework (PTF) to derive managed adapters and script adapters.

Encryption consideration

* Protocol transportation does not require encryption, so there is no encryption consideration for the MS-WDVMODUU test suite.

## Adapter design

### Adapter overview

Only protocol adapter is designed for MS-WDVMODUU test suite.

Protocol adapter

* MS-WDVMODUU adapter
* The MS-WDVMODUU Protocol Adapter is designed to composite some HTTP methods that are used by test cases.
* The MS-WDVMODUU Protocol Adapter receives the response message from SUT and sends the message to the test cases to the test cases. The work flow is as follows:
* The test case drives the MS-WDVMODUU Adapter directly to invoke and get return values of the MS-WDVMODUU operations (some HTTP methods).
* The Protocol Adapter composite some HTTP method requests according to input parameters that are provided by test cases;
* MS-WDVMODUU Adapter communicates with SUT by sending request over HTTP/1.1 and returns the HTTP response from SUT to the test case.

### Technical feasibility of adapter approach

Message generation

MS-WDVMODUU adapter sends message directly by HTTP/1.1. The message is composited by adapter and sent by HTTP/1.1 methods: DELETE, GET, PROPFIND and PUT. The message will be sent by calling HttpWebRequest in .NET framework.

Message consumption

The message returned from the SUT will be consumed by MS-WDVMODUU adapter. The returned message will be parsed by HttpWebResponse in .NET framework first. Then the adapter will parse the message header and body directly.

### Adapter abstract layer

Protocol adapters

MS-WDVMODUU adapter interface

* There are four methods declared in the MS-WDVMODUU adapter interface IMS\_WDVMODUUAdapter, and two properties are defined in the protocol adapter.
* MS-WDVMODUU Adapter class inherits from IMS\_WDVMODUUAdapter and implements these methods and properties.
* The return type for the four methods is class “WDVMODUUResponse”, the class includes nine public properties to return HTTP response data to test cases.

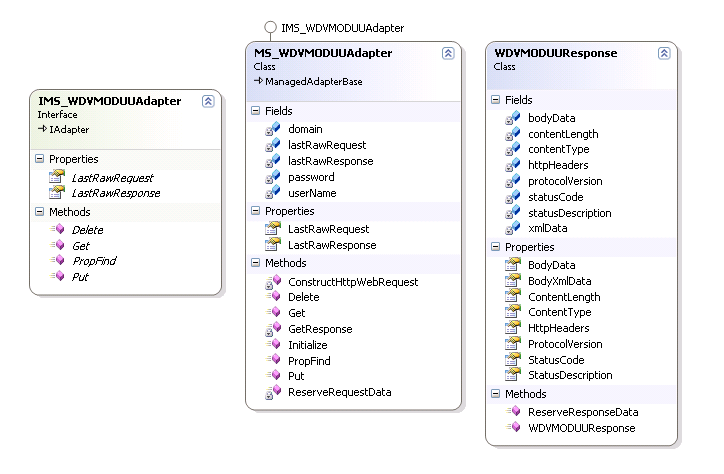
|  |  |
| --- | --- |
| Methods/Properties | Description |
| WDVMODUUResponse Delete(  string requestUri,  NameValueCollection headersCollection) | The method is used to send a HTTP request using DELETE method to the protocol server. As a result, it will return the “HttpWebResponse” object received from the SUT. |
| WDVMODUUResponse Get(  string requestUri,  NameValueCollection headersCollection) | The method is used to send a HTTP request using GET method to the protocol server. As a result, it will return the “HttpWebResponse” object received from the SUT. |
| WDVMODUUResponse PropFind(  string requestUri,  string body,  NameValueCollection headersCollection) | The method is used to send a HTTP request using PROPFIND method to the protocol server. As a result, it will return the “HttpWebResponse” object received from the SUT. |
| WDVMODUUResponse Put(  string requestUri,  byte[] body,  NameValueCollection headersCollection) | The method is used to send a HTTP request using PUT method to the protocol server. As a result, it will return the “HttpWebResponse” object received from the SUT. |
| string LastRawResponse { get; } | Gets the last raw HTTP Response string (include the HTTP headers and XML data if exists) from the protocol server. |
| string LastRawRequest { get; } | Gets the last raw HTTP Request string (include the HTTP headers and body content if exists) that send to the protocol server. |

MS-WDVMODUU adapter interface methods

### Adapter details

#### Protocol adapter

The following figure shows the class diagram of the MS-WDVMODUU adapter.



