

MS-OUTSPS Test Suite Specification

**Abstract:** This document provides information about how to configure the test suite and how the MS-OUTSPS test suite is designed to test the MS-OUTSPS 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](#_Toc356306337)

[1.1 Configuring the test suite client 4](#_Toc356306338)

[1.1.1 Configuring the test suite client manually 4](#_Toc356306339)

[1.1.2 Configuring the test suite client by scripts 4](#_Toc356306340)

[1.2 Configuring the system under test (SUT) 4](#_Toc356306341)

[1.2.1 Configuring the SUT manually 4](#_Toc356306342)

[1.2.2 Configuring the SUT by scripts 5](#_Toc356306343)

[1.3 Configuring the SHOULD/MAY requirements 5](#_Toc356306344)

[2 Test suite design 6](#_Toc356306345)

[2.1 Assumptions, scope and constraints 6](#_Toc356306346)

[Assumptions 6](#_Toc356306347)

[Scope 6](#_Toc356306348)

[In scope 6](#_Toc356306349)

[Out of scope 6](#_Toc356306350)

[Constraints 6](#_Toc356306351)

[2.2 Test suite architecture 6](#_Toc356306352)

[2.3 Technical dependencies and encryption considerations 8](#_Toc356306353)

[Dependencies 8](#_Toc356306354)

[Encryption consideration 8](#_Toc356306355)

[2.4 Adapter design 8](#_Toc356306356)

[2.4.1 Adapter overview 8](#_Toc356306357)

[Protocol adapter 8](#_Toc356306358)

[SUT control adapter 8](#_Toc356306359)

[2.4.2 Technical feasibility of adapter approach 9](#_Toc356306360)

[Message generation 9](#_Toc356306361)

[Message consumption 9](#_Toc356306362)

[SUT control adapter 9](#_Toc356306363)

[2.4.3 Adapter abstract layer 9](#_Toc356306364)

[Protocol adapter 9](#_Toc356306365)

[MS-OUTSPS adapter interface 9](#_Toc356306366)

[SUT control adapter 9](#_Toc356306367)

[SUT control adapter interface 9](#_Toc356306368)

[2.4.4 Adapter details 10](#_Toc356306369)

[*2.4.4.1* Protocol adapter 10](#_Toc356306370)

[2.4.4.1.1 MS-OUTSPS protocol adapter 10](#_Toc356306371)

[Adapter interface 10](#_Toc356306372)

[Adapter implementation 10](#_Toc356306373)

[*2.4.4.2* SUT control adapter 11](#_Toc356306374)

[2.4.4.2.1 MS-OUTSPS SUT control adapter 11](#_Toc356306375)

[2.5 Test scenarios 11](#_Toc356306376)

[2.5.1 S01\_OperateAttachment 11](#_Toc356306377)

[Description 11](#_Toc356306378)

[Operations 11](#_Toc356306379)

[Prerequisites 12](#_Toc356306380)

[Cleanup 12](#_Toc356306381)

[2.5.2 S02\_OperateListItems 12](#_Toc356306382)

[Description 12](#_Toc356306383)

[Operations 12](#_Toc356306384)

[Prerequisites 12](#_Toc356306385)

[Cleanup 12](#_Toc356306386)

[2.5.3 S03\_CheckListDefination 12](#_Toc356306387)

[Description 12](#_Toc356306388)

[Operations 12](#_Toc356306389)

[Prerequisites 12](#_Toc356306390)

[Cleanup 12](#_Toc356306391)

[2.6 Test case design 13](#_Toc356306392)

[2.6.1 Traditional test case design 13](#_Toc356306393)

[Test case selection 13](#_Toc356306394)

[2.6.2 Test case description 14](#_Toc356306395)

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

1. Open MS-OUTSPS\TestSuite\MS-OUTSPS\_TestSuite.deployment.ptfconfig file.
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="MSOUTSPS\_SiteCollection"
* Property name="TimeZoneIDOfPacificTime" value="13"
* Property name ="TargetServiceUrl" value ="[TransportType]://[SUTComputerName]/sites/[SiteCollectionName]/\_vti\_bin/lists.asmx"

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

* Property name="ListFieldText" value="MSOUTSPSTest"
* Property name="DelayBetweenAddItemAndUpdateItem" value="30"
* Property name="ServiceTimeOut" value="10"
* Property name="MessageDataFileName" value="DiscussionItemMessage.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

All the implementation of the SHOULD/MAY and endnote related requirements are pre-configured in the format "<Property name="RXXXEnabled" value="XXXX"/>" for six Microsoft product versions in six SHOULD/MAY PTFConfig files:

* MS-OUTSPS\_WindowsSharePointServices3\_SHOULDMAY.deployment.ptfconfig
* MS-OUTSPS\_SharePointServer2007\_SHOULDMAY.deployment.ptfconfig
* MS-OUTSPS\_SharePointFoundation2010\_SHOULDMAY.deployment.ptfconfig
* MS-OUTSPS\_SharePointServer2010\_SHOULDMAY.deployment.ptfconfig
* MS-OUTSPS\_SharePointFoundation2013\_SHOULDMAY.deployment.ptfconfig
* MS-OUTSPS\_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 SharePoint Foundation 2010 is chosen,user can open **MS-OUTSPS\_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 eight SOAP operations: AddAttachment, DeleteAttachment, AddDiscussionBoardItem, GetAttachmentCollection, GetList, GetListItemChanges, GetListItemChangesSinceToken, and UpdateListItems.
* This test suite will verify two custom headers “Translate” and “IF-Match” for HTTP GET and HTTP PUT operations.
* This test suite will use three SOAP operations: AddList, UpdateList, and DeleteList which are described in [MS-LISTSWS] Open Specification as adapter operations, in order to perform initialization process.
* This test suite will verify types’ schema definition in this 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 Full WSDL of all SOAP operations described in the Open Specification, because all SOAP operations’ detail definitions are described in [MS-LISTSWS] Open Specification. [MS-LISTSWS] Open Specification is the transfer protocol of [MS-OUTSPS].
* 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

The protocol SUT must implement the protocol described by [MS-LISTSWS].

## Test suite architecture

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

 **The architecture of the test suite**

The details of the MS-OUTSPS test suite architecture.

* SUT hosts the Administration Web Service which this test suite runs against.
* SUT is the protocol server implementation from third-party user’s point of view.
* This test suite was used to test the MS-OUTSPS 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
* Test suite acts as the client to communicate with the SUT and validates the requirements gathered from MS-OUTSPS Open Specification.
* The test cases invoke the MS-OUTSPS adapter to call the MS-OUTSPS operations and validate the response from SUT. Test cases also use the SUT control adapter to set the SUT to the test case specific situation.
* MS-OUTSPS adapter is used in the test cases. The test cases call the methods in the adapter interfaces to invoke the MS-OUTSPS operations.
* The test cases also use the SUT control adapter to set/modify the SUT environment by calling the methods in the SUT control adapter interface to configure the SUT.

