

MS-VERSS Test Suite Specification

**Abstract:** This document provides information about how to configure the test suite and how the MS-VERSS test suite is designed to test MS-VERSS 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](#_Toc388541600)

[1.1 Configuring the test suite client 4](#_Toc388541601)

[1.1.1 Configuring the test suite client manually 4](#_Toc388541602)

[1.1.2 Configuring the test suite client by scripts 4](#_Toc388541603)

[1.2 Configuring the system under test (SUT) 4](#_Toc388541604)

[1.2.1 Configuring the SUT manually 4](#_Toc388541605)

[1.2.2 Configuring the SUT by scripts 4](#_Toc388541606)

[1.3 Configuring the SHOULD/MAY requirements 5](#_Toc388541607)

[2 Test suite design 6](#_Toc388541608)

[2.1 Assumptions, scope and constraints 6](#_Toc388541609)

[Assumptions 6](#_Toc388541610)

[Scope 6](#_Toc388541611)

[In scope 6](#_Toc388541612)

[Out of scope 6](#_Toc388541613)

[Constraints 6](#_Toc388541614)

[2.2 Test suite architecture 6](#_Toc388541615)

[2.3 Technical dependencies and encryption considerations 8](#_Toc388541616)

[Technical dependencies 8](#_Toc388541617)

[Encryption consideration 8](#_Toc388541618)

[2.4 Adapter design 8](#_Toc388541619)

[2.4.1 Adapter overview 8](#_Toc388541620)

[Protocol adapter 8](#_Toc388541621)

[SUT control adapters 8](#_Toc388541622)

[2.4.2 Technical feasibility of adapter approach 9](#_Toc388541623)

[Message generation 9](#_Toc388541624)

[Message consumption 9](#_Toc388541625)

[SUT control adapter 9](#_Toc388541626)

[2.4.3 Adapter abstract layer 9](#_Toc388541627)

[Protocol adapters 9](#_Toc388541628)

[MS-VERSS adapter interface 9](#_Toc388541629)

[SUT control adapters 10](#_Toc388541630)

[2.4.4 Adapter details 11](#_Toc388541631)

[*2.4.4.1* Protocol adapter 11](#_Toc388541632)

[2.4.4.1.1 MS-VERSS Protocol adapter 11](#_Toc388541633)

[The following outlines details of the class diagram:Adapter interface 12](#_Toc388541634)

[Adapter implementation 12](#_Toc388541635)

[Other class 12](#_Toc388541636)

[Enumeration 12](#_Toc388541637)

[*2.4.4.2* SUT control adapters 13](#_Toc388541638)

[2.4.4.2.1 LISTSWS SUT control adapter 13](#_Toc388541639)

[2.4.4.2.2 SUT control adapter 13](#_Toc388541640)

[2.5 Test scenarios 14](#_Toc388541641)

[2.5.1 S01\_DeleteVersion 14](#_Toc388541642)

[Description 14](#_Toc388541643)

[Operations 14](#_Toc388541644)

[Prerequisites 15](#_Toc388541645)

[Cleanup 15](#_Toc388541646)

[2.5.2 S02\_RestoreVersion 15](#_Toc388541647)

[Description 15](#_Toc388541648)

[Operations 15](#_Toc388541649)

[Prerequisites 15](#_Toc388541650)

[Cleanup 15](#_Toc388541651)

[2.5.3 S03\_ErrorConditions 15](#_Toc388541652)

[Description 15](#_Toc388541653)

[Operations 15](#_Toc388541654)

[Prerequisites 15](#_Toc388541655)

[Cleanup 16](#_Toc388541656)

[2.6 Test case design 17](#_Toc388541657)

[2.6.1 Traditional test case design 17](#_Toc388541658)

[Test case selection 17](#_Toc388541659)

[2.6.2 Test case description 18](#_Toc388541660)

# 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-VERSS\_TestSuite.deployment.ptfconfig file can also be configured by running the client setup script.

1. Open MS-VERSS\TestSuite\MS-VERSS\_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="MSVERSSServiceUrl" value="[TransportType]://[SutComputerName]/sites/[SiteCollectionName]/\_vti\_bin/versions.asmx"
* Property name="MSLISTSWSServiceUrl" value="[TransportType]://[SutComputerName]/sites/[SiteCollectionName]/\_vti\_bin/lists.asmx"
* Property name="RequestUrl" value="[TransportType]://[SutComputerName]/sites/[SiteCollectionName]/"
* Property name="SiteCollectionName" value="MSVERSS\_SiteCollection"
* Property name="DocumentLibraryName" value="MSVERSS\_DocumentLibrary"
* Property name="FileName" value="MSVERSS\_File.txt"
* Property name="NonexistentFileName" value="MSVERSS\_NonexistentFileName.txt"

### 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).

## 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).

## 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-VERSS\_WindowsSharePointServices3\_SHOULDMAY.deployment.ptfconfig
* MS-VERSS\_SharePointServer2007\_SHOULDMAY.deployment.ptfconfig
* MS-VERSS\_SharePointFoundation2010\_SHOULDMAY.deployment.ptfconfig
* MS-VERSS\_SharePointServer2010\_SHOULDMAY.deployment.ptfconfig
* MS-VERSS\_SharePointFoundation2013\_SHOULDMAY.deployment.ptfconfig
* MS-VERSS\_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-VERSS\_SharePointFoundation2010\_SHOULDMAY.deployment.ptfconfig** and update the RXXXEnabled accordingly.