* Class diagram of MS-WDVMODUU adaptersMS-WDVMODUU documents extension to the Web Distributed Authoring and Versioning (WebDAV) protocols described in [[RFC2518]](http://www.ietf.org/rfc/rfc2518.txt). The extensions include:
* Header updates;
* A property which enables optimized protocol interaction for synchronization;
* A property which allows the server to send clients the virus infection status of a document
* Because this Open Specification is based on WebDAV and it does not define any new method, so the MS-WDVMODUU Adapter is just designed to help the test cases to:
* Construct HTTP methods (such as PROPFIND, GET, PUT and so on) that can include above extensions;
* And then send the HTTP request to the protocol server;
* At last return the whole HTTP response to the test cases.

## Test scenarios

There are three scenarios are designed to cover the in-scope, testable requirements in the MS-WDVMODUU test suite.

The details of the scenarios are as follows:

|  |  |
| --- | --- |
| Scenario | Description |
| S01\_XVirusInfectedHeader | This scenario is designed to test the extension header “XVirusInfectedHeader” in the HTTP GET/PUT response |
| S02\_IgnoredHeaders | Send message to the server by HTTP/1.1 PUT/GET/DELETE requests to test the extension headers that are ignored by protocol server in this protocol. |
| S03\_PropFindExtension | This scenario is designed to test the extension properties in the XML body of HTTP PROPFIND method. |

MS-WDVMODUU Scenarios

### S01\_XVirusInfectedHeader

Description

This scenario is designed to test the extension header “XVirusInfectedHeader” in the HTTP GET/PUT response.

Operations

The extension header “XVirusInfectedHeader”

Prerequisites

* The server has installed valid virus scanner software, and a fake virus file that can be detected by the virus scanner software exists in a URI under the server. The URI should be set as a property value in PTF configuration file.
* And in the client test environment there is a fake virus file that can be detected by the valid virus scanner software. The fake virus file name should be set as a property value in PTF configuration file.

Cleanup

N/A

### S02\_IgnoredHeaders

Description

This scenario is designed to test the seven extension headers that are ignored by the protocol server, and test "SyncMan []" comments that is ignored by the protocol server in the User-Agent header.

Moss-Uid

Moss-Did

Moss-VerFrom

Moss-CBFile

MS-Set-Repl-Uid

MS-BinDiff

X-Office-Version

Operations

* All ignored extension headers.

Prerequisites

N/A

Cleanup

N/A

### S03\_PropFindExtension

Description

This scenario is designed to test the extension properties in the XML body of HTTP PROPFIND method.

Operations

* Repl:collblob Element
* Repl:repl Element Collection

Prerequisites

* In the protocol server, the following resource structure is existed:
* [Root-Folder]
* New\_File001.txt
* New\_File002.txt
* [Sub-Folder]
* New\_File003.txt

Cleanup

N/A

## Test case design

### Traditional test case design

Traditional Testing approach is selected as the testing approach for MS-WDVMODUU.

There are ten test cases designed to cover the three scenarios mentioned in section [2.5 Test scenarios](#_Test_Scenarios).

Details of the traditional test cases are specified in section [2.6.2 test case description](#_Test_Cases_Description).

The scenarios distributions of the test cases are listed in the following table.

