MS-WSP Test Suite User Guide

Microsoft

[1. Test Environment Requirement 2](#_Toc50648758)

[1.1 Generic Requirements 2](#_Toc50648759)

[1.2 Network Setting 3](#_Toc50648760)

[2. Test Suite Deployment 3](#_Toc50648761)

[2.1 How to configure Driver Computer 3](#_Toc50648762)

[2.2 How to configure SUT 4](#_Toc50648763)

[2.3 How to configure Test Suite 4](#_Toc50648764)

[2.4 How to run Test Suite 5](#_Toc50648765)

[3. Test Suite Design 6](#_Toc50648766)

[3.1 Traditional Test Cases 6](#_Toc50648767)

[3.2 Model-based Test Cases 9](#_Toc50648768)

[4. Appendix 12](#_Toc50648769)

[4.1 How to index files on Windows Server 12](#_Toc50648770)

# Test Environment Requirement

## Generic Requirements

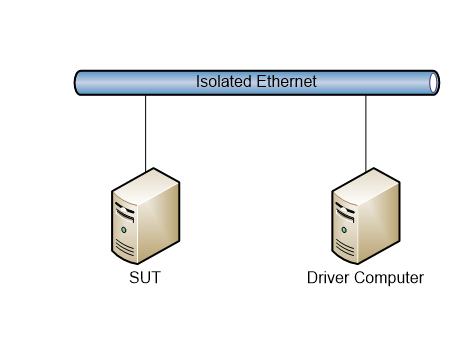
The MS-WSP requires the test environment including:

* 2 machines for servers and clients.
* Generic software installed on the machines.
* Ethernet network.

|  |  |  |
| --- | --- | --- |
| **Device Type** | **Role** | **Description** |
| **Hardware** | SUT | System Under Test. |
| Driver Computer | Driver computer to run the test suite. |
| **Service** | SUT | An implementation of MS-WSP. |
| Driver Computer | None. |
| **Software** | SUT | None |
| Driver Computer | Install the software to enable the running of test suite. |
| **Network** | The network is an isolated hub/switch connected Ethernet network. | |

## Network Setting

SUT and Driver Computer should be under one isolated network.



# Test Suite Deployment

## How to configure Driver Computer

The following software is needed to install on Driver Computer.

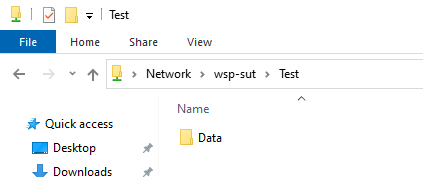
* [.NET Framework 4.7.1](https://www.microsoft.com/en-US/download/details.aspx?id=56116)
* [Protocol Test Framework 1.0.7500.0](https://github.com/Microsoft/ProtocolTestFramework/releases/tag/1.0.7500.0)
* [Spec Explorer 2010 3.5.3146.0](https://marketplace.visualstudio.com/items?itemName=SpecExplorerTeam.SpecExplorer2010VisualStudioPowerTool-5089)
* MS-WSP test suite

## How to configure SUT

Follow the steps to configure SUT:

* Create a Share named **Test** in the SUT.
* When the test suite is installed on the Driver Computer, copy the folder **Data** under C:\MicrosoftProtocolTests\MS-WSP\Server-Endpoint\<version#>\ to the share of the SUT.

Make sure the share can be accessed from Driver Computer.



* Index the files in the **Data** folder. Refer to [Appendix 4.1 for how to index files on Windows Server](#_How_to_install).

## How to configure Test Suite

The only file needs to be modified is **MS-WSP\_ServerTestSuite.deployment.ptfconfig**, under **C:\MicrosoftProtocolTests\MS-WSP\Server-Endpoint\<version#>\Bin** of Driver Computer.

Open **MS-WSP\_TestSuite.deployment.ptfconfig**, those properties’ value is very likely to be changed before running the test suite:

<Property name="ServerComputerName" value="sut" />

<Property name="UserName" value="Administrator" />

<Property name="Password" value="Password01!" />

<Property name="ShareName" value="Test" />

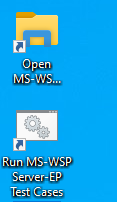
<Property name="ClientName" value="DRIVER" />