## Technical dependencies and encryption considerations

Dependencies

* This test suite depends on the SOAP messaging protocol for exchanging structured data and type information, when using the SOAP operations described in the Open Specification.
* This test suite will use the HTTP or HTTPS protocol directly for exchanging structured data and type information when using the HTTP GET/PUT methods to downloading and uploading files.
* This test suite depends on HTTP or HTTPS protocol to transmit the SOAP messages.
* This test suite will use MS-LISTSWS proxy class, and it is generated by the wsdl.exe tool in the .NET Framework SDK.
* This test suite depends on the Protocol Test Framework (PTF) to drive managed adapters.

Encryption consideration

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

## Adapter design

### Adapter overview

This protocol consists of one protocol adapter and one SUT control adapter.

Protocol adapter

* MS-OUTSPS adapter
* The MS-OUTSPS adapter is a managed adapter, which is derived from the ManagedAdapterBase class in the Protocol Test Framework (PTF).
* The MS-OUTSPS adapter has the following functionalities
* Choose HTTP or HTTPS and SOAP 1.1 or 1.2 for transport;
* Construct requests of three MS-LISTSWS SOAP operations;
* Construct requests of eight MS-OUTSPS SOAP operations.
* Construct requests of two MS-OUTSPS HTTP 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-OUTSPS adapter uses the C# proxy class, which is generated by running the wsdl.exe tool against the full WSDL of this protocol to send SOAP request messages and receive SOAP response messages. The wsdl.exe can be found in Microsoft .NET Framework SDK tools.

SUT control adapter

* The SUT control adapter will be a PowerShell script adapter.
* The SUT Control Adapter is invoked by the test cases.
* The SUT control adapter has the following functionalities
* Upload a file to Document Library on SUT.
* Upload a file to a sub folder of Document Library on SUT.
* Delete a sub folder of Document Library on SUT.
* Make sure interactive mode can be used.

### Technical feasibility of adapter approach

Message generation

The MS-OUTSPS adapter gets the parameter values of the WSDL operations and calls the corresponding operations in MS-OUTSPS proxy class, the MS-OUTSPS 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-OUTSPS proxy class.

Message consumption

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

SUT control adapter

The SUT control adapter is designed to retrieve the site collection property information from the SUT.

### Adapter abstract layer

Protocol adapter

MS-OUTSPS adapter interface

There are 13 methods declared in the MS-OUTSPS adapter interface IMS\_OUTSPSAdapter.

* Eight of the methods correspond to the eight MS-LISTSWS SOAP operations AddAttachment, DeleteAttachment, AddDiscussionBoardItem, GetAttachmentCollection, GetList, GetListItemChanges, GetListItemChangesSinceToken, and UpdateListItems. The operators of the eight methods are abstracted the same as the operations specified in the MS-LISTSWS. These eight methods are referenced in [MS-OUTSPS] Open Specification.
* Three of the methods correspond to the three MS-LISTSWS SOAP operations: AddList, UpdateList, and DeleteList. The operators of the three methods are abstracted the same as the operations specified in the MS-LISTSWS.
* Two of the methods correspond to the two HTTP operations: HTTPGET and HTTPPUT. The operators of the two methods are abstracted the same as the operations PUT and GET methods which are specified in the HTTP protocol. These two methods are referenced in [MS-OUTSPS] Open Specification.

SUT control adapter

SUT control adapter interface

Three methods corresponding with the following there function is declared in the SUT control adapter interface IMS\_OUTSPSSUTControlAdapter.

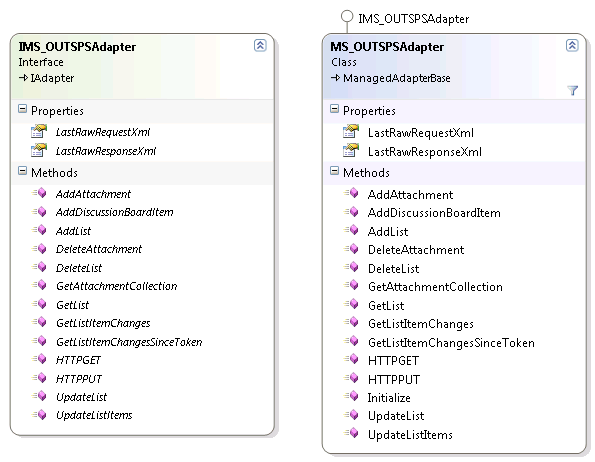
* Upload a file to Document Library on protocol SUT.
* Upload a file to a sub folder of Document Library on protocol SUT.
* Delete a sub folder of Document Library on protocol SUT.

### Adapter details

#### Protocol adapter

##### MS-OUTSPS protocol adapter

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



Protocol adapter class diagram

The following outlines details of the class diagram:

Adapter interface

* IMS\_OUTSPSAdapter is the interface of the protocol adapter.
* IMS\_OUTSPSAdapter defines the 13 methods invoked by test cases. See the list of these 13 methods in section [2.4.3 Adapter Abstract Layer](#_Adapter_abstract_layer).
* IMS\_OUTSPSAdapter defines the LastRawRequestXml and LastRawResponseXml properties to presents the actual SOAP request/response message.

Adapter implementation

* MS\_OUTSPSAdapter is the protocol adapter class of the test suite. It is used to implement IMS\_OUTSPSAdapter.
* The Initialize method is used to initialize the MS-OUTSPS test suite.

#### SUT control adapter

##### MS-OUTSPS SUT control adapter

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



SUT control adapter class diagram

The following outlines details of the class diagram:

The IMS\_OUTSPSSUTControlAdapter 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 is designed to cover the in-scope, testable requirements in the MS-OUTSPS test suite. The details of the scenarios are as follows.

|  |  |
| --- | --- |
| Scenario | Description |
| [S01\_OperateAttachment](#S01_OperateAttchment) | The client tries to add an attachment on a list item, perform update/delete operation on the attachment. |
| [S02\_OperateListItems](#S02_OperateListItems) | The client tries to add list items on a specified list, and perform update/delete operation on these list items, and sync the list items changes from the protocol SUT. |
| [S03\_CheckListDefination](#S03_CheckListDefination) | The client tries to get the list definition from the protocol SUT, and verify the field definition of specified list. |

MS-OUTSPS scenarios

### S01\_OperateAttachment

Description

The client tries to add an attachment on a list item, perform update/delete operation on the attachment.