# Test suite design

## Assumptions, scope and constraints

Assumptions

None.

Scope

In scope

* This 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 four operations.
* This test suite will verify the Full WSDL which is provided in the Open Specification.
* This test suite will verify the server-side and testable requirements by running all the test cases on both HTTP and HTTPS.
* This test suite will verify operations over SOAP 1.1 and SOAP 1.2.

Out of scope

* This test suite will not verify the requirements related to client behaviors.
* This test suite will not verify the requirements related to server internal behaviors.
* This test suite will not verify the internal implementations of its transport protocol stack.

Constraints

None.

## Test suite architecture

This test suite verifies the server-side and testable requirements obtained from Open Specification. The following figure shows the architecture of this test suite.

 **The architecture of the test suite**

The details of the MS-VERSS test suite architecture

* SUT hosts the Versions Web Service which this test suite runs against.
* From a third-party’s point of view, the SUT is the protocol server implementation.
* The following products have been tested with the MS-VERSS test suite on Windows platform.
* 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 communicate with the SUT and validates the requirements gathered from the MS-VERSS Open Specification.
* Test cases use the MS-VERSS Adapter to call and get the results of the MS-VERSS operations. Test cases also use the SUT Control Adapter and the LISTSWS SUT Control Adapter to set the SUT to the test case specific situation.
* MS-VERSS Adapter is used in the test cases. The test cases call the methods in the interfaces to invoke the MS-VERSS operations.
* LISTSWS SUT Control Adapter is used in the test cases. The test cases call the methods in the interfaces to configure the SUT.
* SUT Control Adapter is used in the test cases. The test cases call the methods in the interfaces to configure the SUT.

## Technical dependencies and encryption considerations

Technical dependencies

* This test suite depends on the SOAP messaging protocol for exchanging structured data and type information.
* This test suite depends on HTTP protocol or HTTPS protocol to transmit the messages.
* This test suite depends on the wsdl.exe tool in the .NET Framework SDK to generate the MS-VERSS proxy class.
* This test suite depends on the Protocol Test Framework (PTF) to derive managed adapters.
* This test suite depends on the MS-LISTSWS methods to add/delete/retrieve lists in the SUT, and to check in or check out files on the SUT.

Encryption consideration

* Transportation of MS-VERSS includes HTTP and HTTPS, and encryption will be handled by HTTPS.

## Adapter design

### Adapter overview

One protocol adapter and two SUT control adapters will be designed for this test suite.

Protocol adapter

* MS-VERSS protocol adapter
* The MS-VERSS adapter is a managed adapter, which is derived from the ManagedAdapterBase class in PTF.
* The MS-VERSS adapter has the following functionalities.
* Choose HTTP or HTTPS and SOAP 1.1 or 1.2 for transport.
* Construct requests of 4 MS-VERSS operations.
* Communicate with the SUT by sending requests to the SUT and receive the corresponding responses from the SUT.
* Parse the response messages and validate the messages according to the WSDL schema.
* Generate the result log.
* The MS-VERSS adapter uses the C# proxy class, which is generated by running the wsdl.exe tool against the full WSDL of this protocol to send request SOAP messages and receive SOAP response messages. The wsdl.exe can be found in Microsoft .NET Framework SDK tools.

SUT control adapters

* LISTSWS SUT control adapter
* The LISTSWS SUT control adapter is a managed adapter, which is derived from the ManagedAdapterBase class in PTF.
* The LISTSWS SUT control adapter has the following functionalities.
* Add and delete lists on the SUT.
* Retrieve lists from the SUT.
* Check in or check out files on the SUT.
* The LISTSWS SUT control adapter uses the proxy class in the MS-LISTSWS test suite to send request SOAP messages and receive SOAP response messages.
* The LISTSWS SUT control adapter is invoked by the test cases.
* SUT control adapter
* The SUT control adapter will be a scripted adapter.
* The SUT control adapter has the following functionalities.
* Add, modify, delete and retrieve files on the SUT.
* Add sub folder on the SUT.
* Set a specified file’s status as published or unpublished.
* Configure the SUT to enforce the check-out mechanism or not.
* Enable and disable versioning on the SUT.
* The SUT control adapter is invoked by the test cases.

### Technical feasibility of adapter approach

Message generation

The MS-VERSS adapter gets the parameter values of the WSDL operations and calls the corresponding operations in MS-VERSS proxy class, the MS-VERSS proxy class serializes the parameter values to XML elements to format the SOAP request messages, then the SOAP request messages are sent out by the MS-VERSS proxy class.

