namespace CodedUITestProject

{

using Microsoft.VisualStudio.TestTools.UITesting.WinControls;

using Microsoft.VisualStudio.TestTools.UITesting.HtmlControls;

using System;

using System.Security;

using System.Diagnostics;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Input;

using System.CodeDom.Compiler;

using System.Text;

using System.Text.RegularExpressions;

using System.Globalization;

using Microsoft.VisualStudio.TestTools.UITest.Extension;

using Microsoft.VisualStudio.TestTools.UITesting;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Keyboard = Microsoft.VisualStudio.TestTools.UITesting.Keyboard;

using Mouse = Microsoft.VisualStudio.TestTools.UITesting.Mouse;

using MouseButtons = System.Windows.Forms.MouseButtons;

using Microsoft.Office.Interop.Excel;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Net.Http.Formatting;

using Newtonsoft.Json;

using System.Windows.Forms;

using System.Web;

using Microsoft.Win32;

using System.Data.SqlClient;

using System.Xml.Schema;

using System.Xml.Serialization;

using System.Xml;

using System.Threading;

using System.Threading.Tasks;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.IO;

public partial class UIMap

{

public static Microsoft.Office.Interop.Excel.Application \_xlApp;

public static Microsoft.Office.Interop.Excel.Workbook \_xlWorkBook;

public static Microsoft.Office.Interop.Excel.Worksheet \_xlWorkSheet;

public TestContext ActionLogEntry { get; set; }

public void Keep\_Alive()

{

Keyboard.PressModifierKeys(ModifierKeys.Control);

Keyboard.SendKeys("{ESC}");

Keyboard.ReleaseModifierKeys(ModifierKeys.Control);

Keyboard.SendKeys("{ESC}");

}

public void WebService\_POST\_Test()

{

string listName = "PCSC\_OU\_List";

string viewName = "{5E28FB90-39B1-44C0-B339-4039D0825DB3}";

string[] fields = { "OU\_Strings" };

string rowLimit = "150";

XmlDocument xmlDoc = new System.Xml.XmlDocument();

XmlElement query = xmlDoc.CreateElement("Query");

XmlElement queryOptions = xmlDoc.CreateElement("QueryOptions");

XmlElement viewFields = xmlDoc.CreateElement("ViewFields");

query.InnerXml = "<Where><Gt><FieldRef Name=\"ID\" />" + "<Value Type=\"Counter\">3</Value></Gt></Where>";

viewFields.InnerXml = "<FieldRef Name=\"OU\_Strings\" />";

queryOptions.InnerXml = "";

//SharepointList.ListsSoapClient lists = new SharepointList.ListsSoapClient();

//System.Xml.XmlNode ResponseListItems = lists.GetListItems(listName, viewName, query, viewFields, null, queryOptions, null);

Console.WriteLine("ran successfully");

//POST

//var postData = new List<KeyValuePair<string, string>>();

//postData.Add(new KeyValuePair<string, string>("newSharePath", sharePath));

//postData.Add(new KeyValuePair<string, string>("newUserSoeId", shareSOEID));

//postData.Add(new KeyValuePair<string, string>("hostname", hostname));

//HttpContent content = new FormUrlEncodedContent(postData);

//HttpResponseMessage response = client.PostAsync("CreateShare", content).Result;

//Console.WriteLine(response.StatusCode);

//Console.WriteLine(response.Content.ReadAsStringAsync().Result);

}

public void ShareAutomation\_Web\_API\_GetDomainForHost(string HostName, string ExpectedDomain, string BaseAddress)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

if (Run\_Environment == "OST\_DEV")

client.BaseAddress = new Uri(BaseAddress + "GetDomainForHost");

else if (Run\_Environment == "OST\_UAT")

client.BaseAddress = new Uri(BaseAddress + "GetDomainForHost");

else if (Run\_Environment == "OST\_PROD")

client.BaseAddress = new Uri(BaseAddress + "GetDomainForHost");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?host=" + HostName);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?host=" + HostName);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful.");

Console.WriteLine("Domain: " + response.Result.Content.ReadAsStringAsync().Result.Replace("\"", "").ToUpper());

Assert.AreEqual(ExpectedDomain, response.Result.Content.ReadAsStringAsync().Result.Replace("\"", "").ToUpper(), "Domain Name different than expected.");

}

else

{

var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceDataServerError>(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Message: " + Response.Message);

Console.WriteLine("Exception Message: " + Response.ExceptionMessage);

Console.WriteLine("Exception Type: " + Response.ExceptionType);

//Only 'xyz' host is expected to be not found, should fail on any other host not found than 'xyz'

Assert.AreEqual("Host xyz is not found.", Response.ExceptionMessage, "Unexpected message received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public class ShareAutomationWebserviceDataServerError

{

public string Message;

public string ExceptionMessage;

public string ExceptionType;

}

public void ShareAutomation\_Web\_API\_CreateFolder(string uncPath, string IDriveBaseAddress)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(IDriveBaseAddress + "CreateFolder");

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

if (Run\_Environment == "OST\_DEV")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

else if(Run\_Environment == "OST\_UAT")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\TestTeamCerts\OstCertUat.pkcs12", "citi@123"));

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful.");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceCreateDirectory>(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Success: " + DynamicResult.Success);

Console.WriteLine("FolderName: " + DynamicResult.FolderName);

Console.WriteLine("ParentUncPath: " + DynamicResult.ParentUncPath);

Console.WriteLine("UncPath: " + DynamicResult.UncPath);

Console.WriteLine("Log: " + DynamicResult.Log);

if (DynamicResult.Log.ToString().Contains("already exists") == true)

{

Console.WriteLine("Directory already exists.");

}

if (DynamicResult.Log.ToString().Contains("Directory Created") == true)

{

Console.WriteLine("Directory created.");

}

Assert.AreEqual(uncPath.ToString().ToLower(), DynamicResult.UncPath.ToString().ToLower(), "Directory uncPath does not match input path.");

}

else

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_CreateInvalidDirectory(string uncPath, string BaseAddress)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "CreateFolder");

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

if (Run\_Environment == "OST\_DEV")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

else if (Run\_Environment == "OST\_UAT")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\TestTeamCerts\OstCertUat.pkcs12", "citi@123"));

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "InternalServerError")

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceDataServerError>(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Exception Message: " + Response.ExceptionMessage.ToString().Trim());

if (Response.ExceptionMessage.ToString().Trim().StartsWith("Could not find file '") == true)

{

Assert.AreEqual("Could not find file '" + uncPath + "'.", Response.ExceptionMessage.ToString().Trim(), Response.ExceptionMessage.ToString().Trim());

}

else if (Response.ExceptionMessage.ToString().Trim() == "The network path was not found.")

{

Assert.AreEqual("The network path was not found.", Response.ExceptionMessage.ToString().Trim(), Response.ExceptionMessage.ToString().Trim());

}

else

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

}

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Exception Message: " + DynamicResult.ErrorStr);

StringAssert.Contains(DynamicResult.ErrorStr.ToString(), "does not exist", "Expected error message not found.");

}

else

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("xyz", response.Result.StatusCode.ToString(), "Unexpected message/response received. Make sure proper error code or message was shown for invalid input data, not just OK.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_DeleteDirectory(string uncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "DeleteDirectory");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK" && response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().Replace("\"", "") == "Succesfully deleted")

{

Console.WriteLine("HTTP GET was successful.");

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Replace("\"", ""));

Assert.AreEqual("Succesfully deleted", response.Result.Content.ReadAsStringAsync().Result.Replace("\"", ""), response.Result.Content.ReadAsStringAsync().Result.Replace("\"", ""));

}

else if (response.Result.StatusCode.ToString() == "OK" && response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().Replace("\"", "") == "no folder found")

{

Console.WriteLine("HTTP GET was successful.");

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Replace("\"", ""));

Assert.AreEqual("Successfully deleted", response.Result.Content.ReadAsStringAsync().Result.Replace("\"", ""), "Folder not found.");

}

else if (response.Result.StatusCode.ToString() == "InternalServerError")

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceDataServerError>(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Exception Message: " + Response.ExceptionMessage);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), Response.ExceptionMessage.ToString());

}

else

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_DeleteInvalidDirectory(string uncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "DeleteDirectory");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "InternalServerError")

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceDataServerError>(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Exception Message: " + Response.ExceptionMessage);

Assert.AreEqual("Could not find a part of the path '" + uncPath + "'.", Response.ExceptionMessage.ToString().Trim(), Response.ExceptionMessage.ToString());

}

else if (response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().ToLower().Replace("\"", "") == "no folder found")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().Replace("\"", ""));

Assert.AreEqual("no folder found", response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("Result: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("no folder found", response.Result.Content.ReadAsStringAsync().Result.ToString().Trim().ToLower().Replace("\"", ""), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_AddNTFSPermissionForUNC(string uncPath, string Domain, string newUserSID, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "AddNTFSPermissionForUNC");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("access have provide for the directory", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_AddNTFSPermissionForUNCInvalidAccess(string uncPath, string Domain, string newUserSID, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "AddNTFSPermissionForUNC");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

//var Response = JsonConvert.DeserializeObject<ShareAutomationWebserviceDataServerError>(response.Result.Content.ReadAsStringAsync().Result);

dynamic Response = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Response:" + Response);

//Console.WriteLine("Exception Message: " + Response.ExceptionMessage.Trim());

Assert.AreEqual("The user is not in the domain", Response, "Expected response message: The user is not in the domain");

//Assert.AreEqual("some or all identity references could not be translated.", Response.ExceptionMessage.Trim().ToLower().Replace("\"", ""));

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_AddNTFSPermissionForUNCInvalidPath(string uncPath, string Domain, string newUserSID, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "AddNTFSPermissionForUNC");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("no folder found", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.IsTrue(response.Result.Content.ReadAsStringAsync().Result.Contains("is not found") == true, "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_DeleteNTFSPermissionForUNC(string uncPath, string Domain, string newUserSID, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "DeleteNTFSPermissionForUNC");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath + "&" + "domain=" + Domain + "&" + "newUserSID=" + newUserSID);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("access have delete for the directory", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_CreateShare(string sharename, string sharePath, string shareSOEID, string hostname, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "CreateShare");

DateTime Start = DateTime.Now;

Console.WriteLine("Checkpoint 1. Creating Share.");

var response = client.GetAsync("?newSharePath=" + sharePath + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?newSharePath=" + sharePath + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

//Delete share if it already exists:

if (response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", "") == "sharename already exists")

{

Console.WriteLine("Checkpoint 2. Sharename already exists, attempting to delete.");

var client2 = new HttpClient(handler);

client2.BaseAddress = new Uri(BaseAddress + "DeleteShare");

response = client2.GetAsync("?sharename=" + sharename + "&hostname=" + hostname);

Console.WriteLine("URL: " + client2.BaseAddress.ToString() + "?sharename=" + sharename + "&hostname=" + hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("deleted succesfuly", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""), "Unable to delete Share for some reason.");

Console.WriteLine("Checkpoint 3. Creating Share.");

Start = DateTime.Now;

response = client.GetAsync("?newSharePath=" + sharePath + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?newSharePath=" + sharePath + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

End = DateTime.Now;

DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("succesfully created share", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""), "Unable to create Share for some reason.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("Message: " + DynamicResult.Message.ToString());

Console.WriteLine("Message Detail: " + DynamicResult.MessageDetail.ToString());

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_CreateInvalidShare(string sharename, string sharePath, string shareSOEID, string hostname, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "CreateShare");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?newSharePath=" + sharePath + "non-existing" + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?newSharePath=" + sharePath + "non-existing" + "&newUserSoeId=" + shareSOEID + "&hostname=" + hostname + "&newShareName=" + sharename);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("directory not found", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_DeleteShare(string sharename, string hostname, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "DeleteShare");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?sharename=" + sharename + "&hostname=" + hostname);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?sharename=" + sharename + "&hostname=" + hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("deleted succesfuly", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""), "Unable to delete Share for some reason.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

try { Console.WriteLine(DynamicResult.MessageDetail.ToString()); }

catch { }

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_ValidateValidUNCPath(string uncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "ValidateUNCPath");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "ValidateUNCPath" + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", "\\\""));

var xSerializer = new XmlSerializer(typeof(bool), null, null,

new XmlRootAttribute("boolean"),

"http://schemas.microsoft.com/2003/10/Serialization/");

using (var sr = new StringReader(response.Result.Content.ReadAsStringAsync().Result))

using (var xr = XmlReader.Create(sr))

{

var y = xSerializer.Deserialize(xr);

Console.WriteLine("Result: " + y);

Assert.AreEqual(true, y, "Expected True value.");

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_ValidateInvalidUNCPath(string InvalidUncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "ValidateUNCPath");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?uncPath=" + InvalidUncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "ValidateUNCPath" + "?uncPath=" + InvalidUncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Result: " + response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

Assert.AreEqual("false", response.Result.Content.ReadAsStringAsync().Result.Trim().ToLower().Replace("\"", ""));

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetNTFSPermissionsForUNCPath(string uncPath, string BaseAddress, string AccountName0, string AccessType0)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "GetNTFSPermissionForUNC");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

for (int i = 0; i < DynamicResult.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Account Name: " + DynamicResult[i].AccountName);

Console.WriteLine("Access Type: " + DynamicResult[i].AccessType);

}

if (AccountName0 != "[]" & AccessType0 != "[]")

{

Assert.AreEqual(AccountName0.ToLower().Trim(), DynamicResult[0].AccountName.ToString().ToLower().Trim());

Assert.AreEqual(AccessType0.ToLower().Trim(), DynamicResult[0].AccessType.ToString().ToLower().Trim());

}

else

{

Assert.AreEqual(AccountName0.Trim(), DynamicResult.ToString().Trim());

Assert.AreEqual(AccessType0.Trim(), DynamicResult.ToString().Trim());

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void IDriveCreation\_Web\_API\_GetNTFSPermissionsForUNCPath(string uncPath, string BaseAddress, string IdentityReference0, string InheritanceFlags0,

string IsInherited0, string PropagationFlags0, string AccessControlType0, string FileSystemRights0, string FileSystemRightsString0)

{

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

if (Run\_Environment == "OST\_DEV")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

else if (Run\_Environment == "OST\_UAT")

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\TestTeamCerts\OstCertUat.pkcs12", "citi@123"));

client.BaseAddress = new Uri(BaseAddress + "GetNtfsPerms");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?uncPath=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?uncPath=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = (JsonConvert.DeserializeObject<IDictionary<string, object>>(response.Result.Content.ReadAsStringAsync().Result))["PermList"];

for (int i = 0; i < DynamicResult.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("IdentityReference: " + DynamicResult[i].IdentityReference);

Console.WriteLine("InheritanceFlags: " + DynamicResult[i].InheritanceFlags);

Console.WriteLine("IsInherited: " + DynamicResult[i].IsInherited);

Console.WriteLine("PropagationFlags: " + DynamicResult[i].PropagationFlags);

Console.WriteLine("AccessControlType: " + DynamicResult[i].AccessControlType);

Console.WriteLine("FileSystemRights: " + DynamicResult[i].FileSystemRights);

Console.WriteLine("FileSystemRights: " + DynamicResult[i].FileSystemRightsString);

}

if (IdentityReference0 != "[]" & InheritanceFlags0 != "[]")

{

Assert.AreEqual(IdentityReference0.ToLower().Trim(), DynamicResult[0].IdentityReference.ToString().ToLower().Trim());

Assert.AreEqual(InheritanceFlags0.ToLower().Trim(), DynamicResult[0].InheritanceFlags.ToString().ToLower().Trim());

Assert.AreEqual(IsInherited0.ToLower().Trim(), DynamicResult[0].IsInherited.ToString().ToLower().Trim());

Assert.AreEqual(PropagationFlags0.ToLower().Trim(), DynamicResult[0].PropagationFlags.ToString().ToLower().Trim());

Assert.AreEqual(AccessControlType0.ToLower().Trim(), DynamicResult[0].AccessControlType.ToString().ToLower().Trim());

Assert.AreEqual(FileSystemRights0.ToLower().Trim(), DynamicResult[0].FileSystemRights.ToString().ToLower().Trim());

Assert.AreEqual(FileSystemRightsString0.ToLower().Trim(), DynamicResult[0].FileSystemRightsString.ToString().ToLower().Trim());

}

else

{

Assert.AreEqual(IdentityReference0.ToLower().Trim(), DynamicResult.ToString().Trim());

Assert.AreEqual(InheritanceFlags0.ToLower().Trim(), DynamicResult.ToString().Trim());

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

try { Console.WriteLine(DynamicResult.Message); }

catch { };

try { Console.WriteLine(DynamicResult.MessageDetail); }

catch { };

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetSharedFolderTree(string FullPath, string BaseAddress, string SubFolder)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "SharedDirectory.asmx/GetSharedFolderTree");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?fullPath=" + FullPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?fullPath=" + FullPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

string Passed = "No";

if (DynamicResult.TreeReturn.RetRecSet == null)

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else if (DynamicResult.TreeReturn.RetRecSet.FolderTree.GetType().ToString().Contains("Array") == true)

{

for (int i = 0; i < DynamicResult.TreeReturn.RetRecSet.FolderTree.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Id: " + DynamicResult.TreeReturn.RetRecSet.FolderTree[i].Id);

Console.WriteLine("Name: " + DynamicResult.TreeReturn.RetRecSet.FolderTree[i].Name);

Console.WriteLine("Has Children: " + DynamicResult.TreeReturn.RetRecSet.FolderTree[i].HasChildren);

Console.WriteLine("Can Select: " + DynamicResult.TreeReturn.RetRecSet.FolderTree[i].CanSelect);

if (DynamicResult.TreeReturn.RetRecSet.FolderTree[i].Id.ToString().ToLower() == SubFolder.ToLower())

{

Passed = "Yes";

}

}

}

else if (DynamicResult.TreeReturn.RetRecSet.FolderTree.GetType().ToString().Contains("Object") == true)

{

if (DynamicResult.TreeReturn.RetRecSet.FolderTree.Id.ToString().ToLower() == SubFolder.ToLower())

{

Passed = "Yes";

}

}

Assert.AreEqual("Yes", Passed, "Expected SubFolder not found");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetSharedFolderTreeV2(string FullPath, string BaseAddress, string SubFolder, string dirOptimization)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "SharedDirectory.asmx/GetSharedFolderTreeV2");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?fullPath=" + FullPath + "&dirOptimization=" + dirOptimization);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?fullPath=" + FullPath + "&dirOptimization=" + dirOptimization);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

string Passed = "No";

if (response.Result.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

if (DynamicResult.TreeResponse.RetRecSet == null)

{

Console.WriteLine("No data returned.");

//Testing for nonexisting UNC Path

if (FullPath.ToLower().Contains("nonexisting") == true & DynamicResult.TreeResponse.Log.ToString().ToLower().Contains("uncpath does not exist") == true)

{Passed = "Yes";}

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else if (DynamicResult.TreeResponse.RetRecSet.FolderTree.GetType().ToString().Contains("Array") == true)

{

for (int i = 0; i < DynamicResult.TreeResponse.RetRecSet.FolderTree.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Id: " + DynamicResult.TreeResponse.RetRecSet.FolderTree[i].Id);

Console.WriteLine("Name: " + DynamicResult.TreeResponse.RetRecSet.FolderTree[i].Name);

Console.WriteLine("Has Children: " + DynamicResult.TreeResponse.RetRecSet.FolderTree[i].HasChildren);

Console.WriteLine("Can Select: " + DynamicResult.TreeResponse.RetRecSet.FolderTree[i].CanSelect);

if (DynamicResult.TreeResponse.RetRecSet.FolderTree[i].Id.ToString().ToLower() == SubFolder.ToLower())

{

Console.WriteLine("Success: " + DynamicResult.TreeResponse.Success.ToString().ToLower());

Console.WriteLine("RuntimeInSeconds: " + DynamicResult.TreeResponse.RuntimeInSeconds);

//conditions requested by Sukh

if (DifferenceDateTime.Duration().TotalSeconds < 60 & DynamicResult.TreeResponse.Success.ToString().ToLower().Contains("true") == true)

{ Passed = "Yes"; }

}

}

}

else if (DynamicResult.TreeResponse.RetRecSet.FolderTree.GetType().ToString().Contains("Object") == true)

{

if (DynamicResult.TreeResponse.RetRecSet.FolderTree.Id.ToString().ToLower() == SubFolder.ToLower())

{

Console.WriteLine("Success: " + DynamicResult.TreeResponse.Success.ToString().ToLower());

Console.WriteLine("RuntimeInSeconds: " + DynamicResult.TreeResponse.RuntimeInSeconds);

//conditions requested by Sukh

if (DifferenceDateTime.Duration().TotalSeconds < 60 & DynamicResult.TreeResponse.Success.ToString().ToLower().Contains("true") == true)

{Passed = "Yes";}

}

}

Assert.AreEqual("Yes", Passed, "Expected SubFolder not found");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

Assert.AreEqual("Yes", Passed, "Expected SubFolder not found");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetMatchingShares(string searchPattern, string BaseAddress, string pingTest, string AccessType, string userDomainName)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "SharedDirectory.asmx/GetMatchingShares");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?searchPattern=" + searchPattern + "&userDomainName=" + userDomainName + "&pingTest=" + pingTest + "&AccessType=" + AccessType);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?searchPattern=" + searchPattern + "&userDomainName=" + userDomainName + "&pingTest=" + pingTest + "&AccessType=" + AccessType);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

string Passed = "No";

if (response.Result.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

if (DynamicResult.SharedDirectoryResponse.HostShareDetails == null)

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else if (DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel.GetType().ToString().Contains("Array") == true)

{

Console.WriteLine("Checkpoint 1 (Array)...");

for (int i = 0; i < DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("HostName: " + DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel[i].HostName);

Console.WriteLine("ShareName: " + DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel[i].ShareName);

Console.WriteLine("UncPath: " + DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel[i].UncPath);

Console.WriteLine("HasRead: " + DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel[i].HasRead);

Console.WriteLine("HasModify: " + DynamicResult.SharedDirectoryResponse.HostShareDetails.HostAndShareServiceModel[i].HasModify);

}

Console.WriteLine("Success: " + DynamicResult.SharedDirectoryResponse.Success.ToString().ToLower());

Console.WriteLine("RuntimeInSeconds: " + DynamicResult.SharedDirectoryResponse.RuntimeInSeconds);

//conditions requested by Sukh

if (DifferenceDateTime.Duration().TotalSeconds < 60 & DynamicResult.SharedDirectoryResponse.Success.ToString().ToLower().Contains("true") == true)

{ Passed = "Yes"; }

}

Assert.AreEqual("Yes", Passed, "Expected results not found");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

Assert.AreEqual("Yes", Passed, "Expected SubFolder not found");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetHostEx(string hostname, string BaseAddress, string ResultHost)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "SharedDirectory.asmx/GetHostEx");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?hostname=" + hostname);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?hostname=" + hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Host.HostData == null)

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.Host.HostData.HostInfo.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Host: " + DynamicResult.Host.HostData.HostInfo[i].host);

if (DynamicResult.Host.HostData.HostInfo[i].host == ResultHost)

{

Passed = "Yes";

}

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void ShareAutomation\_Web\_API\_GetGroupByShareAutomation(string hostname, string sharename, string BaseAddress, string FolderPath, string cmpOrderId, string permission, string userDomain, string userSoeId, string PrimaryOwnerSoeId, string SecondaryOwnerSoeId, string ExpectedResult)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "SharedDirectory.asmx/GetGroupByShareAutomation");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?cmpOrderId=" + cmpOrderId + "&shareType=&hostname=" + hostname + "&sharename=" + sharename + "&permission=" + permission + "&folderPath=" + FolderPath + "&userDomain=" + userDomain + "&userSoeId=" + userSoeId + "&onbehalf\_UserDomain=&onbehalf\_UserId=&userDriveLetter=");

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?cmpOrderId=" + cmpOrderId + "&shareType=&hostname=" + hostname + "&sharename=" + sharename + "&permission=" + permission + "&folderPath=" + FolderPath + "&userDomain=" + userDomain + "&userSoeId=" + userSoeId + "&onbehalf\_UserDomain=&onbehalf\_UserId=&userDriveLetter=");

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

try { Console.WriteLine("NoDataGroupFound: " + DynamicResult.GroupByCMPOrderId.NoDataGroupFound.ToString()); } catch { }

try { Console.WriteLine("NoOwnersFound: " + DynamicResult.GroupByCMPOrderId.NoOwnersFound.ToString()); } catch { }

try { Console.WriteLine("NoDriveMappingGrouFound: " + DynamicResult.GroupByCMPOrderId.NoDriveMappingGrouFound.ToString()); } catch { }

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.GroupByCMPOrderId.NoDataGroupFound == "true")

{

Console.WriteLine("No data group found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No data group found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual(ExpectedResult.ToString().Trim(), DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Trim(), "Did not get expected result.");

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "No data group found.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoOwnersFound == "true")

{

Console.WriteLine("No owners found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No owners found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual(ExpectedResult.ToString(), DynamicResult.GroupByCMPOrderId.ReturnString.ToString(), "Did not get expected result.");

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "Return: " + DynamicResult.GroupByCMPOrderId.ReturnString + " Did not get expected result.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoDriveMappingGrouFound == "true")

{

Console.WriteLine("No Drive Mapping Group found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No Drive Mapping Group found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual(ExpectedResult, DynamicResult.GroupByCMPOrderId.ReturnString, "Did not get expected result.");

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "No Drive Mapping Group found.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoOwnersFound == "false")

{

Assert.AreEqual(PrimaryOwnerSoeId.ToLower(), DynamicResult.GroupByCMPOrderId.PrimaryOwnerSoeId.ToString().ToLower());

Console.WriteLine("Primary Owner found and verified.");

try

{

if (DynamicResult.GroupByCMPOrderId.SecondarOwnerSoeId.ToString() != "")

{

Assert.AreEqual(SecondaryOwnerSoeId.ToLower(), DynamicResult.GroupByCMPOrderId.SecondarOwnerSoeId.ToString().ToLower());

Console.WriteLine("Secondary Owner found and verified.");

}

}

catch { }

Passed = "Yes";

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void Operations\_Web\_API\_GetGroupByShareAutomation(string hostname, string sharename, string BaseAddress, string FolderPath, string cmpOrderId, string permission, string userDomain, string userSoeId, string PrimaryOwnerSoeId, string SecondaryOwnerSoeId, string ExpectedResult)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var content = new FormUrlEncodedContent(new[]

{

new KeyValuePair<string, string>("cmpOrderId", cmpOrderId),

new KeyValuePair<string, string>("hostname", hostname),

new KeyValuePair<string, string>("sharename", sharename),

new KeyValuePair<string, string>("folderPath", FolderPath),

new KeyValuePair<string, string>("shareType", ""),

new KeyValuePair<string, string>("permission", permission),

new KeyValuePair<string, string>("userDomain", userDomain),

new KeyValuePair<string, string>("userSoeId", userSoeId),

new KeyValuePair<string, string>("onbehalf\_UserDomain", ""),

new KeyValuePair<string, string>("onbehalf\_UserId", ""),

new KeyValuePair<string, string>("userDriveLetter", ""),

});

var httpContent = new HttpRequestMessage(HttpMethod.Post, BaseAddress);

client.Timeout = new TimeSpan(0, 0, 4, 0, 0);

var response = client.PostAsync("GetGroupByShareAutomation", content).Result;

Console.WriteLine("StatusCode: " + response.Content.ReadAsStringAsync().Status);

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.StatusCode.ToString() == "OK")

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(response.Content.ReadAsStringAsync().Result);

string json = JsonConvert.SerializeXmlNode(doc);

dynamic DynamicResult = JsonConvert.DeserializeObject(json);

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.GroupByCMPOrderId.NoDataGroupFound == "true")

{

Console.WriteLine("No data group found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No data group found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "No data group found.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoOwnersFound == "true")

{

Console.WriteLine("No owners found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No owners found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "No owners found.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoDriveMappingGrouFound == "true")

{

Console.WriteLine("No Drive Mapping Group found.");

if (DynamicResult.GroupByCMPOrderId.ReturnString.ToString().Contains(ExpectedResult) == true)

{

Console.WriteLine("Expected: No Drive Mapping Group found.");

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Passed = "Yes";

}

else

{

Console.WriteLine("Test Case iteration ExpectedResult value: " + ExpectedResult);

Assert.AreEqual("Yes", Passed, "No Drive Mapping Group found.");

}

}

else if (DynamicResult.GroupByCMPOrderId.NoOwnersFound == "false")

{

Assert.AreEqual(PrimaryOwnerSoeId.ToLower(), DynamicResult.GroupByCMPOrderId.PrimaryOwnerSoeId.ToString().ToLower());

Console.WriteLine("Primary Owner found and verified.");

try

{

if (DynamicResult.GroupByCMPOrderId.SecondarOwnerSoeId.ToString() != "")

{

Assert.AreEqual(SecondaryOwnerSoeId.ToLower(), DynamicResult.GroupByCMPOrderId.SecondarOwnerSoeId.ToString().ToLower());

Console.WriteLine("Secondary Owner found and verified.");

}

}

catch { }

Passed = "Yes";

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result);

Assert.AreEqual("OK", response.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_Version(string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var response = client.GetAsync("version");

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "version");

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Name == "OstScanner")

{

Console.WriteLine("Name: " + DynamicResult.Name);

Console.WriteLine("Version: " + DynamicResult.Version);

Passed = "Yes";

}

else

{

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_Sharepoint\_Hostname(string BaseAddress, string Hostname)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("sharepoint?hostName=" + Hostname);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "sharepoint?hostName=" + Hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Hostname == Hostname)

{

Console.WriteLine("SharePoint2003: " + DynamicResult.SharePoint2003);

Console.WriteLine("SharePoint2007: " + DynamicResult.SharePoint2007);

Console.WriteLine("SharePoint2010: " + DynamicResult.SharePoint2010);

Console.WriteLine("SharePoint2013: " + DynamicResult.SharePoint2013);

for (int i = 0; i < DynamicResult.RegistryKeys.Count; i++)

{

Console.Write("Dataset " + i + ": ");

Console.WriteLine("Registry Key found: " + DynamicResult.RegistryKeys[i].IsFound);

if (DynamicResult.RegistryKeys[i].IsFound == "false")

{

Passed = "Yes";

}

}

//Console.WriteLine(DynamicResult.DeviceName[1]);

Assert.AreEqual("Yes", Passed, "Registry Key not found or not returned by scanner.");

}

else

{

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_Sharepoint\_StartSharePointProcessing\_Domain(string BaseAddress, string Domain, string ExpectedHost)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("StartSharePointProcessing?domain=" + Domain);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "StartSharePointProcessing?domain=" + Domain);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Hosts.Count > 1)

{

for (int i = 0; i < DynamicResult.Hosts.Count; i++)

{

if (DynamicResult.Hosts[i] == ExpectedHost)

{

Passed = "Yes";