Operations

* AddAttachment
* DeleteAttachment
* HTTP GET
* HTTP PUT
* GetAttachmentCollection

Prerequisites

N/A

Cleanup

N/A

### S02\_OperateListItems

Description

The client tries to add list items on a specified list, and perform update/delete operation on these list items, and sync the list items changes from the protocol SUT.

Operations

* AddDiscussionBoardItem
* GetListItemChangesSinceToken
* GetListItemChanges
* UpdateListItems

Prerequisites

N/A

Cleanup

N/A

### S03\_CheckListDefination

Description

The client tries to get the list definition from the protocol SUT, and verify the field definition of specified list.

Operations

GetList

Prerequisites

N/A

Cleanup

N/A

## Test case design

### Traditional test case design

Traditional testing approach is selected as the test approach for this test suite. 64 test cases are designed to cover the server-side and testable requirements.

Test case selection

64 traditional test cases are designed to cover the three scenarios mentioned in section [2.5 Test scenarios](#TestScenario). Details of the traditional test cases are specified in section [2.6.2 Test case description](#TestCaseDescription). The scenarios distributions of the test cases are listed in the following table.

|  |  |
| --- | --- |
| Scenario ID | Test case name |
| S01\_OperateAttachment | [MSOUTSPS\_S01\_TC01\_OperateAttachment\_AppointmentTemplateType](#S01_TC01) |
| [MSOUTSPS\_S01\_TC02\_OperateAttachment\_ContactsTemplateType](#S01_TC02) |
| [MSOUTSPS\_S01\_TC03\_OperateAttachment\_DiscussionBoardTemplateType](#S01_TC03) |
| [MSOUTSPS\_S01\_TC04\_OperateAttachment\_TasksTemplateType](#S01_TC04) |
| [MSOUTSPS\_S01\_TC05\_AddAttachment\_Fail](#S01_TC05) |
| S02\_OperateListItems | [MSOUTSPS\_S02\_TC01\_OperationListItemsForAppointment](#S02_TC01) |
| [MSOUTSPS\_S02\_TC02\_OperationListItems\_fAllDayEvent](#S02_TC02) |
| [MSOUTSPS\_S02\_TC03\_OperationListItems\_EventTypeRecurring](#S02_TC03) |
| [MSOUTSPS\_S02\_TC04\_OperationListItems\_EventTypeNotRecurring](#S02_TC04) |
| [MSOUTSPS\_S02\_TC05\_OperationListItems\_fAllDayEventIsTrue](#S02_TC06) |
| [MSOUTSPS\_S02\_TC06\_OperationListItems\_fRecurrenceValues1](#S02_TC07) |
| [MSOUTSPS\_S02\_TC07\_OperationListItems\_fRecurrenceValues0](#S02_TC08) |
| [MSOUTSPS\_S02\_TC08\_OperationListItems\_ExceptionItemForMasterSeriesItemID](#S02_TC09) |
| [MSOUTSPS\_S02\_TC09\_OperationListItems\_ExceptionItemForRecurrenceID](#S02_TC10) |
| [MSOUTSPS\_S02\_TC10\_OperationListItems\_UID](#S02_TC12) |
| [MSOUTSPS\_S02\_TC11\_OperationListItems\_XMLTZoneMissing](#S02_TC13) |
| [MSOUTSPS\_S02\_TC12\_OperationListItems\_UIDIgnored](#S02_TC14) |
| [MSOUTSPS\_S02\_TC13\_OperationListItemsForAppointment\_XMLTZoneValid](#S02_TC13New) |
| [MSOUTSPS\_S02\_TC14\_RecurrenceAppointmentItem\_VerifyUIDField](#S02_TC16) |
| [MSOUTSPS\_S02\_TC15\_OperateOnListItems\_VerifyChangeTypeValue](#S02_TC15) |
| [MSOUTSPS\_S02\_TC16\_RecurrenceAppointmentItem\_RecurrenceDataMIssing](#S02_TC18) |
| [MSOUTSPS\_S02\_TC17\_RecurrenceAppointmentItem\_RecurrenceDataValid](#S02_TC19) |
| [MSOUTSPS\_S02\_TC18\_RecurrenceAppointmentItem\_RecurrenceRulewindowEnd](#S02_TC20) |
| [MSOUTSPS\_S02\_TC19\_RecurrenceAppointmentItem\_RecurrenceRulerepeatForever](#S02_TC21) |
| [MSOUTSPS\_S02\_TC20\_RecurrenceAppointmentItem\_RecurrenceRulerepeatInstances](#S02_TC22) |
| [MSOUTSPS\_S02\_TC21\_RecurrenceAppointmentItem\_RecurrenceDefinition](#S02_TC23) |
| [MSOUTSPS\_S02\_TC22\_RecurrenceAppointmentItem\_RepeatPattern](#S02_TC24) |
| [MSOUTSPS\_S02\_TC23\_OperationListItemsForAppointment\_TransitionDate](#S02_TC25) |
| [MSOUTSPS\_S02\_TC24\_OperationListItems\_TimeZoneRule](#S02_TC26) |
| [MSOUTSPS\_S02\_TC25\_OperationListItems\_DayOfWeekSimpleType](#S02_TC27) |
| [MSOUTSPS\_S02\_TC26\_OperationListItems\_DayOfWeekOrMonthSimpleType](#S02_TC28) |
| [MSOUTSPS\_S02\_TC27\_RecurrenceAppointmentItem\_TrueFalseDOWSimleType](#S02_TC29) |
| [MSOUTSPS\_S02\_TC28\_OperationListItems\_WeekdayOfMonth](#S02_TC30) |
| [MSOUTSPS\_S02\_TC29\_OperationListItems\_UIDUnique](#S02_TC31) |
| [MSOUTSPS\_S02\_TC30\_OperationListItems\_DurationValue](#S02_TC32) |
| [MSOUTSPS\_S02\_TC31\_OperationListItems\_DeleteRecurrence](#S02_TC34) |
| [MSOUTSPS\_S02\_TC32\_OperationListItemsForAppointment\_TimeZone](#S02_TC35) |
| [MSOUTSPS\_S02\_TC33\_OperationListItems\_TimeZoneSetByProtocolSUT](#S02_TC36) |
| [MSOUTSPS\_S02\_TC34\_TriggerExceptionDeletion\_UpdateEndDate](#S02_TC37) |
| [MSOUTSPS\_S02\_TC35\_TriggerExceptionDeletion\_UpdateEventDate](#S02_TC38) |
| [MSOUTSPS\_S02\_TC36\_TriggerExceptionDeletion\_UpdateRecurrenceData](#S02_TC39) |
| [MSOUTSPS\_S02\_TC37\_TriggerExceptionDeletion\_UpdateXMLTZone](#S02_TC40) |
| [MSOUTSPS\_S02\_TC38\_GetListItemChangesSinceToken\_Support](#S02_TC41) |
| [MSOUTSPS\_S02\_TC39\_GetListItemChangesSinceToken\_QueryIsEmpty](#S02_TC42) |
| [MSOUTSPS\_S02\_TC40\_GetListItemChangesSinceToken\_HaveInstances](#S02_TC43) |
| [MSOUTSPS\_S02\_TC41\_AddDiscussionBoardItem](#S02_TC44) |
| [MSOUTSPS\_S02\_TC42\_GetListItemChangesSinceToken\_OptimizeLookups](#S02_TC45) |
| [MSOUTSPS\_S02\_TC43\_DeleteDocumentsAndFolders](#S02_TC46) |
| [MSOUTSPS\_S02\_TC44\_GenericList\_VerifyVtiVersionHistoryValue](#S02_TC47) |
| [MSOUTSPS\_S02\_TC45\_GenericList\_VerifyFieldsInCommonDefinition](#S02_TC48) |
| [MSOUTSPS\_S02\_TC46\_OperateOnListItems\_VerifyContentTypeId](#S02_TC49) |
| [MSOUTSPS\_S02\_TC47\_OperateOnListItems\_VerifyEventTypeInterpretedAsZero](#S02_TC50) |
| [MSOUTSPS\_S02\_TC48\_OperationListItemsForContacts](#S02_TC51) |
| [MSOUTSPS\_S02\_TC49\_OperationListItemsForDiscussion](#S02_TC52) |
| [MSOUTSPS\_S02\_TC50\_OperationListItemsForTasks](#S02_TC53) |
| [MSOUTSPS\_S02\_TC51\_OperationListItemsForDocument](#S02_TC54) |
| [MSOUTSPS\_S02\_TC52\_OperateOnListItems\_VerifyMoreChangeValue](#S02_TC55) |
| [MSOUTSPS\_S02\_TC53\_OperateOnListItems\_VerifyListItemCollectionPositionNextValue](#S02_TC56) |
| S03\_CheckListDefination | [MSOUTSPS\_S03\_TC01\_VerifyAppointmentsList](#S03_TC01) |
| [MSOUTSPS\_S03\_TC02\_VerifyContactsList](#S03_TC02) |
| [MSOUTSPS\_S03\_TC03\_VerifyDiscussionList](#S03_TC03) |
| [MSOUTSPS\_S03\_TC04\_VerifyDocumentLibrary](#S03_TC04) |
| [MSOUTSPS\_S03\_TC05\_VerifyTasksList](#S03_TC05) |
| [MSOUTSPS\_S03\_TC06\_VerifyCHOICESAndMAPPINGSElements\_TasksList](#S03_TC06) |