<Property name="QueryPath" value="file://sut/test/" />

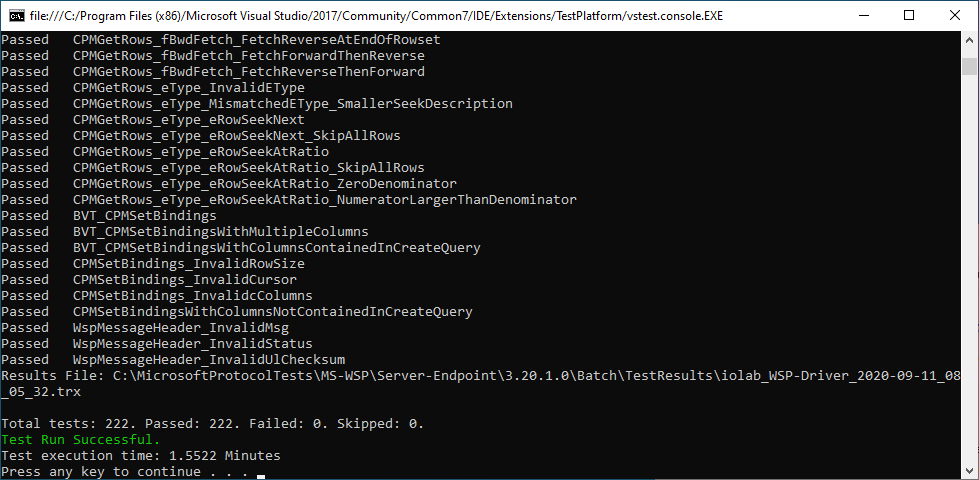
## How to run Test Suite

Once you configure the MS-WSP\_TestSuite.deployment.ptfconfig file. Perform the following steps on Driver Computer to run the test cases.

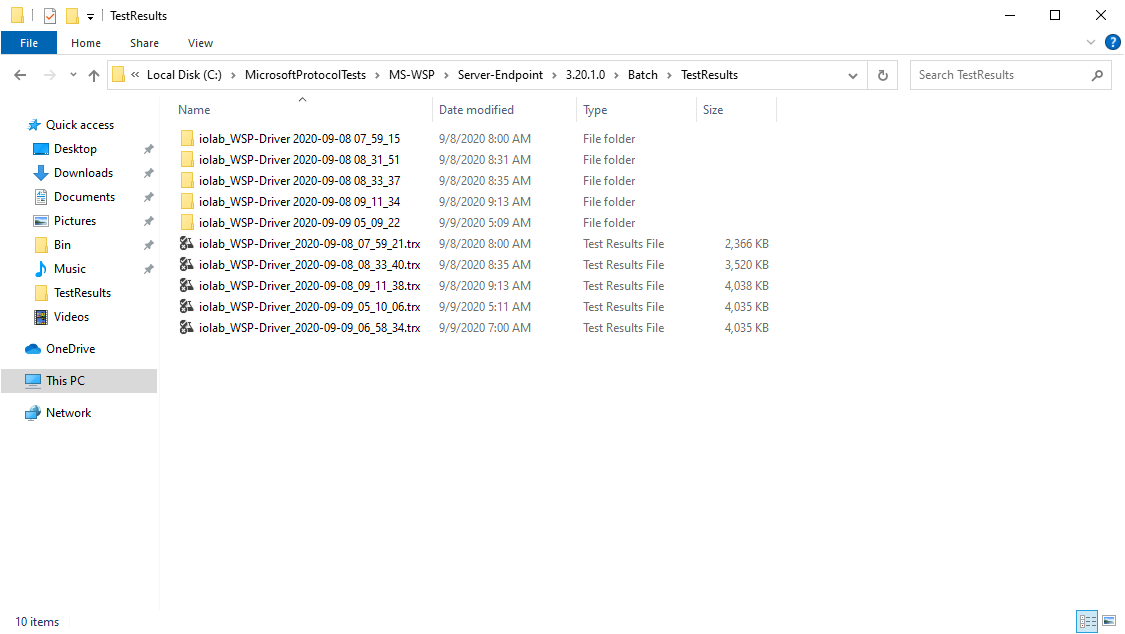
* Double click **Run MS-WSP Server-EP Test Cases** shortcuton **Desktop** to execute the test cases.



* Press any key to close the console, when **Pass** appears on the screen.



* The result can be found under the folder: C:\MicrosoftProtocolTests\MS-WSP\Server-Endpoint\<version#>\Batch\TestResults



# Test Suite Design

MS-WSP test suite consists of two kinds of test cases: traditional test cases and model-based test cases.

There are 189 traditional test cases and 33 model-based test cases, 222 cases in total.

## Traditional Test Cases

Traditional test cases cover the typical message sequences in the figure below.

Machine generated alternative text:



Besides the messages above, there are also test cases on messages CPMGetQueryStatus, CPMGetQueryStatusEx, CPMFetchValue, CPMCiState.