}

}

//Console.WriteLine(DynamicResult.DeviceName[1]);

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

Console.WriteLine("No Hosts found");

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_StartDFSProcessing\_Domain(string BaseAddress, string Domain, string ExpectedHost)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("StartDFSScanningProcessing?domain=" + Domain);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "StartDFSScanningProcessing?domain=" + Domain);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

string Passed = "No";

if (DynamicResult.Hosts.Count > 1)

{

for (int i = 0; i < DynamicResult.Hosts.Count; i++)

{

if (DynamicResult.Hosts[i] == ExpectedHost)

{

Passed = "Yes";

Console.WriteLine("Expected Host found in web service response.");

}

}

//Console.WriteLine(DynamicResult.DeviceName[1]);

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("No Hosts found");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_DFSScanning(string BaseAddress, string Host, string ExpectedHost)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("DFSScanning?hostname=" + Host);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "DFSScanning?hostname=" + Host);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

Assert.AreEqual(ExpectedHost.ToString().ToLower(), DynamicResult.Hostname.ToString().ToLower(), "Expected host not found in web service response.");

string Passed = "No";

if (DynamicResult.Links.Count > 1)

{

for (int i = 0; i < DynamicResult.Links.Count; i++)

{

if (DynamicResult.Links[i].DfsRootHost == ExpectedHost)

{

Passed = "Yes";

}

}

//Console.WriteLine(DynamicResult.DeviceName[1]);

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("No Hosts found");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_NetAppNASDetails(string BaseAddress, string Hostname, string ExpectedDomain, string ExpectedVirtualDeviceIP, string ExpectedVirtualDeviceName)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("netappnasdetails?hostname=" + Hostname);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "netappnasdetails?hostname=" + Hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult);

string Passed = "No";

string Error = "";

if (DynamicResult.LocalNasPhysicalDevice != null)

{

Console.WriteLine("HostName: " + DynamicResult.HostName);

Console.WriteLine("Success: " + DynamicResult.Success);

Console.WriteLine("PhysicalDevice Domain: " + DynamicResult.LocalNasPhysicalDevice.Domain);

try { Console.WriteLine("VirtualDeviceName: " + DynamicResult.LocalNasVirtualDevices[0].VirtualDeviceName); } catch { }

try { Console.WriteLine("VirtualDevice IPAddress: " + DynamicResult.LocalNasVirtualDevices[0].IpAddress); } catch { }

try { Console.WriteLine("VirtualDevice Domain: " + DynamicResult.LocalNasVirtualDevices[0].Domain); } catch { }

try { Console.WriteLine("VirtualDevice Status: " + DynamicResult.LocalNasVirtualDevices[0].Status); } catch { }

try { Console.WriteLine("VirtualDeviceName: " + DynamicResult.LocalNasVirtualDevices[0].VirtualDeviceName); } catch { }

try {

Error = DynamicResult.Errors[0].Message;

if (Error != "")

{

Console.WriteLine("Error message: " + Error);

}

} catch { }

Assert.AreEqual("", Error, "Error message: " + Error);

Assert.AreEqual(Hostname.ToLower(), DynamicResult.LocalNasPhysicalDevice.PhysicalDeviceName.ToString().ToLower(), "Hostname does not match.");

Assert.AreEqual(ExpectedDomain.ToLower(), DynamicResult.LocalNasPhysicalDevice.Domain.ToString().ToLower(), "Domain of Physical Device is not displayed as expected.");

Assert.AreEqual(ExpectedDomain.ToLower(), DynamicResult.LocalNasVirtualDevices[0].Domain.ToString().ToLower(), "Domain of Virtual Device is not displayed as expected.");

Assert.AreEqual(ExpectedVirtualDeviceIP, DynamicResult.LocalNasVirtualDevices[0].IpAddress.ToString(), "IP Address of VirtualDevice is either not displayed or has changed.");

Assert.AreEqual(ExpectedVirtualDeviceName.ToLower(), DynamicResult.LocalNasVirtualDevices[0].VirtualDeviceName.ToString().ToLower(), "VirtualDevice Name is not displayed as expected.");

Assert.AreEqual("true", DynamicResult.Success.ToString().ToLower(), "Success is not True.");

Passed = "Yes";

}

else

{

Passed = "No";

Console.WriteLine("Requested Hostname details not displayed");

try

{

Console.WriteLine("Message: " + DynamicResult.Errors[0].Message);

Error = DynamicResult.Errors[0].Message;

}

catch { }

}

Assert.AreEqual("Yes", Passed, "Expected result not found. Error (if any): " + Error);

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void Node\_JS\_Process\_logging\_message(string BaseAddress)

{

String Passed = "No";

WebRequestHandler handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var ParameterObject = new Node\_JS\_ProcessLog\_JsonParameter

{

Application = "TestApplication2",

Component = "TestComponent2",

TargetAction = "",

TargetSpecific = "",

LogLevel = 0,

Message = "Test Message 2",

LogTime = DateTime.Now,

};

var response = client.PostAsJsonAsync("ProcessLog?=", ParameterObject).Result;

Console.WriteLine("StatusCode: " + response.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

Console.WriteLine("Response: " + response.Content.ReadAsStringAsync().Result);

if (DynamicResult.Contains("Message successfully logged!") == true)

{

Passed = "Yes";

}

}

else

{

Console.WriteLine("Unexpected Result: " + response.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result + "\r\n");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Assert.AreEqual("Yes", Passed, "Received unexpected result.");

}

}

public void Node\_JS\_Process\_SharePointResult\_message(string BaseAddress, string Hostname, string Success, string SharePoint2003, string SharePoint2007, string SharePoint2010, string SharePoint2013, string RegistryKey, string RegistryKeyIsFound, string RegistryKeyValue, string VobId, string UniqueId, string DeviceName, string SerialNumber, string BarcodeNumber, string Manufacturer, string Model, string AssetNumber, string DeviceType, string Description, string Building, string Floor, string Platform, string Os, string MachineCategory, string MachineType, string StorageType, string BackupType, string SystemType, string InstallDate, string Country, string HaClusterNode, string Region, string Cluster, string InstallStatus, string UseStatus, string SpaceId, string GridLocation, string RackName, string BottomRackUnit, string ParentDeviceName, string RemsId, string Fqdn, string ASGVobId, string ASGSupportGroupType, string ASGSupportGroup, string ASGGroupEmail, string ASGGroupHotline, string ASGGroupUrl, string ASGPrimaryContactSoeId, string ASGSecondaryContactSoeId)

{

string Result = "Not Passed";

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

string ProcessSharePointResult\_JSON = "{\"SharePointInfo\": {\"Hostname\":\"" + Hostname + "\",\"Success\":" + Success + ",\"Errors\":[],\"SharePoint2003\":" + SharePoint2003 + ",\"SharePoint2007\":" + SharePoint2007 + ",\"SharePoint2010\":" + SharePoint2010 + ",\"SharePoint2013\":" + SharePoint2013 + ",\"RegistryKeys\":[{\"Key\":\"" + RegistryKey + "\",\"IsFound\":" + RegistryKeyIsFound + ",\"Value\":\"" + RegistryKeyValue + "\",\"RegistryView\":\"regView\"}],\"ApertureDeviceBasicInfo\":[{\"VobId\":\"" + VobId + "\",\"UniqueId\":\"" + UniqueId + "\",\"DeviceName\":\"" + DeviceName + "\",\"SerialNumber\":\"" + SerialNumber + "\",\"BarcodeNumber\":\"" + BarcodeNumber + "\",\"Manufacturer\":\"" + Manufacturer + "\",\"Model\":\"" + Model + "\",\"AssetNumber\":\"" + AssetNumber + "\",\"DeviceType\":\"" + DeviceType + "\",\"Description\":\"" + Description + "\",\"Building\":\"" + Building + "\",\"Floor\":\"" + Floor + "\",\"Platform\":\"" + Platform + "\",\"Os\":\"" + Os + "\",\"MachineCategory\":\"" + MachineCategory + "\",\"MachineType\":\"" + MachineType + "\",\"StorageType\":\"" + StorageType + "\",\"BackupType\":\"" + BackupType + "\",\"SystemType\":\"" + SystemType + "\",\"InstallDate\":" + InstallDate + ",\"Country\":\"" + Country + "\",\"HaClusterNode\":\"" + HaClusterNode + "\",\"Region\":\"" + Region + "\",\"Cluster\":\"" + Cluster + "\",\"InstallStatus\":\"" + InstallStatus + "\",\"UseStatus\":\"" + UseStatus + "\",\"SpaceId\":\"" + SpaceId + "\",\"GridLocation\":\"" + GridLocation + "\",\"RackName\":\"" + RackName + "\",\"BottomRackUnit\":\"" + BottomRackUnit + "\",\"ParentDeviceName\":\"" + ParentDeviceName + "\",\"RemsId\":\"" + RemsId + "\",\"Fqdn\":\"" + Fqdn + "\",\"ApertureSupportGroups\":[{\"VobId\":\"" + ASGVobId + "\",\"SupportGroupType\":\"" + ASGSupportGroupType + "\",\"SupportGroup\":\"" + ASGSupportGroup + "\",\"GroupEmail\":\"" + ASGGroupEmail + "\",\"GroupHotline\":\"" + ASGGroupHotline + "\",\"GroupUrl\":\"" + ASGGroupUrl + "\",\"PrimaryContactSoeId\":\"" + ASGPrimaryContactSoeId + "\",\"SecondaryContactSoeId\":\"" + ASGSecondaryContactSoeId + "\"}]}]}}";

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

var httpContent = new HttpRequestMessage(HttpMethod.Post, BaseAddress);

httpContent.Content = new StringContent(ProcessSharePointResult\_JSON, Encoding.UTF8, "application/json");

client.Timeout = new TimeSpan(0, 0, 4, 0, 0);

var response = client.PostAsync("ProcessSharePointResult", httpContent.Content).Result;

Console.WriteLine("Input: " + ProcessSharePointResult\_JSON);

Console.WriteLine("StatusCode: " + response.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

Console.WriteLine("Response: " + response.Content.ReadAsStringAsync().Result);

if (

response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("sharepointinfo for host " + Hostname.ToLower()) == true

&

(

response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("updated") == true

| response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("saved") == true

)

)

{

Result = "Passed";

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

else

{

Console.WriteLine("Unexpected Result: " + response.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result + "\r\n");

}

Assert.AreEqual("Passed", Result, "Expected entry not found.");

}

}

public void Node\_JS\_Process\_NetAppNASDetailsResult\_message(string BaseAddress, string Hostname, string OsVersion, string SerialNo, string IpAddress, string Vendor, string Domain, string Model, string ClusterPartner, string ArchivedDate, string Virtual0DeviceName, string Virtual0IpAddress, string Virtual0Domain, string Virtual0Status, string Virtual0ArchivedDate, string LocalVolume1VirtualDeviceName, string LocalVolume1VolumeName, string LocalVolume1Security, string LocalVolume1DedupeStatus, string LocalVolume1Used, string LocalVolume1Size, string LocalVolume1SisSavings, string LocalVolume1CobPath, string LocalVolume1ArchivedDate, string QTree0QtreeName, string QTree0VolumeName, string QTree0Security, string QTree0Used, string QTree0Size, string QTree0ArchivedDate, string Priority, string TryCount, string Expiration, string ApertureApertureDeviceBasicInfoId, string ApertureVobId, string ApertureUniqueId, string ApertureDeviceName, string ApertureSerialNumber, string ApertureBarcodeNumber, string ApertureManufacturer, string ApertureAssetNumber, string ApertureDeviceType, string ApertureDescription, string ApertureBuilding, string ApertureFloor, string AperturePlatform, string ApertureOs, string ApertureMachineCategory, string ApertureMachineType, string ApertureStorageType, string ApertureBackupType, string ApertureSystemType, string ApertureInstallDate, string ApertureCountry, string ApertureHaClusterNode, string ApertureRegion, string ApertureCluster, string ApertureInstallStatus, string ApertureUseStatus, string ApertureSpaceId, string ApertureGridLocation, string ApertureRackName, string ApertureBottomRackUnit, string ApertureParentDeviceName, string ApertureRemsId, string ApertureFqdn, string ApertureArchivedDate, string ApertureSupportGroupId, string ApertureDeviceBasicInfoId, string ApertureSupportVobId, string ApertureSupportGroupType, string ApertureSupportGroup, string ApertureGroupEmail, string ApertureGroupHotline, string ApertureGroupUrl, string AperturePrimaryContactSoeId, string ApertureSecondaryContactSoeId, string ApertureModel, string MainArchivedDate)

{

string Result = "Not Passed";

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

string ProcessSharePointResult\_JSON = "{\"NasDetails\":{\"HostName\":\"" + Hostname + "\",\"ArchivedDate\":\"" + MainArchivedDate + "\",\"LocalNasPhysicalDevice\":{\"PhysicalDeviceName\":\"" + Hostname + "\",\"OsVersion\":\"" + OsVersion + "\",\"SerialNo\":\"" + SerialNo + "\",\"IpAddress\":\"" + IpAddress + "\",\"Vendor\":\"" + Vendor + "\",\"Domain\":\"" + Domain + "\",\"Model\":\"" + Model + "\",\"ClusterPartner\":\"" + ClusterPartner + "\",\"ArchivedDate\":\"" + ArchivedDate + "\"},\"LocalNasVirtualDevices\":[{\"VirtualDeviceName\":\"" + Virtual0DeviceName + "\",\"PhysicalDeviceName\":\"" + Hostname + "\",\"IpAddress\":\"" + Virtual0IpAddress + "\",\"Domain\":\"" + Virtual0Domain + "\",\"Status\":\"" + Virtual0Status + "\",\"ArchivedDate\":\"" + Virtual0ArchivedDate + "\"},{\"VirtualDeviceName\":\"ldnvnasgrp0002\",\"PhysicalDeviceName\":\"" + Hostname + "\",\"IpAddress\":\"169.182.224.43\",\"Domain\":\"eur.nsroot.net\",\"Status\":\"running\",\"ArchivedDate\":null}],\"LocalNasVolumes\":[{\"VirtualDeviceName\":\"vfiler0\",\"VolumeName\":\"vol0\",\"Security\":\"ntfs\",\"DedupeStatus\":null,\"Used\":11150098432.0,\"Size\":335007449088.0,\"SisSavings\":null,\"CobPath\":null,\"ArchivedDate\":null},{\"VirtualDeviceName\":\"" + LocalVolume1VirtualDeviceName + "\",\"VolumeName\":\"" + LocalVolume1VolumeName + "\",\"Security\":\"" + LocalVolume1Security + "\",\"DedupeStatus\":\"" + LocalVolume1DedupeStatus + "\",\"Used\":\"" + LocalVolume1Used + "\",\"Size\":\"" + LocalVolume1Size + "\",\"SisSavings\":\"" + LocalVolume1SisSavings + "\",\"CobPath\":\"" + LocalVolume1CobPath + "\",\"ArchivedDate\":\"" + LocalVolume1ArchivedDate + "\"}],\"LocalNasQTrees\":[{\"QtreeName\":\"" + QTree0QtreeName + "\",\"VolumeName\":\"" + QTree0VolumeName + "\",\"Security\":\"" + QTree0Security + "\",\"Used\":\"" + QTree0Used + "\",\"Size\":\"" + QTree0Size + "\",\"ArchivedDate\":\"" + QTree0ArchivedDate + "\"},{\"QtreeName\":\"grpqt0002\_grid\_deploy\_prd\",\"VolumeName\":\"ldngrpv0002\",\"Security\":\"unix\",\"Used\":null,\"Size\":null,\"ArchivedDate\":null}],\"Success\":true,\"Errors\":[],\"ApertureDeviceBasicInfo\":{\"ApertureDeviceBasicInfoId\":67639,\"VobId\":\"" + ApertureVobId + "\",\"UniqueId\":\"" + ApertureUniqueId + "\",\"DeviceName\":\"" + ApertureDeviceName + "\",\"SerialNumber\":\"" + ApertureSerialNumber + "\",\"BarcodeNumber\":\"" + ApertureBarcodeNumber + "\",\"Manufacturer\":\"" + ApertureManufacturer + "\",\"Model\":\"" + ApertureModel + "\",\"AssetNumber\":\"" + ApertureAssetNumber + "\",\"DeviceType\":\"" + ApertureDeviceType + "\",\"Description\":\"" + ApertureDescription + "\",\"Building\":\"" + ApertureBuilding + "\",\"Floor\":\"" + ApertureFloor + "\",\"Platform\":\"" + AperturePlatform + "\",\"Os\":\"" + ApertureOs + "\",\"MachineCategory\":\"" + ApertureMachineCategory + "\",\"MachineType\":\"" + ApertureMachineType + "\",\"StorageType\":\"" + ApertureStorageType + "\",\"BackupType\":\"" + ApertureBackupType + "\",\"SystemType\":\"" + ApertureSystemType + "\",\"InstallDate\":\"" + ApertureInstallDate + "\",\"Country\":\"" + ApertureCountry + "\",\"HaClusterNode\":\"" + ApertureHaClusterNode + "\",\"Region\":\"" + ApertureRegion + "\",\"Cluster\":\"" + ApertureCluster + "\",\"InstallStatus\":\"" + ApertureInstallStatus + "\",\"UseStatus\":\"" + ApertureUseStatus + "\",\"SpaceId\":\"" + ApertureSpaceId + "\",\"GridLocation\":\"" + ApertureGridLocation + "\",\"RackName\":\"" + ApertureRackName + "\",\"BottomRackUnit\":\"" + ApertureBottomRackUnit + "\",\"ParentDeviceName\":\"" + ApertureParentDeviceName + "\",\"RemsId\":\"" + ApertureRemsId + "\",\"Fqdn\":\"" + ApertureFqdn + "\",\"ArchivedDate\":\"" + ApertureArchivedDate + "\",\"ApertureSupportGroups\":[{\"ApertureSupportGroupId\":15715,\"ApertureDeviceBasicInfoId\":67639,\"VobId\":\"" + ApertureSupportVobId + "\",\"SupportGroupType\":\"" + ApertureSupportGroupType + "\",\"SupportGroup\":\"" + ApertureSupportGroup + "\",\"GroupEmail\":\"" + ApertureGroupEmail + "\",\"GroupHotline\":\"" + ApertureGroupHotline + "\",\"GroupUrl\":\"" + ApertureGroupUrl + "\",\"PrimaryContactSoeId\":\"" + AperturePrimaryContactSoeId + "\",\"SecondaryContactSoeId\":\"" + ApertureSecondaryContactSoeId + "\"}]}},\"Priority\":\"" + Priority + "\",\"TryCount\":\"" + TryCount + "\",\"Expiration\":\"" + Expiration + "\"}";

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

var httpContent = new HttpRequestMessage(HttpMethod.Post, BaseAddress);

httpContent.Content = new StringContent(ProcessSharePointResult\_JSON, Encoding.UTF8, "application/json");

client.Timeout = new TimeSpan(0, 0, 4, 0, 0);

var response = client.PostAsync("ProcessNetAppNasDetailsResult", httpContent.Content).Result;

Console.WriteLine("Input: " + ProcessSharePointResult\_JSON);

Console.WriteLine("StatusCode: " + response.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

Console.WriteLine("Response: " + response.Content.ReadAsStringAsync().Result);

if (

response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("nasdetails for host " + Hostname.ToLower()) == true

&

(

response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("updated") == true

| response.Content.ReadAsStringAsync().Result.ToString().ToLower().Contains("saved") == true

)

)

{

Result = "Passed";

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

else

{

Console.WriteLine("Unexpected Result: " + response.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result + "\r\n");

}

Assert.AreEqual("Passed", Result, "Expected entry not found.");

}

}

public void Node\_JS\_Get\_SharePoint\_Information\_For\_Host(string BaseAddress, string Hostname, string SharePoint2003, string SharePoint2007, string SharePoint2010, string SharePoint2013, string RegistryKey, string RegistryKeyIsFound, string RegistryKeyValue, string VobId, string UniqueId, string DeviceName, string SerialNumber, string BarcodeNumber, string Manufacturer, string Model, string AssetNumber, string DeviceType, string Description, string Building, string Floor, string Platform, string Os, string MachineCategory, string MachineType, string StorageType, string BackupType, string SystemType, string InstallDate, string Country, string HaClusterNode, string Region, string Cluster, string InstallStatus, string UseStatus, string SpaceId, string GridLocation, string RackName, string BottomRackUnit, string ParentDeviceName, string RemsId, string Fqdn, string ASGVobId, string ASGSupportGroupType, string ASGSupportGroup, string ASGGroupEmail, string ASGGroupHotline, string ASGGroupUrl, string ASGPrimaryContactSoeId, string ASGSecondaryContactSoeId)

{

int Result = 0;

int NumberOfFields = 48;

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var response = client.GetAsync("GetSharePointInformationForHost/" + Hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

//Console.WriteLine("Response: " + response.Result.Content.ReadAsStringAsync().Result);

//1

if (DynamicResult[0].SharePoint2003.ToString().ToLower() == SharePoint2003.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 1"); }

else { Console.WriteLine("Expected: " + SharePoint2003.ToString().ToLower() + " Actual: " + DynamicResult[0].SharePoint2003.ToString().ToLower()); }

//2

if (DynamicResult[0].SharePoint2007.ToString().ToLower() == SharePoint2007.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 2"); }

else { Console.WriteLine("Expected: " + SharePoint2007.ToString().ToLower() + " Actual: " + DynamicResult[0].SharePoint2007.ToString().ToLower()); }

//3

if (DynamicResult[0].SharePoint2010.ToString().ToLower() == SharePoint2010.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 3"); }

else { Console.WriteLine("Expected: " + SharePoint2010.ToString().ToLower() + " Actual: " + DynamicResult[0].SharePoint2010.ToString().ToLower()); }

//4

if (DynamicResult[0].SharePoint2013.ToString().ToLower() == SharePoint2013.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 4"); }

else { Console.WriteLine("Expected: " + SharePoint2013.ToString().ToLower() + " Actual: " + DynamicResult[0].SharePoint2013.ToString().ToLower()); }

//5

if (DynamicResult[0].RegistryKeys[0].Key.ToString().ToLower() == RegistryKey.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 5"); }

else { Console.WriteLine("Expected: " + RegistryKey.ToString().ToLower() + " Actual: " + DynamicResult[0].RegistryKeys[0].Key.ToString().ToLower()); }

//6

if (DynamicResult[0].RegistryKeys[0].IsFound.ToString().ToLower() == RegistryKeyIsFound.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 6"); }

else { Console.WriteLine("Expected: " + RegistryKeyIsFound.ToString().ToLower() + " Actual: " + DynamicResult[0].RegistryKeys[0].IsFound.ToString().ToLower()); }

//7

if (DynamicResult[0].RegistryKeys[0].Value.ToString().ToLower() == RegistryKeyValue.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 7"); }

else { Console.WriteLine("Expected: " + RegistryKeyValue.ToString().ToLower() + " Actual: " + DynamicResult[0].RegistryKeys[0].Value.ToString().ToLower()); }

//8

if (DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower() == VobId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 8"); }

else { Console.WriteLine("Expected: " + VobId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower()); }

//9

if (DynamicResult[0].ApertureDeviceBasicInfo[0].UniqueId.ToString().ToLower() == UniqueId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 9"); }

else { Console.WriteLine("Expected: " + UniqueId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].UniqueId.ToString().ToLower()); }

//10

if (DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceName.ToString().ToLower() == DeviceName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 10"); }

else { Console.WriteLine("Expected: " + DeviceName.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceName.ToString().ToLower()); }

//11

if (DynamicResult[0].ApertureDeviceBasicInfo[0].SerialNumber.ToString().ToLower() == SerialNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 11"); }

else { Console.WriteLine("Expected: " + SerialNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].SerialNumber.ToString().ToLower()); }

//12

if (DynamicResult[0].ApertureDeviceBasicInfo[0].BarcodeNumber.ToString().ToLower() == BarcodeNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 12"); }

else { Console.WriteLine("Expected: " + BarcodeNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].BarcodeNumber.ToString().ToLower()); }

//13

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Manufacturer.ToString().ToLower() == Manufacturer.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 13"); }

else { Console.WriteLine("Expected: " + Manufacturer.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Manufacturer.ToString().ToLower()); }

//14

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Model.ToString().ToLower() == Model.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 14"); }

else { Console.WriteLine("Expected: " + Model.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Model.ToString().ToLower()); }

//15

if (DynamicResult[0].ApertureDeviceBasicInfo[0].AssetNumber.ToString().ToLower() == AssetNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 15"); }

else { Console.WriteLine("Expected: " + AssetNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].AssetNumber.ToString().ToLower()); }

//16

if (DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceType.ToString().ToLower() == DeviceType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 16"); }

else { Console.WriteLine("Expected: " + DeviceType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceType.ToString().ToLower()); }

//17

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Description.ToString().ToLower() == Description.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 17"); }

else { Console.WriteLine("Expected: " + Description.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Description.ToString().ToLower()); }

//18

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Building.ToString().ToLower() == Building.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 18"); }

else { Console.WriteLine("Expected: " + Building.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Building.ToString().ToLower()); }

//19

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Floor.ToString().ToLower() == Floor.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 19"); }

else { Console.WriteLine("Expected: " + Floor.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Floor.ToString().ToLower()); }

//20

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Platform.ToString().ToLower() == Platform.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 20"); }

else { Console.WriteLine("Expected: " + Platform.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Platform.ToString().ToLower()); }

//21

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Os.ToString().ToLower() == Os.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 21"); }

else { Console.WriteLine("Expected: " + Os.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Os.ToString().ToLower()); }

//22

if (DynamicResult[0].ApertureDeviceBasicInfo[0].MachineCategory.ToString().ToLower() == MachineCategory.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 22"); }

else { Console.WriteLine("Expected: " + MachineCategory.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].MachineCategory.ToString().ToLower()); }

//23

if (DynamicResult[0].ApertureDeviceBasicInfo[0].MachineType.ToString().ToLower() == MachineType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 23"); }

else { Console.WriteLine("Expected: " + MachineType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].MachineType.ToString().ToLower()); }

//24

if (DynamicResult[0].ApertureDeviceBasicInfo[0].StorageType.ToString().ToLower() == StorageType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 24"); }

else { Console.WriteLine("Expected: " + StorageType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].StorageType.ToString().ToLower()); }

//25

if (DynamicResult[0].ApertureDeviceBasicInfo[0].BackupType.ToString().ToLower() == BackupType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 25"); }

else { Console.WriteLine("Expected: " + BackupType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].BackupType.ToString().ToLower()); }

//26

if (DynamicResult[0].ApertureDeviceBasicInfo[0].SystemType.ToString().ToLower() == SystemType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 26"); }

else { Console.WriteLine("Expected: " + SystemType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].SystemType.ToString().ToLower()); }

//27

var epoch = new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

if (DynamicResult[0].ApertureDeviceBasicInfo[0].InstallDate == epoch.AddMilliseconds(Convert.ToInt64(InstallDate.ToString())))

{ Result++; Console.WriteLine("Passed test No. 27"); }

else { Console.WriteLine("Expected: " + epoch.AddMilliseconds(Convert.ToInt64(InstallDate.ToString())) + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].InstallDate); }

//28

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Country.ToString().ToLower() == Country.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 28"); }

else { Console.WriteLine("Expected: " + Country.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Country.ToString().ToLower()); }

//29

if (DynamicResult[0].ApertureDeviceBasicInfo[0].HaClusterNode.ToString().ToLower() == HaClusterNode.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 29"); }

else { Console.WriteLine("Expected: " + HaClusterNode.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].HaClusterNode.ToString().ToLower()); }

//30

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Region.ToString().ToLower() == Region.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 30"); }

else { Console.WriteLine("Expected: " + Region.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Region.ToString().ToLower()); }

//31

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Cluster.ToString().ToLower() == Cluster.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 31"); }

else { Console.WriteLine("Expected: " + Cluster.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Cluster.ToString().ToLower()); }

//32

if (DynamicResult[0].ApertureDeviceBasicInfo[0].InstallStatus.ToString().ToLower() == InstallStatus.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 32"); }

else { Console.WriteLine("Expected: " + InstallStatus.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].InstallStatus.ToString().ToLower()); }

//33

if (DynamicResult[0].ApertureDeviceBasicInfo[0].UseStatus.ToString().ToLower() == UseStatus.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 33"); }

else { Console.WriteLine("Expected: " + UseStatus.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].UseStatus.ToString().ToLower()); }

//34

if (DynamicResult[0].ApertureDeviceBasicInfo[0].SpaceId.ToString().ToLower() == SpaceId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 34"); }

else { Console.WriteLine("Expected: " + SpaceId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].SpaceId.ToString().ToLower()); }

//35

if (DynamicResult[0].ApertureDeviceBasicInfo[0].GridLocation.ToString().ToLower() == GridLocation.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 35"); }

else { Console.WriteLine("Expected: " + GridLocation.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].GridLocation.ToString().ToLower()); }

//36

if (DynamicResult[0].ApertureDeviceBasicInfo[0].RackName.ToString().ToLower() == RackName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 36"); }

else { Console.WriteLine("Expected: " + RackName.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].RackName.ToString().ToLower()); }

//37

if (DynamicResult[0].ApertureDeviceBasicInfo[0].BottomRackUnit.ToString().ToLower() == BottomRackUnit.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 37"); }

else { Console.WriteLine("Expected: " + BottomRackUnit.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].BottomRackUnit.ToString().ToLower()); }

//38

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ParentDeviceName.ToString().ToLower() == ParentDeviceName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 38"); }

else { Console.WriteLine("Expected: " + ParentDeviceName.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ParentDeviceName.ToString().ToLower()); }

//39

if (DynamicResult[0].ApertureDeviceBasicInfo[0].RemsId.ToString().ToLower() == RemsId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 39"); }

else { Console.WriteLine("Expected: " + RemsId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].RemsId.ToString().ToLower()); }

//40

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Fqdn.ToString().ToLower() == Fqdn.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 40"); }

else { Console.WriteLine("Expected: " + Fqdn.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Fqdn.ToString().ToLower()); }

//41

if (DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower() == ASGVobId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 41"); }

else { Console.WriteLine("ASGVobId Expected: " + ASGVobId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower()); }

//42

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SupportGroupType.ToString().ToLower() == ASGSupportGroupType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 42"); }

else { Console.WriteLine("Expected: " + ASGSupportGroupType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SupportGroupType.ToString().ToLower()); }

//43

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SupportGroup.ToString().ToLower() == ASGSupportGroup.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 43"); }

else { Console.WriteLine("Expected: " + ASGSupportGroup.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SupportGroup.ToString().ToLower()); }

//44

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupEmail.ToString().ToLower() == ASGGroupEmail.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 44"); }

else { Console.WriteLine("Expected: " + ASGGroupEmail.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupEmail.ToString().ToLower()); }

//45

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupHotline.ToString().ToLower() == ASGGroupHotline.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 45"); }

else { Console.WriteLine("Expected: " + ASGGroupHotline.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupHotline.ToString().ToLower()); }