Test case scenario distribution

The test cases are designed to verify the MS-OUTSPS response messages and the core operations of this protocol. For example, the request message sent to server is actually verified by the server and the response is sent back to the client with correct result.

### Test case description

There are 64 traditional test cases designed in this test suite.

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

The following tables describe the traditional test cases.

|  |  |
| --- | --- |
| Common Cleanup | Delete all lists added by "AddListToSUT" helper method. |

Common cleanup

|  |  |
| --- | --- |
| S01\_OperateAttachment | |
| Test case ID | [MSOUTSPS\_S01\_TC01\_OperateAttachment\_AppointmentTemplateType](#S01_TC01) |
| Description | This test case is used to verify AddAttachment operation, DeleteAttachment operation, GetAttachment operation, and GetAttachmentCollection operation with Appointment template type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Add a ListItem, return the listId. 3. Call method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listItemId * fileName: fileName * attachment: attachmentContent  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent  1. Call method **GetAttachmentCollection** to get the list of all attachments on a single item in one list.   **Input parameters:**   * listName: listName * listItemID: listId  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * Camloptions:   camloptions.QueryOptions.IncludeAttachmentUrls = bool.TrueString;  camloptions.QueryOptions.IncludeAttachmentVersion = bool.TrueString;   1. Call method **HTTP PUT** to overwrite files that are already on the protocol server.   **Input parameters:**   * requestResourceUrl: fullUrlOfAttachment * ifmatchHeader: ifmatchHeaderValue * contentData: newattachmentContents  1. Call method **DeleteAttachment** to delete attachments from an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listId * url: fullUrlOfAttachment  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent |
| Cleanup | Common clean up |

MSOUTSPS\_S01\_TC01\_OperateAttachment\_AppointmentTemplateType

|  |  |
| --- | --- |
| S01\_OperateAttachment | |
| Test case ID | [MSOUTSPS\_S01\_TC02\_OperateAttachment\_ContactsTemplateType](#S01_TC02) |
| Description | This test case is used to verify AddAttachment operation, DeleteAttachment operation, GetAttachment operation, and GetAttachmentCollection operation with Contacts template type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Add a ListItem, return the listId. 3. Call method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listItemId * fileName: fileName * attachment: attachmentContent  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent  1. Call method **GetAttachmentCollection** to get the list of all attachments on a single item in one list.   **Input parameters:**   * listName: listName * listItemID: listId  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * Camloptions:   camloptions.QueryOptions.IncludeAttachmentUrls = bool.TrueString;  camloptions.QueryOptions.IncludeAttachmentVersion= bool.TrueString;   1. Call method **HTTP PUT** to overwrite files that are already on the protocol server.   **Input parameters:**   * requestResourceUrl: fullUrlOfAttachment * ifmatchHeader: ifmatchHeaderValue * contentData: newattachmentContents  1. Call method **DeleteAttachment** to delete attachments from an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listId * url: fullUrlOfAttachment  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent |
| Cleanup | Common clean up |

MSOUTSPS\_S01\_TC02\_OperateAttachment\_ContactsTemplateType

|  |  |
| --- | --- |
| S01\_OperateAttachment | |
| Test case ID | [MSOUTSPS\_S01\_TC03\_OperateAttachment\_DiscussionBoardTemplateType](#S01_TC03) |
| Description | This test case is used to verify AddAttachment operation, DeleteAttachment operation, GetAttachment operation, and GetAttachmentCollection operation with Discussion\_Board template type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Add a ListItem, return the listId. 3. Call method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listItemId * fileName: fileName * attachment: attachmentContent  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent  1. Call method **GetAttachmentCollection** to get the list of all attachments on a single item in one list.   **Input parameters:**   * listName: listName * listItemID: listId  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * Camloptions:   camloptions.QueryOptions.IncludeAttachmentUrls = bool.TrueString;  camloptions.QueryOptions.IncludeAttachmentVersion = bool.TrueString;   1. Call method **HTTP PUT** to overwrite files that are already on the protocol server.   **Input parameters:**   * requestResourceUrl: fullUrlOfAttachment * ifmatchHeader: ifmatchHeaderValue * contentData: newattachmentContents  1. Call method **DeleteAttachment** to delete attachments from an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listId * url: fullUrlOfAttachment  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent |
| Cleanup | Common clean up |