Message consumption

The messages received from the SUT will be parsed in the MS-VERSS proxy class and be passed upon to the MS-VERSS protocol adapter. Then these messages are consumed in the MS-VERSS protocol adapter to validate the message format and to validate the logic-related requirements in the test cases.

SUT control adapter

The LISTSWS SUT control adapter and the SUT control adapter are designed to remotely control the SUT to set up and clean up the test environment of the SUT and to retrieve file version information from the SUT.

### Adapter abstract layer

Protocol adapters

MS-VERSS adapter interface

There are five methods declared in the MS-VERSS adapter interface IMS\_VERSSAdapter.

* Four of these methods correspond to the four MS-VERSS operations: DeleteAllVersions, DeleteVersion, GetVersions and RestoreVersion. The operators of the four methods are abstracted the same as the operations specified in the MS-VERSS.
* One more method named InitializeUnauthorizedService is used to initialize a protocol web service using incorrect authorization information.

The following table shows the details of these five interface methods.

|  |  |
| --- | --- |
| Interface method name | Description |
| DeleteAllVersions | This method is used to delete all the previous versions of the specified file except the published version and the current version. |
| DeleteVersion | This method is used to delete a specific version of the specified file. |
| GetVersions | This method is used to get details about all versions of the specified file that the user can access. |
| RestoreVersion | This method is used to restore the specified file to a specific version. |
| InitializeUnauthorizedService | This method is used to initialize a protocol web service using incorrect authorization information. |

SUT control adapters

* LISTSWS SUT control adapter interface
* There are five methods declared in the LISTSWS SUT Control Adapter interface ILISTSWSSUTControlAdapter.
* These five methods correspond to the five MS-LISTSWS operations: AddList, DeleteList, GetList, CheckInFile and CheckOutFile. The operators of the five methods are abstracted the same as the operations specified in the MS-LISTSWS.
* The following table shows the details of these five interface methods.

|  |  |
| --- | --- |
| Interface method name | Description |
| AddList | This method is used to create a list in site. |
| CheckInFile | This method is used to check in a file to a document library. |
| CheckoutFile | This method is used to check out a document in a document library. |
| DeleteList | This method is used to delete the specified list in site. |
| GetListID | This method is used to get the id of specified list. |

* SUT Control Adapter Interface
* There are 10 methods declared in the SUT control adapter interface ISUTControlAdapter, the following table shows the details of these ten interface methods.

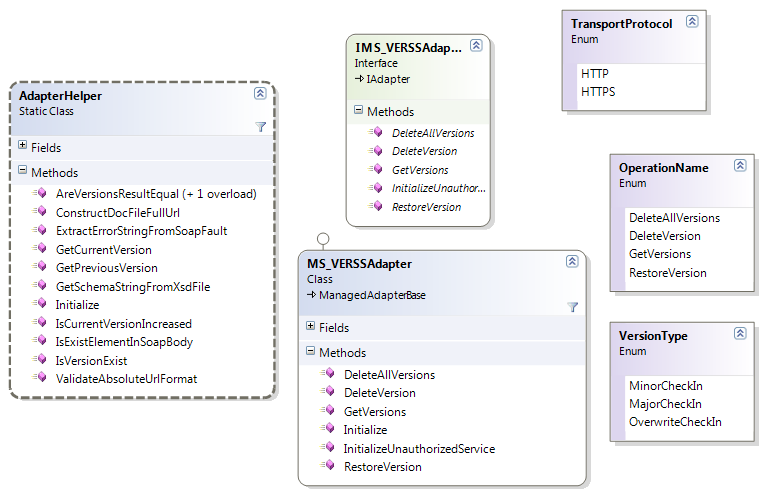
|  |  |
| --- | --- |
| Interface method name | Description |
| AddFile | This method is used to upload a file to the specified list. |
| AddFolder | This method is used to Create a sub folder into the specified list. |
| DeleteItemsInListFromRecycleBin | This method is used to delete the items whose original locations were in the specified list from Recycle Bin. |
| GetFileVersionAttributes | This method is used to get the attributes of the specified version of the file. |
| GetFileVersions | This method is used to get all versions of the specified file in the specified list. |
| IsFileExistInRecycleBin | This method is used to check whether the specified file with the specified version exists in Recycle Bin. |
| SetFilePublish | This method is used to publish a specified file. |
| SetRecycleBinEnable | This method is used to enable or disable the Recycle Bin in the site. |
| SetEnforceCheckout | This method is used to enforce or revoke the check-out mechanism of a specified list. |
| SetVersioning | This method is used to enable or disable the versioning in the list. |

### Adapter details

#### Protocol adapter

##### MS-VERSS Protocol adapter

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



Protocol adapter class diagram

The following outlines details of the class diagram:Adapter interface

* IMS\_VERSSAdapter is the interface of the protocol adapter.
* IMS\_VERSSAdapter defines the methods invoked by test cases, including DeleteAllVersions, DeleteVersion, GetVersions, RestoreVersion and InitializeUnauthorizedService methods.