//46

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupUrl.ToString().ToLower() == ASGGroupUrl.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 46"); }

else { Console.WriteLine("Expected: " + ASGGroupUrl.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].GroupUrl.ToString().ToLower()); }

//47

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].PrimaryContactSoeId.ToString().ToLower() == ASGPrimaryContactSoeId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 47"); }

else { Console.WriteLine("Expected: " + ASGPrimaryContactSoeId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].PrimaryContactSoeId.ToString().ToLower()); }

//48

if (DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SecondaryContactSoeId.ToString().ToLower() == ASGSecondaryContactSoeId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 48"); }

else { Console.WriteLine("Expected: " + ASGSecondaryContactSoeId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].ApertureSupportGroups[0].SecondaryContactSoeId.ToString().ToLower()); }

}

else

{

Console.WriteLine("Unexpected Result: " + response.Result.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result + "\r\n");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Assert.AreEqual(NumberOfFields, Result, "Expected number of fields passing verification did not match.");

}

}

public void Node\_JS\_Get\_NetAppNASDetails\_For\_Host(string BaseAddress, string Hostname, string OsVersion, string SerialNo, string IpAddress, string Vendor, string Domain, string Model, string ClusterPartner, string ArchivedDate, string Virtual0DeviceName, string Virtual0IpAddress, string Virtual0Domain, string Virtual0Status, string Virtual0ArchivedDate, string LocalVolume1VirtualDeviceName, string LocalVolume1VolumeName, string LocalVolume1Security, string LocalVolume1DedupeStatus, string LocalVolume1Used, string LocalVolume1Size, string LocalVolume1SisSavings, string LocalVolume1CobPath, string LocalVolume1ArchivedDate, string QTree0QtreeName, string QTree0VolumeName, string QTree0Security, string QTree0Used, string QTree0Size, string QTree0ArchivedDate, string Priority, string TryCount, string Expiration, string ApertureApertureDeviceBasicInfoId, string ApertureVobId, string ApertureUniqueId, string ApertureDeviceName, string ApertureSerialNumber, string ApertureBarcodeNumber, string ApertureManufacturer, string ApertureAssetNumber, string ApertureDeviceType, string ApertureDescription, string ApertureBuilding, string ApertureFloor, string AperturePlatform, string ApertureOs, string ApertureMachineCategory, string ApertureMachineType, string ApertureStorageType, string ApertureBackupType, string ApertureSystemType, string ApertureInstallDate, string ApertureCountry, string ApertureHaClusterNode, string ApertureRegion, string ApertureCluster, string ApertureInstallStatus, string ApertureUseStatus, string ApertureSpaceId, string ApertureGridLocation, string ApertureRackName, string ApertureBottomRackUnit, string ApertureParentDeviceName, string ApertureRemsId, string ApertureFqdn, string ApertureArchivedDate, string ApertureSupportGroupId, string ApertureDeviceBasicInfoId, string ApertureSupportVobId, string ApertureSupportGroupType, string ApertureSupportGroup, string ApertureGroupEmail, string ApertureGroupHotline, string ApertureGroupUrl, string AperturePrimaryContactSoeId, string ApertureSecondaryContactSoeId, string ApertureModel, string MainArchivedDate)

{

int Result = 0;

int NumberOfFields = 50;

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var response = client.GetAsync("GetNasDetailsForHost/" + Hostname);

Console.WriteLine("URL: " + BaseAddress + "GetNasDetailsForHost/" + Hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK" & response.Result.Content.ReadAsStringAsync().Result != "[]")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine("DynamicResult: " + DynamicResult);

//Console.WriteLine("Response: " + response.Result.Content.ReadAsStringAsync().Result);

//1

if (DynamicResult[0].HostName.ToString().ToLower() == Hostname.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 1"); }

else { Console.WriteLine("Expected: " + Hostname.ToString().ToLower() + " Actual: " + DynamicResult[0].HostName.ToString().ToLower()); }

//2

if (DynamicResult[0].LocalNasPhysicalDevice.OsVersion.ToString().ToLower() == OsVersion.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 2"); }

else { Console.WriteLine("Expected: " + OsVersion.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.OsVersion.ToString().ToLower()); }

//3

if (DynamicResult[0].LocalNasPhysicalDevice.SerialNo.ToString().ToLower() == SerialNo.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 3"); }

else { Console.WriteLine("Expected: " + SerialNo.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.SerialNo.ToString().ToLower()); }

//4

if (DynamicResult[0].LocalNasPhysicalDevice.IpAddress.ToString().ToLower() == IpAddress.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 4"); }

else { Console.WriteLine("Expected: " + IpAddress.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.IpAddress.ToString().ToLower()); }

//5

if (DynamicResult[0].LocalNasPhysicalDevice.Vendor.ToString().ToLower() == Vendor.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 5"); }

else { Console.WriteLine("Expected: " + Vendor.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.Vendor.ToString().ToLower()); }

//6

if (DynamicResult[0].LocalNasPhysicalDevice.Domain.ToString().ToLower() == Domain.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 6"); }

else { Console.WriteLine("Expected: " + Domain.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.Domain.ToString().ToLower()); }

//7

if (DynamicResult[0].LocalNasPhysicalDevice.Model.ToString().ToLower() == Model.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 7"); }

else { Console.WriteLine("Expected: " + Model.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.Model.ToString().ToLower()); }

//8

if (DynamicResult[0].LocalNasPhysicalDevice.ClusterPartner.ToString().ToLower() == ClusterPartner.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 8"); }

else { Console.WriteLine("Expected: " + ClusterPartner.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.ClusterPartner.ToString().ToLower()); }

//9

if (DateTime.Parse(DynamicResult[0].LocalNasPhysicalDevice.ArchivedDate.ToString()) == DateTime.Parse(ArchivedDate.ToString()))

{ Result++; Console.WriteLine("Passed test No. 9"); }

else { Console.WriteLine("Expected: " + ArchivedDate.ToString() + " Actual: " + DynamicResult[0].LocalNasPhysicalDevice.ArchivedDate.ToString()); }

//10

if (DynamicResult[0].LocalNasVirtualDevices[0].VirtualDeviceName.ToString().ToLower() == Virtual0DeviceName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 10"); }

else { Console.WriteLine("Expected: " + Virtual0DeviceName.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVirtualDevices[0].VirtualDeviceName.ToString().ToLower()); }

//11

if (DynamicResult[0].LocalNasVirtualDevices[0].IpAddress.ToString().ToLower() == Virtual0IpAddress.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 11"); }

else { Console.WriteLine("Expected: " + Virtual0IpAddress.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVirtualDevices[0].IpAddress.ToString().ToLower()); }

//12

if (DynamicResult[0].LocalNasVirtualDevices[0].Domain.ToString().ToLower() == Virtual0Domain.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 12"); }

else { Console.WriteLine("Expected: " + Virtual0Domain.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVirtualDevices[0].Domain.ToString().ToLower()); }

//13

if (DynamicResult[0].LocalNasVirtualDevices[0].Status.ToString().ToLower() == Virtual0Status.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 13"); }

else { Console.WriteLine("Expected: " + Virtual0Status.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVirtualDevices[0].Status.ToString().ToLower()); }

//14

if (DateTime.Parse(DynamicResult[0].LocalNasVirtualDevices[0].ArchivedDate.ToString()) == DateTime.Parse(Virtual0ArchivedDate.ToString()))

{ Result++; Console.WriteLine("Passed test No. 14"); }

else { Console.WriteLine("Expected: " + Virtual0ArchivedDate.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVirtualDevices[0].ArchivedDate.ToString().ToLower()); }

//15

if (DynamicResult[0].LocalNasVolumes[1].VirtualDeviceName.ToString().ToLower() == LocalVolume1VirtualDeviceName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 15"); }

else { Console.WriteLine("Expected: " + LocalVolume1VirtualDeviceName.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].VirtualDeviceName.ToString().ToLower()); }

//16

if (DynamicResult[0].LocalNasVolumes[1].VolumeName.ToString().ToLower() == LocalVolume1VolumeName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 16"); }

else { Console.WriteLine("Expected: " + LocalVolume1VolumeName.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].VolumeName.ToString().ToLower()); }

//17

if (DynamicResult[0].LocalNasVolumes[1].Security.ToString().ToLower() == LocalVolume1Security.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 17"); }

else { Console.WriteLine("Expected: " + LocalVolume1Security.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].Security.ToString().ToLower()); }

//18

if (DynamicResult[0].LocalNasVolumes[1].DedupeStatus.ToString().ToLower() == LocalVolume1DedupeStatus.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 18"); }

else { Console.WriteLine("Expected: " + LocalVolume1DedupeStatus.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].DedupeStatus.ToString().ToLower()); }

//19

if (DynamicResult[0].LocalNasVolumes[1].Used.ToString().ToLower() == LocalVolume1Used.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 19"); }

else { Console.WriteLine("Expected: " + LocalVolume1Used.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].Used.ToString().ToLower()); }

//20

if (DynamicResult[0].LocalNasVolumes[1].Size.ToString().ToLower() == LocalVolume1Size.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 20"); }

else { Console.WriteLine("Expected: " + LocalVolume1Size.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].Size.ToString().ToLower()); }

//21

if (DynamicResult[0].LocalNasVolumes[1].SisSavings.ToString().ToLower() == LocalVolume1SisSavings.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 21"); }

else { Console.WriteLine("Expected: " + LocalVolume1SisSavings.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].SisSavings.ToString().ToLower()); }

//22

if (DynamicResult[0].LocalNasVolumes[1].CobPath.ToString().ToLower() == LocalVolume1CobPath.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 22"); }

else { Console.WriteLine("Expected: " + LocalVolume1CobPath.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].CobPath.ToString().ToLower()); }

//23

if (DateTime.Parse(DynamicResult[0].LocalNasVolumes[1].ArchivedDate.ToString()) == DateTime.Parse(LocalVolume1ArchivedDate.ToString()))

{ Result++; Console.WriteLine("Passed test No. 23"); }

else { Console.WriteLine("Expected: " + LocalVolume1ArchivedDate.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasVolumes[1].ArchivedDate.ToString().ToLower()); }

//24

if (DynamicResult[0].LocalNasQTrees[0].QtreeName.ToString().ToLower() == QTree0QtreeName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 24"); }

else { Console.WriteLine("Expected: " + QTree0QtreeName.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasQTrees[0].QtreeName.ToString().ToLower()); }

//25

if (DynamicResult[0].LocalNasQTrees[0].VolumeName.ToString().ToLower() == QTree0VolumeName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 25"); }

else { Console.WriteLine("Expected: " + QTree0VolumeName.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasQTrees[0].VolumeName.ToString().ToLower()); }

//26

if (DynamicResult[0].LocalNasQTrees[0].Security.ToString().ToLower() == QTree0Security.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 26"); }

else { Console.WriteLine("Expected: " + QTree0Security.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasQTrees[0].Security.ToString().ToLower()); }

//27

if (DynamicResult[0].LocalNasQTrees[0].Used.ToString().ToLower() == QTree0Used.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 27"); }

else { Console.WriteLine("Expected: " + QTree0Used.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasQTrees[0].Used.ToString().ToLower()); }

//28

if (DynamicResult[0].LocalNasQTrees[0].Size.ToString().ToLower() == QTree0Size.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 28"); }

else { Console.WriteLine("Expected: " + QTree0Size.ToString().ToLower() + " Actual: " + DynamicResult[0].LocalNasQTrees[0].Size.ToString().ToLower()); }

//29

if (DateTime.Parse(DynamicResult[0].LocalNasQTrees[0].ArchivedDate.ToString()) == DateTime.Parse(QTree0ArchivedDate.ToString()))

{ Result++; Console.WriteLine("Passed test No. 29"); }

else { Console.WriteLine("Expected: " + DateTime.Parse(QTree0ArchivedDate.ToString()) + " Actual: " + DateTime.Parse(DynamicResult[0].LocalNasQTrees[0].ArchivedDate.ToString())); }

//value not required and not returned, skipping

//if (DynamicResult[0].Priority.ToString().ToLower() == Priority.ToString().ToLower())

//{ Result++; Console.WriteLine("Passed test No. 30"); }

//else { Console.WriteLine("Expected: " + Priority.ToString().ToLower() + " Actual: " + DynamicResult[0].Priority.ToString().ToLower()); }

//value not required and not returned, skipping

//if (DynamicResult[0].TryCount.ToString().ToLower() == TryCount.ToString().ToLower())

//{ Result++; Console.WriteLine("Passed test No. 31"); }

//else { Console.WriteLine("Expected: " + TryCount.ToString().ToLower() + " Actual: " + DynamicResult[0].TryCount.ToString().ToLower()); }

//value not required and not returned, skipping

//if (DynamicResult[0].Expiration.ToString().ToLower() == Expiration.ToString().ToLower())

//{ Result++; Console.WriteLine("Passed test No. 32"); }

//else { Console.WriteLine("Expected: " + Expiration.ToString().ToLower() + " Actual: " + DynamicResult[0].Expiration.ToString().ToLower()); }

//30

if (DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower() == ApertureVobId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 30"); }

else { Console.WriteLine("Expected: " + ApertureVobId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].VobId.ToString().ToLower()); }

//31

if (DynamicResult[0].ApertureDeviceBasicInfo[0].UniqueId.ToString().ToLower() == ApertureUniqueId.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 31"); }

else { Console.WriteLine("Expected: " + ApertureUniqueId.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].UniqueId.ToString().ToLower()); }

//32

if (DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceName.ToString().ToLower() == ApertureDeviceName.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 32"); }

else { Console.WriteLine("Expected: " + ApertureDeviceName.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceName.ToString().ToLower()); }

//33

if (DynamicResult[0].ApertureDeviceBasicInfo[0].SerialNumber.ToString().ToLower() == ApertureSerialNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 33"); }

else { Console.WriteLine("Expected: " + ApertureSerialNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].SerialNumber.ToString().ToLower()); }

//34

if (DynamicResult[0].ApertureDeviceBasicInfo[0].BarcodeNumber.ToString().ToLower() == ApertureBarcodeNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 34"); }

else { Console.WriteLine("Expected: " + ApertureBarcodeNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].BarcodeNumber.ToString().ToLower()); }

//35

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Manufacturer.ToString().ToLower() == ApertureManufacturer.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 35"); }

else { Console.WriteLine("Expected: " + ApertureManufacturer.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Manufacturer.ToString().ToLower()); }

//36

if (DynamicResult[0].ApertureDeviceBasicInfo[0].AssetNumber.ToString().ToLower() == ApertureAssetNumber.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 36"); }

else { Console.WriteLine("Expected: " + ApertureAssetNumber.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].AssetNumber.ToString().ToLower()); }

//37

if (DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceType.ToString().ToLower() == ApertureDeviceType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 37"); }

else { Console.WriteLine("Expected: " + ApertureDeviceType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].DeviceType.ToString().ToLower()); }

//38

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Description.ToString().ToLower() == ApertureDescription.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 38"); }

else { Console.WriteLine("Expected: " + ApertureDescription.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Description.ToString().ToLower()); }

//39

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Building.ToString().ToLower() == ApertureBuilding.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 39"); }

else { Console.WriteLine("Expected: " + ApertureBuilding.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Building.ToString().ToLower()); }

//40

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Floor.ToString().ToLower() == ApertureFloor.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 40"); }

else { Console.WriteLine("Expected: " + ApertureFloor.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Floor.ToString().ToLower()); }

//41

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Platform.ToString().ToLower() == AperturePlatform.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 41"); }

else { Console.WriteLine("Expected: " + AperturePlatform.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Platform.ToString().ToLower()); }

//42

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Os.ToString().ToLower() == ApertureOs.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 42"); }

else { Console.WriteLine("Expected: " + ApertureOs.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Os.ToString().ToLower()); }

//43

if (DynamicResult[0].ApertureDeviceBasicInfo[0].MachineCategory.ToString().ToLower() == ApertureMachineCategory.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 43"); }

else { Console.WriteLine("Expected: " + ApertureMachineCategory.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].MachineCategory.ToString().ToLower()); }

//44

if (DynamicResult[0].ApertureDeviceBasicInfo[0].MachineType.ToString().ToLower() == ApertureMachineType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 44"); }

else { Console.WriteLine("Expected: " + ApertureMachineType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].MachineType.ToString().ToLower()); }

//45

if (DynamicResult[0].ApertureDeviceBasicInfo[0].StorageType.ToString().ToLower() == ApertureStorageType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 45"); }

else { Console.WriteLine("Expected: " + ApertureStorageType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].StorageType.ToString().ToLower()); }

//46

if (DynamicResult[0].ApertureDeviceBasicInfo[0].BackupType.ToString().ToLower() == ApertureBackupType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 46"); }

else { Console.WriteLine("Expected: " + ApertureBackupType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].BackupType.ToString().ToLower()); }

//47

if (DynamicResult[0].ApertureDeviceBasicInfo[0].SystemType.ToString().ToLower() == ApertureSystemType.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 47"); }

else { Console.WriteLine("Expected: " + ApertureSystemType.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].SystemType.ToString().ToLower()); }

//48

if (DateTime.Parse(DynamicResult[0].ApertureDeviceBasicInfo[0].InstallDate.ToString()) == DateTime.Parse(ApertureInstallDate.ToString()))

{ Result++; Console.WriteLine("Passed test No. 48"); }

else { Console.WriteLine("Expected: " + DateTime.Parse(ApertureInstallDate.ToString()) + " Actual: " + DateTime.Parse(DynamicResult[0].ApertureDeviceBasicInfo[0].InstallDate.ToString())); }

//49

if (DynamicResult[0].ApertureDeviceBasicInfo[0].Country.ToString().ToLower() == ApertureCountry.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 49"); }

else { Console.WriteLine("Expected: " + ApertureCountry.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].Country.ToString().ToLower()); }

//50

if (DynamicResult[0].ApertureDeviceBasicInfo[0].HaClusterNode.ToString().ToLower() == ApertureHaClusterNode.ToString().ToLower())

{ Result++; Console.WriteLine("Passed test No. 50"); }

else { Console.WriteLine("Expected: " + ApertureHaClusterNode.ToString().ToLower() + " Actual: " + DynamicResult[0].ApertureDeviceBasicInfo[0].HaClusterNode.ToString().ToLower()); }

if (MainArchivedDate != "")

{

//51

NumberOfFields++;

Console.WriteLine("MainArchivedDate.ToString().Length: " + MainArchivedDate.ToString().Length + " " + "DynamicResult[0].ArchivedDate.ToString().Length: " + DynamicResult[0].ArchivedDate.ToString().Length);

if (DateTime.Parse(DynamicResult[0].ArchivedDate.ToString().Trim()) == DateTime.Parse(MainArchivedDate.ToString().Trim()))

{ Result++; Console.WriteLine("Passed test No. 51"); }

else { Console.WriteLine("Expected: " + DateTime.Parse(MainArchivedDate.ToString()) + " Actual: " + DateTime.Parse(DynamicResult[0].ArchivedDate.ToString())); }

}

}

else

{

Console.WriteLine("Unexpected Result: " + response.Result.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result + "\r\n");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Assert.AreEqual(NumberOfFields, Result, "Expected number of fields passing verification did not match.");

}

}

public void Node\_JS\_Get\_All\_SharePoint\_Information(string BaseAddress)

{

string Passed = "No";

int Count\_of\_archived = 0;

int Nulls = 0;

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

handler.ReadWriteTimeout = 600000;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

//ServicePointManager.SecurityProtocol = SecurityProtocolType.Ssl3;

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

client.Timeout = new TimeSpan(0, 0, 5, 0, 0); ;

DateTime Start = DateTime.Now;

//using (Stream stm = req.GetRequestStream())

//using (StreamWriter sw = new StreamWriter(stm))

//{

// sw.Write(postData);

//}

var response = client.GetAsync("GetAllSharePointInformation");

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine("DynamicResult: " + DynamicResult);

Console.WriteLine("Number of results: " + DynamicResult.Count);

Console.WriteLine("First result Hostname: " + DynamicResult[0].Hostname);

Console.WriteLine("First result ArchivedDate: " + DynamicResult[0].ArchivedDate);

Console.WriteLine("Last result Hostname: " + DynamicResult[DynamicResult.Count-1].Hostname);

Console.WriteLine("Last result ArchivedDate: " + DynamicResult[DynamicResult.Count - 1].ArchivedDate);

Passed = "Yes";

for (int i = 0; i < DynamicResult.Count; i++)

{

if (DynamicResult[0].ArchivedDate != null)

{ Count\_of\_archived++; }

if (DynamicResult[0].ArchivedDate == null)

{ Nulls++; }

}

Console.WriteLine("Count of Archived: " + Count\_of\_archived);

Console.WriteLine("Count of Nulls: " + Nulls);

}

else

{

Console.WriteLine("Unexpected Result: " + response.Result.StatusCode.ToString() + "\r\n");

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result + "\r\n");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Assert.AreEqual("Yes", Passed, "Failed to get all sharepoint information for some reason.");

}

}

public void Node\_JS\_Get\_All\_NASDetails(string BaseAddress)

{

int verified = 0;

int expected = 2;

int Count\_of\_archived = 0;

int Nulls = 0;

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var response = client.GetAsync("GetAllNasDetails").Result;

Console.WriteLine("StatusCode: " + response.StatusCode.ToString() + "\r\n");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Content.ReadAsStringAsync().Result);

//Console.WriteLine("DynamicResult: " + DynamicResult);

Console.WriteLine("Number of results: " + DynamicResult.Count);

Console.WriteLine("First result Hostname: " + DynamicResult[0].HostName);

if (DynamicResult[0].HostName != "") { verified++; }

Console.WriteLine("Last result Hostname: " + DynamicResult[DynamicResult.Count - 1].HostName);

if (DynamicResult[DynamicResult.Count - 1].HostName != "") { verified++; }

}

else

{

Console.WriteLine("Unexpected Result: " + response.StatusCode.ToString() + "\r\n");

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result + "\r\n");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Assert.AreEqual(expected, verified, "Failed to get all NASDetails information for some reason.");

}

}

public void OST\_Scanner\_Api\_GetShares\_For\_Host(string BaseAddress, string HostName)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 5, 0, 0);

var response = client.GetAsync("shares?hostname=" + HostName);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "shares?hostname=" + HostName);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Shares.ToString() == "[]")

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.Shares.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("ShareName: " + DynamicResult.Shares[i].ShareName);

Console.WriteLine("PhysicalPath: " + DynamicResult.Shares[i].PhysicalPath);

Console.WriteLine("Description: " + DynamicResult.Shares[i].Description);

Console.WriteLine("Drive: " + DynamicResult.Shares[i].Drive);

Console.WriteLine("Partition: " + DynamicResult.Shares[i].Partition);

Console.WriteLine("RootPath: " + DynamicResult.Shares[i].RootPath);

Console.WriteLine("UncPath: " + DynamicResult.Shares[i].UncPath);

Console.WriteLine("CurrentUserCount: " + DynamicResult.Shares[i].CurrentUserCount);

Assert.AreEqual(HostName.ToLower(), DynamicResult.Hostname.ToString().ToLower(), "Hostname does not match.");

//DynamicResult.Shares[i].Partition.ToString().Contains(HostName);

//DynamicResult.Shares[i].UncPath.ToString().Contains(HostName);

Assert.AreEqual("true", DynamicResult.Success.ToString().ToLower(), "Success is not True.");

Passed = "Yes";

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_EmcNASDetails(string BaseAddress, string Hostname)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("emcnasdetails?hostname=" + Hostname);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "emcnasdetails?hostname=" + Hostname);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult);

string Passed = "NotPassed";

string Error = "";

if (DynamicResult.LocalNasPhysicalDevice != null)

{

Console.WriteLine("HostName: " + DynamicResult.HostName);

Console.WriteLine("Success: " + DynamicResult.Success);

Console.WriteLine("PhysicalDevice Domain: " + DynamicResult.LocalNasPhysicalDevice.Domain);

Console.WriteLine("VirtualDeviceName: " + DynamicResult.LocalNasVirtualDevices[0].VirtualDeviceName);

Console.WriteLine("VirtualDevice IPAddress: " + DynamicResult.LocalNasVirtualDevices[0].IpAddress);

Console.WriteLine("VirtualDevice Domain: " + DynamicResult.LocalNasVirtualDevices[0].Domain);

Console.WriteLine("VirtualDevice Status: " + DynamicResult.LocalNasVirtualDevices[0].Status);

Console.WriteLine("VirtualDeviceName: " + DynamicResult.LocalNasVirtualDevices[0].VirtualDeviceName);

Assert.AreEqual(Hostname.ToLower(), DynamicResult.LocalNasPhysicalDevice.PhysicalDeviceName.ToString().ToLower(), "Hostname does not match.");

Assert.AreEqual("true", DynamicResult.Success.ToString().ToLower(), "Success is not True.");

Passed = "Passed";

}

else

{

Passed = "NotPassed";

Console.WriteLine("Requested Hostname details not displayed");

try

{

Console.WriteLine("Message: " + DynamicResult.Errors[0].Message);

Error = DynamicResult.Errors[0].Message;

}

catch { }

}

Assert.AreEqual("Passed", Passed, "Expected result not found. Error (if any): " + Error);

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_NASDetails\_PhysicalFiler(string BaseAddress, string Domain)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "nasphysicaldevices");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("?domain=" + Domain);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?domain=" + Domain);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.PhysicalDevices.ToString() == null)

{

Console.WriteLine("No data returned.");

Console.WriteLine(DynamicResult.ErrorMessage);

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.PhysicalDevices.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("NAS Device Name: " + DynamicResult.PhysicalDevices[i]);

if ((DynamicResult.PhysicalDevices[i].ToString()).Contains("NAS")) ;

{

Passed = "Yes";

}

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_NASDetails\_RegionSpecificQueue(string BaseAddress, string Domain)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("RegionBasedScannerQueue?domain=" + Domain);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "RegionBasedScannerQueue?domain=" + Domain);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult == "QueueOstScanner"+Domain)

{

Passed = "Yes";

}

else

{

Console.WriteLine(DynamicResult.ErrorMessage);

Passed = "No";

Assert.AreEqual("Yes", Passed, "No data returned.");

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("An error has occurred.", DynamicResult.Message.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_NASDetails\_TestHealth(string BaseAddress)

{

WebRequestHandler handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

//ServicePointManager.SecurityProtocol = SecurityProtocolType.Ssl3;

handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

var response = client.GetAsync("testhealth");

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "testhealth");

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult);

string Passed = "No";

String[] Uptime = DynamicResult.upTime.ToString().Split(' ');

double time\_ms = Convert.ToDouble(Uptime[0]);

var ts = TimeSpan.FromMilliseconds(time\_ms);

var parts = string

.Format("{0:D2}d:{1:D2}h:{2:D2}m:{3:D2}s:{4:D3}ms",

ts.Days, ts.Hours, ts.Minutes, ts.Seconds, ts.Milliseconds)

.Split(':')

.SkipWhile(s => Regex.Match(s, @"00\w").Success) // skip zero-valued components

.ToArray();

var result = string.Join(" ", parts); // combine the result

Console.WriteLine(result);

Console.WriteLine("Server Name:" + DynamicResult.server.ToString());

Console.WriteLine("Server Type:" + DynamicResult.type.ToString());

Console.WriteLine("Server is Up for :" + result);

Console.WriteLine("Server's Total Memory:" + DynamicResult.totalMemory.ToString());

Console.WriteLine("Server's Free memory:" + DynamicResult.freeMemory.ToString());

if (time\_ms >= 0)

Passed = "Yes";

Assert.AreEqual("Yes", Passed, "Server is Down");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("An error has occurred.", DynamicResult.Message.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void GetLiveNTFSPermissionsForUNCPath\_Web\_API\_NtfsPermData(string uncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "/GetLiveNTFSPermissionsForUNCPath");

DateTime Start = DateTime.Now;

client.Timeout = new TimeSpan(0, 0, 3, 0, 0);

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

var response = client.GetAsync("?path=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?path=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

//XmlDocument doc = new XmlDocument();

//doc.LoadXml(response.Result.Content.ReadAsStringAsync().Result);

//string json = JsonConvert.SerializeXmlNode(doc);

//dynamic DynamicResult = JsonConvert.DeserializeObject(json);

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Permissions.ToString() == "[]")

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.Permissions.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Account: " + DynamicResult.Permissions[i].Account);

Console.WriteLine("RightText: " + DynamicResult.Permissions[i].RightText);

Console.WriteLine("Domain: " + DynamicResult.Permissions[i].Domain);

Console.WriteLine("FolderPath: " + DynamicResult.Permissions[i].FolderPath);

Console.WriteLine("InheritanceText: " + DynamicResult.Permissions[i].InheritanceText);

Console.WriteLine("PropagationText: " + DynamicResult.Permissions[i].PropagationText);

Console.WriteLine("IsInherited: " + DynamicResult.Permissions[i].IsInherited);

if ((DynamicResult.Permissions[i].InheritanceText.ToString()!= "") && (DynamicResult.Permissions[i].InheritanceText.ToString()!= ""));

{

Passed = "Yes";

}

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void OST\_Scanner\_Api\_GetNTFSPermissions\_For\_UNC(string uncPath, string BaseAddress)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.BaseAddress = new Uri(BaseAddress + "/NtfsPermission");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?path=" + uncPath);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?path=" + uncPath);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

//Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.Permissions.ToString() == "[]")

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.Permissions.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Account: " + DynamicResult.Permissions[i].Account);

Console.WriteLine("RightText: " + DynamicResult.Permissions[i].RightText);

Console.WriteLine("Domain: " + DynamicResult.Permissions[i].Domain);

Console.WriteLine("FolderPath: " + DynamicResult.Permissions[i].FolderPath);

Console.WriteLine("InheritanceText: " + DynamicResult.Permissions[i].InheritanceText);

Console.WriteLine("PropagationText: " + DynamicResult.Permissions[i].PropagationText);

Console.WriteLine("IsInherited: " + DynamicResult.Permissions[i].IsInherited);

if ((DynamicResult.Permissions[i].InheritanceText.ToString() != "") && (DynamicResult.Permissions[i].InheritanceText.ToString() != "")) ;

{

Passed = "Yes";

}

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public void Node\_JS\_Get\_Archived\_ApertureDevices(string BaseAddress, string FromDate, string ToDate)

{

HttpClientHandler handler = new HttpClientHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

client.BaseAddress = new Uri(BaseAddress + "/GetArchivedApertureDevices");

DateTime Start = DateTime.Now;

var response = client.GetAsync("?fromDateTime=" + FromDate + "&toDateTime=" + ToDate);

Console.WriteLine("URL: " + client.BaseAddress.ToString() + "?fromDateTime=" + FromDate + "&toDateTime=" + ToDate);

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

DateTime ArchivedDate, ArchivedFromDate, ArchivedToDate;

bool DateRange;

if (response.Result.StatusCode.ToString() == "OK")