MSOUTSPS\_S01\_TC03\_OperateAttachment\_DiscussionBoardTemplateType

|  |  |
| --- | --- |
| S01\_OperateAttachment | |
| Test case ID | [MSOUTSPS\_S01\_TC04\_OperateAttachment\_TasksTemplateType](#S01_TC04) |
| Description | This test case is used to verify AddAttachment operation, DeleteAttachment operation, GetAttachment operation, and GetAttachmentCollection operation with Tasks template type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Add a ListItem, return the listId. 3. Call method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listItemId * fileName: fileName * attachment: attachmentContent  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent  1. Call method **GetAttachmentCollection** to get the list of all attachments on a single item in one list.   **Input parameters:**   * listName: listName * listItemID: listId  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * Camloptions:   camloptions.QueryOptions.IncludeAttachmentUrls = bool.TrueString;  camloptions.QueryOptions.IncludeAttachmentVersion = bool.TrueString;   1. Call method **HTTP PUT** to overwrite files that are already on the protocol server.   **Input parameters:**   * requestResourceUrl: fullUrlOfAttachment * ifmatchHeader: ifmatchHeaderValue * contentData: newattachmentContents  1. Call method **DeleteAttachment** to delete attachments from an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listId * url: fullUrlOfAttachment  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * listName: listName * listItemID: listItemId * arrachmentFileName: attachmentContent |
| Cleanup | Common clean up |

MSOUTSPS\_S01\_TC04\_OperateAttachment\_TasksTemplateType

|  |  |
| --- | --- |
| S01\_OperateAttachment | |
| Test case ID | [MSOUTSPS\_S01\_TC05\_AddAttachment\_Fail](#S01_TC05) |
| Description | This test case is used to verify AddAttachment operation, create a new attachment failed. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Add a ListItem, return the listId. 3. Call method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listItemId * fileName: fileName * attachment: attachmentContent  1. Recall method **AddAttachment** to create a new attachment on an item on the protocol server.   **Input parameters:**   * listName: listName * listItemID: listId * fileName: fileName * attachment: attachmentContent |
| Cleanup | Common clean up |

MSOUTSPS\_S01\_TC05\_AddAttachment\_Fail

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC01\_OperationListItemsForAppointment |
| Description | This test case is used to verify Appointment template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Description, Duration, Editor, EndDate, EventDate, EventType, fAllDayEvent, fRecurrence, Location, MasterSeriesItemID, RecurrenceData, RecurrenceID, TimeZone, Title, UID, XMLTZon. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC01\_OperationListItemsForAppointment

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC02\_OperationListItems\_fAllDayEvent |
| Description | This test case is used to verify if the fAllDayEvent property is 1 then the time portion of the EventDate MUST be 0 hours UTC. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set EventType values 0, fAllDayEvent values 1. EventDate values UTC fomate. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC02\_OperationListItems\_fAllDayEvent

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC03\_OperationListItems\_EventTypeRecurring |
| Description | This test case is used to verify if the EventType indicates a recurring event, then fRecurrence MUST be 1. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set EventType values 1. fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC03\_OperationListItems\_EventTypeRecurring

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC04\_OperationListItems\_EventTypeNotRecurring |
| Description | This test case is used to verify if the EventType does not indicate a recurring event, then fRecurrence MUST be 0. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set EventType values 0. fRecurrence values 0. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC04\_OperationListItems\_EventTypeNotRecurring

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC05\_OperationListItems\_fAllDayEventIsTrue |
| Description | This test case is used to verify the value of fAllDayEvent 1 means true. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set fAllDayEvent values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC05\_OperationListItems\_fAllDayEventIsTrue

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC06\_OperationListItems\_fRecurrenceValues1 |
| Description | This test case is used to verify the value of fRecurrence 1 means the event is recurring |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC06\_OperationListItems\_fRecurrenceValues1

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC07\_OperationListItems\_fRecurrenceValues0 |
| Description | This test case is used to verify the value of fRecurrence 0 means the event is not recurring. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set fRecurrence values 0. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC07\_OperationListItems\_fRecurrenceValues0

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC08\_OperationListItems\_ExceptionItemForMasterSeriesItemID |
| Description | This test case is used to verify MasterSeriesItemID exists only for exception items. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4,fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add exception items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC08\_OperationListItems\_ExceptionItemForMasterSeriesItemID

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC09\_OperationListItems\_ExceptionItemForRecurrenceID |
| Description | This test case is used to verify RecurrenceID is equal to the starting date and time of one instance of a recurrence when the EventType indicates an exception or deleted instance. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4,fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC09\_OperationListItems\_ExceptionItemForRecurrenceID

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC10\_OperationListItems\_UID |
| Description | This test case is used to verify if fRecurrence is true, UID MUST contain a valid stringGUID. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC10\_OperationListItems\_UID

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC11\_OperationListItems\_XMLTZoneMissing |
| Description | This test case is used to verify if fRecurrence is FALSE, TimeZoneXML can be empty. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC11\_OperationListItems\_XMLTZoneMissing

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC12\_OperationListItems\_UIDIgnored |
| Description | This test case is used to verify if fRecurrence is FALSE, UID is ignored. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0, UID values an invalid GUID string. 3. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. 4. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC12\_OperationListItems\_UIDIgnored

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC13\_OperationListItemsForAppointment\_XMLTZoneValid |
| Description | This test case is used to verify if EventType is 1, then this property MUST contain a valid TimeZoneXML. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   TimeZoneXML values “<timeZoneRule><standardBias>0</standardBias><additionalDaylightBias>0</additionalDaylightBias></timeZoneRule>”.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC13\_OperationListItemsForAppointment\_XMLTZoneValid

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC14\_RecurrenceAppointmentItem\_VerifyUIDField |
| Description | This test case is used to verify UID MUST be changed if the recurrence has been changed or added. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Set the values for the specified fields: UID values a valid and unique GUID value. 3. Call method **UpdateListItem** to add recurrence appointment item that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Set the values for the specified fields: UID values a valid and unique GUID value which is different from the UID value in step2;   Title values unique title string.   1. Call method **UpdateListItem** to update added recurrence appointment item’s Title and UID.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC14\_RecurrenceAppointmentItem\_VerifyUIDField