|  |  |
| --- | --- |
| Scenario ID | Test case name |
| S01\_XVirusInfectedHeader | [MSWDVMODUU\_S01\_TC01\_XVirusInfectedHeader\_Get](#S01_TC01) |
| [MSWDVMODUU\_S01\_TC02\_XVirusInfectedHeader\_Put](#S01_TC02) |
| S02\_IgnoredHeaders | [MSWDVMODUU\_S02\_TC01\_IgnoredHeaders\_Get](#S02_TC01) |
| [MSWDVMODUU\_S02\_TC02\_IgnoredHeaders\_Put](#S02_TC02) |
| [MSWDVMODUU\_S02\_TC03\_IgnoredHeaders\_UserAgent](#S02_TC03) |
| S03\_PropFindExtension | [MSWDVMODUU\_S03\_TC01\_PropFindExtension\_Resource](#S03_TC01) |
| [MSWDVMODUU\_S03\_TC02\_PropFindExtension\_Resource\_5Minutes](#S03_TC02) |
| [MSWDVMODUU\_S03\_TC03\_PropFindExtension\_Resource\_Depth](#S03_TC03) |
| [MSWDVMODUU\_S03\_TC04\_PropFindExtension\_Resource\_Descendant](#S03_TC04) |
| [MSWDVMODUU\_S03\_TC05\_PropFindExtension\_Repl](#S03_TC05) |

Test case scenario distribution

### Test case description

The steps in the following test cases definitions use methods and parameters in the adapter interfaces directly and all test cases use HTTP/1.1 as transport protocol.

The following tables describe the traditional test cases.

|  |  |
| --- | --- |
| S01\_XVirusInfectedHeader | |
| Test case ID | MSWDVMODUU\_S01\_TC01\_XVirusInfectedHeader\_Get |
| Description | This test case is used to verify the "X-Virus-Infected header" in the response to an HTTP GET request. |
| Prerequisites | The server has installed valid virus scanner software, and a fake virus file that can be detected by the virus scanner software has existed in a URI under the server. The URI should be set as a property value in PTF configuration file. |
| Test execution steps | 1. Call HTTP GET method with the URI that is existed special path for the fake virus file in the server. |
| Cleanup | N/A |

MSWDVMODUU\_S01\_TC01\_XVirusInfectedHeader\_Get

|  |  |
| --- | --- |
| S01\_XVirusInfectedHeader | |
| Test case ID | MSWDVMODUU\_S01\_TC02\_XVirusInfectedHeader\_Put |
| Description | This test case is used to verify the "X-Virus-Infected header" in the response to an HTTP PUT request. |
| Prerequisites | The server has installed valid virus scanner software and in the client test environment there is a fake virus file that can be detected by the valid virus scanner software. The fake virus file path should be set as a property value in PTF configuration file. |
| Test execution steps | 1. Call HTTP PUT method to upload the fake virus file to the server. |
| Cleanup | N/A |

MSWDVMODUU\_S01\_TC02\_XVirusInfectedHeader\_Put

|  |  |
| --- | --- |
| S02\_IgnoredHeaders | |
| Test case ID | MSWDVMODUU\_S02\_TC01\_IgnoredHeaders\_Get |
| Description | This test case is used to partially test that the server ignores some headers in HTTP GET request. |
| Prerequisites | N/A |
| Test execution steps | 1. Call HTTP GET method with following headers:   Moss-Uid  Moss-Did  Moss-VerFrom  Moss-CBFile  MS-Set-Repl-Uid  MS-BinDiff   1. Record the response in the step 1. 2. Call HTTP GET method without following headers:   Moss-Uid  Moss-Did  Moss-VerFrom  Moss-CBFile  MS-Set-Repl-Uid  MS-BinDiff   1. Record the response in the step 3. |
| Cleanup | N/A |

MSWDVMODUU\_S02\_TC01\_IgnoredHeaders\_Get

|  |  |
| --- | --- |
| S02\_IgnoredHeaders | |
| Test case ID | MSWDVMODUU\_S02\_TC02\_IgnoredHeaders\_Put |
| Description | This test case is used to partially test that the server ignores some headers in HTTP PUT request. |
| Prerequisites | N/A |
| Test execution steps | 1. Call HTTP PUT method with following headers to upload a test file to the server:   Moss-Uid  Moss-Did  Moss-VerFrom  Moss-CBFile  MS-Set-Repl-Uid  MS-BinDiff   1. Record the response in the step 1. 2. Call HTTP DELETE method to delete the test file that uploaded in the step 1. 3. Call HTTP PUT method without following headers to upload the same test file in step 1 to the server:   Moss-Uid  Moss-Did  Moss-VerFrom  Moss-CBFile  MS-Set-Repl-Uid  MS-BinDiff   1. Record the response in the step 4. 2. Call HTTP DELETE method to delete the test file that uploaded in the step 4. |
| Cleanup | N/A |