{

dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Test

Console.WriteLine(DynamicResult);

string Passed = "No";

if (DynamicResult.ToString() == "[]")

{

Console.WriteLine("No data returned.");

Assert.AreEqual("Yes", Passed, "No data returned.");

}

else

{

for (int i = 0; i < DynamicResult.Count; i++)

{

Console.WriteLine("Dataset: " + i);

Console.WriteLine("Device Name: " + DynamicResult[i].DeviceName);

Console.WriteLine("Device Type: " + DynamicResult[i].DeviceType);

Console.WriteLine("Installed Date: " + DynamicResult[i].InstallDate);

Console.WriteLine("Cluster: " + DynamicResult[i].Cluster);

Console.WriteLine("UseStatus: " + DynamicResult[i].UseStatus);

Console.WriteLine("FDQN: " + DynamicResult[i].Fqdn);

Console.WriteLine("Archived Date: " + DynamicResult[i].ArchivedDate);

ArchivedDate = DateTime.Parse(DynamicResult[i].ArchivedDate.ToString());

ArchivedFromDate = DateTime.Parse(FromDate);

ArchivedToDate = DateTime.Parse(ToDate);

DateRange = ArchivedDate >= ArchivedFromDate && ArchivedDate < ArchivedToDate;

if ((DynamicResult[i].DeviceName.ToString() != "") && (DateRange));

{

Passed = "Yes";

}

}

}

Assert.AreEqual("Yes", Passed, "Expected result not found.");

Console.WriteLine("");

}

else

{

Console.WriteLine("StatusCode: " + response.Result.StatusCode.ToString());

Console.WriteLine("Entire response message: " + response.Result.Content.ReadAsStringAsync().Result);

//dynamic DynamicResult = JsonConvert.DeserializeObject(response.Result.Content.ReadAsStringAsync().Result);

//Console.WriteLine(DynamicResult.Message);

//Console.WriteLine(DynamicResult.MessageDetail);

Assert.AreEqual("OK", response.Result.StatusCode.ToString(), "Unexpected message/response received.");

}

Console.WriteLine("Web Service response time: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

}

}

public class GetEmployeeDataError

{

public string Message;

public string ExceptionMessage;

public string ExceptionType;

public string StackTrace;

}

public class Node\_JS\_ProcessLog\_JsonParameter

{

public string Application { get; set; }

public string Component { get; set; }

public string TargetAction { get; set; }

public string TargetSpecific { get; set; }

public int LogLevel { get; set; }

public string Message { get; set; }

public DateTime LogTime { get; set; }

}

public void Node\_JS\_Status(string BaseAddress)

{

var handler = new WebRequestHandler();

handler.UseDefaultCredentials = true;

using (var client = new HttpClient(handler))

{

ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) => { return true; };

//handler.ClientCertificates.Add(new X509Certificate2(@"\\vm-364b-4c65\CodedUITesting\OstCert.pkcs12", "Marcin1"));

client.BaseAddress = new Uri(BaseAddress);

DateTime Start = DateTime.Now;

var response = client.GetAsync("Status").Result;

Console.WriteLine("StatusCode: " + response.StatusCode.ToString() + "\r\n");

string Passed = "NotPassed";

if (response.StatusCode.ToString() == "OK")

{

Console.WriteLine("HTTP GET was successful." + "\r\n");

Console.WriteLine("Response: " + response.Content.ReadAsStringAsync().Result);

Console.WriteLine("Node.JS node is up and running, processing 55 PB/s");

Passed = "Passed";

}

else

{

Console.WriteLine("Unexpected Result: " + response.StatusCode.ToString() + "\r\n");

Console.WriteLine("Entire response message: " + response.Content.ReadAsStringAsync().Result + "\r\n");

}

Assert.AreEqual("Passed", Passed, "Unexpected Status response.");

}

}

/// <summary>

/// Verify\_Shares\_Dropdown - Use 'Verify\_Shares\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Shares\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uISharesHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UISharesHyperlink;

HtmlHyperlink uIDfsSharesHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIDfsSharesHyperlink;

HtmlHyperlink uINtfsPermissionsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UINtfsPermissionsHyperlink;

HtmlHyperlink uIShareComplianceHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIShareComplianceHyperlink;

HtmlHyperlink uIOwnersHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIOwnersHyperlink;

HtmlHyperlink uIGlobalComplianceHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIGlobalComplianceHyperlink;

HtmlHyperlink uIGlobalAccountSearchHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIGlobalAccountSearchHyperlink;

#endregion

// Verify that the 'InnerText' property of 'Shares' link equals 'Shares'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UISharesHyperlinkInnerText, uISharesHyperlink.InnerText, true);

Console.WriteLine("Shares Menu Item found.");

// Verify that the 'InnerText' property of 'Dfs Shares' link equals 'Dfs Shares'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UIDfsSharesHyperlinkInnerText, uIDfsSharesHyperlink.InnerText, true);

Console.WriteLine("Dfs Shares Menu Item found.");

// Verify that the 'InnerText' property of 'Ntfs Permissions' link equals 'Ntfs Permissions'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UINtfsPermissionsHyperlinkInnerText, uINtfsPermissionsHyperlink.InnerText, true);

Console.WriteLine("Ntfs Permissions Menu Item found.");

// Verify that the 'InnerText' property of 'Share Compliance' link equals 'Share Compliance'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UIShareComplianceHyperlinkInnerText, uIShareComplianceHyperlink.InnerText, true);

Console.WriteLine("Share Compliance Menu Item found.");

// Verify that the 'InnerText' property of 'Owners' link equals 'Owners'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UIOwnersHyperlinkInnerText, uIOwnersHyperlink.InnerText, true);

Console.WriteLine("Owners Menu Item found.");

// Verify that the 'InnerText' property of 'Global Compliance' link equals 'Global Compliance'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UIGlobalComplianceHyperlinkInnerText, uIGlobalComplianceHyperlink.InnerText, true);

Console.WriteLine("Global Compliance Menu Item found.");

// Verify that the 'InnerText' property of 'Global Account Search' link equals 'Global Account Search'

Assert.AreEqual(this.Verify\_Shares\_DropdownExpectedValues.UIGlobalAccountSearchHyperlinkInnerText, uIGlobalAccountSearchHyperlink.InnerText, true);

Console.WriteLine("Global Account Entitlement Search Menu Item found.");

}

public virtual Verify\_Shares\_DropdownExpectedValues Verify\_Shares\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Shares\_DropdownExpectedValues == null))

{

this.mVerify\_Shares\_DropdownExpectedValues = new Verify\_Shares\_DropdownExpectedValues();

}

return this.mVerify\_Shares\_DropdownExpectedValues;

}

}

private Verify\_Shares\_DropdownExpectedValues mVerify\_Shares\_DropdownExpectedValues;

/// <summary>

/// Verify\_Printers\_Dropdown - Use 'Verify\_Printers\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Printers\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIPrintQueueHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane2.UIPrintQueueHyperlink;

HtmlHyperlink uIPrintQueueHistoryHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane2.UIPrintQueueHistoryHyperlink;

HtmlHyperlink uIPrintQueueStatisticsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane2.UIPrintQueueStatisticsHyperlink;

#endregion

// Verify that the 'InnerText' property of 'Print Queue' link equals 'Print Queue'

Assert.AreEqual(this.Verify\_Printers\_DropdownExpectedValues.UIPrintQueueHyperlinkInnerText, uIPrintQueueHyperlink.InnerText, true);

Console.WriteLine("Print Queue menu item found.");

// Verify that the 'InnerText' property of 'Print Queue History' link equals 'Print Queue History'

Assert.AreEqual(this.Verify\_Printers\_DropdownExpectedValues.UIPrintQueueHistoryHyperlinkInnerText, uIPrintQueueHistoryHyperlink.InnerText, true);

Console.WriteLine("Print Queue History menu item found.");

// Verify that the 'InnerText' property of 'Print Queue Statistics' link equals 'Print Queue Statistics'

Assert.AreEqual(this.Verify\_Printers\_DropdownExpectedValues.UIPrintQueueStatisticsHyperlinkInnerText, uIPrintQueueStatisticsHyperlink.InnerText, true);

Console.WriteLine("Print Queue Statistics menu item found.");

}

public virtual Verify\_Printers\_DropdownExpectedValues Verify\_Printers\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Printers\_DropdownExpectedValues == null))

{

this.mVerify\_Printers\_DropdownExpectedValues = new Verify\_Printers\_DropdownExpectedValues();

}

return this.mVerify\_Printers\_DropdownExpectedValues;

}

}

private Verify\_Printers\_DropdownExpectedValues mVerify\_Printers\_DropdownExpectedValues;

/// <summary>

/// Verify\_Report\_Dropdown - Use 'Verify\_Report\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Report\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIExceptionReportsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIExceptionReportsHyperlink;

HtmlHyperlink uINASGlobalCoverageHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UINASGlobalCoverageHyperlink;

HtmlHyperlink uIGlobalIDriveHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIGlobalIDriveHyperlink;

HtmlHyperlink uIReportRequestsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIReportRequestsHyperlink;

HtmlHyperlink uIDLPDataProtectionReportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIDLPDataProtectionReportHyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Exception Reports' link equals 'Exception Reports'

Assert.AreEqual(this.Verify\_Report\_DropdownExpectedValues.UIExceptionReportsHyperlinkInnerText, uIExceptionReportsHyperlink.InnerText, true);

Console.WriteLine("Exception Reports menu item found.");

if (Run\_Environment == "OST\_DEV")

{

// Verify that the 'InnerText' property of 'NAS Global Coverage' link equals 'NAS Global Coverage'

Assert.AreEqual(this.Verify\_Report\_DropdownExpectedValues.UINASGlobalCoverageHyperlinkInnerText, uINASGlobalCoverageHyperlink.InnerText, true);

Console.WriteLine("NAS Global Coverage menu item found.");

}

// Verify that the 'InnerText' property of 'Global IDrive' link equals 'Global IDrive'

Assert.AreEqual(this.Verify\_Report\_DropdownExpectedValues.UIGlobalIDriveHyperlinkInnerText, uIGlobalIDriveHyperlink.InnerText, true);

Console.WriteLine("Global IDrive menu item found.");

// Verify that the 'InnerText' property of 'Report Requests' link equals 'Report Requests'

Assert.AreEqual(this.Verify\_Report\_DropdownExpectedValues.UIReportRequestsHyperlinkInnerText, uIReportRequestsHyperlink.InnerText, true);

Console.WriteLine("Report Requests menu item found.");

Assert.AreEqual("DLP/Data Protection Report".ToLower(), uIDLPDataProtectionReportHyperlink.InnerText.ToLower(), true);

Console.WriteLine("DLP/Data Protection Report menu item found.");

}

public virtual Verify\_Report\_DropdownExpectedValues Verify\_Report\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Report\_DropdownExpectedValues == null))

{

this.mVerify\_Report\_DropdownExpectedValues = new Verify\_Report\_DropdownExpectedValues();

}

return this.mVerify\_Report\_DropdownExpectedValues;

}

}

private Verify\_Report\_DropdownExpectedValues mVerify\_Report\_DropdownExpectedValues;

/// <summary>

/// Verify\_Support\_Dropdown - Use 'Verify\_Support\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Support\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIMonitorHealthHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIMonitorHealthHyperlink;

HtmlHyperlink uIManageMonitorAccountHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIManageMonitorAccountHyperlink;

HtmlHyperlink UIUserRoleslink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIUserRoleslink;

HtmlHyperlink UISystemInformationlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UISystemInformationlink;

HtmlHyperlink uIHostMaintanceHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIHostMaintanceHyperlink;

HtmlHyperlink uIOwnerInfoHyperlink = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UIOwnerInfoHyperlink;

HtmlHyperlink uIHostImportHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIMainnavPane1.UIHostImportHyperlink;

HtmlHyperlink UIScannerHealthlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIScannerHealthlink;

HtmlHyperlink UIMongoDBHealthlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIMongoDBHealthlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Monitor Health' link equals 'Monitor Health'

Assert.AreEqual(this.Verify\_Support\_DropdownExpectedValues.UIMonitorHealthHyperlinkInnerText, uIMonitorHealthHyperlink.InnerText, true);

Console.WriteLine("Monitor Health menu item found.");

// Verify that the 'InnerText' property of 'Manage Monitor Account' link equals 'Manage Monitor Account'

Assert.AreEqual(this.Verify\_Support\_DropdownExpectedValues.UIManageMonitorAccountHyperlinkInnerText, uIManageMonitorAccountHyperlink.InnerText, true);

Console.WriteLine("Manage Monitor Account menu item found.");

Assert.AreEqual("User Roles And Teams", UIUserRoleslink.InnerText, true);

Console.WriteLine("User Roles menu item found.");

//Assert.AreEqual("System Information", UISystemInformationlink.InnerText, true);

//Console.WriteLine("System Information menu item found.");

Assert.AreEqual("Host Maintenance", uIHostMaintanceHyperlink.InnerText, true);

Console.WriteLine("Host Maintenance menu item found.");

//Assert.AreEqual("Host Import/Update", uIHostImportHyperlink.InnerText, true);

//Console.WriteLine("Host Import menu item found.");

Assert.AreEqual("Owners for UNC Path", uIOwnerInfoHyperlink.InnerText.Trim(), true);

Console.WriteLine("Owner Info menu item found.");

Assert.AreEqual("Health Page", UIScannerHealthlink.InnerText.Trim(), true);

Console.WriteLine("Health Page menu item found");

//Assert.AreEqual("MongoDB Health", UIMongoDBHealthlink.InnerText.Trim(), true);

//Console.WriteLine("MongoDB Helath menu item found");

}

public virtual Verify\_Support\_DropdownExpectedValues Verify\_Support\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Support\_DropdownExpectedValues == null))

{

this.mVerify\_Support\_DropdownExpectedValues = new Verify\_Support\_DropdownExpectedValues();

}

return this.mVerify\_Support\_DropdownExpectedValues;

}

}

private Verify\_Support\_DropdownExpectedValues mVerify\_Support\_DropdownExpectedValues;

/// <summary>

/// Verify\_Chargeback\_Dropdown - Use 'Verify\_Chargeback\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Chargeback\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIChargebackReportsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane1.UIChargebackReportsHyperlink;

HtmlHyperlink uIChargebackExceptionsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane1.UIChargebackExceptionsHyperlink;

HtmlHyperlink uIChargebackCostCodeHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane1.UIChargebackCostCodeHyperlink;

#endregion

// Verify that the 'InnerText' property of 'Chargeback Reports' link equals 'Chargeback Reports'

Assert.AreEqual(this.Verify\_Chargeback\_DropdownExpectedValues.UIChargebackReportsHyperlinkInnerText, uIChargebackReportsHyperlink.InnerText, true);

Console.WriteLine("Chargeback Reports menu item found.");

// Verify that the 'InnerText' property of 'Chargeback Exceptions' link equals 'Chargeback Exceptions'

Assert.AreEqual(this.Verify\_Chargeback\_DropdownExpectedValues.UIChargebackExceptionsHyperlinkInnerText, uIChargebackExceptionsHyperlink.InnerText, true);

Console.WriteLine("Chargeback Exceptions menu item found.");

// Verify that the 'InnerText' property of 'Chargeback Cost Code' link equals 'Chargeback Cost Code'

Assert.AreEqual(this.Verify\_Chargeback\_DropdownExpectedValues.UIChargebackCostCodeHyperlinkInnerText, uIChargebackCostCodeHyperlink.InnerText, true);

Console.WriteLine("Chargeback Cost Code menu item found.");

}

public virtual Verify\_Chargeback\_DropdownExpectedValues Verify\_Chargeback\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Chargeback\_DropdownExpectedValues == null))

{

this.mVerify\_Chargeback\_DropdownExpectedValues = new Verify\_Chargeback\_DropdownExpectedValues();

}

return this.mVerify\_Chargeback\_DropdownExpectedValues;

}

}

private Verify\_Chargeback\_DropdownExpectedValues mVerify\_Chargeback\_DropdownExpectedValues;

/// <summary>

/// Search\_Print\_Queue\_History

/// </summary>

public void Search\_Print\_Queue\_History(String Host)

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UIItemEdit;

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UISearchButton;

HtmlCustom bfscrpsp002 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.bfscrpsp002;

HtmlCustom fftctisvps002ua = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.fftctisvps002ua;

HtmlCustom namthoaksapps01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.namthoaksapps01;

#endregion

uIHostNameEdit.WaitForControlExist(5000);

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

uIAvailableRegionsDdlComboBox.SelectedItem = "EMEA";

}

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIHostNameEdit.Text = strhost;

if (Run\_Environment == "OST\_DEV")

{

bfscrpsp002.WaitForControlExist(5000);

Mouse.Click(bfscrpsp002);

}

else if (Run\_Environment == "OST\_UAT")

{

fftctisvps002ua.WaitForControlExist(5000);

Mouse.Click(fftctisvps002ua);

}

else if (Run\_Environment == "OST\_PROD")

{

namthoaksapps01.WaitForControlExist(5000);

Mouse.Click(namthoaksapps01);

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Print\_Queue\_History

/// </summary>

public void Search\_Print\_Queue\_History\_by\_Queue(String Queue)

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UIItemEdit;

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UISearchButton;

HtmlCustom bfscrpsp002 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.bfscrpsp002;

HtmlCustom fftctisvps002ua = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.fftctisvps002ua;

HtmlCustom namthoaksapps01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.namthoaksapps01;

#endregion

uIItemEdit.WaitForControlExist(5000);

uIItemEdit.Text = Queue;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Print\_Queue\_History

/// </summary>

public void Search\_All\_Print\_Queue\_History()

{

#region Variable Declarations

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UISearchButton;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIHostNameEdit;

#endregion

uIHostNameEdit.WaitForControlExist(5000);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Print\_Queue

/// </summary>

public void Search\_Print\_Queue(String Host)

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIHostNameEdit;

HtmlEdit uIQueueInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIQueueInputEdit;

HtmlEdit uILocationInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UILocationInputEdit;

HtmlEdit uIIpAddressInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIIpAddressInputEdit;

HtmlComboBox uIDriverTypeSelectComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDriverTypeSelectComboBox;

HtmlButton uISearchButton = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchButton;

HtmlCustom ldncfisp001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.ldncfisp001;

HtmlCustom milctisvsp007 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.milctisvsp007;

HtmlCustom apaccitisgap381 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apaccitisgap381;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIHostNameEdit.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

ldncfisp001.WaitForControlExist(5000);

Mouse.Click(ldncfisp001);

}

else if (Run\_Environment == "OST\_UAT")

{

apaccitisgap381.WaitForControlExist(5000);

Mouse.Click(apaccitisgap381);

}

else if (Run\_Environment == "OST\_PROD")

{

milctisvsp007.WaitForControlExist(10000);

Mouse.Click(milctisvsp007);

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Print\_Queue

/// </summary>

public void Search\_Print\_Queue\_All()

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIHostNameEdit;

HtmlEdit uIQueueInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIQueueInputEdit;

HtmlEdit uILocationInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UILocationInputEdit;

HtmlEdit uIIpAddressInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIIpAddressInputEdit;

HtmlComboBox uIDriverTypeSelectComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDriverTypeSelectComboBox;

HtmlButton uISearchButton = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchButton;

#endregion

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Print\_Queue\_Statistics

/// </summary>

public void Search\_Print\_Queue\_Statistics(String Queue)

{

#region Variable Declarations

HtmlControl uIPrintQueueStatisticsPane = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIPrintQueueStatisticsPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIHostNameEdit;

HtmlEdit uIQueueInputEdit = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIQueueInputEdit;

HtmlButton uISearchButton = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UISearchFormCustom.UISearchButton;

#endregion

uIQueueInputEdit.WaitForControlExist(5000);

uIQueueInputEdit.Text = Queue;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_All\_Print\_Queue\_Statistics

/// </summary>

public void Search\_All\_Print\_Queue\_Statistics(String Queue)

{

#region Variable Declarations

HtmlButton uISearchButton = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UISearchFormCustom.UISearchButton;

#endregion

Mouse.Click(uISearchButton);

}

/// <summary>

/// Verify\_Compliance\_Dropdown - Use 'Verify\_Compliance\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Compliance\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIAntiVirusHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UIAntiVirusHyperlink;

HtmlHyperlink uISecurityChecksHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISecurityChecksHyperlink;

HtmlHyperlink uILocalUsersHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UILocalUsersHyperlink;

HtmlHyperlink uILocalGroupsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UILocalGroupsHyperlink;

HtmlHyperlink uIUserRightsHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UIUserRightsHyperlink;

HtmlHyperlink uIOrphansHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UIOrphansHyperlink;

HtmlHyperlink uIServerScheduledTasksHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UIServerScheduledTasksHyperlink;

HtmlHyperlink uIShareExclusionHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UIShareExclusionHyperlink;

HtmlHyperlink uiSharepointInfoHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISharepointInfoHyperlink;

HtmlHyperlink UINasPhysicalDeviceHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UINasPhysicalDeviceHyperlink;

#endregion

String Run\_Environment = Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Anti Virus' link equals 'Anti Virus'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UIAntiVirusHyperlinkInnerText, uIAntiVirusHyperlink.InnerText, true);

Console.WriteLine("Anti Virus menu item found.");

// Verify that the 'InnerText' property of 'Security Checks' link equals 'Security Checks'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UISecurityChecksHyperlinkInnerText, uISecurityChecksHyperlink.InnerText, true);

Console.WriteLine("Security Checks menu item found.");

// Verify that the 'InnerText' property of 'Local Users' link equals 'Local Users'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UILocalUsersHyperlinkInnerText, uILocalUsersHyperlink.InnerText, true);

Console.WriteLine("Local Users menu item found.");

// Verify that the 'InnerText' property of 'Local Groups' link equals 'Local Groups'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UILocalGroupsHyperlinkInnerText, uILocalGroupsHyperlink.InnerText, true);

Console.WriteLine("Local Groups menu item found.");

// Verify that the 'InnerText' property of 'User Rights' link equals 'User Rights'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UIUserRightsHyperlinkInnerText, uIUserRightsHyperlink.InnerText, true);

Console.WriteLine("User Rights menu item found.");

// Verify that the 'InnerText' property of 'Orphans' link equals 'Orphans'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UIOrphansHyperlinkInnerText, uIOrphansHyperlink.InnerText, true);

Console.WriteLine("Orphans menu item found.");

// Verify that the 'InnerText' property of 'Server Scheduled Tasks' link equals 'Server Scheduled Tasks'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UIServerScheduledTasksHyperlinkInnerText, uIServerScheduledTasksHyperlink.InnerText, true);

Console.WriteLine("Server Scheduled Tasks menu item found.");

// Verify that the 'InnerText' property of 'Share Exclusion' link equals 'Share Exclusion'

Assert.AreEqual(this.Verify\_Compliance\_DropdownExpectedValues.UIShareExclusionHyperlinkInnerText, uIShareExclusionHyperlink.InnerText, true);

Console.WriteLine("Share Exclusion menu item found.");

// Verify that the 'InnerText' property of 'Sharepoint Info' link equals 'Sharepoint Info'

Assert.AreEqual("Sharepoint Info", uiSharepointInfoHyperlink.InnerText, true);

Console.WriteLine("Sharepoint Info menu item found.");

// Verify that the 'InnerText' property of 'NAS Physical Device Information' link equals 'Sharepoint Info'

Assert.AreEqual("Nas Physical Device", UINasPhysicalDeviceHyperlink.InnerText, true);

Console.WriteLine("Nas Physical Device Info menu item found.");

}

public virtual Verify\_Compliance\_DropdownExpectedValues Verify\_Compliance\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_Compliance\_DropdownExpectedValues == null))

{

this.mVerify\_Compliance\_DropdownExpectedValues = new Verify\_Compliance\_DropdownExpectedValues();

}

return this.mVerify\_Compliance\_DropdownExpectedValues;

}

}

private Verify\_Compliance\_DropdownExpectedValues mVerify\_Compliance\_DropdownExpectedValues;

/// <summary>

/// Verify\_DataStats\_Dropdown - Use 'Verify\_DataStats\_DropdownExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_DataStats\_Dropdown()

{

#region Variable Declarations

HtmlHyperlink uIOverviewHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane3.UIOverviewHyperlink;

HtmlHyperlink uIFindHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane3.UIFindHyperlink;

HtmlHyperlink uIFileUsageDLPHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane3.UIFileUsageDLPHyperlink;

HtmlHyperlink uIPSTReportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane3.UIPSTReportHyperlink;

#endregion

// Verify that the 'InnerText' property of 'Overview' link equals 'Overview'

Assert.AreEqual(this.Verify\_DataStats\_DropdownExpectedValues.UIOverviewHyperlinkInnerText, uIOverviewHyperlink.InnerText, true);

Console.WriteLine("Overview menu item found.");

// Verify that the 'InnerText' property of 'Find' link equals 'Find'

Assert.AreEqual(this.Verify\_DataStats\_DropdownExpectedValues.UIFindHyperlinkInnerText, uIFindHyperlink.InnerText, true);

Console.WriteLine("Find menu item found.");

// Verify that the 'InnerText' property of 'File Usage (DLP)' link equals 'File Usage (DLP)'

Assert.AreEqual(this.Verify\_DataStats\_DropdownExpectedValues.UIFileUsageDLPHyperlinkInnerText, uIFileUsageDLPHyperlink.InnerText, true);

Console.WriteLine("File Usage (DLP) menu item found.");

// Verify that the 'InnerText' property of 'PST Report' link equals 'PST Report'

Assert.AreEqual(this.Verify\_DataStats\_DropdownExpectedValues.UIPSTReportHyperlinkInnerText, uIPSTReportHyperlink.InnerText, true);

Console.WriteLine("PST Report menu item found.");

}

public virtual Verify\_DataStats\_DropdownExpectedValues Verify\_DataStats\_DropdownExpectedValues

{

get

{

if ((this.mVerify\_DataStats\_DropdownExpectedValues == null))

{

this.mVerify\_DataStats\_DropdownExpectedValues = new Verify\_DataStats\_DropdownExpectedValues();

}

return this.mVerify\_DataStats\_DropdownExpectedValues;

}

}

private Verify\_DataStats\_DropdownExpectedValues mVerify\_DataStats\_DropdownExpectedValues;

/// <summary>

/// Verify\_Elements\_Share\_Compliance - Use 'Verify\_Elements\_Share\_ComplianceExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Share\_Compliance()

{

#region Variable Declarations

HtmlDiv uIShareCompliancePane = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIShareCompliancePane;

HtmlLabel uIRegionLabel = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIFilterLabel = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UISearchFormCustom.UIFilterLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIHostNameEdit;

HtmlComboBox uIItemComboBox = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UISearchFormCustom.UIItemComboBox;

HtmlButton uISearchButton = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Share Compliance' pane starts with 'Share Compliance'

StringAssert.StartsWith(uIShareCompliancePane.InnerText, this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIShareCompliancePaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays metrics for teams’ share compliance in percentages.");

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UITeamLabelInnerText, uITeamLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Filter:' label starts with 'Filter:'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIFilterLabelInnerText, uIFilterLabel.InnerText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'InnerText' property of combo box equals 'AllCompliantNon-Compliant'

StringAssert.Contains(uIItemComboBox.InnerText, this.Verify\_Elements\_Share\_ComplianceExpectedValues.UIItemComboBoxInnerText);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Share\_ComplianceExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public virtual Verify\_Elements\_Share\_ComplianceExpectedValues Verify\_Elements\_Share\_ComplianceExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Share\_ComplianceExpectedValues == null))

{

this.mVerify\_Elements\_Share\_ComplianceExpectedValues = new Verify\_Elements\_Share\_ComplianceExpectedValues();

}

return this.mVerify\_Elements\_Share\_ComplianceExpectedValues;

}

}

private Verify\_Elements\_Share\_ComplianceExpectedValues mVerify\_Elements\_Share\_ComplianceExpectedValues;

/// <summary>

/// Verify\_Elements\_Owners - Use 'Verify\_Elements\_OwnersExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Owners()

{

#region Variable Declarations

HtmlDiv uIOwnersPane = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIOwnersPane;

HtmlLabel uIRegionLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIShareNameLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIShareNameLabel;

HtmlLabel uIOwnerNameLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIOwnerNameLabel;

HtmlLabel uISOEIdLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UISOEIdLabel;

HtmlLabel uIFolderPathLabel = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIFolderPathLabel;

HtmlCell uIThisreportscomplexitCell = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemTable.UIThisreportscomplexitCell;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit;

HtmlEdit uIItemEdit1 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit1;

HtmlEdit uIItemEdit2 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit2;

HtmlEdit uIItemEdit3 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit3;

HtmlButton uISearchButton = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Owners' pane equals 'Owners'

StringAssert.StartsWith(uIOwnersPane.InnerText, this.Verify\_Elements\_OwnersExpectedValues.UIOwnersPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays primary and secondary owner information for shares. Owners are updated in DNA and fed into OST. You can search by host, share name, owner name, SOE ID or folder path.");

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UITeamLabelInnerText, uITeamLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Share Name:' label starts with 'Share Name:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIShareNameLabelInnerText, uIShareNameLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Owner Name:' label starts with 'Owner Name:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIOwnerNameLabelInnerText, uIOwnerNameLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'SOE Id:' label starts with 'SOE Id:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UISOEIdLabelInnerText, uISOEIdLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Folder Path:' label starts with 'Folder Path:'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIFolderPathLabelInnerText, uIFolderPathLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'This report's complexity makes it slow t' cell equals 'This report's complexity makes it slow to retrieve. Please be patient after submitting a query.'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIThisreportscomplexitCellInnerText, uIThisreportscomplexitCell.InnerText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIItemEditClass, uIItemEdit.Class);

// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIItemEdit1TagName, uIItemEdit1.TagName);

// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIItemEdit2TagName, uIItemEdit2.TagName);

// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UIItemEdit3TagName, uIItemEdit3.TagName);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_OwnersExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public virtual Verify\_Elements\_OwnersExpectedValues Verify\_Elements\_OwnersExpectedValues

{

get

{

if ((this.mVerify\_Elements\_OwnersExpectedValues == null))

{

this.mVerify\_Elements\_OwnersExpectedValues = new Verify\_Elements\_OwnersExpectedValues();

}

return this.mVerify\_Elements\_OwnersExpectedValues;

}

}

private Verify\_Elements\_OwnersExpectedValues mVerify\_Elements\_OwnersExpectedValues;

/// <summary>

/// Verify\_Elements\_Shares - Use 'Verify\_Elements\_SharesExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Shares()

{

#region Variable Declarations

HtmlDiv uIShareInformationPane = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIShareInformationPane;

HtmlLabel uIRegionLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uIPartitionLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIPartitionLabel;

HtmlLabel uIShareNameLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIShareNameLabel;

HtmlLabel uITeamLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIShareTypeLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIShareTypeLabel;

HtmlLabel uIIncludeArchivedLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIIncludeArchivedLabel;