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC15\_OperateOnListItems\_VerifyChangeTypeValue |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Generic List1 on the server, return listId. 2. Add a Generic List2 on the server, return listId. 3. Call method GetListItemChangesSinceToken to return changeToken.   Input parameters:   * listName: listId of list1 added in step1.  1. Add 10 list item into the list2 added in step2. 2. Call method GetListItemChangesSinceToken with invalid changeToken values   Input parameters:   * listName: listId of list2 added in step2. * viewFields: <ViewFields /> * changeToken: the value get from list1 in step3 |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC15\_OperateOnListItems\_VerifyChangeTypeValue

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC16\_RecurrenceAppointmentItem\_RecurrenceDataMIssing |
| Description | This test case is used to verify if fRecurrence is FALSE, RecurrenceData can be empty or missing. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set EventType values 0, fRecurrence values 0. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC16\_RecurrenceAppointmentItem\_RecurrenceDataMIssing

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC17\_RecurrenceAppointmentItem\_RecurrenceDataValid |
| Description | This test case is used to verify if EventType is 1, RecurrenceData MUST contain a valid RecurrenceXML. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RecurrenceData values validRecurrenceData. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC17\_RecurrenceAppointmentItem\_RecurrenceDataValid

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC18\_RecurrenceAppointmentItem\_RecurrenceRulewindowEnd |
| Description | This test case is used to verify RecurrenceRule complex type and windowEnd element. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RecurrenceRule. windowEnd values validDateTime. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC18\_RecurrenceAppointmentItem\_RecurrenceRulewindowEnd

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC19\_RecurrenceAppointmentItem\_RecurrenceRulerepeatForever |
| Description | This test case is used to verify RecurrenceRule complex type and repeatForever element. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RecurrenceRule. repeatForever values “FALSE”. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC19\_RecurrenceAppointmentItem\_RecurrenceRulerepeatForever

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC20\_RecurrenceAppointmentItem\_RecurrenceRulerepeatInstances |
| Description | This test case is used to verify RecurrenceRule complex type and repeatInstances element. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RecurrenceRule. repeatInstances values 10. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC20\_RecurrenceAppointmentItem\_RecurrenceRulerepeatInstances

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC21\_RecurrenceAppointmentItem\_RecurrenceDefinition |
| Description | This test case is used to verify RecurrenceDefinition complex type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RecurrenceDefinition.rule as valid value. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC21\_RecurrenceAppointmentItem\_RecurrenceDefinition

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC22\_RecurrenceAppointmentItem\_RepeatPattern |
| Description | This test case is used to verify RepeatPattern complex type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 0. Set RepeatPattern.daily values valid data. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC22\_RecurrenceAppointmentItem\_RepeatPattern

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC23\_OperationListItemsForAppointment\_TransitionDate |
| Description | This test case is used to verify TransitionDate complex type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate values valid transitionDate.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC23\_OperationListItemsForAppointment\_TransitionDate

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC24\_OperationListItems\_TimeZoneRule |
| Description | This test case is used to verify TimeZoneRule complex type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   TimeZoneRule values valid timeZoneRule.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC24\_OperationListItems\_TimeZoneRule

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC25\_OperationListItems\_DayOfWeekSimpleType |
| Description | This test case is used to verify DayOfWeek simple type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “su”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “mo”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “tu”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “we”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “th”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “fr”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   TransitionDate.day values valid “sa”.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC25\_OperationListItems\_DayOfWeekSimpleType

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC26\_OperationListItems\_DayOfWeekOrMonthSimpleType |
| Description | This test case is used to verify DayOfWeekOrMonth simple type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “su”. 3. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “mo”. 4. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “tu”. 5. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “we”. 6. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “th”. 7. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “fr”. 8. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “sa”. 9. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “day”. 10. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “weekday”. 11. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RecurrenceRule. firstDayOfWeek values “weekend\_day”. 12. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC26\_OperationListItems\_DayOfWeekOrMonthSimpleType

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC27\_RecurrenceAppointmentItem\_TrueFalseDOWSimleType |
| Description | This test case is used to verify TrueFalseDOW simple type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RepeatPattern. Daily.su values TRUE. 3. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RepeatPattern. Daily.su values FALSE. 4. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RepeatPattern. Daily.su values true. 5. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. Set RepeatPattern. Daily.su values false. 6. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC27\_RecurrenceAppointmentItem\_TrueFalseDOWSimleType

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC28\_OperationListItems\_WeekdayOfMonth |
| Description | This test case is used to verify WeekdayOfMonth simple type. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   RepeatPattern. monthlyByDay.weekdayOfMonth values “ first”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   RepeatPattern. monthlyByDay.weekdayOfMonth values “second”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   RepeatPattern. monthlyByDay.weekdayOfMonth values “third”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   RepeatPattern. monthlyByDay.weekdayOfMonth values “fourth”.   1. Create a Recurrence AppointMent settings: Set EventType values 1,   RepeatPattern. monthlyByDay.weekdayOfMonth values “last”.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC28\_OperationListItems\_WeekdayOfMonth

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC29\_OperationListItems\_UIDUnique |
| Description | This test case is used to verify UID is unique among all other recurring events on this list. |
| Prerequisites | Windows SharePoint Services 3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID1. 3. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID2. 4. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID3. 5. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID4. 6. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID5. 7. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID6. 8. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID7. 9. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID8. 10. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID9. 11. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1, UID values a valid GUID0. 12. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC29\_OperationListItems\_UIDUnique

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC30\_OperationListItems\_DurationValue |
| Description | This test case is used to verify Appointments have a duration value and an ending date and time |
| Prerequisites | Windows SharePoint Services 3.0 and above products |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Single AppointMent settings: Set EndDate values endDataTime, eventDate values eventDataTime, Duration values endDate – eventDate. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC30\_OperationListItems\_DurationValue

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC31\_OperationListItems\_DeleteRecurrence |
| Description | This test case is used to verify When a recurrence is deleted, all exceptions to that recurrence also is deleted as optional behaviors. |
| Prerequisites | Microsoft SharePoint Foundation 2010, Microsoft SharePoint Foundation 2013 |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Call method **UpdateListItem** to delete items that are not discussion items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC31\_OperationListItems\_DeleteRecurrence

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC32\_OperationListItemsForAppointment\_TimeZone |
| Description | This test case is used to verify if fRecurrence is TRUE, TimeZone contains an integer index into a list of time zones. |
| Prerequisites | Windows SharePoint Services 3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   fRecurrence values 1, TimeZone values 2.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC32\_OperationListItemsForAppointment\_TimeZone

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC33\_OperationListItems\_TimeZoneSetByProtocolSUT |
| Description | This test case is used to verify TimeZone doesn’t be empty. |
| Prerequisites | Windows SharePoint Services 3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1,   fRecurrence values 1, TimeZoneXML values “<timeZoneRule><standardBias>0</standardBias><additionalDaylightBias>0</additionalDaylightBias></timeZoneRule>”.   1. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC33\_OperationListItems\_TimeZoneIsEmpty