The table below listed all the traditional cases.

|  |  |  |
| --- | --- | --- |
| Test Focus | Description | Case Count |
| CPMCiState | Test cases send out CPMCiStateIn in different scenarios (send after CPMConnectIn/CPMCreateQueryIn/CPMSetBindingsIn…) and verify responses. | 6 |
| CPMConnect | Test cases send out CPMConnectIn with different fields, including ClientVersion, ClientRemote, CatalogName, xtPropSet and verify responses. | 8 |
| CPMCreateQuery | Test cases send out CPMCreateQueryIn with different fields, including Size, RestrictionArray, ColumnSet, SortSet, CategorizationSet (with Aggregation), PidMapper (with different CFullPropSpec) and RowSetProperties. | 105 |
| CPMDisconnect | Test cases send out CPMDisconnect in different scenarios (send after CPMConnectIn/CPMCreateQueryIn/CPMSetBindingsIn…) and verify responses. | 7 |
| CPMFetchValue | Test cases send out CPMFetchValueIn to query values of different properties or query values with single/multiple chunks, and verify responses. | 11 |
| CPMFreeCursor | Test cases send out CPMFreeCursorIn in different scenarios (send after CPMConnectIn/CPMCreateQueryIn/CPMSetBindingsIn…) and verify responses. | 8 |
| CPMGetQueryStatus | Test cases send out CPMGetQueryStatusIn in different scenarios (send after CPMConnectIn/CPMCreateQueryIn/CPMSetBindingsIn…) and verify responses. | 4 |
| CPMGetQueryStatusEx | Test cases send out CPMGetQueryStatusExIn in different scenarios (send after CPMConnectIn/CPMCreateQueryIn/CPMSetBindingsIn…) and verify responses. | 5 |
| CPMGetRows | Test cases send out CPMGetRowsIn with different fields, including RowsToTransfer, RowWidth, ReadBuffer, FetchType, SeekDescription and verify responses. | 25 |
| CPMSetBindings | Test cases send out CPMSetBindingsIn with different fields, including Columns (multiple Columns, with Columns contained or not contained in CPMCreateQueryIn), RowSize and cursors, then verify responses. | 7 |
| InvalidWspMessageHeader | Test cases send out invalid fields of WspMessageHeader for CPMConnectIn and verify responses. | 3 |

## Model-based Test Cases

|  |  |  |
| --- | --- | --- |
| Test Focus | Description | Case Count |
| QueryAdminMessages | Validates error code returned by the server when connected client doesn’t have a connection to send a CPMCiStateInOut request to the server. | 1 |
| ConnectOut\_StatusInvalidParameter | Validates the error code returned by the server when already connected client sends ConnectIn request. | 1 |
| FetchValue\_StatusInvalidParameterErr | Validates the error code returned by the server when FetchValueIn request is sent for the non-existing query. | 2 |
| FindIndicesSuccessfully | Validates the server responses for getting rowset position of the next occurrence of a document identifier | 1 |
| FindIndicesInvalidParameter | Validates the error code returned by the server when FindIndicesIn request is sent for the non-existing query. | 1 |
| QueryNotAssociatedCase | Validates the error code returned by the server when the client sends different query status requests for any non-existing query. | 1 |
| QueryOut\_CreatingExistingQueryCase | Validates the error code returned by the server when the client sends the request to create an already existing query. | 1 |
| FreeCursorIn\_E\_FailError | Validates the error code returned by the server when client sends FreeCursorIn request with cursor handle passed are not in the list of the client's cursor handles. | 1 |
| GetRowsOut\_E\_FailError | Validates the error code returned by the server when client sends GetRowsIn request with cursor handle passed are not in the list of the client's cursor handles. | 2 |
| GetRowsetNotifySuccessfully | Validates the server responses for getting the next rowset event from the server with different eventType and there is or there is no another event of any non-PROPAGATE\_NONE type immediately available. | 4 |
| GetRowsetNotifyInvalidParameter | Validates the error code returned by the server when the client sends request for any non-existing query with different eventType and there is or there is no another event of any non-PROPAGATE\_NONE type immediately available. | 4 |
| GetScopeStatisticsSuccessfully | Validates the server responses for statistic number of indexed items when setting the DBPROP\_ENABLEROWSETEVENTS property to TRUE or FALSE. | 2 |
| GetScopeStatisticsInvalidParameter | Validates the error code returned by the server when GetScopeStatisticsIn request is sent for the non-existing query. | 1 |
| QueryForDifferentQueryStatus | Validates the response for different query status requests sent by the client to the server. | 1 |
| QueryForResultSets | Validates the response for the request sent by the client to connect to the server, query a catalog for result sets, release the cursor and finally free the connection from the server. | 1 |
| SetScopePrioritizationInvalidParameter | Validates the error code returned by the server when SetScopePrioritizationIn request is sent for the non-existing query with the priority set to different value. | 4 |
| SetScopePrioritizationSuccessfully | Validates the server response for modifying the indexing priority of documents that may be relevant to query in this action for different value. | 4 |
| QueryStatus\_E\_FailError | Validates the error code returned by the server when client sends CPMGetQueryStatusIn request with cursor handle passed are not in the list of the client's cursor handles. | 1 |