HtmlLabel uIHostLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIAvailableRegionsDdlComboBox;

HtmlEdit uIItemEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIItemEdit;

HtmlEdit uIShareNameTxtEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIShareNameTxtEdit;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIAvailableTeamsDdlComboBox;

HtmlComboBox uIItemComboBox = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIItemComboBox;

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIHostNameEdit;

HtmlCheckBox uIItemCheckBox = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIItemCheckBox;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

#endregion

// Verify that the 'InnerText' property of 'Share Information' pane equals 'Share Information'

StringAssert.StartsWith(uIShareInformationPane.InnerText, this.Verify\_Elements\_SharesExpectedValues.UIShareInformationPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays a list of shares for each host along with some share details. You can search by host, partition, share type (home/group) and share name. You can include archived information as well if you wish.");

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Partition:' label equals 'Partition:'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIPartitionLabelInnerText, uIPartitionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Share Name:' label equals 'Share Name:'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIShareNameLabelInnerText, uIShareNameLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label equals 'Team:'

Assert.AreEqual(uITeamLabel.InnerText.Trim(), this.Verify\_Elements\_SharesExpectedValues.UITeamLabelInnerText);

// Verify that the 'InnerText' property of 'Share Type:' label equals 'Share Type:'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIShareTypeLabelInnerText, uIShareTypeLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Include Archived:' label equals 'Include Archived: '

Assert.AreEqual(uIIncludeArchivedLabel.InnerText.Trim(), this.Verify\_Elements\_SharesExpectedValues.UIIncludeArchivedLabelInnerText);

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIItemEditClass, uIItemEdit.Class);

// Verify that the 'TagName' property of text box equals 'INPUT'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIItemEditTagName, uIItemEdit.TagName);

// Verify that the 'Id' property of 'shareNameTxt' text box equals 'shareNameTxt'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIShareNameTxtEditId, uIShareNameTxtEdit.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'TagName' property of combo box equals 'SELECT'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIItemComboBoxTagName, uIItemComboBox.TagName);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'TagName' property of check box equals 'INPUT'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIItemCheckBoxTagName, uIItemCheckBox.TagName);

// Verify that the 'ControlType' property of check box equals 'CheckBox'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UIItemCheckBoxControlType, uIItemCheckBox.ControlType.ToString());

// Verify that the 'ControlType' property of 'Search' button equals 'Button'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UISearchButtonControlType, uISearchButton.ControlType.ToString());

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_SharesExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public void FindColumn()

{

FindColumnNumber("Hostname");

}

public void Encrypt(String Text)

{

Console.WriteLine("Encrypted: " + Playback.EncryptText(Text));

}

public virtual Verify\_Elements\_SharesExpectedValues Verify\_Elements\_SharesExpectedValues

{

get

{

if ((this.mVerify\_Elements\_SharesExpectedValues == null))

{

this.mVerify\_Elements\_SharesExpectedValues = new Verify\_Elements\_SharesExpectedValues();

}

return this.mVerify\_Elements\_SharesExpectedValues;

}

}

private Verify\_Elements\_SharesExpectedValues mVerify\_Elements\_SharesExpectedValues;

/// <summary>

/// Perform\_search\_on\_NTFS\_page

/// </summary>

public void Search\_Ntfs\_Permissions(String Host, String Share, String Account, String InitialScanDate, String FolderPath)

{

#region Variable Declarations

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIHostNameEdit;

HtmlEdit uIShareNameTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIShareNameTxtEdit;

HtmlEdit uIAccountTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIAccountTxtEdit;

HtmlEdit uIPathTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIPathTxtEdit;

HtmlEdit uIScanDateDivDtPickEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIScanDateDivDtPickEdit;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

HtmlCustom naschi06v4fut = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.naschi06v4fut;

HtmlCustom gtdvnasgcg0001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.gtdvnasgcg0001;

HtmlCustom data\_efcs = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.data\_efcs;

HtmlButton uISearchButton1 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UISearchButton1;

HtmlCustom AB10978 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.AB10978;

HtmlCustom dn01598 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.dn01598;

HtmlDiv Column\_Initial\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Initial\_Scan\_Date;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

Console.WriteLine("Entering " + Host + " into host field.");

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIHostNameEdit.Text = strhost;

if (Run\_Environment == "OST\_DEV")

{

amsgtssf01.WaitForControlExist(5000);

Mouse.Click(amsgtssf01);

}

else if ((Run\_Environment == "OST\_UAT") || (Run\_Environment == "OST\_PROD"))

{

gtdvnasgcg0001.WaitForControlExist(5000);

Mouse.Click(gtdvnasgcg0001);

}

//else if (Run\_Environment == "OST\_PROD")

//{

// naschi06v4fut.WaitForControlExist(5000);

// Mouse.Click(naschi06v4fut);

//}

Console.WriteLine("Entering " + Share + " into share field.");

//string strshare = Share;

//strshare = strshare.Remove(strshare.Length - 1, 1);

uIShareNameTxtEdit.Text = Share;

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT"))

{

//Mouse.Click(data\_efcs);

uIPathTxtEdit.Text = FolderPath;

uIScanDateDivDtPickEdit.Text = InitialScanDate;

}

else if (Run\_Environment == "OST\_PROD")

{

//Mouse.Click(dn01598);

uIPathTxtEdit.Text = FolderPath;

uIScanDateDivDtPickEdit.Text = InitialScanDate;

}

//else if (Run\_Environment == "OST\_UAT")

//{

// Mouse.Click(AB10978);

//}

Console.WriteLine("Entering " + Account + " into account field.");

uIAccountTxtEdit.Text = Account;

//Excel export is sorted by Initial Scan Date column, so sort by the column after searching if more than one record in results.

Mouse.Click(Column\_Initial\_Scan\_Date);

// Click 'Search' button

Mouse.Click(uISearchButton1);

}

/// <summary>

/// Perform\_search\_on\_Shares\_page - Use 'Perform\_search\_on\_Shares\_pageParams' to pass parameters into this method.

/// </summary>

public void Perform\_search\_on\_Shares\_page(string Host, string Partition)

{

#region Variable Declarations

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIHostNameEdit;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

HtmlEdit uIItemEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIItemEdit;

HtmlCustom uIUiid122Custom = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIUiid122Custom;

HtmlButton uISearchButton1 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UISearchButton1;

HtmlCustom naschi05v2ccc = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.naschi05v2ccc;

HtmlCustom nastam02ctam1 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nastam02ctam1;

HtmlCustom nastpa11 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nastpa11;

HtmlCustom chi500soe\_3y = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.chi500soe\_3y;

HtmlCustom data\_v5 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.data\_v5;

#endregion

// Type 'nasspr904rv1cr' in 'hostName' text box

//uIHostNameEdit.Text = this.Perform\_search\_on\_Shares\_pageParams.UIHostNameEditText;

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIHostNameEdit.Text = strhost;

// Click 'HostAutocomplete' custom control

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

amsgtssf01.WaitForControlExist(5000);

Mouse.Click(amsgtssf01);

}

else if (Run\_Environment == "OST\_UAT")

{

naschi05v2ccc.WaitForControlExist(5000);

Mouse.Click(naschi05v2ccc);

}

else if (Run\_Environment == "OST\_PROD")

{

nastpa11.WaitForControlExist();

Mouse.Click(nastpa11);

}

// Type 'dat' in text box

uIItemEdit.Text = Partition;

// Click 'PartitionAutocomplete' custom control

if (Run\_Environment == "OST\_DEV")

{

//Mouse.Click(uIUiid122Custom);

}

else if (Run\_Environment == "OST\_UAT")

{

chi500soe\_3y.WaitForControlExist();

Mouse.Click(chi500soe\_3y);

}

else if (Run\_Environment == "OST\_PROD")

{

data\_v5.WaitForControlExist();

Mouse.Click(data\_v5);

}

// Click 'Search' button

Mouse.Click(uISearchButton1);

}

public void Perform\_search\_Muliple\_times\_on\_Shares\_page()

{

#region Variable Declarations

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIHostNameEdit;

HtmlCustom uIUiid120Custom = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIUiid120Custom;

HtmlEdit uIItemEdit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UIItemEdit;

HtmlCustom uIUiid122Custom = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIUiid122Custom;

HtmlButton uISearchButton1 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchFormCustom.UISearchButton1;

HtmlCustom naschi05v2ccc = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.naschi05v2ccc;

HtmlCustom nastam02ctam1 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nastam02ctam1;

HtmlCustom nastpa11 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nastpa11;

HtmlCustom chi500soe\_3y = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.chi500soe\_3y;

HtmlCustom data\_v5 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.data\_v5;

HtmlDiv UITableRowcount = this.UIShareInformationOSTOWindow2.UIShareInformationOSTODocument.UIView120of44Cell.UITableRowcount;

#endregion

Mouse.Click(uISearchButton1);

Verify\_search\_finished();

//if (Compare\_Result != 0)

// Console.WriteLine("Duplicate data has resulted on clicking search multiple times");

//else

// Console.WriteLine("Duplicate data has not resulted on clicking search multiple times");

}

/// <summary>

/// Get\_Record\_Number

/// </summary>

public void Get\_Record\_Number()

{

#region Variable Declarations

HtmlDiv UITableRowcount = this.UIShareInformationOSTOWindow2.UIShareInformationOSTODocument.UIView120of44Cell.UITableRowcount;

#endregion

// Click 'View 1 - 20 of 44' pane

}

/// <summary>

/// Perform\_search\_on\_Shares\_page - Use 'Perform\_search\_on\_Shares\_pageParams' to pass parameters into this method.

/// </summary>

public void Perform\_search\_on\_Share\_Compliance(string Host)

{

#region Variable Declarations

HtmlEdit uIHostNameEdit = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UIHostNameEdit;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

HtmlComboBox uIItemComboBox = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UISearchFormCustom.UIItemComboBox;

HtmlButton uISearchButton = this.UINTFSPermissionsOSTOpWindow.UIShareComplianceOSTOpDocument.UISearchFormCustom.UISearchButton;

HtmlCustom apausmondbd02 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apausmondbd02;

HtmlCustom apaccitikrap077 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apaccitikrap077;

#endregion

// Type 'nasspr904rv1cr' in 'hostName' text box

//uIHostNameEdit.Text = this.Perform\_search\_on\_Shares\_pageParams.UIHostNameEditText;

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIHostNameEdit.Text = strhost;

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'HostAutocomplete' custom control

if (Run\_Environment == "OST\_DEV")

{

amsgtssf01.WaitForControlExist(5000);

Mouse.Click(amsgtssf01);

}

else if (Run\_Environment == "OST\_UAT")

{

apausmondbd02.WaitForControlExist(5000);

Mouse.Click(apausmondbd02);

}

else if (Run\_Environment == "OST\_PROD")

{

apaccitikrap077.WaitForControlExist(5000);

Mouse.Click(apaccitikrap077);

}

// Click 'Search' button

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Owners

/// </summary>

public void Perform\_Search\_Owners(String FolderPath, String SoeID)

{

#region Variable Declarations

HtmlDiv uIOwnersPane = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIOwnersPane;

HtmlCell uIThisreportscomplexitCell = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemTable.UIThisreportscomplexitCell;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit;

HtmlEdit uIItemEdit1 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit1;

HtmlEdit uIItemEdit2 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit2;

HtmlEdit uIItemEdit3 = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UIItemEdit3;

HtmlButton uISearchButton = this.UINTFSPermissionsOSTOpWindow.UIOwnersOSTOperationsSDocument.UISearchFormCustom.UISearchButton;

#endregion

//uIAvailableRegionsDdlComboBox.SelectedItem = "North America";

//uIAvailableTeamsDdlComboBox.SelectedItem = "Corporate NA Client Support";

uIHostNameEdit.Text = "cpna111ap001";

uIItemEdit2.Text = SoeID;

uIItemEdit3.Text = FolderPath;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Global\_Account

/// </summary>

public void Perform\_Search\_Global\_Account(String Account)

{

#region Variable Declarations

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

#endregion

uIAccountTxtEdit.Text = Account;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Global\_Account

/// </summary>

public void Perform\_Search\_Global\_Account\_using\_suggestion\_dropdowns(String Domain, String User)

{

#region Variable Declarations

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlEdit uIDomainTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIDomainTxtEdit;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

HtmlCustom EUR = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.EUR;

HtmlCustom mw13910 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.mw13910;

#endregion

uIDomainTxtEdit.Text = Domain;

EUR.WaitForControlExist(60000);

Mouse.Click(EUR);

DateTime Start = DateTime.Now;

uIAccountTxtEdit.Text = User;

mw13910.WaitForControlExist(60000);

Assert.IsTrue(mw13910.Exists == true, "Suggestion dropdown did not appear.");

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Dropdown time to appear: " + DifferenceDateTime.Duration().TotalSeconds + " seconds." + "\n");

Mouse.Click(mw13910);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Global\_Account

/// </summary>

public void Perform\_Search\_Global\_Account\_by\_Domain\_Account(String Domain, String Account, bool IncludeGroups)

{

#region Variable Declarations

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlEdit uIDomainTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIDomainTxtEdit;

HtmlCheckBox UIIncludeGroupsCheckBox = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIIncludeGroupsCheckBox;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

#endregion

uIDomainTxtEdit.Text = Domain;

uIAccountTxtEdit.Text = Account;

if (IncludeGroups == true)

{ UIIncludeGroupsCheckBox.Checked = true; }

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Owners

/// </summary>

public void Perform\_Search\_Global\_Compliance(String Region)

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableTeamsDdlComboBox;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UISearchFormCustom.UISearchButton;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIAvailableRegionsDdlComboBox.WaitForControlReady(5000);

//uIAvailableRegionsDdlComboBox.SelectedItem = Region;

if (Run\_Environment == "OST\_DEV")

{

uIAvailableRegionsDdlComboBox.SelectedItem = Region;

Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, "{DOWN}" + "{DOWN}" + "{DOWN}");

}

else if (Run\_Environment == "OST\_UAT")

{

Keyboard.SendKeys(uIAvailableRegionsDdlComboBox, "N");

}

if (Run\_Environment == "OST\_PROD")

{

uIAvailableRegionsDdlComboBox.SelectedItem = Region;

//Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, "{DOWN}");

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Perform\_Search\_Owners

/// </summary>

public void Perform\_Search\_Global\_Compliance\_All(String Region)

{

#region Variable Declarations

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableTeamsDdlComboBox;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UISearchFormCustom.UISearchButton;

#endregion

uIAvailableRegionsDdlComboBox.WaitForControlReady(5000);

uIAvailableRegionsDdlComboBox.SelectedItem = Region;

Mouse.Click(uISearchButton);

}

public virtual Perform\_search\_on\_Shares\_pageParams Perform\_search\_on\_Shares\_pageParams

{

get

{

if ((this.mPerform\_search\_on\_Shares\_pageParams == null))

{

this.mPerform\_search\_on\_Shares\_pageParams = new Perform\_search\_on\_Shares\_pageParams();

}

return this.mPerform\_search\_on\_Shares\_pageParams;

}

}

private Perform\_search\_on\_Shares\_pageParams mPerform\_search\_on\_Shares\_pageParams;

/// <summary>

/// Verify\_search\_results\_on\_Shares\_page - Use 'Verify\_search\_results\_on\_Shares\_pageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_search\_results\_on\_Shares\_page(String Host)

{

Verify\_search\_criteria(3, Host);

}

/// <summary>

/// Verify\_search\_results\_on\_Dfs\_Shares\_page

/// </summary>

public void Verify\_search\_results\_on\_Dfs\_Shares\_page(String Parameter)

{

Verify\_search\_criteria(1, Parameter);

}

public void Verify\_Partial\_results\_on\_Dfs\_Shares\_Page(String Parameter)

{

Verify\_search\_criteria\_contains(1, Parameter);

}

/// <summary>

/// Verify\_search\_results\_on\_NTFS\_page

/// </summary>

public void Verify\_search\_results\_on\_NTFS\_page(String Host, String Share, String Account)

{

Verify\_search\_criteria(5, Host);

Verify\_search\_criteria(6, Share);

Verify\_search\_criteria(3, Account);

}

/// <summary>

/// Verify\_search\_results

/// </summary>

public void Verify\_search\_results(int Column, String Data)

{

Verify\_search\_criteria(Column, Data);

}

/// <summary>

/// Verify\_search\_results\_on\_Host\_Maintenance

/// </summary>

public void Verify\_search\_results\_on\_Host\_Maintenance(String TeamName, String Division, String COBPartner, String MonitoringServer, String DataCenter, String Value)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

if (uIJqGridTable.GetCell(1, 1) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(1, 1);

Console.WriteLine("Expected: '" + TeamName + "'. Actual: '" + row.GetCell(2).InnerText.ToString() + "'.");

Assert.AreEqual(TeamName, row.GetCell(2).InnerText.ToString());

Console.WriteLine("Expected: '" + Division + "'. Actual: '" + row.GetCell(3).InnerText.ToString() + "'.");

Assert.AreEqual(Division, row.GetCell(3).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(4).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(4).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(5).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(5).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(6).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(6).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(7).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(7).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(8).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(8).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(9).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(9).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(10).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(10).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(11).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(11).InnerText.ToString());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + row.GetCell(12).InnerText.ToString() + "'.");

Assert.AreEqual(Value, row.GetCell(12).InnerText.ToString());

Console.WriteLine("Expected: '" + COBPartner + "'. Actual: '" + row.GetCell(14).InnerText.ToString() + "'.");

Assert.AreEqual(COBPartner, row.GetCell(14).InnerText.ToString());

Console.WriteLine("Expected: '" + MonitoringServer + "'. Actual: '" + row.GetCell(15).InnerText.ToString() + "'.");

Assert.AreEqual(MonitoringServer, row.GetCell(15).InnerText.ToString());

Console.WriteLine("Expected: '" + DataCenter + "'. Actual: '" + row.GetCell(18).InnerText.ToString() + "'.");

Assert.AreEqual(DataCenter, row.GetCell(18).InnerText.ToString());

}

else

{

Console.WriteLine("Expected: '" + TeamName + "'. Actual: '" + uIJqGridTable.GetCell(1, 2).InnerText.ToString() + "'.");

Assert.AreEqual(TeamName, uIJqGridTable.GetCell(1, 2).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Division + "'. Actual: '" + uIJqGridTable.GetCell(1, 3).InnerText.ToString() + "'.");

Assert.AreEqual(Division, uIJqGridTable.GetCell(1, 3).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 4).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 4).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 5).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 5).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 6).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 6).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 7).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 7).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 8).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 8).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 9).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 9).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 10).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 10).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 11).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 11).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + Value + "'. Actual: '" + uIJqGridTable.GetCell(1, 12).InnerText.ToString() + "'.");

Assert.AreEqual(Value, uIJqGridTable.GetCell(1, 12).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + COBPartner + "'. Actual: '" + uIJqGridTable.GetCell(1, 14).InnerText.ToString() + "'.");

Assert.AreEqual(COBPartner, uIJqGridTable.GetCell(1, 14).InnerText.ToString().Trim());

Console.WriteLine("Expected: '" + MonitoringServer + "'. Actual: '" + uIJqGridTable.GetCell(1, 15).InnerText.ToString() + "'.");

Assert.AreEqual(MonitoringServer, uIJqGridTable.GetCell(1, 15).InnerText.ToString());

Console.WriteLine("Expected: '" + DataCenter + "'. Actual: '" + uIJqGridTable.GetCell(1, 18).InnerText.ToString() + "'.");

Assert.AreEqual(DataCenter, uIJqGridTable.GetCell(1, 18).InnerText.ToString().Trim());

}

Console.WriteLine("Search results verification finished.");

}

/// <summary>

/// Verify\_search\_results\_on\_NTFS\_page

/// </summary>

public void Verify\_search\_results\_on\_Share\_Compliance(String Host)

{

Verify\_search\_criteria(1, Host);

}

public void Verify\_filtered\_search\_results\_on\_Shares\_page(String Data)

{

Verify\_search\_criteria(2, Data);

}

public void Verify\_filtered\_search\_results\_on\_Dfs\_Shares\_page(String Data)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

Verify\_search\_criteria(5, Data);

}

else if(Run\_Environment == "OST\_UAT")

{

Verify\_search\_criteria(3, Data);

}

else if (Run\_Environment == "OST\_PROD")

{

Verify\_search\_criteria(4, Data);

}

}

public void Verify\_filtered\_search\_results\_on\_NTFS\_page(String Data)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

Verify\_search\_criteria(4, Data);

}

else if (Run\_Environment == "OST\_UAT")

{

Verify\_search\_criteria(7, Data);

}

}

public void Verify\_filtered\_search\_results\_on\_Share\_Compliance(String Data)

{

Verify\_search\_criteria(2, Data);

}

public virtual Verify\_search\_results\_on\_Shares\_pageExpectedValues Verify\_search\_results\_on\_Shares\_pageExpectedValues

{

get

{

if ((this.mVerify\_search\_results\_on\_Shares\_pageExpectedValues == null))

{

this.mVerify\_search\_results\_on\_Shares\_pageExpectedValues = new Verify\_search\_results\_on\_Shares\_pageExpectedValues();

}

return this.mVerify\_search\_results\_on\_Shares\_pageExpectedValues;

}

}

private Verify\_search\_results\_on\_Shares\_pageExpectedValues mVerify\_search\_results\_on\_Shares\_pageExpectedValues;

/// <summary>

/// Verify\_Shares\_search\_finished - Use 'Verify\_Shares\_search\_finishedExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_search\_finished()

{

#region Variable Declarations

HtmlLabel uIProcessingLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIProcessingPane.UIProcessingLabel;

HtmlDiv uIProgressBar = this.UIShareInformationOSTOWindow2.UIShareInformationOSTODocument1.UIProgressBar;

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlControl uIMessageSearchMoreResults = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.uIMessageSearchMoreResults;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

Playback.Wait(3000);

if (Run\_Environment == "OST\_DEV")

{

try

{

while (uIProgressBar.GetProperty("InnerHTML").ToString().Trim() != "")

{

Playback.Wait(5000);

}

}

catch {}

}

else

{

//uIProcessingLabel.WaitForControlNotExist(70000);

try

{

while (uIProgressBar.GetProperty("InnerHTML").ToString().Trim() != "")

{

Playback.Wait(5000);

}

}

catch { }

}

uIJqGridTable.Find();

Console.WriteLine("Search finished, records in table on first page: " + (uIJqGridTable.RowCount - 1));

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table or a timeout occured.");

}

//public String Get\_grid\_table\_records\_count()

//{

// #region Variable Declarations

// HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

// HtmlCell UITotalGridCount = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UITotalGridCount;

// HtmlComboBox uIRowsDropdown = this.UIGlobalAccountEntitleWindow.UIGlobalAccountEntitleDocument.UIPageofFooter.UIRowsDropdown;

// HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

// #endregion

// uIJqGridTable.Find();

// uIRowsDropdown.SelectedItem = "100";

// Console.WriteLine("Total records in table: " + (uIJqGridTable.RowCount - 1));

// DateTime Start = DateTime.Now;

// String[] HeaderArray = new String[header.CellCount];

// for (int i = 0; i < header.CellCount; i++)

// {

// HeaderArray[i] = header.GetCell(0, i).InnerText.Trim();

// Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

// }

// List<string> ShareName = new List<string>();

// var ShareName = new List<string>;

// int GridColumnOffset = -1;

// int I = 1;

// for (I = 1; I < uIJqGridTable.RowCount; I++)

// {

// for (int J = 1; J < uIJqGridTable.GetRow(1).GetChildren().Count - GridColumnOffset; J++)

// {

// if (HeaderArray[J + GridColumnOffset] == "Share Name")

// {

// ShareName.Add(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim());

// }

// }

// }

// DateTime End = DateTime.Now;

// TimeSpan DifferenceDateTime = (End - Start);

// DifferenceDateTime = DifferenceDateTime.Duration();

// Console.WriteLine("Duration for Row Iteration: " + DifferenceDateTime);

// Console.WriteLine("Number of records captured from table: " + ShareName.Count);

// Assert.AreEqual(uIJqGridTable.RowCount - 1, ShareName.Count, "Number of captured records does not match number of records in table.");

// Start = DateTime.Now;

// var DistinctShareName = ShareName.GroupBy(s => s).Select(group => new { Word = group.Key, Count = group.Count() });

// Console.WriteLine("Number of distinct records captured from table: " + DistinctShareName.ToList().Count.ToString());

// End = DateTime.Now;

// DifferenceDateTime = (End - Start);

// DifferenceDateTime = DifferenceDateTime.Duration();

// Console.WriteLine("Duration for distint value calc: " + DifferenceDateTime);

// foreach (var Value in DistinctShareName)

// {

// Console.WriteLine("Value: " + Value);

// }

// return (DistinctShareName.ToList().Count.ToString());

//}

public String Get\_grid\_table\_records\_count()

{

#region Variable Declarations

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlCell UITotalGridCount = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UITotalGridCount;

HtmlComboBox uIRowsDropdown = this.UIGlobalAccountEntitleWindow.UIGlobalAccountEntitleDocument.UIPageofFooter.UIRowsDropdown;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

#endregion

int j = 0;

int Counter = 0;

int ColShareName = 0;

uIJqGridTable.Find();

uIRowsDropdown.SelectedItem = "100";

Console.WriteLine("Total records in table: " + (uIJqGridTable.RowCount - 1));

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

int RowCnt = uIJqGridTable.RowCount;

int ColCnt = uIJqGridTable.GetRow(1).GetChildren().Count;

String[] DataArray = new String[RowCnt];

UITestControlCollection utc = uIJqGridTable.Rows;

foreach (HtmlRow row in utc)

{

Counter++;

if (Counter == 1)

{

for (int k = 0; k < header.CellCount; k++)

{

String HeaderText = header.GetCell(0, k).InnerText.Trim().ToString();

Console.WriteLine("Header " + k + ": " + HeaderText.ToString());

DataArray[j] = DataArray[j] + "^#$@~" + HeaderText;

}

j++;

}

else

{

foreach (HtmlCell cell in row.Cells)

{

String Data = cell.InnerText;

DataArray[j] = DataArray[j] + "^#$@~" + Data;

}

j++;

}

}

List<string> ShareName = new List<string>();

string[] stringSeparators = new string[] { "^#$@~" };

String[] HeaderArray = DataArray[0].Split(stringSeparators, StringSplitOptions.None);

for(int k=0; k<HeaderArray.Count(); k++)

{

if (HeaderArray[k] == "Share Name")

{

ColShareName = k;

break;

}

}

for(int i =1; i< DataArray.Count(); i++)

{

String[] RowValues = DataArray[i].Split(stringSeparators, StringSplitOptions.None);

ShareName.Add(RowValues[ColShareName].Trim());

}

Console.WriteLine("Number of records captured from table: " + ShareName.Count);

Assert.AreEqual(uIJqGridTable.RowCount - 1, ShareName.Count, "Number of captured records does not match number of records in table.");

var DistinctShareName = ShareName.GroupBy(s => s).Select(group => new { Word = group.Key, Count = group.Count() });

Console.WriteLine("Number of distinct records captured from table: " + DistinctShareName.ToList().Count.ToString());

foreach (var Value in DistinctShareName)

{

Console.WriteLine("Value: " + Value);

}

return (DistinctShareName.ToList().Count.ToString());

}

public virtual Verify\_Shares\_search\_finishedExpectedValues Verify\_Shares\_search\_finishedExpectedValues

{

get

{

if ((this.mVerify\_Shares\_search\_finishedExpectedValues == null))

{

this.mVerify\_Shares\_search\_finishedExpectedValues = new Verify\_Shares\_search\_finishedExpectedValues();

}

return this.mVerify\_Shares\_search\_finishedExpectedValues;

}

}

private Verify\_Shares\_search\_finishedExpectedValues mVerify\_Shares\_search\_finishedExpectedValues;

public void Verify\_search\_criteria(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

if (BrowserWindow.CurrentBrowser != "Chrome")

{

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

}

//while (intNextPageDisabled == 0)

//{

DateTime Start = DateTime.Now;

UITestControlCollection rowcontrol = uIJqGridTable.Rows;

int rowcount = 0;

foreach (UITestControl item in rowcontrol)

{

int colCount = 0;

if (item is HtmlRow)

{

rowcount++;

if (BrowserWindow.CurrentBrowser == "Chrome")

{

if (rowcount > 1 & rowcount < 7)

{

HtmlRow rowchrome = (HtmlRow)item;

StrCellContent = rowchrome.GetCell(ColumnNumber).InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

Assert.AreEqual(Data.ToLower(), StrCellContent.TrimEnd().ToLower());

}

}

else

{

foreach (HtmlControl cell in item.GetChildren())

{

if (rowcount > 1 & rowcount < 7 & colCount == ColumnNumber)

{

StrCellContent = cell.InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

Assert.AreEqual(Data.ToLower(), StrCellContent.Trim().ToLower());

}

colCount++;

}

}

}

}

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Duration: " + DifferenceDateTime);

// DateTime Start2 = DateTime.Now;

//for (int I = 1; I < uIJqGridTable.RowCount && I < 6; I++)

//{

// if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

// {

// HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

// StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

// Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

// Assert.AreEqual(Data, StrCellContent);

// }

// else

// {

// StrCellContent = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

// Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

// Assert.AreEqual(Data, StrCellContent);

// }

// //}

// //intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

//}

//Console.WriteLine("Search results verification finished.");

////ActionLogEntry.WriteLine("ActionLogEntry: Search results verification finished.");

//DateTime End2 = DateTime.Now;

//TimeSpan DifferenceDateTime2 = (End2 - Start2);

//DifferenceDateTime2 = DifferenceDateTime2.Duration();

//Console.WriteLine("Duration: " + DifferenceDateTime2);

}

public void Verify\_search\_criteria\_contains(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

String StrCellContent = "";

Playback.Wait(2000);

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

if (BrowserWindow.CurrentBrowser != "Chrome")

{

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

}

DateTime Start = DateTime.Now;

UITestControlCollection rowcontrol = uIJqGridTable.Rows;

int rowcount = 0;

foreach (UITestControl item in rowcontrol)

{

int colCount = 0;

if (item is HtmlRow)

{

rowcount++;

if (BrowserWindow.CurrentBrowser == "Chrome")

{

if (rowcount > 1 & rowcount < 7)

{

HtmlRow rowchrome = (HtmlRow)item;

StrCellContent = rowchrome.GetCell(ColumnNumber).InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

StringAssert.Contains(StrCellContent.TrimEnd().ToLower(), Data.ToLower());

}

}

else

{

foreach (HtmlControl cell in item.GetChildren())

{

if (rowcount > 1 & rowcount < 7 & colCount == ColumnNumber)

{

StrCellContent = cell.InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

StringAssert.Contains(StrCellContent.TrimEnd().ToLower(), Data.ToLower());

}

colCount++;

}

}

}

}

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Duration: " + DifferenceDateTime);