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC34\_TriggerExceptionDeletion\_UpdateEndDate |
| Description | This test case is used to verify protocol servers will trigger exception deletion when EndDate is updated. |
| Prerequisites | Windows SharePoint Services3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Setting “deleteExceptions ” element present in the value of “RecurrenceData” field. 2. Call method **UpdateListItem** to update the EndDatevalue.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC34\_TriggerExceptionDeletion\_UpdateEndDate

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC35\_TriggerExceptionDeletion\_UpdateEventDate |
| Description | This test case is used to verify protocol servers will trigger exception deletion when EventDate is updated. |
| Prerequisites | Windows SharePoint Services3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Setting “deleteExceptions ” element present in the value of “RecurrenceData” field. 2. Call method **UpdateListItem** to update the EventDate value.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC35\_TriggerExceptionDeletion\_UpdateEventDate

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC36\_TriggerExceptionDeletion\_UpdateRecurrenceData |
| Description | This test case is used to verify protocol servers will trigger exception deletion when RecurrenceData is updated. |
| Prerequisites | Windows SharePoint Services3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Setting “deleteExceptions ” element present in the value of “RecurrenceData” field. 2. Call method **UpdateListItem** to update the EventDate value.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC36\_TriggerExceptionDeletion\_UpdateRecurrenceData

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC37\_TriggerExceptionDeletion\_UpdateXMLTZone |
| Description | This test case is used to verify protocol servers will trigger exception deletion when XMLTZone is updated. |
| Prerequisites | Windows SharePoint Services3.0 and above products. |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Setting “deleteExceptions ” element present in the value of “RecurrenceData” field. 2. Call method **UpdateListItem** to update the XMLTZone value.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC37\_TriggerExceptionDeletion\_UpdateXMLTZone

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC38\_GetListItemChangesSinceToken\_Support |
| Description | This test case is used to verify the value of the server version is "12.0.0.4326" or greater indicates the server supports GetListItemChangesSinceToken. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a generic List on the server, return listId. 2. Add a list item. 3. Call method **GetList** to get information about the list.   **Input parameters:**   * listName: listName  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC38\_GetListItemChangesSinceToken\_Support

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC39\_GetListItemChangesSinceToken\_QueryIsEmpty |
| Description | This test case is used to verify in Windows SharePoint Services 3.0 and SharePoint Foundation 2010, if there is no "query" element in the request, the server will sort items by the ID field, in ascending order. |
| Prerequisites | Windows SharePoint Services 3.0 and above products. |
| Test execution steps | 1. Add a generic List on the server, return listId. 2. Add three list item. 3. get list items changes from SUT |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC39\_GetListItemChangesSinceToken\_queryIsEmpty

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC40\_GetListItemChangesSinceToken\_HaveInstances |
| Description | This test case is used to verify the instance of a recurrence can have zero or one total exceptions and deleted instances |
| Prerequisites | Windows SharePoint Services 3.0, and SharePoint Foundation 2010 |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Create a Recurrence AppointMent settings: Set EventType values 1, fRecurrence values 1. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create an Exception AppointMent setting: EventType values 4, fRecurrence values 1, RecurrenceID values EventDate+1, MasterSeriesItemID values the list item id of the Recurrence Appointment. 2. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Create a deleted AppointMent setting: EventType values 3. 2. Call method **UpdateListItem** to update the XMLTZone value.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateDat |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC40\_GetListItemChangesSinceToken\_HaveInstances

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC41\_AddDiscussionBoardItem |
| Description | This test case is used to verify AddDiscussionBoardItem operation to add a new discussion item to a list. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a discussion board item. 2. Call method AddDiscussionBoardItem to add a new discussion item to a list.   **Input parameters:**   * listName: listName * message: validMessage  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC41\_AddDiscussionBoardItem

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC42\_GetListItemChangesSinceToken\_OptimizeLookups |
| Description | This test case is used to verify if include queryOptions.OptimizeLookups, must not change the contents of a successful protocol server response. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a generic List on the server, return listId. 2. Add three list item. 3. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC42\_GetListItemChangesSinceToken\_OptimizeLookups

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC43\_DeleteDocumentsAndFolders |
| Description | This test case is used to verify if the folder item is deleted, then all documents and folders in it MUST be deleted too. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a document list. 2. Add a folder. 3. Call SUT method UploadFileWithFolder to upload a file whose content is generated in the format "MSOUTSPS Test on [HHmmss\_fff] into the specified folder of a Docuemnt Library on the SUT.   **Input parameters:**   * listTitle: listTitle * subfolderName: folderName * fileName: fileName  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Call SUT method DeleteFolder to delete the specified folder in the specified list.   **Input parameters:**   * listtitle: listTitle * subfolderName: folderName  1. Call method **HTTPGET** to get the attachment content.   **Input parameters:**   * requestResourceUrl: url for uploaded file. * translateHeaderValue: f |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC43\_DeleteDocumentsAndFolders

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC44\_GenericList\_VerifyVtiVersionHistoryValue |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Generice List on the server, return listId. 2. Add a list item into the list. 3. Call method **GetListItemChangesSinceToken to** return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields Properties="True"><FieldRef Name="MetaInfo" /></ViewFields></viewFields> ”  1. Call method **UpdateListItem** to update the title of the list item added in step2.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method **GetListItemChangesSinceToken** toreturn changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields Properties="True"><FieldRef Name="MetaInfo" /></ViewFields></viewFields> ” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC44\_GenericList\_VerifyVtiVersionHistoryValue

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC45\_GenericList\_VerifyFieldsInCommonDefinition |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Generice List on the server, return listId. 2. Add two list item into the list. 3. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Call method UpdateListItem to update the title of the first list item added in step2.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC45\_GenericList\_VerifyFieldsInCommonDefinition

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC46\_OperateOnListItems\_VerifyContentTypeId |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Appointment list on the server, return listId. 2. Add one list item into this Appointment list. 3. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Add a Contacts list on the server, return listId. 2. Add one list item into this Appointment list. 3. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Add a Discussion Board list on the server, return listId. 2. Call method AddDiscussionBoardItem to add a new discussion item to a list.   **Input parameters:**   * listName: listName * message: validMessage  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Add a Document Library on the server, return listId. 2. Add a folder into this Document Library. 3. Upload one document item into this folder under the Document Library. 4. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />”  1. Add a Task list on the server, return listId. 2. Add a task item into this list. 3. Call method **GetListItemChangesSinceToken to** return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC46\_OperateOnListItems\_VerifyContentTypeId