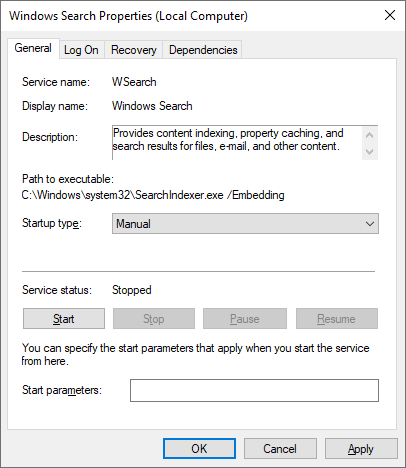
# Appendix

## How to index files on Windows Server

* Go to **Start** and type “services.msc” in the search field, press **Enter**.

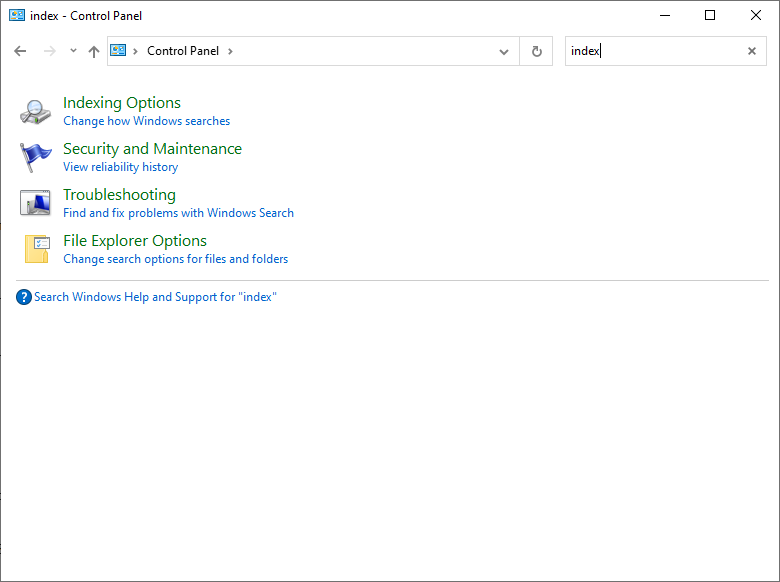
Find out **Windows Search** service and right-click it.

Change Startup type to **Manual** and Click **Start**.

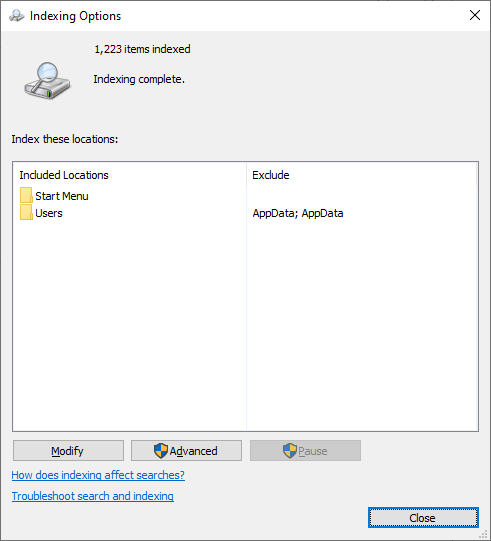


* Go to **Start** and type “control panel” in the search field, press **Enter**.

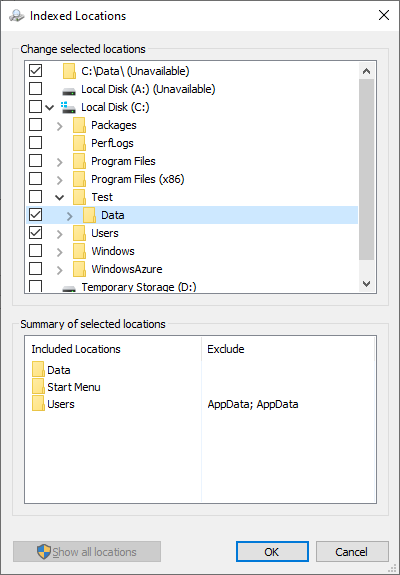
Search **index** in the right upper search box, press **Enter**



* Click Indexing Options, then click Modify



In the pop-out dialog, select **Data** folder and click **OK**



Then files under **Data** will be indexed.