}

public void Verify\_search\_criteria\_Sharepoint\_Info(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

if (BrowserWindow.CurrentBrowser != "Chrome")

{

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

}

//while (intNextPageDisabled == 0)

//{

DateTime Start = DateTime.Now;

UITestControlCollection rowcontrol = uIJqGridTable.Rows;

int rowcount = 0;

foreach (UITestControl item in rowcontrol)

{

int colCount = 0;

if (item is HtmlRow)

{

rowcount++;

if (BrowserWindow.CurrentBrowser == "Chrome")

{

if (rowcount > 1 & rowcount < 7)

{

HtmlRow rowchrome = (HtmlRow)item;

StrCellContent = rowchrome.GetCell(ColumnNumber).InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

Assert.AreEqual(Data.ToLower(), StrCellContent.TrimEnd().ToLower());

}

}

else

{

foreach (HtmlControl cell in item.GetChildren())

{

if (rowcount > 1 & rowcount < 7 & colCount == ColumnNumber)

{

StrCellContent = cell.InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

Assert.AreEqual(Data.ToLower(), StrCellContent.Trim().ToLower());

}

colCount++;

}

}

}

}

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Duration: " + DifferenceDateTime);

}

public void Verify\_LastUpdatedDate\_dates\_on\_Sharepoint\_Info(int ColumnNumber)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

if (BrowserWindow.CurrentBrowser != "Chrome")

{

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

}

DateTime Start = DateTime.Now;

UITestControlCollection rowcontrol = uIJqGridTable.Rows;

int rowcount = 0;

foreach (UITestControl item in rowcontrol)

{

int colCount = 0;

if (item is HtmlRow)

{

rowcount++;

if (BrowserWindow.CurrentBrowser == "Chrome")

{

if (rowcount > 1 & rowcount < 7)

{

HtmlRow rowchrome = (HtmlRow)item;

StrCellContent = rowchrome.GetCell(ColumnNumber).InnerText;

Console.WriteLine("Row: " + rowcount + ". Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

//Assert.AreEqual(Data.ToLower(), StrCellContent.TrimEnd().ToLower());

TimeSpan UpdatedDate = (DateTime.Parse(StrCellContent.Trim(), System.Globalization.CultureInfo.InvariantCulture) - Start);

UpdatedDate = UpdatedDate.Duration();

Assert.IsTrue(UpdatedDate.TotalHours < 24, "LastUpdatedDate is more than 24 hours from now.");

}

}

else

{

foreach (HtmlControl cell in item.GetChildren())

{

if (rowcount > 1 & rowcount < 7 & colCount == ColumnNumber)

{

StrCellContent = cell.InnerText;

Console.WriteLine("Row: " + rowcount + ". Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

//Assert.AreEqual(Data.ToLower(), StrCellContent.Trim().ToLower());

TimeSpan UpdatedDate = (DateTime.Parse(StrCellContent.Trim(), System.Globalization.CultureInfo.InvariantCulture) - Start);

UpdatedDate = UpdatedDate.Duration();

//Console.WriteLine("Duration: " + UpdatedDate.ToString());

//Console.WriteLine("Age: " + UpdatedDate.TotalHours.ToString());

Assert.IsTrue(UpdatedDate.TotalHours < 24, "LastUpdatedDate is more than 24 hours from now.");

}

colCount++;

}

}

}

}

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Duration: " + DifferenceDateTime);

}

public void Verify\_search\_results\_on\_NAS\_PhysicalDevice(String Host)

{

Verify\_search\_criteria(0, Host);

}

public void Verify\_search\_results\_on\_NAS\_PhysicalDevice\_Domain(String Region)

{

String Domain= "";

Region = Region.ToUpper();

switch (Region)

{

case "NA":

Domain = "nam.nsroot.net";

break;

case "LATAM":

Domain = "nam.nsroot.net";

break;

case "EMEA":

Domain = "eur.nsroot.net";

break;

case "ASIAPAC":

Domain = "apac.nsroot.net";

break;

default:

Console.WriteLine("Select any valid Domain");

break;

}

Verify\_search\_criteria(4, Domain);

}

public void Verify\_search\_results\_on\_NAS\_PhysicalDevice\_ErrMsg(String SearchData)

{

String ErrMsg = "";

if (SearchData == "false")

ErrMsg = "Error getting NAS Details.";

Verify\_search\_criteria\_contains(9, ErrMsg);

}

public void Verify\_Date(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

if (BrowserWindow.CurrentBrowser != "Chrome")

{

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

}

DateTime Start = DateTime.Now;

UITestControlCollection rowcontrol = uIJqGridTable.Rows;

int rowcount = 0;

foreach (UITestControl item in rowcontrol)

{

int colCount = 0;

if (item is HtmlRow)

{

rowcount++;

if (BrowserWindow.CurrentBrowser == "Chrome")

{

if (rowcount > 1 & rowcount < 7)

{

HtmlRow rowchrome = (HtmlRow)item;

StrCellContent = rowchrome.GetCell(ColumnNumber).InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

DateTime Dtdel = DateTime.Parse(StrCellContent, System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy");

StrCellContent = Datedel.ToLower();

Assert.AreEqual(Data.ToLower(), StrCellContent.TrimEnd().ToLower());

}

}

else

{

foreach (HtmlControl cell in item.GetChildren())

{

if (rowcount > 1 & rowcount < 7 & colCount == ColumnNumber)

{

StrCellContent = cell.InnerText;

Console.WriteLine("Row: " + rowcount + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

//Console.WriteLine("StrCellContent: " + StrCellContent);

DateTime Dtdel = DateTime.Parse(StrCellContent, System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy");

StrCellContent = Datedel.ToLower();

Assert.AreEqual(Data.ToLower(), StrCellContent.Trim().ToLower());

}

colCount++;

}

}

}

}

DateTime End = DateTime.Now;

TimeSpan DifferenceDateTime = (End - Start);

DifferenceDateTime = DifferenceDateTime.Duration();

Console.WriteLine("Duration: " + DifferenceDateTime);

}

public void Verify\_search\_criteria\_second\_table(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeadersExclusionsForTeams;

int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

for (int i = 0; i < header.CellCount; i++)

{

Console.WriteLine("Header " + i + ": " + header.GetCell(0, i).InnerText.ToString());

}

while (intNextPageDisabled == 0)

{

for (int I = 1; I < uIJqGridTable.RowCount && I < 6; I++)

{

if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

else

{

StrCellContent = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

}

intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

}

Trace.WriteLine("Trace: Search results verification finished.");

Debug.WriteLine("Debug: Search results verification finished.");

Console.WriteLine("Search results verification finished.");

//ActionLogEntry.WriteLine("ActionLogEntry: Search results verification finished.");

}

public void Verify\_Anti\_Virus\_search\_criteria(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

while (intNextPageDisabled == 0)

{

for (int I = 1; I < uIJqGridTable.RowCount; I++)

{

if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

{

DateTime DateTime = DateTime.Parse(Data, System.Globalization.CultureInfo.InvariantCulture);

Data = DateTime.ToString("MM/dd/yyyy").ToLower();

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

DateTime DateTimeCellContent = DateTime.Parse(StrCellContent, System.Globalization.CultureInfo.InvariantCulture);

StrCellContent = DateTimeCellContent.ToString("MM/dd/yyyy").ToLower();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

else

{

DateTime DateTime = DateTime.Parse(Data, System.Globalization.CultureInfo.InvariantCulture);

Data = DateTime.ToString("MM/dd/yyyy").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime DateTimeCellContent = DateTime.Parse(uIJqGridTableTemp.ToString(), System.Globalization.CultureInfo.InvariantCulture);

StrCellContent = DateTimeCellContent.ToString("MM/dd/yyyy").ToLower();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

} intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

}

Console.WriteLine("Search results verification finished.");

}

public void Verify\_Anti\_Virus\_search\_criteria\_PROD(int ColumnNumber, string Data)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

int intNextPageDisabled = 0;

String StrCellContent = "";

Playback.Wait(2000);

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

for (int i = 1; i <= 2; i++)

{

if(Run\_Environment == "OST\_PROD")

Sort\_Anti\_Virus\_Page\_by\_Host\_column\_Asc();

for (int I = 1; I < uIJqGridTable.RowCount; I++)

{

if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

{

DateTime DateTime = DateTime.Parse(Data, System.Globalization.CultureInfo.InvariantCulture);

Data = DateTime.ToString("MM/dd/yyyy").ToLower();

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

DateTime DateTimeCellContent = DateTime.Parse(StrCellContent, System.Globalization.CultureInfo.InvariantCulture);

StrCellContent = DateTimeCellContent.ToString("MM/dd/yyyy").ToLower();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

else

{

DateTime DateTime = DateTime.Parse(Data, System.Globalization.CultureInfo.InvariantCulture);

Data = DateTime.ToString("MM/dd/yyyy").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime DateTimeCellContent = DateTime.Parse(uIJqGridTableTemp.ToString(), System.Globalization.CultureInfo.InvariantCulture);

StrCellContent = DateTimeCellContent.ToString("MM/dd/yyyy").ToLower();

Console.WriteLine("Row: " + I + ". Expected: '" + Data + "'. Actual: '" + StrCellContent + "'.");

Assert.AreEqual(Data, StrCellContent);

}

}

}

Console.WriteLine("Search results verification finished.");

}

public void Verify\_Deleted\_Volume\_Exception\_search\_criteria(DateTime DateTime, String Host, String Volume, String CsiID, String Comment)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

HtmlDiv Column\_Date\_Deleted = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Date\_Deleted;

string GridCellContent = "";

String CurrentDateTime = DateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Mouse.Click(Column\_Date\_Deleted);

Mouse.Click(Column\_Date\_Deleted);

if (uIJqGridTable.GetCell(1, 1) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(1, 1);

GridCellContent = row.GetCell(2).InnerText.Trim();

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 5).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + row.GetCell(0).InnerText);

Console.WriteLine("Column: " + 1 + " Value: " + row.GetCell(1).InnerText);

Console.WriteLine("Column: " + 3 + " Value: " + row.GetCell(3).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + row.GetCell(4).InnerText);

DateTime GridDateTime = DateTime.Parse(row.GetCell(5).InnerText, System.Globalization.CultureInfo.InvariantCulture);

TimeSpan DifferenceDateTime = (GridDateTime - DateTime);

DifferenceDateTime = DifferenceDateTime.Duration();

String DeleteDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Assert.AreEqual(Host, row.GetCell(0).InnerText);

Assert.AreEqual(Volume, row.GetCell(1).InnerText);

Assert.AreEqual(CsiID, row.GetCell(3).InnerText);

Assert.AreEqual(Comment, row.GetCell(4).InnerText);

Assert.IsTrue(DifferenceDateTime.TotalSeconds < 30);

Console.WriteLine("Volume Exception Delete Date/Time verified successully: " + DeleteDateTime);

Console.WriteLine("Measured Click/Delete time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

else

{

GridCellContent = uIJqGridTable.GetCell(1, 2).InnerText;

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 5).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + uIJqGridTable.GetCell(1, 0).InnerText);

Console.WriteLine("Column: " + 1 + " Value: " + uIJqGridTable.GetCell(1, 1).InnerText);

Console.WriteLine("Column: " + 3 + " Value: " + uIJqGridTable.GetCell(1, 3).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + uIJqGridTable.GetCell(1, 4).InnerText);

String uIJqGridTableTemp = uIJqGridTable.GetCell(1, 5).InnerText;

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

TimeSpan DifferenceDateTime = (GridDateTime - DateTime);

DifferenceDateTime = DifferenceDateTime.Duration();

String DeleteDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Assert.AreEqual(Host, uIJqGridTable.GetCell(1, 0).InnerText);

Assert.AreEqual(Volume, uIJqGridTable.GetCell(1, 1).InnerText);

Assert.AreEqual(CsiID, uIJqGridTable.GetCell(1, 3).InnerText);

Assert.AreEqual(Comment, uIJqGridTable.GetCell(1, 4).InnerText);

Assert.IsTrue(DifferenceDateTime.TotalSeconds < 30);

Console.WriteLine("Volume Exception Delete Date/Time verified successully: " + DeleteDateTime);

Console.WriteLine("Measured Click/Delete time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

}

public void Verify\_Deleted\_Share\_Exception\_search\_criteria(DateTime DateTime, String Host, String Share, String CsiID, String Comment)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

HtmlDiv Column\_Date\_Deleted = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Date\_Deleted;

string GridCellContent = "";

String CurrentDateTime = DateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Mouse.Click(Column\_Date\_Deleted);

Mouse.Click(Column\_Date\_Deleted);

if (uIJqGridTable.GetCell(1, 1) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(1, 1);

GridCellContent = row.GetCell(2).InnerText.Trim();

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 5).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + row.GetCell(0).InnerText);

Console.WriteLine("Column: " + 2 + " Value: " + row.GetCell(2).InnerText);

Console.WriteLine("Column: " + 3 + " Value: " + row.GetCell(3).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + row.GetCell(4).InnerText);

Console.WriteLine("Column: " + 5 + " Value: " + row.GetCell(5).InnerText);

DateTime GridDateTime = DateTime.Parse(row.GetCell(5).InnerText, System.Globalization.CultureInfo.InvariantCulture);

TimeSpan DifferenceDateTime = (GridDateTime - DateTime);

DifferenceDateTime = DifferenceDateTime.Duration();

String DeleteDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Assert.AreEqual(Host, row.GetCell(0).InnerText);

Assert.AreEqual(Share, row.GetCell(2).InnerText);

Assert.AreEqual(CsiID, row.GetCell(3).InnerText);

Assert.AreEqual(Comment, row.GetCell(4).InnerText);

Assert.AreEqual(DeleteDateTime, CurrentDateTime);

Console.WriteLine("Share Exception Delete Date/Time verified successully: " + DeleteDateTime);

Console.WriteLine("Measured Click/Delete time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

else

{

GridCellContent = uIJqGridTable.GetCell(1, 2).InnerText;

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 5).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + uIJqGridTable.GetCell(1, 0).InnerText);

Console.WriteLine("Column: " + 2 + " Value: " + uIJqGridTable.GetCell(1, 2).InnerText);

Console.WriteLine("Column: " + 3 + " Value: " + uIJqGridTable.GetCell(1, 3).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + uIJqGridTable.GetCell(1, 4).InnerText);

Console.WriteLine("Column: " + 5 + " Value: " + uIJqGridTable.GetCell(1, 5).InnerText);

String uIJqGridTableTemp = uIJqGridTable.GetCell(1, 5).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

TimeSpan DifferenceDateTime = (GridDateTime - DateTime);

DifferenceDateTime = DifferenceDateTime.Duration();

String DeleteDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

Assert.AreEqual(Host, uIJqGridTable.GetCell(1, 0).InnerText);

Assert.AreEqual(Share, uIJqGridTable.GetCell(1, 2).InnerText);

Assert.AreEqual(CsiID, uIJqGridTable.GetCell(1, 3).InnerText);

Assert.AreEqual(Comment, uIJqGridTable.GetCell(1, 4).InnerText);

Assert.AreEqual(DeleteDateTime, CurrentDateTime);

Console.WriteLine("Share Exception Delete Date/Time verified successully: " + DeleteDateTime);

Console.WriteLine("Measured Click/Delete time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

}

public int FindColumnNumber(String ColumnName)

{

int col = 0;

HtmlTable table = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//Console.WriteLine(table.GetCell(0, 0).InnerText.ToString());

//Console.WriteLine(table.GetCell(0, 0).ControlType.ToString());

if (table.GetCell(0, 0).ControlType.ToString() == "RowHeader")

{

Console.WriteLine("Is row.");

HtmlRow row = (HtmlRow)table.GetCell(0, 0);

for (col = 0; col < row.CellCount; col++)

{

if (row.GetCell(col).InnerText.ToString().Trim().Equals(ColumnName))

{

Console.WriteLine("Found column " + ColumnName + " in column number " + col);

Assert.AreEqual(ColumnName, row.GetCell(col).InnerText.ToString().Trim());

return col;

}

}

}

else

{

for (col = 0; col < table.CellCount; col++)

{

if (table.GetCell(0, col).InnerText.Trim().Equals(ColumnName))

{

Console.WriteLine("Found column " + ColumnName + " in column number " + col);

Assert.AreEqual(ColumnName, table.GetCell(0, col).InnerText.Trim().ToString());

return col;

}

}

}

Console.WriteLine("Column number not found.");

return 0;

}

public int BrowseTableNextButtonEnabled()

{

HtmlCell uINext\_gridPagerCell = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UINext\_gridPagerCell;

if (uINext\_gridPagerCell.GetProperty("Class").ToString().EndsWith("disabled") == false)

{

Mouse.Click(uINext\_gridPagerCell);

Console.WriteLine("Navigated to the next page of the table.");

return 0;

}

else

{

Console.WriteLine("End of Table reached.");

return 1;

}

}

public virtual Verify\_Shares\_TableExpectedValues Verify\_Shares\_TableExpectedValues

{

get

{

if ((this.mVerify\_Shares\_TableExpectedValues == null))

{

this.mVerify\_Shares\_TableExpectedValues = new Verify\_Shares\_TableExpectedValues();

}

return this.mVerify\_Shares\_TableExpectedValues;

}

}

private Verify\_Shares\_TableExpectedValues mVerify\_Shares\_TableExpectedValues;

/// <summary>

/// Filter\_Shares - Use 'Filter\_SharesParams' to pass parameters into this method.

/// </summary>

public void Filter\_search\_results(String Criteria, String Operator, String SearchData, String ExpectedMessage)

{

#region Variable Declarations

HtmlButton uIFilterButton = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIFilterButton;

HtmlEdit uIJqg1Edit = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqg1Edit;

HtmlComboBox UIOperatorSelect = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIOperatorSelect;

HtmlComboBox UICriteriaSelect = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UICriteriaSelect;

HtmlHyperlink uISearchHyperlink = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISearchHyperlink;

HtmlHyperlink uIVoid0Hyperlink = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIAdvancedSearchPane.UIVoid0Hyperlink;

HtmlLabel uIFilterShareNameisequLabel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIFilterShareNameisequLabel;

#endregion

// Click 'Filter' button

Playback.Wait(3000);

uIFilterButton.WaitForControlReady(5000);

Mouse.Click(uIFilterButton);

if (UICriteriaSelect.Exists == false)

{ Mouse.Click(uIFilterButton); }

UICriteriaSelect.WaitForControlEnabled(2000);

//Keyboard.SendKeys(UICriteriaSelect, Criteria + "{Enter}");

UICriteriaSelect.SelectedItem = Criteria;

//UIOperatorSelect.SelectedItem = Operator;

Keyboard.SendKeys(UIOperatorSelect, Operator + "{Enter}");

uIJqg1Edit.WaitForControlEnabled(2000);

Mouse.Click(uIJqg1Edit);

Keyboard.SendKeys(uIJqg1Edit, SearchData + "{Enter}");

uIJqg1Edit.WaitForControlReady(5000);

// Click 'Search' link

Mouse.Click(uISearchHyperlink);

if (uISearchHyperlink.Exists == true)

{

Mouse.Click(uISearchHyperlink);

}

// Close Filter popup

while (uIVoid0Hyperlink.Exists == true)

{

try

{

Mouse.Click(uIVoid0Hyperlink);

}

catch { return; }

}

// Verify that the 'InnerText' property of 'Filter: [ShareName is equal to 'autoaudit\_archives...' label equals 'Filter: [ShareName is equal to 'autoaudit\_archives']'

Console.WriteLine("Expected: " + ExpectedMessage + " Actual: " + uIFilterShareNameisequLabel.InnerText);

Assert.AreEqual(ExpectedMessage, uIFilterShareNameisequLabel.InnerText);

}

public void Verify\_Excel(int GridColumnOffset, int ExcelColumnOffset, String StrFilepath)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(StrFilepath, 0, true);

\_xlWorkSheet = \_xlWorkBook.Worksheets.get\_Item(1);

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

// int intNextPageDisabled = 0;

int Page = 1;

int I = 1;

string GridCellContent = "";

string ExcelCellContent = "";

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

double Temp;

double Temp2;

String[] HeaderArray = new String[header.CellCount];

//String[,] GridArray = new String[uIJqGridTable.RowCount, uIJqGridTable.GetRow(1).GetChildren().Count];

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

for (int i = 0; i < header.CellCount; i++)

{

HeaderArray[i] = header.GetCell(0, i).InnerText.Trim();

Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

}

//Console.WriteLine("Starting to get Grid table values: " + DateTime.Now);

//for (int i = 1; i < uIJqGridTable.RowCount; i++)

//{

// for (int j = 0; j < uIJqGridTable.GetRow(1).GetChildren().Count; j++)

// {

// if (uIJqGridTable.GetCell(i, j).InnerText != null)

// {

// GridArray[i, j] = uIJqGridTable.GetCell(i, j).InnerText.Trim();

// Console.WriteLine("Row: " + i + "Column: " + j + " Value: " + GridArray[i, j]);

// }

// }

//}

//Console.WriteLine("Finished to get Grid table values: " + DateTime.Now);

//while (intNextPageDisabled == 0)

//{

for (I = 1; I < uIJqGridTable.RowCount && I < 10; I++)

{

for (int J = 1; J < uIJqGridTable.GetRow(1).GetChildren().Count - GridColumnOffset; J++)

{

if (uIJqGridTable.GetCell(I, J + GridColumnOffset).InnerText != null)

{

ExcelCellContent = "";

GridCellContent = "";

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

try

{

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString();

}

catch {}

if (ExcelCellContent == null) { ExcelCellContent = ""; }

if (GridCellContent == null) { GridCellContent = ""; }

if (HeaderArray[J + GridColumnOffset] == "Compliance %" || (HeaderArray[J + GridColumnOffset] == "Free (%)") || (HeaderArray[J + GridColumnOffset] == "Used (%)"))//(header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Compliance %")

{

Temp = Math.Truncate(double.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString()) \* 1000) / 1000;

ExcelCellContent = String.Format("{0:N2}", Temp);

Console.WriteLine("Checkpoint: 1");

}

else if (\_xlWorkSheet.Cells[1, 5].Value2 == "Group Member")

{

if (J > 4)

{

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset + 2].Value2.ToString();

if (HeaderArray[J + GridColumnOffset] == "Inheritance")

{

GridCellContent = "Skipping Inheritance comparison";

ExcelCellContent = "Skipping Inheritance comparison";

Console.WriteLine("Checkpoint: 2");

}

else if (HeaderArray[J + GridColumnOffset] == "Initial Scan Date")

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + 2].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

Console.WriteLine("Checkpoint: 3");

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

}

else ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString();

Console.WriteLine("Checkpoint: else");

}

else if (HeaderArray[J + GridColumnOffset] == "UoWType"

|| HeaderArray[J + GridColumnOffset] == "UserReportRequestId"

|| HeaderArray[J + GridColumnOffset] == "Actions"

|| HeaderArray[J + GridColumnOffset] == "GroupID"

|| HeaderArray[J + GridColumnOffset] == "TeamId"

|| HeaderArray[J + GridColumnOffset] == "ArchiveDate"

|| HeaderArray[J + GridColumnOffset] == "ArchivedDate"

|| HeaderArray[J + GridColumnOffset] == "ArchivedBy"

|| HeaderArray[J + GridColumnOffset] == "Action"

|| HeaderArray[J + GridColumnOffset] == "Migrated")

//(header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "UoWType" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "UserReportRequestId" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Actions" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "GroupID" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "TeamId" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "ArchiveDate" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "ArchivedBy" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Action")

{

GridCellContent = "";

ExcelCellContent = "";

Console.WriteLine("Skipping hidden grid column: " + header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim());

}

else if ((HeaderArray[J + GridColumnOffset] == "Status" || HeaderArray[J + GridColumnOffset] == "File Name") && HeaderArray[1] == "Report Name")

{

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset - 1].Value2.ToString();

Console.WriteLine("Checkpoint: 5");

}

else if (HeaderArray[J + GridColumnOffset] == "Data Center" && HeaderArray[0] == "Action")

{

Console.WriteLine("Checkpoint: 6");

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset - 1].Value2.ToString();

}

else if (HeaderArray[J + GridColumnOffset] == "Started (UTC)" || HeaderArray[J + GridColumnOffset] == "Finished (UTC)" || HeaderArray[J + GridColumnOffset] == "Requested (UTC)")

//(header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Started (UTC)" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Finished (UTC)" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Requested (UTC)")

{

Console.WriteLine("Checkpoint: 7");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = ExcelDateTime.AddHours(-1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else if (HeaderArray[J + GridColumnOffset] == "Total" || HeaderArray[J + GridColumnOffset] == "Size (MB)" || HeaderArray[J + GridColumnOffset] == "Junk GB" || HeaderArray[J + GridColumnOffset] == "Left GB" || HeaderArray[J + GridColumnOffset] == "TotalStorage" || HeaderArray[J + GridColumnOffset] == "Used GB" || HeaderArray[J + GridColumnOffset] == "Total GB")

//(header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Total" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Size (MB)" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Junk GB" || header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Left GB" || header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "TotalStorage")

{

Console.WriteLine("Checkpoint: 8");

Temp = Math.Truncate(double.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString()) \* 1000) / 1000;

Temp2 = Math.Truncate(double.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim()) \* 1000) / 1000;

ExcelCellContent = String.Format("{0:N2}", Temp);

GridCellContent = String.Format("{0:N2}", Temp2);

}

else if ((J + GridColumnOffset) > 5 && ((HeaderArray[3] == "User ID" && HeaderArray[4] == "Account Disabled") || HeaderArray[3] == "Local Group"))

{

if (HeaderArray[J + GridColumnOffset] == "Last Scan" && HeaderArray[3] == "User ID" && HeaderArray[4] == "Account Disabled")

{

Console.WriteLine("Checkpoint: 9");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

//ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

Console.WriteLine("GridDateTime after date time parse value: " + GridDateTime.ToString());

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

Console.WriteLine("Final GridCellContent: " + GridCellContent);

}

else if (HeaderArray[J + GridColumnOffset] == "Last Scan" && HeaderArray[3] == "Local Group")

{

Console.WriteLine("Checkpoint: 17");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset - 1].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else

{

GridCellContent = "Skipping hidden column.";

ExcelCellContent = "Skipping hidden column.";

}

}

else if (HeaderArray[J + GridColumnOffset] == "Last Scan" && J > 6)

{

//t120 Excel Export Local Groups

//Console.WriteLine("J: " + J);

//Console.WriteLine("Grid: " + uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim());

//Console.WriteLine("Excel: " + \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset - 1].Value2.ToString());

Console.WriteLine("Checkpoint: 10");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset - 1].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

//AntiVirus

else if (HeaderArray[J + GridColumnOffset] == "Log Date" && (J + GridColumnOffset == 0))

{

Console.WriteLine("AntiVirus page.");

var culture = new CultureInfo("en-US");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, culture);

//ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

Console.WriteLine("Excel: " + ExcelCellContent);

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), culture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

Console.WriteLine("Grid before: " + GridDateTime);

Console.WriteLine("Grid after: " + GridCellContent);

}

//User Rights and Security Checks

else if (HeaderArray[J + GridColumnOffset] == "Log Date" && (J + GridColumnOffset == 3))

{

Console.WriteLine("Checkpoint: 11");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

//ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

//DataStatsOverview

else if (HeaderArray[J + GridColumnOffset] == "Log Date" && J > 8)

{

//if (J + GridColumnOffset == 9)

//{

Console.WriteLine("Checkpoint: 12");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 5].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

//}

}

//else if ((HeaderArray[J + GridColumnOffset] == "Log Date" && J == 8)

// || (HeaderArray[J + GridColumnOffset] == "LogDate" && J == 7))

//{

// DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

// ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

// ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

// DateTime GridDateTime = DateTime.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim(), System.Globalization.CultureInfo.InvariantCulture);

// GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

// GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

//}

//else if (header.GetCell(0, 0).InnerText.Trim() == "Account Name")

//{

// if (header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Initial Scan Date")

// {

// DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J].Value2, System.Globalization.CultureInfo.InvariantCulture);

// ExcelDateTime = ExcelDateTime.AddHours(1);

// ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

// ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

// DateTime GridDateTime = DateTime.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText, System.Globalization.CultureInfo.InvariantCulture);

// GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

// GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

// }

// else

// {

// GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

// ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J].Value2.ToString();

// }

//}

//Global Account Entitlement Search

else if ((HeaderArray[J + GridColumnOffset] == "Initial Scan Date" && HeaderArray[J + GridColumnOffset - 1] == "IsInherited"))

{

Console.WriteLine("Checkpoint: 16");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

if (Run\_Environment == "OST\_DEV")

{

//Adding 1h for DEV server time difference

ExcelDateTime = ExcelDateTime.AddHours(1);

}

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else if ((HeaderArray[J + GridColumnOffset] == "Created")

|| (HeaderArray[J + GridColumnOffset] == "Log Date" && J == 8)

|| (HeaderArray[J + GridColumnOffset] == "LogDate" && J == 7)

|| (HeaderArray[J + GridColumnOffset] == "NextRun")

)

{

Console.WriteLine("Checkpoint: 13");

Console.WriteLine("Excel: " + \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2);

if (\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2 != "")

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

if (Run\_Environment == "OST\_DEV")

{

//Adding 1h for DEV server time difference

Console.WriteLine("Checkpoint386.");

//Orphans fail with +1

//ExcelDateTime = ExcelDateTime.AddHours(1);

}

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

if (uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim() != "")

{

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

}

else if (HeaderArray[J + GridColumnOffset] == "Archived Date" || HeaderArray[J + GridColumnOffset] == "Initial Scan Date")

{

Console.WriteLine("Checkpoint: 14");

Console.WriteLine("Value: " + \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2);

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

Console.WriteLine("Value: " + NewGridCellContent.ToString());

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

//DateTime GridDateTime = DateTime.ParseExact(NewGridCellContent.ToString(), "M/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

Console.WriteLine("Excel: " + ExcelCellContent);

Console.WriteLine("Grid: " + GridCellContent);

}

else if (HeaderArray[J + GridColumnOffset] == "LastUpdatedDate")

{

Console.WriteLine("Checkpoint: 15");

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.CurrentCulture);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

//DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

Console.WriteLine("Excel: " + ExcelCellContent);

Console.WriteLine("Grid: " + GridCellContent);

}

else if (HeaderArray[J + GridColumnOffset] == "Daily MB" || HeaderArray[J + GridColumnOffset] == "Weekly MB" || HeaderArray[J + GridColumnOffset] == "Monthly MB" || HeaderArray[J + GridColumnOffset] == "Yearly MB" || HeaderArray[J + GridColumnOffset] == "Home Share Count" || HeaderArray[J + GridColumnOffset] == "Group Share Count")

{

Console.WriteLine("Checkpoint: 16");

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Trim();

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString();

if (ExcelCellContent == "") { ExcelCellContent = "-"; }

}

else

{ //Console.WriteLine("Column: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim()); Console.WriteLine("Else route.");

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), (J + ExcelColumnOffset)].Value2.ToString();