Adapter implementation

* MS\_VERSSAdapter is the Protocol Adapter class of the test suite. It is used to implement IMS\_VERSSAdapter. It sends the SOAP requests and gets corresponding SOAP responses over HTTP/HTTPS.
* The Initialize method is used to initialize the MS-VERSS test suite.
* The DeleteAllVersions, DeleteVersion, GetVersions and RestoreVersion methods are implemented by generating SOAP request, and then invoking the DeleteAllVersions, DeleteVersion, GetVersions and RestoreVersion methods provided by the MS-VERSS proxy class to send SOAP requests; getting corresponding de-serialized response and verifying related adapter requirements.

Other class

* The AdapterHelper class provides the methods of getting and verifying version information, getting and verifying URL, etc.

Enumeration

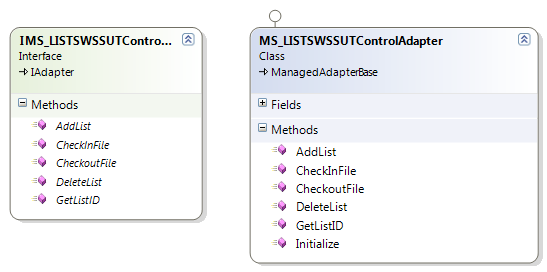
* OperationName is used to specify the MS-VERSS protocol operation names.
* TransportProtocol is used to specify the transport type that the test suite is based on.
* VersionType is used to specify the type of a check in item.

#### SUT control adapters

The following outlines details of the class diagram:

##### LISTSWS SUT control adapter

The following figure show the class diagram of the LISTSWS SUT Control Adapter.



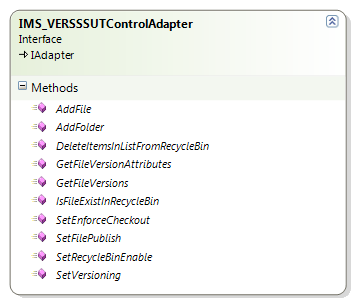
LISTSWS SUT control adapter class diagram

The following outlines details of the class diagram:

* Adapter interface
* ILISTSWSUTControlAdapter is the interface of LISTSWSUTControlAdapter.
* ILISTSWSUTControlAdapter defines the methods invoked by test cases, including AddList, CheckInFile, CheckoutFile, DeleteList and GetListInformation methods.
* Adapter implementation
* LISTSWSUTControlAdapter is the LISTSWS SUT Control Adapter class of the test suite. It is used to implement ILISTSWSUTControlAdapter. It sends the SOAP requests and gets corresponding SOAP responses over HTTP/HTTPS.
* The AddList, CheckInFile, CheckoutFile, DeleteList and GetListInformation methods are implemented by generating SOAP request, invoking the AddList, CheckInFile, CheckOutFile, DeleteList and GetListInformation methods provided by the MS-LISTSWS proxy class to send SOAP requests; getting corresponding de-serialized response.
* The Initialize method is used to initialize the LISTSWS SUT Control Adapter.
* The LISTSWSUTControlAdapter is implemented by managed codes. The implementation can be substituted by other implementation or be configured as interactive mode for the third party’s need.

##### SUT control adapter

The following figure show the class diagram of the SUT control adapter.



SUT control adapter class diagram

* The following outlines details of the class diagram:

The ISUTContorolAdapter is the interface of the SUT control adapter which is implemented by Microsoft PowerShell script. The implementation can be substituted by other implementation for the third party’s need.

## Test scenarios

Three scenarios are designed to cover the server-side, testable requirements in the MS-VERSS test suite. The details of the scenarios are as follows:

|  |  |
| --- | --- |
| Scenario | Description |
| [S01\_DeleteVersion](#_S1_DeleteVersion:_Get_and) | Get and delete versions for a specified file with valid input parameters. |
| [S02\_RestoreVersion](#_S2_RestoreVersion:_Get_and) | Get and restore versions for a specified file with valid input parameters. |
| [S03\_ErrorConditions](#_S3_ErrorConditions:_This_scenario) | Verify various error conditions of the 4 operations. |

MS-VERSS Scenarios

### S01\_DeleteVersion

Description

This scenario is designed to get and delete versions for a specified file with valid input parameters.

Operations

* GetVersions
* DeleteVersion
* DeleteAllVersions

Prerequisites

* Create one document list in an existing MS-VERSS site collection on the server.

Call the LISTSWS SUT Control Adapter method GetListID to get the ID of the created list and save it at client side.

Cleanup

* Delete the lists on the server.
* Delete items in the list from recycle bin on the server.

### S02\_RestoreVersion

Description

This scenario is designed to Get and restore versions for a specified file with valid input parameters.

Operations

* GetVersions
* RestoreVersion

Prerequisites

* Create one document list in an existing MS-VERSS site collection on the server.
* Call the LISTSWS SUT Control Adapter method GetListID to get the ID of the created list and save it at client side.