MSWDVMODUU\_S02\_TC02\_IgnoredHeaders\_Put

|  |  |
| --- | --- |
| S02\_IgnoredHeaders | |
| Test case ID | MSWDVMODUU\_S02\_TC03\_IgnoredHeaders\_UserAgent |
| Description | This test case is used to verify that, in Windows SharePoint Services 3.0, the reply of implementation does be the same whether "SyncMan []" is included in the User-Agent Header or not. |
| Prerequisites | The test case is only valid in Windows SharePoint Services 3.0. |
| Test execution steps | 1. In Windows SharePoint Services 3.0, call HTTP GET method with User-Agent header that include "SyncMan []". 2. Record the response in step 1. 3. In Windows SharePoint Services 3.0, call HTTP GET method with User-Agent header that does not include "SyncMan []". 4. Record the response in step 3. |
| Cleanup | N/A |

MSWDVMODUU\_S02\_TC03\_IgnoredHeaders\_UserAgent

|  |  |
| --- | --- |
| S03\_PropFindExtension | |
| Test case ID | MSWDVMODUU\_S03\_TC01\_PropFindExtension\_Resource |
| Description | When the server receives a PROPFIND request with the "Repl:collblob" element set to a timestamp, it includes a "response" element for each resource in the "multistatus" element. This test case is used to verify that each "response" element is a descendant of the "Request-URI" in the PROPFIND request. |
| Prerequisites | N/A |
| Test execution steps | 1. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “infinity”; * Set the time stamp to the default time stamp “ 1969-01-01T12:00:00Z”; |
| Cleanup | N/A |

MSWDVMODUU\_S03\_TC01\_PropFindExtension\_Resonce

|  |  |
| --- | --- |
| S03\_PropFindExtension | |
| Test case ID | MSWDVMODUU\_S03\_TC02\_PropFindExtension\_Resource\_5Minutes |
| Description | When the server receives a PROPFIND request with the "Repl:collblob" element set to a timestamp, it includes a "response" element for each resource in the "multistatus" element. And these return resources should conform to following rules: Rule 1: The resource was last modified later than 5 minutes before the timestamp. OR Rule 2: The resource is a descendant of a resource that was last modified later than 5 minutes before the timestamp. This test case is used to verify the Rule 1. |
| Prerequisites | N/A |
| Test execution steps | 1. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “0”; * Set the time stamp to the default time stamp “ 1969-01-01T12:00:00Z”; * Set the “Request-URI” to a non-collection resource;  1. In the response of step 1, record the last modified date time for the non-collection resource. 2. Call HTTP PROPFIND request again with following settings:  * Set “Depth” header to “0”; * Set the time stamp to the last modified date time of the non-collection resource plus 5 minutes; * Set the “Request-URI” to the non-collection resource;  1. In the response of step 3, make sure the non-collection resource is returned. 2. Call HTTP PROPFIND request in the third time with following settings:  * Set “Depth” header to “0”; * Set the time stamp to the last modified date time of the non-collection resource plus 5 minutes and 1 second; * Set the “Request-URI” to the non-collection resource;  1. In the response of step 5, make sure the non-collection resource is NOT returned. |
| Cleanup | N/A |