Console.WriteLine("Checkpoint: last else.");

}

//Console.WriteLine("Loop Date/Time: " + DateTime.Now);

Console.WriteLine("Header: " + HeaderArray[J + GridColumnOffset]);

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Value: " + GridCellContent);

Console.WriteLine("Excel Row: " + (I + 1 + ((Page - 1) \* 20)) + " Column " + (J + ExcelColumnOffset) + ": " + " Value: " + ExcelCellContent);

//test

//Console.WriteLine("Table row 1 column 0: " + uIJqGridTable.GetCell(1, 0).InnerText);

//Console.WriteLine("Table row 1 column 1: " + uIJqGridTable.GetCell(1, 1).InnerText);

//Console.WriteLine("Table row 1 column 2: " + uIJqGridTable.GetCell(1, 2).InnerText);

//Console.WriteLine("Table row 1 column 3: " + uIJqGridTable.GetCell(1, 3).InnerText);

//test

if (ExcelCellContent == null) { ExcelCellContent = ""; }

if (GridCellContent == null) { GridCellContent = ""; }

Assert.AreEqual(GridCellContent.Normalize().Trim().ToString().ToLower().Replace("\r", ""), ExcelCellContent.Normalize().Trim().ToString().ToLower().Replace("\n", ""), "Value in excel appears to be not as expected.");

}

else

{

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Cell is empty.");

}

}

}

//intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

//Page++;

//}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

Console.WriteLine("Excel export successfully compared with grid table.");

}

public void VerifyExcel\_DataStatsOverview(string FileName, string[] ArrayData)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(FileName, 0, true);//5, "", "", true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

\_xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)\_xlWorkBook.Worksheets.get\_Item(1);

String strCellValue, strCellValue2, strheaderval;

int i, j, k;

bool flagheader, flagvalidcol;

//: int rownum = 1;

int intFailCnt = 0;

int ArraySize = ArrayData.Length;

string[] stringSeparators = new string[] { "^#$@~" };

//String[] SplitWrds = EleEnabled.Split(' ');

String[] header = ArrayData[0].Split(stringSeparators, StringSplitOptions.None);

int intheadercnt = header.Length;

for (i = 0; i < intheadercnt; i++)

{

flagheader = false;

flagvalidcol = false;

for (j = 1; j <= intheadercnt; j++)

{

strCellValue = \_xlWorkSheet.Cells[1, j].Value2.ToString();

if (strCellValue != null)

{

strCellValue = strCellValue.ToLower();

strheaderval = header[i].ToLower();

if (strheaderval == "account/group")

{

strheaderval = "account name";

}

if (strheaderval == "log date")

{

strheaderval = "log date(utc)";

}

if (strheaderval == "free gb")

{

strheaderval = "available gb";

}

if ((strheaderval == "") || (strheaderval == " details"))

{

flagheader = true;

flagvalidcol = false;

break;

}

else if (strheaderval.Trim() == strCellValue.Trim())

{

Console.WriteLine("Column header" + header[i], "Column " + header[i] + " found", 0);

flagvalidcol = true;

flagheader = true;

for (k = 1; k < ArrayData.Length; k++)

{

String[] arrresult = ArrayData[k].Split(stringSeparators, StringSplitOptions.None);

strCellValue2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

strCellValue2 = strCellValue2.ToLower();

string strArrresult = arrresult[i].ToLower();

if (header[i] == "Log Date")

{

//DateTime Dtdel = DateTime.ParseExact(strArrresult, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i] == "Inheritance")

{

strArrresult = "Skipping Inheritance comparison";

strCellValue2 = "Skipping Inheritance comparison";

}

if (header[i] == "Initial Scan Date")

{

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if ((header[i].Trim() == "Daily MB") || (header[i].Trim() == "Weekly MB") || (header[i].Trim() == "Monthly MB") || (header[i].Trim() == "Yearly MB") || (header[i].Trim() == "Home Share Count") || (header[i].Trim() == "Group Share Count"))

{

//if (strCellValue2 != "")

//{

//}

//if (strArrresult != "-")

//{

//}

if ((strArrresult == "-") && (strCellValue2 == ""))

{

strCellValue2 = "-";

}

}

if (header[i].Trim() == "Free (%)" || header[i].Trim() == "Used (%)" || header[i].Trim() == "Total GB" || header[i].Trim() == "Used GB" || header[i].Trim() == "Free GB" || header[i].Trim() == "Junk GB" || header[i].Trim() == "Left GB")

{

double DblValue = double.Parse(\_xlWorkSheet.Cells[k + 1, j].Value2.ToString());

DblValue = Math.Truncate(DblValue \* 1000) / 1000;

strCellValue2 = String.Format("{0:N2}", DblValue);

}

//if (\_Function.Compare("Cell value in " + header[i] + " Row no " + (k + 1), strCellValue2.Trim(), strArrresult.Trim(), false) == false)

Assert.AreEqual(strCellValue2.Trim(), strArrresult.Trim());

Console.WriteLine("Cell value \"" + strArrresult.Trim() + "\" and Excel Cell value \"" + strCellValue2.Trim() + "\" are compared");

}

Console.WriteLine("Column header" + header[i], "Values in Column " + header[i] + " verified", 0);

break;

}

}

}

if (flagvalidcol == true)

{

Console.WriteLine("Column header" + header[i], "Column " + header[i] + " found and verified");

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

}

public void VerifyExcel2(string FileName, string[] ArrayData)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(FileName, 0, true);//5, "", "", true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

\_xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)\_xlWorkBook.Worksheets.get\_Item(1);

String strCellValue, strCellValue2, strheaderval;

int i, j, k;

bool flagheader, flagvalidcol;

//: int rownum = 1;

int intFailCnt = 0;

int ArraySize = ArrayData.Length;

string[] stringSeparators = new string[] { "^#$@~" };

//String[] SplitWrds = EleEnabled.Split(' ');

String[] header = ArrayData[0].Split(stringSeparators, StringSplitOptions.None);

int intheadercnt = header.Length;

for (i = 0; i < intheadercnt; i++)

{

flagheader = false;

flagvalidcol = true;

for (j = 1; j <= intheadercnt; j++)

{

strCellValue = \_xlWorkSheet.Cells[1, j].Value2.ToString();

if (strCellValue != null)

{

strCellValue = strCellValue.ToLower();

strheaderval = header[i].ToLower();

if (strheaderval == "account/group")

{

strheaderval = "account name";

}

else if ((strheaderval == "queue") && (strCellValue == "queuename"))

{

strheaderval = "queuename";

}

else if ((strheaderval == "ip address") && (strCellValue == "ipaddr"))

{

strheaderval = "ipaddr";

}

else if (strheaderval == "port name")

{

strheaderval = "portname";

}

else if (strheaderval == "printed jobs")

{

strheaderval = "prtjobs";

}

else if (strheaderval == "q length")

{

strheaderval = "prtqlength";

}

else if (strheaderval == "q status")

{

strheaderval = "prtstatus";

}

else if (strheaderval == "log date" && strCellValue == "logdate")

{

strheaderval = "logdate";

}

else if (strheaderval == "taskname")

{

strheaderval = "scheduledtaskname";

}

else if (strheaderval == "nextrun")

{

strheaderval = "nextruntime";

}

else if (strheaderval == "status")

{

strheaderval = "taskstatus";

}

else if (strheaderval == "host name" && strCellValue == "hostname")

{

strheaderval = "hostname";

}

else if (strheaderval == "teamname" && strCellValue == "team name")

{

strheaderval = "team name";

}

if ((strheaderval == "") || (strheaderval.Trim() == "details") || (strheaderval.Trim() == "ntfs") || (strheaderval.Trim() == "action") || (strheaderval.Trim() == "migrated"))

{

flagheader = true;

flagvalidcol = false;

break;

}

else if (strheaderval.Trim() == strCellValue.Trim())

{

Console.WriteLine("Column " + header[i] + " found", 0);

flagvalidcol = true;

flagheader = true;

for (k = 1; k < ArrayData.Length; k++)

{

String[] arrresult = ArrayData[k].Split(stringSeparators, StringSplitOptions.None);

strCellValue2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

strCellValue2 = strCellValue2.ToLower();

string strArrresult = arrresult[i].ToLower();

if ((header[i].Trim() == "Created") || (header[i] == "Log Date") || (header[i] == "LogDate") || (header[i] == "LastUpdatedDate"))

{

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in strArrresult)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent.Append(ch);

}

}

//DateTime Dtdel = DateTime.ParseExact(strArrresult, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtdel = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i].Trim() == "NextRun" && strArrresult.Trim() != "")

{

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in strArrresult)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent.Append(ch);

}

}

DateTime Dtdel = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

Dtcelval = TimeZoneInfo.ConvertTimeToUtc(Dtcelval);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i] == "Initial Scan Date")

{

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in strArrresult)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent.Append(ch);

}

}

DateTime Dtdel = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

// Dtcelval = TimeZoneInfo.ConvertTimeToUtc(Dtcelval);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i].Trim() == "Compliance %" || header[i].Trim() == "Free (%)" || header[i].Trim() == "Used (%)")

{

// double DblValue = double.ParseDouble(\_xlWorkSheet.Cells[k + 1, j].Value2());

//string Value2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

double DblValue = double.Parse(\_xlWorkSheet.Cells[k + 1, j].Value2.ToString());

DblValue = Math.Truncate(DblValue \* 1000) / 1000;

strCellValue2 = String.Format("{0:N2}", DblValue);

}

//if (\_Function.Compare("Cell value in " + header[i] + " Row no " + (k + 1), strCellValue2.Trim(), strArrresult.Trim(), false) == false)

strCellValue2 = string.Join("", strCellValue2.Split(default(string[]), StringSplitOptions.RemoveEmptyEntries));

strArrresult = string.Join("", strArrresult.Split(default(string[]), StringSplitOptions.RemoveEmptyEntries));

Assert.AreEqual(strCellValue2, strArrresult);

Console.WriteLine("Cell value \"" + strArrresult.Trim() + "\" and Excel Cell value \"" + strCellValue2.Trim() + "\" are compared");

}

Console.WriteLine("Values in Column " + header[i] + " verified", 0);

break;

}

}

}

if (flagvalidcol == true)

{

if (flagheader == true)

{

Console.WriteLine("Column " + header[i] + " found and verified");

}

else

{

Console.WriteLine("Column " + header[i] + "not found");

Assert.Fail();

}

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

}

public void VerifyExcel\_Global\_AccountPage(string FileName, string[] ArrayData)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(FileName, 0, true);//5, "", "", true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

\_xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)\_xlWorkBook.Worksheets.get\_Item(1);

String strCellValue, strCellValue2, strheaderval;

int i, j, k;

bool flagheader, flagvalidcol;

//: int rownum = 1;

int intFailCnt = 0;

int ArraySize = ArrayData.Length;

string[] stringSeparators = new string[] { "^#$@~" };

//String[] SplitWrds = EleEnabled.Split(' ');

String[] header = ArrayData[0].Split(stringSeparators, StringSplitOptions.None);

int intheadercnt = header.Length;

for (i = 0; i < intheadercnt; i++)

{

flagheader = false;

flagvalidcol = false;

for (j = 1; j <= intheadercnt; j++)

{

strCellValue = \_xlWorkSheet.Cells[1, j].Value2.ToString();

if (strCellValue != null)

{

strCellValue = strCellValue.ToLower();

strheaderval = header[i].ToLower();

if ((strheaderval == "") || (strheaderval == " details"))

{

flagheader = true;

flagvalidcol = false;

break;

}

else if (strheaderval.Trim() == strCellValue.Trim())

{

Console.WriteLine("Column header" + header[i], "Column " + header[i] + " found", 0);

flagvalidcol = true;

flagheader = true;

for (k = 1; k < ArrayData.Length; k++)

{

String[] arrresult = ArrayData[k].Split(stringSeparators, StringSplitOptions.None);

strCellValue2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

strCellValue2 = strCellValue2.ToLower();

string strArrresult = arrresult[i].ToLower();

if ((header[i].Trim() == "Created") || (header[i] == "Log Date") || (header[i] == "LogDate"))

{

//DateTime Dtdel = DateTime.ParseExact(strArrresult, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if ((header[i].Trim() == "NextRun") || (header[i] == "LogDate"))

{

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

Dtcelval = TimeZoneInfo.ConvertTimeToUtc(Dtcelval);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i].Trim() == "Compliance %" || header[i].Trim() == "Free (%)" || header[i].Trim() == "Used (%)")

{

double DblValue = double.Parse(\_xlWorkSheet.Cells[k + 1, j].Value2());

DblValue = Math.Truncate(DblValue \* 1000) / 1000;

strCellValue2 = String.Format("{0:N2}", DblValue);

}

//if (\_Function.Compare("Cell value in " + header[i] + " Row no " + (k + 1), strCellValue2.Trim(), strArrresult.Trim(), false) == false)

Assert.AreEqual(strCellValue2.Trim(), strArrresult.Trim());

Console.WriteLine("Cell value \"" + strArrresult.Trim() + "\" and Excel Cell value \"" + strCellValue2.Trim() + "\" are compared");

}

Console.WriteLine("Column header" + header[i], "Values in Column " + header[i] + " verified", 0);

break;

}

}

}

if (flagvalidcol == true)

{

Console.WriteLine("Column header" + header[i], "Column " + header[i] + " found and verified");

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

}

public void VerifyExcel\_PrintQueueHitory(string FileName, string[] ArrayData)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(FileName, 0, true);//5, "", "", true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

\_xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)\_xlWorkBook.Worksheets.get\_Item(1);

String strCellValue, strCellValue2, strheaderval;

int i, j, k;

bool flagheader, flagvalidcol;

//: int rownum = 1;

int intFailCnt = 0;

int ArraySize = ArrayData.Length;

string[] stringSeparators = new string[] { "^#$@~" };

//String[] SplitWrds = EleEnabled.Split(' ');

String[] header = ArrayData[0].Split(stringSeparators, StringSplitOptions.None);

int intheadercnt = header.Length;

for (i = 0; i < intheadercnt; i++)

{

flagheader = false;

flagvalidcol = true;

for (j = 1; j <= intheadercnt; j++)

{

strCellValue = \_xlWorkSheet.Cells[1, j].Value2.ToString();

if (strCellValue != null)

{

strCellValue = strCellValue.ToLower();

strheaderval = header[i].ToLower();

if (strheaderval == "queue")

{

strheaderval = "queuename";

}

else if (strheaderval == "ip address")

{

strheaderval = "ipaddr";

}

else if (strheaderval == "port name")

{

strheaderval = "portname";

}

if ((strheaderval == "") || (strheaderval.Trim() == "details") || (strheaderval.Trim() == "id") || (strheaderval.Trim() == "teamid"))

{

flagheader = true;

flagvalidcol = false;

break;

}

else if (strheaderval.Trim() == strCellValue.Trim())

{

Console.WriteLine("Column " + header[i] + " found", 0);

flagvalidcol = true;

flagheader = true;

for (k = 1; k < ArrayData.Length; k++)

{

String[] arrresult = ArrayData[k].Split(stringSeparators, StringSplitOptions.None);

strCellValue2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

strCellValue2 = strCellValue2.ToLower();

string strArrresult = arrresult[i].ToLower();

if ((header[i].Trim() == "Created") || (header[i] == "Log Date") || (header[i] == "LogDate"))

{

//DateTime Dtdel = DateTime.ParseExact(strArrresult, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in strArrresult)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent.Append(ch);

}

}

DateTime Dtdel = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

//if (\_Function.Compare("Cell value in " + header[i] + " Row no " + (k + 1), strCellValue2.Trim(), strArrresult.Trim(), false) == false)

Assert.AreEqual(strCellValue2.Trim(), strArrresult.Trim());

Console.WriteLine("Cell value \"" + strArrresult.Trim() + "\" and Excel Cell value \"" + strCellValue2.Trim() + "\" are compared");

}

Console.WriteLine("Values in Column " + header[i] + " verified", 0);

break;

}

}

}

if (flagvalidcol == true)

{

if (flagheader == true)

{

Console.WriteLine("Column " + header[i] + " found and verified");

}

else

{

Console.WriteLine("Column " + header[i] + "not found");

Assert.Fail();

}

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

}

public void VerifyExcel\_Security\_Group(string FileName, string[] ArrayData)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(FileName, 0, true);//5, "", "", true, Excel.XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);

\_xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)\_xlWorkBook.Worksheets.get\_Item(1);

String strCellValue, strCellValue2, strheaderval;

int i, j, k;

bool flagheader, flagvalidcol;

//: int rownum = 1;

int intFailCnt = 0;

int ArraySize = ArrayData.Length;

string[] stringSeparators = new string[] { "^#$@~" };

//String[] SplitWrds = EleEnabled.Split(' ');

String[] header = ArrayData[0].Split(stringSeparators, StringSplitOptions.None);

int intheadercnt = header.Length;

for (i = 0; i < intheadercnt; i++)

{

flagheader = false;

flagvalidcol = true;

for (j = 1; j <= intheadercnt; j++)

{

strCellValue = \_xlWorkSheet.Cells[1, j].Value2.ToString();

if (strCellValue != null)

{

strCellValue = strCellValue.ToLower();

strheaderval = header[i].ToLower();

if ((strheaderval == "") || (strheaderval == "group members") || (strheaderval.Trim() == "id") || (strheaderval.Trim() == "groupid") || (strheaderval.Trim() == "teamid") || (strheaderval.Trim() == "archivedate") || (strheaderval.Trim() == "archivedby") || (strheaderval.Trim() == "action"))

{

flagheader = true;

flagvalidcol = false;

break;

}

else if (strheaderval.Trim() == strCellValue.Trim())

{

Console.WriteLine("Column " + header[i] + " found", 0);

flagvalidcol = true;

flagheader = true;

for (k = 1; k < ArrayData.Length; k++)

{

String[] arrresult = ArrayData[k].Split(stringSeparators, StringSplitOptions.None);

strCellValue2 = \_xlWorkSheet.Cells[k + 1, j].Value2.ToString();

strCellValue2 = strCellValue2.ToLower();

string strArrresult = arrresult[i].ToLower();

if ((header[i].Trim() == "Created") || (header[i] == "Log Date") || (header[i] == "LogDate"))

{

//DateTime Dtdel = DateTime.ParseExact(strArrresult, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i].Trim() == "NextRun")

{

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

Dtcelval = TimeZoneInfo.ConvertTimeToUtc(Dtcelval);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i] == "Initial Scan Date")

{

DateTime Dtdel = DateTime.Parse(strArrresult, System.Globalization.CultureInfo.InvariantCulture);

Dtdel = TimeZoneInfo.ConvertTimeToUtc(Dtdel);

String Datedel = Dtdel.ToString("MM/dd/yyyy HH:mm:ss");

strArrresult = Datedel.ToLower();

//DateTime Dtcelval = DateTime.ParseExact(strCellValue2, "MM/dd/yyyy hh:mm:ss tt", System.Globalization.CultureInfo.InvariantCulture);

DateTime Dtcelval = DateTime.Parse(strCellValue2, System.Globalization.CultureInfo.InvariantCulture);

// Dtcelval = TimeZoneInfo.ConvertTimeToUtc(Dtcelval);

String Dtcelvalstr = Dtcelval.ToString("MM/dd/yyyy HH:mm:ss");

strCellValue2 = Dtcelvalstr.ToLower();

}

if (header[i].Trim() == "Compliance %" || header[i].Trim() == "Free (%)" || header[i].Trim() == "Used (%)")

{

double DblValue = double.Parse(\_xlWorkSheet.Cells[k + 1, j].Value2());

DblValue = Math.Truncate(DblValue \* 1000) / 1000;

strCellValue2 = String.Format("{0:N2}", DblValue);

}

//if (\_Function.Compare("Cell value in " + header[i] + " Row no " + (k + 1), strCellValue2.Trim(), strArrresult.Trim(), false) == false)

Assert.AreEqual(strCellValue2.Trim(), strArrresult.Trim());

Console.WriteLine("Cell value \"" + strArrresult.Trim() + "\" and Excel Cell value \"" + strCellValue2.Trim() + "\" are compared");

}

Console.WriteLine("Values in Column " + header[i] + " verified", 0);

break;

}

}

}

if (flagvalidcol == true)

{

if (flagheader == true)

{

Console.WriteLine("Column " + header[i] + " found and verified");

}

else

{

Console.WriteLine("Column " + header[i] + "not found");

Assert.Fail();

}

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

}

public void Verify\_Excel\_Directories(int GridColumnOffset, int ExcelColumnOffset, String StrFilepath)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(StrFilepath, 0, true);

\_xlWorkSheet = \_xlWorkBook.Worksheets.get\_Item(1);

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

int intNextPageDisabled = 0;

int Page = 1;

string GridCellContent = "";

string ExcelCellContent = "";

double Temp;

double Temp2;

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

while (intNextPageDisabled == 0)

{

for (int I = 1; I < uIJqGridTable.RowCount; I++)

{

for (int J = 1; J < uIJqGridTable.GetRow(1).GetChildren().Count - GridColumnOffset; J++)

{

if (uIJqGridTable.GetCell(I, J + GridColumnOffset).InnerText != null)

{

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString();

if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Size (MB)")

{

Temp = Math.Truncate(double.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString()) \* 1000) / 1000;

Temp2 = Math.Truncate(double.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText) \* 1000) / 1000;

ExcelCellContent = String.Format("{0:N2}", Temp);

GridCellContent = String.Format("{0:N2}", Temp2);

}

else if (header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Last Access Time"

|| header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Log Date"

|| header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Last Modified Time")

{

if (ExcelCellContent == "2/6/2012 10:30:17 AM")

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = ExcelDateTime.AddHours(1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

//else if (header.GetCell(0, J + GridColumnOffset).InnerText.Trim() == "Last Modified Time")

//{

// Console.WriteLine("Excel value: " + ExcelCellContent);

// if (ExcelCellContent == "2/6/2012 10:30:17 AM")

// {

// DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

// ExcelDateTime = ExcelDateTime.AddMonths(1);

// ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

// ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

// }

// else

// {

// DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

// ExcelDateTime = ExcelDateTime.AddHours(1);

// ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

// ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

// }

// DateTime GridDateTime = DateTime.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText, System.Globalization.CultureInfo.InvariantCulture);

// GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

// GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

//}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Total Offending files")

{

GridCellContent = "";

ExcelCellContent = "";

Console.WriteLine("Skipping hidden grid column: " + header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim());

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Total files & folders")

{

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), (J - 1)].Value2.ToString();

}

else

{ //Console.WriteLine("Column: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim()); Console.WriteLine("Else route.");

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), (J + ExcelColumnOffset)].Value2.ToString();

}

Console.WriteLine("Header: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim());

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Value: " + GridCellContent);

Console.WriteLine("Excel Row: " + (I + 1 + ((Page - 1) \* 20)) + " Column " + (J + ExcelColumnOffset) + ": " + " Value: " + ExcelCellContent);

Assert.AreEqual(GridCellContent.ToString().Trim().ToLower(), ExcelCellContent.ToString().Trim().ToLower());

}

else

{

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Cell is empty.");

}

}

}

intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

Page++;

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

Console.WriteLine("Excel export successfully compared with grid table.");

}

public void Verify\_Excel\_Files(int GridColumnOffset, int ExcelColumnOffset, String StrFilepath)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(StrFilepath, 0, true);

\_xlWorkSheet = \_xlWorkBook.Worksheets.get\_Item(1);

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders1;

int intNextPageDisabled = 0;

int Page = 1;

string GridCellContent = "";

string ExcelCellContent = "";

double Temp;

double Temp2;

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

while (intNextPageDisabled == 0)

{

for (int I = 1; I < uIJqGridTable.RowCount; I++)

{

for (int J = 1; J < uIJqGridTable.GetRow(1).GetChildren().Count - GridColumnOffset; J++)

{

if (uIJqGridTable.GetCell(I, J + GridColumnOffset).InnerText != null)

{

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString();

if (J + GridColumnOffset == 6 || J + GridColumnOffset == 7)

{

if (ExcelCellContent == "7/14/2009 6:14:46 AM")

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = ExcelDateTime.AddHours(1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2, System.Globalization.CultureInfo.InvariantCulture);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.CurrentCulture);

GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else if (J + GridColumnOffset == 4)

{

Temp = Math.Truncate(double.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J + ExcelColumnOffset].Value2.ToString()) \* 1000) / 1000;

Temp2 = Math.Truncate(double.Parse(uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText) \* 1000) / 1000;

ExcelCellContent = String.Format("{0:N2}", Temp);

GridCellContent = String.Format("{0:N2}", Temp2);

}

else

{ //Console.WriteLine("Column: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim()); Console.WriteLine("Else route.");

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), (J + ExcelColumnOffset)].Value2.ToString();

}

Console.WriteLine("Header: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim());

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Value: " + GridCellContent);

Console.WriteLine("Excel Row: " + (I + 1 + ((Page - 1) \* 20)) + " Column " + (J + ExcelColumnOffset) + ": " + " Value: " + ExcelCellContent);

Assert.AreEqual(ExcelCellContent.ToString().Trim().ToLower(), GridCellContent.ToString().Trim().ToLower());

}

else

{

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Cell is empty.");

}

}

}

intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

Page++;

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

Console.WriteLine("Excel export successfully compared with grid table.");

}

public void Verify\_Report\_Requests\_Excel(int GridColumnOffset, String StrFilepath)

{

\_xlApp = new Microsoft.Office.Interop.Excel.Application();

\_xlWorkBook = \_xlApp.Workbooks.Open(StrFilepath, 0, true);

\_xlWorkSheet = \_xlWorkBook.Worksheets.get\_Item(1);

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

int Page = 1;

string GridCellContent = "";

string ExcelCellContent = "";

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

for (int I = 1; (I < uIJqGridTable.RowCount && I < 6); I++)

{

for (int J = 1; J < uIJqGridTable.GetRow(1).GetChildren().Count - GridColumnOffset; J++)

{

if (uIJqGridTable.GetCell(I, J + GridColumnOffset).InnerText != null)

{

//GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

//ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J].Value2.ToString();

if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "UoWType")

{

GridCellContent = "";

ExcelCellContent = "";

Console.WriteLine("Skipping hidden grid column: " + header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim());

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Status")

{

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J - 1].Value2.ToString();

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "File Name")

{

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J - 1].Value2.ToString();

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Requested (UTC)")

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J - 1].Value2, System.Globalization.CultureInfo.InvariantCulture);

//ExcelDateTime = ExcelDateTime.AddHours(-1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

Console.WriteLine("NewGridCellContent: " + NewGridCellContent);

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Started (UTC)")

{

if (uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Contains("\\") == true)

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J - 1].Value2, System.Globalization.CultureInfo.InvariantCulture);

//ExcelDateTime = ExcelDateTime.AddHours(-1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Finished (UTC)")

{

if (uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText.Contains("\\") == true)

{

DateTime ExcelDateTime = DateTime.Parse(\_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J - 1].Value2, System.Globalization.CultureInfo.InvariantCulture);

//ExcelDateTime = ExcelDateTime.AddHours(-1);

ExcelDateTime = TimeZoneInfo.ConvertTimeToUtc(ExcelDateTime);

ExcelCellContent = ExcelDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

String uIJqGridTableTemp = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

Console.WriteLine("uIJqGridTableTemp: " + uIJqGridTableTemp);

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

Console.WriteLine("NewGridCellContent: " + NewGridCellContent.ToString());

DateTime GridDateTime = DateTime.ParseExact(NewGridCellContent.ToString().Trim(), "MM/dd/yyyy HH:mm:ss", System.Globalization.CultureInfo.InvariantCulture);

//GridDateTime = TimeZoneInfo.ConvertTimeToUtc(GridDateTime);

GridCellContent = GridDateTime.ToString("MM/dd/yyyy HH:mm:ss").ToLower();

}

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "Actions")

{

GridCellContent = "";

ExcelCellContent = "";

Console.WriteLine("Skipping hidden grid column: " + header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim());

}

else if (header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim() == "UserReportRequestId")

{

GridCellContent = "";

ExcelCellContent = "";

Console.WriteLine("Skipping hidden grid column: " + header.GetCell(0, (J + GridColumnOffset)).InnerText.Trim());

}

else

{ //Console.WriteLine("Column: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim()); Console.WriteLine("Else route.");

GridCellContent = uIJqGridTable.GetCell(I, (J + GridColumnOffset)).InnerText;

ExcelCellContent = \_xlWorkSheet.Cells[(I + 1 + ((Page - 1) \* 20)), J].Value2.ToString();

}

Console.WriteLine("Header: " + header.GetCell(0, J + GridColumnOffset).InnerText.Trim());

Console.WriteLine("Table Row: " + I + " Column: " + (J + GridColumnOffset) + " Value: " + GridCellContent);

Console.WriteLine("Excel Row: " + (I + 1 + ((Page - 1) \* 20)) + " Column " + J + ": " + " Value: " + ExcelCellContent);

Console.WriteLine("Excel: " + ExcelCellContent);

Console.WriteLine("Grid: " + GridCellContent);

Assert.AreEqual(ExcelCellContent.ToString().Trim().ToLower(), GridCellContent.ToString().Trim().ToLower());

}

else

{

Console.WriteLine("Table Page: " + Page + " Row: " + I + " Column: " + (J + GridColumnOffset) + " Cell is empty.");

}

}

}

//Close excel

\_xlWorkBook.Close(false); // misValue, misValue);

\_xlApp.Quit();

Console.WriteLine("First 10 records from excel export successfully compared with grid table.");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//A method to sort the search results:

//Variables:

// intStrtRow = Row from which the serarch is to be started

// intEndrows = No of Rows before which search is to be stopped

// intResultCol = Column to be verified.