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC47\_OperateOnListItems\_VerifyEventTypeInterpretedAsZero |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Event List on the server, return listId. 2. Setting fields setting:   EventDate: validate DatetimeEndDate: EventDate + 1 hourEventType: 2Title: unique appointment title   1. Call method **UpdateListItem** to add a appointment item with fields’ settings in step2.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method **GetListItemChangesSinceToken** toreturn changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields Properties="True"><FieldRef Name="MetaInfo" /></ViewFields></viewFields> ” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC47\_OperateOnListItems\_VerifyEventTypeInterpretedAsZero

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC48\_OperationListItemsForContacts |
| Description | This test case is used to verify Contacts list's fields when updating a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Set the values for the specified fields: CellPhone, Comments, Company, CompanyPhonetic, Editor, Email, FirstName, FirstNamePhonetic, FullName, HomePhone, JobTitle, LastNamePhonetic, Title, WebPage, WorkAddress, WorkCity, WorkCountry, WorkFax, WorkPhone, WorkState, WorkZip. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC48\_OperationListItemsForContacts

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC49\_OperationListItemsForDiscussion |
| Description | This test case is used to verify DiscussionBoard list's fields when updating a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Set the values for the specified fields: Author, Body, DiscussionTitle, Editor, ThreadIndex, Titl. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC49\_OperationListItemsForDiscussion

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC50\_OperationListItemsForTasks |
| Description | This test case is used to verify Tasks list's fields when updating a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Set the values for the specified fields: AssignedTo, Body, DueDate, Editor, PercentComplete, Priority, StartDate, Status, Title. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData |
| Cleanup | N/A |

MSOUTSPS\_S02\_TC50\_OperationListItemsForTasks

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC51\_OperationListItemsForDocument |
| Description | This test case is used to verify Document list's fields when updating a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a event List on the server, return listId. 2. Set the values for the specified fields: Author, Editor, EncodedAbsUrl, FileDirRef, FileSizeDisplay, LinkCheckedOutTitle, LinkFilename. 3. Call method **UpdateListItem** to add items that are not discussion items and to update items.   **Input parameters:**   * listName: listName * UpdateListItemsUpdates: updateData  1. Call method GetListItemChangesSinceToken to return changes made to a specified list after the event expressed by the change token.   **Input parameters:**   * listName: listName * viewFields: <ViewFields />” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC51\_OperationListItemsForDocument

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC52\_OperateOnListItems\_VerifyMoreChangeValue |
| Description | This test case is used to verify MoreChange value when updating a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Generice List on the server, return listId. 2. Call method **GetListItemChangesSinceToken to** return changes made to a specified list after the event expressed by the change token.   Input parameters:   * listName: listId   3. Add 10 list item into the list.  4. Call method **GetListItemChangesSinceToken to** return changes with row limite equal to “2”.  Input parameters:   * listName: listId * viewFields: <ViewFields /> * rowLimit: “2” * changeToken: the value get in step2. |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC52\_OperateOnListItems\_VerifyMoreChangeValue

|  |  |
| --- | --- |
| S02\_OperateListItems | |
| Test case ID | MSOUTSPS\_S02\_TC53\_OperateOnListItems\_VerifyListItemCollectionPositionNextValue |
| Description | This test case is used to verify Tasks template when update a list item. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Generice List on the server, return listId. 2. Add 10 list item into the list. 3. Call method GetListItemChangesSinceToken to return changes with row limite equal to “2”.   Input parameters:   * listName: listId * viewFields: <ViewFields /> * rowLimit: “2” |
| Cleanup | Common clean up |

MSOUTSPS\_S02\_TC53\_OperateOnListItems\_VerifyListItemCollectionPositionNextValue

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC01\_VerifyAppointmentsList |
| Description | This test case is used to verify definition of Event list which contains Appointments list items. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Appointment List on the server, return listId. 2. Call method **GetList** to the appointment list definition.   **Input parameters:**   * listname: listId  1. Veirfy below fields’ id and type whether equal to their expected values:   [Description, Duration, Editor, EndDate, EventDate, EventType, fAllDayEvent, fRecurrence, Location, MasterSeriesItemID, RecurrenceData, RecurrenceID, TimeZone, Title, UID, XMLTZone] |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC01\_VerifyAppointmentsList

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC02\_VerifyContactsList |
| Description | This test case is used to verify definition of Contacts list which contains contact list. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Contacts List on the server, return listId. 2. Call method GetList to the appointment list definition.   **Input parameters:**   * listname: listId  1. Veirfy below fields’ id and type whether equal to their expected values:   [CellPhone, Comments, Company, CompanyPhonetic, Editor, Email, FirstName, FirstNamePhonetic, FullName, HomePhone, JobTitle, LastNamePhonetic, Title, WebPage, WorkAddress, WorkCity, WorkCountry, WorkFax, WorkPhone, WorkState, WorkZip] |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC02\_VerifyContactsList

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC03\_VerifyDiscussionList |
| Description | This test case is used to verify definition of Discussion list which contains contact list items. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Discussion Board List on the server, return listId. 2. Call method **GetList** to the appointment list definition.   **Input parameters:**   * listname: listId  1. Veirfy below fields’ id and type whether equal to their expected values:   [Author, Body, DiscussionTitle, Editor, ThreadIndex, Title] |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC03\_VerifyDiscussionList

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC04\_VerifyDocumentLibrary |
| Description | This test case is used to verify definition of DocumentLibrary which contains contact list items. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Document Library on the server, return listId. 2. Call method **GetList** to the appointment list definition.   **Input parameters:**   * listname: listId  1. Veirfy below fields’ id and type whether equal to their expected values:   [Author, Editor, EncodedAbsUrl, FileDirRef, FileSizeDisplay, LinkCheckedOutTitle, LinkFilename] |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC04\_VerifyDocumentLibrary

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC05\_VerifyTasksList |
| Description | This test case is used to verify definition of Tasks list which contains contact list items. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Task List on the server, return listId. 2. Call method **GetList** to the appointment list definition.   **Input parameters:**   * listname: listId  1. Veirfy below fields’ id and type whether equal to their expected values:   [AssignedTo, Body, DueDate, Editor, PercentComplete, Priority, StartDate, Status, Title] |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC05\_VerifyTasksList

|  |  |
| --- | --- |
| S03\_CheckListDefination | |
| Test case ID | MSOUTSPS\_S03\_TC06\_VerifyCHOICESAndMAPPINGSElements\_TasksList |
| Description | This test case is used to verify CHOICES and MAPPINGS element of Tasks list. |
| Prerequisites | N/A |
| Test execution steps | 1. Add a Task List on the server, return listId. 2. Call method **GetList** to the appointment list definition.   **Input parameters:**   * listname: listId |
| Cleanup | Common clean up |

MSOUTSPS\_S03\_TC06\_VerifyCHOICESAndMAPPINGSElements\_TasksList