Cleanup

* Delete the lists on the server.
* Delete items in the list from recycle bin on the server.

### S03\_ErrorConditions

Description

This scenario is aimed to verify various error conditions of the four operations.

Operations

* DeleteAllVersions
* DeleteVersion
* GetVersions
* RestoreVersion

Prerequisites

* Create one document list in an existing MS-VERSS site collection on the server.
* Create a sub folder in an existing list on the server.
* Call the LISTSWS SUT Control Adapter method GetListID to get the ID of the created list and save it at client side.

Cleanup

* Delete the lists on the server.
* Delete items in the list from recycle bin on the server.

## Test case design

### Traditional test case design

Traditional testing approach is selected as the test approach for this test suite. 18 test cases are designed to cover the server-side and testable requirements.

Test case selection

18 traditional test cases are 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\_DeleteVersion | [MSVERSS\_S01\_TC01\_DeleteVersionUsingRelativeUrl](#S1_TC1) |
| [MSVERSS\_S01\_TC02\_DeleteVersionUsingAbsoluteUrl](#S1_TC2) |
| [MSVERSS\_S01\_TC03\_DeleteAllVersionsUsingRelativeUrl](#S1_TC3) |
| [MSVERSS\_S01\_TC04\_DeleteAllVersionsUsingAbsoluteUrl](#S1_TC4) |
| S02\_RestoreVersion | [MSVERSS\_S02\_TC01\_RestoreVersionUsingRelativeUrl](#S2_TC1) |
| [MSVERSS\_S02\_TC02\_RestoreVersionUsingAbsoluteUrl](#S2_TC2) |
| [MSVERSS\_S02\_TC03\_RestoreVersionWithoutEnforceCheckout](#S2_TC3) |
| S03\_ErrorConditions | [MSVERSS\_S03\_TC01\_FileNotFound](#S3_TC1) |
| [MSVERSS\_S03\_TC02\_FileVersionNotFound](#S3_TC2) |
| [MSVERSS\_S03\_TC03\_DeleteCurrentFileVersion](#S3_TC3) |
| [MSVERSS\_S03\_TC04\_DeletePublishedFileVersion](#S3_TC4) |
| [MSVERSS\_S03\_TC05\_VersioningDisabled](#S3_TC5) |
| [MSVERSS\_S03\_TC06\_RestoreVersionNotCheckout](#S3_TC6) |
| [MSVERSS\_S03\_TC07\_RestoreDeletedVersion](#S3_TC7) |
| [MSVERSS\_S03\_TC08\_DeleteAllVersionsUsingInvalidCharacters](#S3_TC8) |
| [MSVERSS\_S03\_TC09\_DeleteVersionUsingInvalidCharacters](#S3_TC9) |
| [MSVERSS\_S03\_TC10\_RestoreVersionUsingInvalidCharacters](#S3_TC10) |
| [MSVERSS\_S03\_TC11\_VerifyHTTPFault](#S3_TC11) |

Test case scenario distribution

Negative testing is used in S03\_ErrorConditions. The client will send invalid messages to the server or correct messages to the server that is in a wrong state, expecting to get a SOAP fault message or a HTTP status code which is used to verify negative requirements as described in the Open Specification.

### Test case description

There are 18 traditional test cases designed in this test suite. The following table describes common prerequisites, common cleanup and common steps for all the test cases:

|  |  |  |
| --- | --- | --- |
| Common prerequisites | The client calls the LISTSWS SUT Control Adapter method AddList to create one document list in the site.  The client calls the LISTSWS SUT Control Adapter method GetListID to get the ID of the list and save it at client side. | |
| Common cleanup | 1. The client calls the LISTSWS SUT Control Adapter method DeleteList to delete the list on the server.   The client calls the LISTSWS SUT Control Adapter method DeleteItemsInListFromRecycleBin to delete the items in list from recycle bin on the server. | |
| Common steps | Verify the response of DeleteAllVersions when versioning is enabled | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |
| Verify the response of DeleteAllVersions when versioning is disabled | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |
| Verify the response of DeleteVersion | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |
| Verify the response of GetVersions when versioning is enabled | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the SUT Control Adapter method GetFileVersions to get all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |
| Verify the response of GetVersions when versioning is disabled | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the SUT Control Adapter method GetFileVersions to get all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |
| Verify the response of RestoreVersion | 1. Client gets the ID of the list saved in Common Prerequisites step 2. 2. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 3. The client calls the SUT Control Adapter method GetFileVersionAttributes to get the attributes of each version of the file. |

Test cases common steps

The steps in the following test case use methods and parameters in the adapter interfaces directly.