MSWDVMODUU\_S03\_TC02\_PropFindExtension\_Resonce\_5 Minutes

|  |  |
| --- | --- |
| S03\_PropFindExtension | |
| Test case ID | MSWDVMODUU\_S03\_TC03\_PropFindExtension\_Resource\_Depth |
| Description | This test case is used to verify that when the server receives a PROPFIND request with the "Repl:collblob" element set to a timestamp, it includes a "response" element for each resource in the "multistatus" element, and the resource is limited by the Depth header specified in [RFC2518]. |
| Prerequisites | In the server, the following resource structure is existed before running this test case.  [Root-Folder]  --- New\_File001.txt  --- New\_File002.txt  --- [Sub-Folder]  --- New\_File003.txt |
| Test execution steps | 1. Call HTTP PROPFIND request with following settings:   Set “Depth” header to “Infinity”;  Set the time stamp to the default time stamp “1969-01-01T12:00:00Z”;  Set the “Request-URI” to “[Root-Folder]”.   1. In the response of step 1, make sure following “resource” are returned:   [Root-Folder]  New\_File001.txt  New\_File002.txt  [Sub-Folder]  New\_File003.txt   1. Call HTTP PROPFIND request with following settings:   Set “Depth” header to “1”;  Set the time stamp to the default time stamp “1969-01-01T12:00:00Z”;  Set the “Request-URI” to “[Root-Folder]”.   1. In the response of step 3, make sure following “resource” are returned:   [Root-Folder]  New\_File001.txt  New\_File002.txt  [Sub-Folder]   1. Call HTTP PROPFIND request with following settings:   Set “Depth” header to “0”;  Set the time stamp to the default time stamp “1969-01-01T12:00:00Z”;  Set the “Request-URI” to “New\_File001.txt”.   1. In the response of step 5, make sure only “resource” “New\_File001.txt” is returned. |
| Cleanup | N/A |

MSWDVMODUU\_S03\_TC03\_PropFindExtension\_Resonce\_Depth

|  |  |
| --- | --- |
| S03\_PropFindExtension | |
| Test case ID | MSWDVMODUU\_S03\_TC04\_PropFindExtension\_Resource\_Descendant |
| Description | When the server receives a PROPFIND request with the "Repl:collblob" element set to a timestamp, it includes a response element for each resource in the "multistatus" element. And these return resources should conform to following rules: Rule 1: The resource was last modified later than 5 minutes before the timestamp. OR Rule 2: The resource is a descendant of a resource that was last modified later than 5 minutes before the timestamp. This test case is used to verify the Rule 2. |
| Prerequisites | In the server, the following resource structure is existed before running this test case.  [Root-Folder]  --- [Sub-Folder]  --- New\_File003.txt |
| Test execution steps | 1. Call HTTP PUT request with following settings:  * Set the “Request-URI” to “[Sub-Folder]\New\_File004.txt”.  1. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “1”; * Set the time stamp to the default time stamp “ 1969-01-01T12:00:00Z”; * Set the “Request-URI” to “[Sub-Folder]”.  1. Based on the response of step 2, calculate the critical time stamp for resource “[Sub-Folder]” and resource “New\_File003.txt”. The critical time stamp is the last modified date time of the resource plus 5 minutes and 1 second. 2. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “1”; * Set the time stamp to the critical time stamp for resource “New\_File003.txt”; * Set the “Request-URI” to “[Sub-Folder]”.  1. In the response of step 4, make sure the resource “New\_File003.txt” is still returned. Because its parent resource “[Sub-Folder]” is returned.   For the resource “New\_File003.txt” , its last modified date time plus 5 minutes is still before the critical time stamp for resource “New\_File003.txt”, but the last modified date time of its parent resource “[Sub-Folder]” is later than 5 minutes before the critical time stamp timestamp for resource “New\_File003.txt”, so the resource “New\_File003.txt” should be returned.   1. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “1”; * Set the time stamp to the critical time stamp for resource “[Sub-Folder]”; * Set the “Request-URI” to “[Sub-Folder]”.  1. In the response of step 6, make sure resource “[Sub-Folder]”, “New\_File003.txt” and “New\_File004.txt” are all not returned. 2. Call HTTP DELETE method to delete the test file “New\_File004.txt” that uploaded in the step 1. |
| Cleanup | N/A |

MSWDVMODUU\_S03\_TC04\_PropFindExtension\_Resonce\_Descendant

|  |  |
| --- | --- |
| S03\_PropFindExtension | |
| Test case ID | MSWDVMODUU\_S03\_TC05\_PropFindExtension\_Repl |
| Description | This test case is used to verify the relevant requirements about the element collection "Repl:repl" and the element "Repl:collblob". |
| Prerequisites | N/A |
| Test execution steps | 1. Call HTTP PROPFIND request with following settings:  * Set “Depth” header to “1”; * Set the time stamp to the default time stamp “ 1969-01-01T12:00:00Z”; |
| Cleanup | N/A |

MSWDVMODUU\_S03\_TC05\_PropFindExtension\_Repl