The following tables describe the traditional test cases.

|  |  |
| --- | --- |
| S01\_DeleteVersion | |
| Test case ID | MSVERSS\_S01\_TC01\_DeleteVersionUsingRelativeUrl |
| Description | A test case used to test that the client can get expected DeleteVersionSoapOut and GetVersionsSoapOut messages by calling DeleteVersion and GetVersions operations with the relative URL of a file. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 5. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 6. The client calls the SUT Control Adapter method SetRecycleBinEnable to enable the Recycle Bin. 7. The client calls the MS-VERSS Protocol Adapter method DeleteVersion to delete a specific version of the file by using the relative filename. 8. [Common Steps: Verify the response of DeleteVersion](#CSDeleteVersion). 9. The client calls the SUT Control Adapter method IsFileExistInRecycleBin to check whether the deleted version exists in the Recycle Bin. |
| Cleanup | Common Cleanup |

MSVERSS\_S01\_TC01\_DeleteVersionUsingRelativeUrl

|  |  |
| --- | --- |
| S01\_DeleteVersion | |
| Test case ID | MSVERSS\_S01\_TC02\_DeleteVersionUsingAbsoluteUrl |
| Description | A test case used to test that the client can get expected DeleteVersionSoapOut and GetVersionsSoapOut messages by calling DeleteVersion and GetVersions operations with the absolute URL of a file. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file. 5. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 6. The client calls the SUT Control Adapter method SetRecycleBinEnable to enable the Recycle Bin. 7. The client calls the MS-VERSS Protocol Adapter method DeleteVersion to delete a specific version of the file by using the absolute filename. 8. [Common Steps: Verify the response of DeleteVersion](#CSDeleteVersion). 9. The client calls the SUT Control Adapter method IsFileExistInRecycleBin to check whether the deleted version exists in the Recycle Bin. |
| Cleanup | Common Cleanup |

MSVERSS\_S01\_TC02\_DeleteVersionUsingAbsoluteUrl

|  |  |
| --- | --- |
| S01\_DeleteVersion | |
| Test case ID | MSVERSS\_S01\_TC03\_DeleteAllVersionsUsingRelativeUrl |
| Description | A test case used to test that the client can get expected DeleteAllVersionsSoapOut and GetVersionsSoapOut messages by calling DeleteAllVersions and GetVersions operations with the relative URL of a file. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the LISTSWS SUT Control Adapter method SetFilePublish to publish the current version of the file. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. So the current version, which is the new version, is not same as the published version in step 4. 8. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file, which includes the current version and the published version. 9. The client calls the SUT Control Adapter method SetRecycleBinEnable to enable the Recycle Bin. 10. The client calls the MS-VERSS Protocol Adapter method DeleteAllVersions to delete all the versions of the file except the published version and the current version by using the relative filename. 11. [Common Steps: Verify the response of DeleteAllVersions when versioning is enabled](#CSDeleteAllVersionsWithVersioning). 12. The client calls the SUT Control Adapter method IsFileExistInRecycleBin to check whether the deleted versions exist in the Recycle Bin. |
| Cleanup | Common Cleanup |

MSVERSS\_S01\_TC03\_DeleteAllVersionsUsingRelativeUrl

|  |  |
| --- | --- |
| S01\_DeleteVersion | |
| Test case ID | MSVERSS\_S01\_TC04\_DeleteAllVersionsUsingAbsoluteUrl |
| Description | A test case used to test that the client can get expected DeleteAllVersionsSoapOut and GetVersionsSoapOut messages by calling DeleteAllVersions and GetVersions operations with the absolute URL of a file. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the LISTSWS SUT Control Adapter method SetFilePublish to publish the current version of the file. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. So the current version, which is the new version, is not same as the published version in step 4. 8. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file, which includes the current version and the published version. 9. The client calls the SUT Control Adapter method SetRecycleBinEnable to enable the Recycle Bin. 10. The client calls the MS-VERSS Protocol Adapter method DeleteAllVersions to delete all the versions of the file except the published version and the current version by using the absolute filename. 11. [Common Steps: Verify the response of DeleteAllVersions when versioning is enabled](#CSDeleteAllVersionsWithVersioning). 12. The client calls the SUT Control Adapter method IsFileExistInRecycleBin to check whether the deleted versions exist in the Recycle Bin. |
| Cleanup | Common Cleanup |

MSVERSS\_S01\_TC04\_DeleteAllVersionsUsingAbsoluteUrl

|  |  |
| --- | --- |
| S02\_RestoreVersion | |
| Test case ID | MSVERSS\_S02\_TC01\_RestoreVersionUsingRelativeUrl |
| Description | A test case used to test that the client uses the RestoreVersion operation to successfully restore the file specified by a relative URL to a specific version when check out is enforced. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the SUT Control Adapter method SetServerEnforceCheckOut to enforce that only checked out files can be modified. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile to check out the file. 8. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the relative filename to restore the file to a previous version. 9. [Common Steps: Verify the response of RestoreVersion](#CSRestoreVersion). 10. The client calls the LISTSWS SUT Control Adapter methods CheckInFile to check in the file. |
| Cleanup | Common Cleanup |

MSVERSS\_S02\_TC01\_RestoreVersionUsingRelativeUrl

|  |  |
| --- | --- |
| S02\_RestoreVersion | |
| Test case ID | MSVERSS\_S02\_TC02\_RestoreVersionUsingAbsoluteUrl |
| Description | A test case used to test that the client uses the RestoreVersion operation to successfully restore the file specified by an absolute URL to a specific version when check out is enforced. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the SUT Control Adapter method SetServerEnforceCheckOut to enforce that only checked out files can be modified. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile to check out the file. 8. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the absolute filename to restore the file to a previous version. 9. [Common Steps: Verify the response of RestoreVersion](#CSRestoreVersion). 10. The client calls the LISTSWS SUT Control Adapter methods CheckInFile to check in the file. |
| Cleanup | Common Cleanup |

MSVERSS\_S02\_TC02\_RestoreVersionUsingAbsoluteUrl

|  |  |
| --- | --- |
| S02\_RestoreVersion | |
| Test case ID | MSVERSS\_S02\_TC03\_RestoreVersionWithoutEnforceCheckout |
| Description | A test case used to test that the client uses the RestoreVersion operation to successfully restore the file specified by an absolute URL to a specific version when check out is not enforced. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method SetServerEnforceCheckOut to set that check-out is not enforced. 3. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 4. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the MS-VERSS Protocol Adapter method RestoreVersion to restore the file to a previous version. 8. [Common Steps: Verify the response of RestoreVersion](#CSRestoreVersion). 9. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile to check out the file. 10. The client calls the MS-VERSS Protocol Adapter method GetVersions with the absolute filename to get details about all versions of the file. 11. The client calls the MS-VERSS Protocol Adapter method RestoreVersion to restore the file to a previous version. 12. The client calls the LISTSWS SUT Control Adapter methods CheckInFile to check in the second file. |
| Cleanup | Common Cleanup |

MSVERSS\_S02\_TC03\_RestoreVersionWithoutEnforceCheckout

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC01\_FileNotFound |
| Description | A test case used to test that the server returns soap fault message with corresponding error code when the specified file can't be found on server. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the MS-VERSS Protocol Adapter method DeleteAllVersions with a non-exist file name. 2. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with a non-exist file name. 3. The client calls the MS-VERSS Protocol Adapter method GetVersions with a non-exist file name. 4. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with a non-exist file name. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC01\_FileNotFound

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC02\_FileVersionNotFound |
| Description | A test case used to test that the server returns soap fault message with corresponding error code when the specific version of file can't be found on server. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile three times to create three new versions of the file. 4. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 5. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 6. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with a non-exist version. 7. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with a non-exist version. 8. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with an exist version. 9. [Common Steps: Verify the response of DeleteVersion](#CSDeleteVersion). 10. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with the version deleted in step 8. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC02\_FileVersionNotFound

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC03\_DeleteCurrentFileVersion |
| Description | A test case used to test that the current version of a file cannot be deleted. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 5. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 6. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with current version. 7. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file, and verify that the current version has not been deleted. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC03\_DeleteCurrentFileVersion

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC04\_DeletePublishedFileVersion |
| Description | A test case used to test that the published version of a file cannot be deleted. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the LISTSWS SUT Control Adapter method SetFilePublish to publish the current version of the file. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with published version. 8. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file, and verify that the published version has not been deleted. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC04\_DeletePublishedFileVersion

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC05\_VersioningDisabled |
| Description | A test case used to test the server responses when the versioning is disabled. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to disable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 4. [Common Steps: Verify the response of GetVersions when versioning is disabled](#CSGetVersionsWithoutVersioning). 5. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with current version. 6. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with current version. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC05\_VersioningDisabled

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC06\_RestoreVersionNotCheckout |
| Description | A test case used to test that, if the file is not already checked out, and the protocol server enforces that only checked out files can be modified, the RestoreVersion operation can't allow restoration of the file. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile three times to create three new versions of the file. 4. The client calls the SUT Control Adapter method SetServerEnforceCheckOut to enforce only check out file can be restore. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the MS-VERSS Protocol Adapter method RestoreVersion to restore the file to a specific version without checking it out. 8. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file, and verify that the file has not been restored. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC06\_RestoreVersionNotCheckout

|  |  |
| --- | --- |
| **S03\_ErrorConditions** | |
| **Test case ID** | MSVERSS\_S03\_TC07\_RestoreDeletedVersion |
| **Description** | A test case used to test that a file cannot be restored to a deleted version. |
| **Prerequisites** | Common Prerequisites |
| **Test execution steps** | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile three times to create three new versions of the file. 4. The client calls the SUT Control Adapter method SetServerEnforceCheckOut to de-enforce the check out. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. [Common Steps: Verify the response of GetVersions when versioning is enabled](#CSGetVersionsWithVersioning). 7. The client calls the SUT Control Adapter method SetRecycleBinEnable to enable the Recycle Bin. 8. The client calls the MS-VERSS Protocol Adapter method DeleteVersion to delete a specific version. 9. [Common Steps: Verify the response of DeleteVersion](#CSDeleteVersion). 10. The client calls the SUT Control Adapter method IsFileExistInRecycleBin to confirm that the deleted version exists in the Recycle Bin. 11. The client calls the MS-VERSS Protocol Adapter method RestoreVersion to restore the file to the deleted version which exists in the Recycle Bin. |
| **Cleanup** | Common Cleanup |

MSVERSS\_S03\_TC07\_RestoreDeletedVersion

|  |  |
| --- | --- |
| **S03\_ErrorConditions** | |
| **Test case ID** | MSVERSS\_S03\_TC08\_DeleteAllVersionsUsingInvalidCharacters |
| **Description** | A test case used to test that the server returns soap fault message with corresponding error code when the client calls the DeleteAllVersions operation with fileName contains invalid characters. |
| **Prerequisites** | Common Prerequisites |
| **Test execution steps** | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile three times to create three new versions of the file. 4. The client calls the MS-VERSS Protocol Adapter method DeleteAllVersions with invalid characters contain pound sign (#), question mark (?), tab (\t) or backward slash (\), the characters before the first pound sign (#) or question mark (?) are the valid relative filename, and the tab (\t) or backward slash (\) is ignored at the end of the filename. 5. [Common Steps: Verify the response of DeleteAllVersions](#CSDeleteAllVersionsWithVersioning). 6. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file after DeleteAllVersions operation execute successfully. 7. The client calls the MS-VERSS Protocol Adapter method DeleteAllVersions with invalid characters. |
| **Cleanup** | Common Cleanup |

MSVERSS\_S03\_TC08\_DeleteAllVersionsUsingInvalidCharacters

|  |  |
| --- | --- |
| **S03\_ErrorConditions** | |
| **Test case ID** | MSVERSS\_S03\_TC09\_DeleteVersionUsingInvalidCharacters |
| **Description** | A test case used to verify that the server returns soap fault message with corresponding error code when the client calls the DeleteVersion operation with fileName contains invalid characters. |
| **Prerequisites** | Common Prerequisites |
| **Test execution steps** | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create three new versions of the file. 4. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create three new versions of the file. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 6. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with the URL contains pound sign (#), question mark (?), tab (\t) or backward slash (\), the characters before the first pound sign (#) or question mark (?) are the valid relative filename, and the tab (\t) or backward slash (\) is ignored at the end of the filename. 7. [Common Steps: Verify the response of DeleteVersion](#CSDeleteVersion). 8. The client calls the MS-VERSS Protocol Adapter method DeleteVersion with invalid characters. |
| **Cleanup** | Common Cleanup |

MSVERSS\_S03\_TC09\_DeleteVersionUsingInvalidCharacters

|  |  |
| --- | --- |
| **S03\_ErrorConditions** | |
| **Test case ID** | MSVERSS\_S03\_TC10\_RestoreVersionUsingInvalidCharacters |
| **Description** | A test case used to verify that the server returns soap fault message with corresponding error code when the client calls the RestoreVersion operation with fileName contains invalid characters. |
| **Prerequisites** | Common Prerequisites |
| **Test execution steps** | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the SUT Control Adapter method AddFolder to create a sub folder into the list. 4. The client calls the SUT Control Adapter method AddFile to upload a file into the new sub folder. 5. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile three times to create three new versions of the file. 6. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create one new versions of the file in the new sub folder. 7. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 8. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the file name that contains Forward slash (/) or Two consecutive dots (..) in the end of fileName. 9. [Common Steps: Verify the response of RestoreVersion](#CSRestoreVersion). 10. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the file name that contains Two consecutive dots (..) in the end of folder name. 11. The client calls the MS-VERSS Protocol Adapter method GetVersions with the relative filename to get details about all versions of the file. 12. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the file name that contains invalid characters. 13. The client calls the MS-VERSS Protocol Adapter method RestoreVersion with the folder name that contains invalid characters. |
| **Cleanup** | Common Cleanup |

MSVERSS\_S03\_TC10\_RestoreVersionUsingInvalidCharacters

|  |  |
| --- | --- |
| S03\_ErrorConditions | |
| Test case ID | MSVERSS\_S03\_TC11\_VerifyHTTPFault |
| Description | A test case used to verify that the server faults can be returned via HTTP status codes. |
| Prerequisites | Common Prerequisites |
| Test execution steps | 1. The client calls the SUT Control Adapter method SetVersioning to enable the versioning of the list. 2. The client calls the SUT Control Adapter method AddFile to upload a file into the list. 3. The client calls the LISTSWS SUT Control Adapter methods CheckOutFile and CheckInFile to create a new version of the file. 4. The client calls the MS-VERSS Protocol Adapter method InitializeUnauthorizedService to initialize the web service by using incorrect authorization information. 5. The client calls the MS-VERSS Protocol Adapter method GetVersions, but since the authorization is incorrect, the server returns HTTPS status code. |
| Cleanup | Common Cleanup |

MSVERSS\_S03\_TC11\_VerifyHTTPStatus