// strSortItemType = Data type of the field to be sorted eg: Date / String / Int

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void Verify\_Sort(int intStrtRow, int intEndrows, int ColumnNumber, string strSortItemType)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

//int intNextPageDisabled = 0;

int intRetValue = 0;

int intfailcnt = 0;

string StrCellContent, StrCellContent1;

//while (intNextPageDisabled == 0)

//{

for (int I = 2; I < uIJqGridTable.RowCount && I < 6; I++)

{

if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I - 1, ColumnNumber);

HtmlRow row2 = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

StrCellContent1 = row2.GetCell(ColumnNumber).InnerText.ToString();

switch (strSortItemType)

{

case "Date":

DateTime Date1 = DateTime.Parse(StrCellContent);

DateTime Date2 = DateTime.Parse(StrCellContent1);

intRetValue = DateTime.Compare(Date1, Date2);

break;

case "String":

intRetValue = StrCellContent.CompareTo(StrCellContent1);

Console.WriteLine("Row: " + I + ". Result (" + StrCellContent + "/" + StrCellContent1 + "): " + intRetValue);

break;

}

if (intRetValue < 0)

{

intfailcnt = intfailcnt + 1;

}

}

else

{

StrCellContent = uIJqGridTable.GetCell((I - 1), ColumnNumber).InnerText.ToString();

StrCellContent1 = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

switch (strSortItemType)

{

case "Date":

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in StrCellContent)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent.Append(ch);

}

}

DateTime Date1 = DateTime.Parse(NewGridCellContent.ToString());

System.Text.StringBuilder NewGridCellContent1 = new System.Text.StringBuilder();

foreach (var ch in StrCellContent1)

{

if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true)

{

NewGridCellContent1.Append(ch);

}

}

DateTime Date2 = DateTime.Parse(NewGridCellContent1.ToString());

intRetValue = DateTime.Compare(Date1, Date2);

break;

case "String":

intRetValue = StrCellContent.CompareTo(StrCellContent1);

Console.WriteLine("Row: " + I + ". Result (" + StrCellContent + "/" + StrCellContent1 + "): " + intRetValue);

break;

}

if (intRetValue < 0)

{

intfailcnt = intfailcnt + 1;

}

}

//}

//intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

}

if (intfailcnt == 0)

{

Console.WriteLine("Sort order verified successfully.");

}

else

{

Console.WriteLine("Sort order verification failed.");

}

Assert.AreEqual(intfailcnt, 0, "Sort order not as expected.");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//A method to sort the search results:

//Variables:

// intStrtRow = Row from which the serarch is to be started

// intEndrows = No of Rows before which search is to be stopped

// intResultCol = Column to be verified.

// strSortItemType = Data type of the field to be sorted eg: Date / String / Int

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void Verify\_Sort\_Asc(int intStrtRow, int intEndrows, int ColumnNumber, string strSortItemType)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

//int intNextPageDisabled = 0;

int intRetValue = 0;

int intfailcnt = 0;

string StrCellContent, StrCellContent1;

//while (intNextPageDisabled == 0)

//{

for (int I = 2; I < uIJqGridTable.RowCount && I < 6; I++)

{

if (uIJqGridTable.GetCell(I, ColumnNumber) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(I - 1, ColumnNumber);

HtmlRow row2 = (HtmlRow)uIJqGridTable.GetCell(I, ColumnNumber);

StrCellContent = row.GetCell(ColumnNumber).InnerText.ToString();

StrCellContent1 = row2.GetCell(ColumnNumber).InnerText.ToString();

switch (strSortItemType)

{

case "Date":

DateTime Date1 = DateTime.Parse(StrCellContent);

DateTime Date2 = DateTime.Parse(StrCellContent1);

intRetValue = DateTime.Compare(Date1, Date2);

break;

case "String":

intRetValue = StrCellContent.CompareTo(StrCellContent1);

Console.WriteLine("Row: " + I + ". Result (" + StrCellContent + "/" + StrCellContent1 + "): " + intRetValue);

break;

}

if (intRetValue > 0)

{

intfailcnt = intfailcnt + 1;

}

}

else

{

StrCellContent = uIJqGridTable.GetCell((I - 1), ColumnNumber).InnerText.ToString();

StrCellContent1 = uIJqGridTable.GetCell(I, ColumnNumber).InnerText.ToString();

switch (strSortItemType)

{

case "Date":

DateTime Date1 = DateTime.Parse(StrCellContent);

DateTime Date2 = DateTime.Parse(StrCellContent1);

intRetValue = DateTime.Compare(Date1, Date2);

break;

case "String":

intRetValue = StrCellContent.CompareTo(StrCellContent1);

Console.WriteLine("Row: " + I + ". Result (" + StrCellContent + "/" + StrCellContent1 + "): " + intRetValue);

break;

}

if (intRetValue > 0)

{

intfailcnt = intfailcnt + 1;

}

}

//}

//intNextPageDisabled = intNextPageDisabled + BrowseTableNextButtonEnabled();

}

if (intfailcnt == 0)

{

Console.WriteLine("Sort order verified successfully.");

}

else

{

Console.WriteLine("Sort order verification failed.");

}

Assert.AreEqual(intfailcnt, 0, "Sort order not as expected.");

}

/// <summary>

/// Sort\_by\_Sharename\_column

/// </summary>

public void Sort\_by\_Sharename\_column()

{

#region Variable Declarations

HtmlDiv uISharenamePane = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UISharenamePane;

#endregion

// Click 'Sharename' pane

Mouse.Click(uISharenamePane);

Verify\_Sort(2, 0, 2, "String");

}

/// <summary>

/// Sort\_by\_Share\_column

/// </summary>

public void Sort\_by\_Share\_column()

{

#region Variable Declarations

HtmlDiv Column\_Share = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Share;

#endregion

// Click 'Share'

Mouse.Click(Column\_Share);

}

/// <summary>

/// Sort\_by\_DFS\_Hostname\_column

/// </summary>

public void Sort\_by\_DFS\_LinkShare\_column()

{

#region Variable Declarations

HtmlDiv uISharenamePane = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIDFSLinkShare;

#endregion

// Click 'Sharename' pane

Mouse.Click(uISharenamePane);

Mouse.Click(uISharenamePane);

Verify\_Sort(2, 0, 4, "String");

}

/// <summary>

/// Sort\_NTFS\_by\_Folder\_Path\_column

/// </summary>

public void Sort\_NTFS\_by\_Folder\_Path()

{

#region Variable Declarations

HtmlDiv Column\_Folder\_Path = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Folder\_Path;

#endregion

// Click 'Sharename' pane

Mouse.Click(Column\_Folder\_Path);

Mouse.Click(Column\_Folder\_Path);

Verify\_Sort(2, 0, 4, "String");

}

/// <summary>

/// Sort\_Share\_Compliance\_by\_Folder\_Path\_column

/// </summary>

public void Sort\_Share\_Compliance\_by\_Share\_Name()

{

#region Variable Declarations

HtmlDiv Column\_Share\_Name = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Share\_Name;

#endregion

// Click 'Sharename' pane

Mouse.Click(Column\_Share\_Name);

Verify\_Sort(2, 0, 2, "String");

}

/// <summary>

/// Sort\_Share\_Compliance\_by\_ShareComplianceError

/// </summary>

public void Sort\_Share\_Compliance\_by\_Share\_ComplianceErrror()

{

#region Variable Declarations

HtmlDiv Column\_ShareComplianceError = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_ShareComplianceError;

#endregion

// Click 'Sharename' pane

Mouse.Click(Column\_ShareComplianceError);

Verify\_Sort(2, 0, 3, "String");

}

/// <summary>

/// Sort\_Owners\_by\_Domain\_column

/// </summary>

public void Sort\_Owners\_by\_Domain()

{

#region Variable Declarations

HtmlDiv Column\_Domain = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Domain;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Domain);

Mouse.Click(Column\_Domain);

Verify\_Sort(2, 0, 4, "String");

}

/// <summary>

/// Sort\_Global\_Compliance\_by\_TeamName\_column

/// </summary>

public void Sort\_Global\_Compliance\_by\_TeamName()

{

#region Variable Declarations

HtmlDiv Column\_TeamName = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_TeamName;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_TeamName);

Mouse.Click(Column\_TeamName);

Verify\_Sort(2, 0, 1, "String");

}

public void Sort\_Global\_Compliance\_by\_TeamName\_Asc()

{

#region Variable Declarations

HtmlDiv Column\_TeamName = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_TeamName;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_TeamName);

//Mouse.Click(Column\_TeamName);

}

public void Sort\_Global\_Compliance\_by\_Compliance()

{

#region Variable Declarations

HtmlDiv Column\_Compliance = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Compliance;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Compliance);

Mouse.Click(Column\_Compliance);

Verify\_Sort\_Asc(2, 0, 2, "String");

}

/// <summary>

/// Sort\_Global\_Account\_by\_Initial\_Scan\_Date\_column

/// </summary>

public void Sort\_Global\_Account\_by\_Initial\_Scan\_Date\_column()

{

#region Variable Declarations

HtmlDiv Column\_Initial\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Initial\_Scan\_Date;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_Initial\_Scan\_Date);

Mouse.Click(Column\_Initial\_Scan\_Date);

if(Run\_Environment == "OST\_DEV")

Verify\_Sort(2, 0, 3, "String");

else

Verify\_Sort(2, 0, 8, "Date");

}

public void Sort\_Cost\_Code\_by\_Share\_column()

{

#region Variable Declarations

HtmlDiv Column\_Share = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Share;

#endregion

//string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_Share);

Mouse.Click(Column\_Share);

Verify\_Sort(2, 0, 3, "String");

}

public void Sort\_Cost\_Code\_by\_Owner\_column()

{

#region Variable Declarations

HtmlDiv Column\_Owner = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Owner;

#endregion

//string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_Owner);

Mouse.Click(Column\_Owner);

Verify\_Sort(2, 0, 5, "String");

}

public void Sort\_Global\_Account\_by\_Scan\_Date\_column()

{

#region Variable Declarations

HtmlDiv Column\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Scan\_Date;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

Mouse.Click(Column\_Scan\_Date);

Mouse.Click(Column\_Scan\_Date);

Verify\_Sort(2, 0, 8, "Date");

}

public void Sort\_NTFS\_by\_Scan\_Date\_column()

{

#region Variable Declarations

HtmlDiv Column\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Scan\_Date;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

Mouse.Click(Column\_Scan\_Date);

//Mouse.Click(Column\_Scan\_Date);

Verify\_Sort(2, 0, 11, "Date");

}

/// <summary>

/// Sort\_Print\_Queue\_by\_Share\_Name\_column

/// </summary>

public void Sort\_Print\_Queue\_by\_Share\_Name\_column()

{

#region Variable Declarations

HtmlDiv Column\_Share\_Name = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Share\_Name;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_Share\_Name);

Mouse.Click(Column\_Share\_Name);

Verify\_Sort(2, 0, 4, "String");

}

/// <summary>

/// Sort\_Print\_Queue\_by\_IP\_Address\_column

/// </summary>

public void Sort\_Print\_Queue\_by\_IP\_Address\_column()

{

#region Variable Declarations

HtmlDiv Column\_IP\_Address = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_IP\_Address;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_IP\_Address);

Mouse.Click(Column\_IP\_Address);

Verify\_Sort(2, 0, 6, "String");

}

public void Sort\_Print\_Queue\_by\_IP\_Address\_column\_Acsc()

{

#region Variable Declarations

HtmlDiv Column\_IP\_Address = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_IP\_Address;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Domain' pane

Mouse.Click(Column\_IP\_Address);

}

/// <summary>

/// Sort\_Print\_Queue\_History\_by\_Queue\_column

/// </summary>

public void Sort\_Print\_Queue\_History\_by\_Queue\_column()

{

#region Variable Declarations

HtmlDiv Column\_Queue = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Queue;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Queue);

Mouse.Click(Column\_Queue);

Verify\_Sort(2, 0, 3, "String");

}

/// <summary>

/// Sort\_Print\_Queue\_Statistics\_by\_Queue\_column

/// </summary>

public void Sort\_Print\_Queue\_Statistics\_by\_Queue\_column()

{

#region Variable Declarations

HtmlDiv Column\_Queue = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Queue;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Queue);

Mouse.Click(Column\_Queue);

Verify\_Sort(2, 0, 1, "String");

}

/// <summary>

/// Sort\_Chargeback\_Cost\_Code\_Page\_by\_Volume\_column

/// </summary>

public void Sort\_Chargeback\_Cost\_Code\_Page\_by\_Volume\_column()

{

#region Variable Declarations

HtmlDiv Column\_Volume = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Volume;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Volume);

Mouse.Click(Column\_Volume);

Verify\_Sort(2, 0, 2, "String");

}

/// <summary>

/// Sort\_Datastats\_Overview\_Page\_by\_Partition\_column

/// </summary>

public void Sort\_Datastats\_Overview\_Page\_by\_Partition\_column()

{

#region Variable Declarations

HtmlDiv Column\_Partition = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Partition;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Partition);

Mouse.Click(Column\_Partition);

Verify\_Sort(2, 0, 1, "String");

}

/// <summary>

/// Sort\_Datastats\_File\_Usage\_DLP\_Page\_by\_Total\_column

/// </summary>

public void Sort\_Datastats\_File\_Usage\_DLP\_Page\_by\_Total\_column()

{

#region Variable Declarations

HtmlDiv Column\_Total = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Total;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Total);

Mouse.Click(Column\_Total);

Verify\_Sort(2, 0, 4, "String");

}

/// <summary>

/// Sort\_Anti\_Virus\_Page\_by\_Host\_column

/// </summary>

public void Sort\_Anti\_Virus\_Page\_by\_Host\_column()

{

#region Variable Declarations

HtmlDiv Column\_Host = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Host;

#endregion

// Click 'Domain' pane

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

Mouse.Click(Column\_Host);

//Mouse.Click(Column\_Host); //- default sort on AntiVirus page is by Host so only need to click column once.

Verify\_Sort(2, 0, 1, "String");

}

public void Sort\_Anti\_Virus\_Page\_by\_Host\_column\_Asc()

{

#region Variable Declarations

HtmlDiv Column\_Host = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Host;

#endregion

// Click 'Domain' pane

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Mouse.Click(Column\_Host);

Mouse.Click(Column\_Host);

}

/// <summary>

/// Sort\_Anti\_Virus\_Page\_by\_Host\_column

/// </summary>

public void Sort\_Anti\_Virus\_Page\_by\_Date\_column\_PROD()

{

#region Variable Declarations

HtmlDiv Column\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Date;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Date);

//Verify\_Sort(2, 0, 0, "String");

}

/// <summary>

/// Sort\_Security\_Check\_by\_Host\_column

/// </summary>

public void Sort\_Security\_Check\_by\_Host\_column()

{

#region Variable Declarations

HtmlDiv Column\_Version = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Version;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Version);

Mouse.Click(Column\_Version);

Verify\_Sort(2, 0, 1, "String");

}

public void Sort\_Sharepoint\_Info\_by\_Host\_column()

{

#region Variable Declarations

HtmlDiv Column\_Host = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Host;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Host);

Verify\_Sort(2, 0, 0, "String");

}

public void Sort\_Sharepoint\_Info\_by\_LastUpdatedDate\_column()

{

#region Variable Declarations

HtmlDiv Column\_LastUpdatedDate = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_LastUpdatedDate;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_LastUpdatedDate);

Mouse.Click(Column\_LastUpdatedDate);

}

/// <summary>

/// Sort\_Local\_Users\_by\_UserID\_column

/// </summary>

public void Sort\_Local\_Users\_by\_UserID\_column()

{

#region Variable Declarations

HtmlDiv Column\_UserID = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_UserID;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_UserID);

Playback.Wait(1000);

Mouse.Click(Column\_UserID);

Verify\_Sort(2, 0, 3, "String");

}

/// <summary>

/// Sort\_Local\_Groups\_by\_MemberAccount\_column

/// </summary>

public void Sort\_Local\_Groups\_by\_MemberAccount\_column()

{

#region Variable Declarations

HtmlDiv Column\_MemberAccount = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_MemberAccount;

#endregion

// Click 'MemberAccount' column

//Mouse.Click(Column\_MemberAccount); - column already clicked once in Search step.

Mouse.Click(Column\_MemberAccount);

Verify\_Sort(2, 0, 5, "String");

}

/// <summary>

/// Sort\_User\_Rights\_by\_Rights\_Trustee\_column

/// </summary>

public void Sort\_User\_Rights\_by\_Rights\_Trustee\_column()

{

#region Variable Declarations

HtmlDiv Column\_Rights\_trustee = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Rights\_trustee;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Rights\_trustee);

Mouse.Click(Column\_Rights\_trustee);

Verify\_Sort(2, 0, 2, "String");

}

public void Sort\_User\_Rights\_by\_LogDate\_column()

{

#region Variable Declarations

HtmlDiv Column\_LogDate = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_LogDate;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_LogDate);

Mouse.Click(Column\_LogDate);

// Verify\_Sort(2, 0, 3, "String");

}

public void Sort\_by\_Created\_column()

{

#region Variable Declarations

HtmlDiv Column\_Created = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Created;

#endregion

Mouse.Click(Column\_Created);

Mouse.Click(Column\_Created);

}

/// <summary>

/// Sort\_Orphans\_by\_UserID\_column

/// </summary>

public void Sort\_Orphans\_by\_UserID\_column()

{

#region Variable Declarations

HtmlDiv Column\_UserID = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_UserID;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_UserID);

Mouse.Click(Column\_UserID);

Verify\_Sort(2, 0, 3, "String");

}

/// <summary>

/// Sort\_Server\_Scheduled\_Tasks\_by\_TaskName\_column

/// </summary>

public void Sort\_Server\_Scheduled\_Tasks\_by\_TaskName\_column()

{

#region Variable Declarations

HtmlDiv Column\_TaskName = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_TaskName;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_TaskName);

Verify\_Sort(2, 0, 1, "String");

}

/// <summary>

/// Sort\_Host\_Maintenance\_by\_Host\_Name\_column

/// </summary>

public void Sort\_Host\_Maintenance\_by\_Host\_Name\_column()

{

#region Variable Declarations

HtmlDiv Column\_Host\_Name = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Host\_Name;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Host\_Name);

Verify\_Sort(2, 0, 1, "String");

}

/// <summary>

/// Sort\_Host\_Maintenance\_by\_Host\_Name\_column

/// </summary>

public void Sort\_Host\_Maintenance\_by\_Division\_column()

{

#region Variable Declarations

HtmlDiv Column\_Division = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Division;

#endregion

// Click 'Domain' pane

Mouse.Click(Column\_Division);

Verify\_Sort(2, 0, 3, "String");

}

/// <summary>

/// Perform\_search\_on\_Dfs\_Shares\_page - Use 'Perform\_search\_on\_Dfs\_Shares\_pageParams' to pass parameters into this method.

/// </summary>

public void Perform\_search\_on\_Dfs\_Shares\_page(String Parameter)

{

#region Variable Declarations

HtmlEdit uIHostNameEdit = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIHostNameEdit;

HtmlButton uISearchButton1 = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UISearchButton;

HtmlCustom vmdd1dbb2c = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.vmdd1dbb2c;

HtmlEdit uIItemEdit = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemEdit;

HtmlComboBox uIItemComboBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemComboBox;

HtmlCheckBox uIItemCheckBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemCheckBox;

HtmlDiv Column\_Initial\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Initial\_Scan\_Date;

#endregion

//Keyboard.SendKeys(uIItemComboBox, "{Down}" + "{Down}" + "{Down}");

//Sort by column before exporting

Mouse.Click(Column\_Initial\_Scan\_Date);

uIItemComboBox.SelectedItem = "DFS Shares";

uIItemEdit.Text = Parameter;

// Click 'Search' button

Console.WriteLine("Clicking Search button");

Mouse.Click(uISearchButton1);

}

public virtual Perform\_search\_on\_Dfs\_Shares\_pageParams Perform\_search\_on\_Dfs\_Shares\_pageParams

{

get

{

if ((this.mPerform\_search\_on\_Dfs\_Shares\_pageParams == null))

{

this.mPerform\_search\_on\_Dfs\_Shares\_pageParams = new Perform\_search\_on\_Dfs\_Shares\_pageParams();

}

return this.mPerform\_search\_on\_Dfs\_Shares\_pageParams;

}

}

private Perform\_search\_on\_Dfs\_Shares\_pageParams mPerform\_search\_on\_Dfs\_Shares\_pageParams;

/// <summary>

/// Export\_to\_Excel - Use 'Export\_to\_ExcelParams' to pass parameters into this method.

/// </summary>

public void Export\_to\_Excel(int GridColumnOffset, int ExcelColumnOffset)

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

string SaveAsShown = "No";

Playback.PlaybackSettings.SearchTimeout = 5000;

for (int i = 0; i < 30; i++)

{

try

{

if (uIII11NotificationBar.Exists == true)

{

SaveAsShown = "Yes";

i = 30;

}

}

catch {}

Playback.Wait(1000);

}

Playback.PlaybackSettings.ResetToDefault();

Assert.AreEqual("Yes", SaveAsShown, "Save-As dialog was not shown.");

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

string SaveAsShown = "No";

Playback.PlaybackSettings.SearchTimeout = 5000;

for (int i = 0; i < 30; i++)

{

try

{

if (uISaveButton.Exists == true)

{

SaveAsShown = "Yes";

i = 30;

}

}

catch { }

Playback.Wait(1000);

}

Playback.PlaybackSettings.ResetToDefault();

Assert.AreEqual("Yes", SaveAsShown, "Save-As dialog was not shown.");

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Verify\_Excel(GridColumnOffset, ExcelColumnOffset, StrFilepath);

}

public void Export\_to\_Excel\_DfsShares(int GridColumnOffset, int ExcelColumnOffset)

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

HtmlDiv Column\_Initial\_Scan\_Date = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Initial\_Scan\_Date;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

Mouse.Click(Column\_Initial\_Scan\_Date);

Mouse.Click(Column\_Initial\_Scan\_Date);

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

VerifyExcel2(StrFilepath, ArrayTableData);

}

public void Export\_to\_Excel\_2()

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Playback.Wait(5000);

//Console.WriteLine("Report saved as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

VerifyExcel2(StrFilepath, ArrayTableData);

}

public void Export\_to\_Excel\_PrintQueue\_History()

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Playback.Wait(5000);

//Console.WriteLine("Report saved as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

VerifyExcel\_PrintQueueHitory(StrFilepath, ArrayTableData);

}

public void Export\_to\_Excel\_Security\_Group()

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Playback.Wait(5000);

//Console.WriteLine("Report saved as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

VerifyExcel\_Security\_Group(StrFilepath, ArrayTableData);

}

public void Export\_to\_Excel\_Antivirus(int GridColumnOffset, int ExcelColumnOffset)

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

//Verify\_Excel(GridColumnOffset, ExcelColumnOffset, StrFilepath);

VerifyExcel2(StrFilepath, ArrayTableData);

}

public void Export\_to\_Excel\_DataStatsOverview()

{

#region Variable Declarations

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

//Fetch data from application.

String[] ArrayTableData = Flow\_GetTableData();

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Playback.Wait(5000);

//Console.WriteLine("Report saved as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

VerifyExcel\_DataStatsOverview(StrFilepath, ArrayTableData);

}

public String[] Flow\_GetTableData()

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

int j = 0;

int Counter = 0;

//String[] HeaderArray = new String[header.CellCount];

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

int RowCnt = uIJqGridTable.RowCount;

int ColCnt = uIJqGridTable.GetRow(1).GetChildren().Count;

//IList<IWebElement> rowCollections = table.FindElements(By.TagName("tr"));

//int RowCnt = rowCollections.Count;

String[] DataArray = new String[RowCnt];

UITestControlCollection utc = uIJqGridTable.Rows;

foreach (HtmlRow row in utc)

{

Counter++;

if (Counter == 1)

{

for (int k = 0; k < header.CellCount; k++)

{

String HeaderText = header.GetCell(0, k).InnerText.Trim().ToString();

Console.WriteLine("Header " + k + ": " + HeaderText.ToString());

DataArray[j] = DataArray[j] + "^#$@~" + HeaderText;

}

j++;

}

else

{

foreach (HtmlCell cell in row.Cells)

{

String Data = cell.InnerText;

DataArray[j] = DataArray[j] + "^#$@~" + Data;

}

j++;

}

}

//DataArray = RemoveElement(DataArray, 1);

return DataArray;

}

public String[] Flow\_GetTableData\_Second()

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

int j = 0;

int Counter = 0;

//String[] HeaderArray = new String[header.CellCount];

Console.WriteLine("Row count: " + uIJqGridTable.RowCount);

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

int RowCnt = uIJqGridTable.RowCount;

int ColCnt = uIJqGridTable.GetRow(1).GetChildren().Count;

//IList<IWebElement> rowCollections = table.FindElements(By.TagName("tr"));

//int RowCnt = rowCollections.Count;

String[] DataArray = new String[RowCnt];

UITestControlCollection utc = uIJqGridTable.Rows;

foreach (HtmlRow row in utc)

{

Counter++;

if (Counter == 1)

{

for (int k = 0; k < header.CellCount; k++)

{

String HeaderText = header.GetCell(0, k).InnerText.Trim().ToString();

Console.WriteLine("Header " + k + ": " + HeaderText.ToString());

DataArray[j] = DataArray[j] + "^#$@~" + HeaderText;

}

j++;

}

else

{

foreach (HtmlCell cell in row.Cells)

{

String Data = cell.InnerText;

DataArray[j] = DataArray[j] + "^#$@~" + Data;

}

j++;

}

}

//DataArray = RemoveElement(DataArray, 1);

return DataArray;

}

/// <summary>

/// Export\_to\_Excel

/// </summary>

public void Export\_Directories\_to\_Excel(int GridColumnOffset, int ExcelColumnOffset)

{

#region Variable Declarations

HtmlButton uIExportfoldersButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfoldersButton;

HtmlButton uIExportfilesButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfilesButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportfoldersButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Verify\_Excel\_Directories(GridColumnOffset, ExcelColumnOffset, StrFilepath);

}

/// <summary>

/// Export\_to\_Excel

/// </summary>

public void Export\_Files\_to\_Excel(int GridColumnOffset, int ExcelColumnOffset)

{

#region Variable Declarations

HtmlButton uIExportfilesButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfilesButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportfilesButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Verify\_Excel\_Files(GridColumnOffset, ExcelColumnOffset, StrFilepath);

}

/// <summary>

/// Export\_to\_Excel - Use 'Export\_to\_ExcelParams' to pass parameters into this method.

/// </summary>

public void Export\_Report\_Requests\_to\_Excel(int GridColumnOffset)

{

#region Variable Declarations

HtmlTable table = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Verify\_Report\_Requests\_Excel(GridColumnOffset, StrFilepath);

}

/// <summary>

/// Export\_to\_Excel - Use 'Export\_to\_ExcelParams' to pass parameters into this method.

/// </summary>

public void Generate\_Exception\_Report\_Excel(int GridColumnOffset)

{

#region Variable Declarations

HtmlTable table = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlButton uIExportButton = this.UIShareInformationOSTOWindow1.UIShareInformationOSTODocument.UIExportButton;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

#endregion

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

// Click 'Export' button

Mouse.Click(uIExportButton);

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

Verify\_Excel(GridColumnOffset, 0, StrFilepath);

}

public virtual Export\_to\_ExcelParams Export\_to\_ExcelParams

{

get

{

if ((this.mExport\_to\_ExcelParams == null))

{

this.mExport\_to\_ExcelParams = new Export\_to\_ExcelParams();

}

return this.mExport\_to\_ExcelParams;

}

}

private Export\_to\_ExcelParams mExport\_to\_ExcelParams;

/// <summary>

/// Verify\_Elements\_Chargeback\_Reports - Use 'Verify\_Elements\_Chargeback\_ReportsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Chargeback\_Reports()

{

#region Variable Declarations

HtmlDiv uIChargebackReportsPane = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIChargebackReportsPane;

HtmlLabel uIRegionLabel = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UITeamLabel;

HtmlLabel uIHostLabel = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIHostLabel;

HtmlLabel uIVolumeLabel = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIVolumeLabel;

HtmlLabel uIShareLabel = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIShareLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostInputBoxEdit = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIHostInputBoxEdit;

HtmlEdit uIVolumeInputBoxEdit = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIVolumeInputBoxEdit;

HtmlEdit uIShareInputBoxEdit = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIShareInputBoxEdit;

HtmlButton uIGenerateReportButton = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIGenerateReportButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIChargebackReportsPane.WaitForControlExist(5000);

// Verify that the 'InnerText' property of 'Chargeback Reports' pane equals 'Chargeback Reports'

StringAssert.StartsWith(uIChargebackReportsPane.InnerText, this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIChargebackReportsPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides report detailing user and group, volume and application chargebacks.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Volume:' label equals 'Volume:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIVolumeLabelDisplayText, uIVolumeLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Share:' label equals 'Share:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIShareLabelDisplayText, uIShareLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostInputBox' text box equals 'hostInputBox'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIHostInputBoxEditId, uIHostInputBoxEdit.Id);

// Verify that the 'Id' property of 'volumeInputBox' text box equals 'volumeInputBox'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIVolumeInputBoxEditId, uIVolumeInputBoxEdit.Id);

// Verify that the 'Id' property of 'shareInputBox' text box equals 'shareInputBox'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIShareInputBoxEditId, uIShareInputBoxEdit.Id);

// Verify that the 'Id' property of 'Generate Report' button equals 'generateReportButton'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIGenerateReportButtonId, uIGenerateReportButton.Id);

// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ReportsExpectedValues.UIGenerateReportButtonDisplayText, uIGenerateReportButton.DisplayText.Trim());

}

/// <summary>

/// Generate\_Chargeback\_Report

/// </summary>

public DateTime Generate\_Chargeback\_Report()

{

#region Variable Declarations

HtmlDiv uIChargebackReportsPane = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIChargebackReportsPane;

HtmlButton uIGenerateReportButton = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

uIChargebackReportsPane.WaitForControlExist(5000);

Mouse.Click(uIGenerateReportButton);

DateTime CurrentDateTime = DateTime.Now;

uIReportRequestsforEURPane.WaitForControlExist(5000);

return CurrentDateTime;

}

/// <summary>

/// Generate\_Global\_Account\_Entitlement\_Subordinates\_Report

/// </summary>

public DateTime Generate\_Global\_Account\_Entitlement\_Subordinates\_Report(String Domain, String Account)

{

#region Variable Declarations

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlEdit uIDomainTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIDomainTxtEdit;

HtmlCheckBox UIIncludeGroupsCheckBox = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIIncludeGroupsCheckBox;

HtmlCheckBox UIIncludeSubordinatesCheckBox = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIIncludeSubordinatesCheckBox;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

HtmlButton uIGenerateButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

UIIncludeSubordinatesCheckBox.Checked = true;

uIDomainTxtEdit.Text = Domain;

uIAccountTxtEdit.Text = Account;

Keyboard.SendKeys("{Tab}");

Mouse.Click(uIGenerateButton);

DateTime CurrentDateTime = DateTime.Now;

uIReportRequestsforEURPane.WaitForControlExist(5000);

return CurrentDateTime;

}

public void Verify\_Global\_Account\_Entitlement\_dynamic\_Buttons(String Domain, String Account)

{

#region Variable Declarations

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlEdit uIDomainTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIDomainTxtEdit;

HtmlCheckBox UIIncludeGroupsCheckBox = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIIncludeGroupsCheckBox;

HtmlCheckBox UIIncludeSubordinatesCheckBox = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIIncludeSubordinatesCheckBox;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

HtmlButton uIGenerateButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

HtmlCustom UI3800CC\_TPA\_WWSS\_GCCS = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UI3800CC\_TPA\_WWSS\_GCCS;

HtmlCustom UIAutoCompleteNAM = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAutoCompleteNAM;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIIncludeGroupsCheckBox.Checked = true;

uIDomainTxtEdit.Text = Domain;

if (Run\_Environment == "OST\_UAT")

{

UIAutoCompleteNAM.WaitForControlExist(10000);

Mouse.Click(UIAutoCompleteNAM);

}

uIAccountTxtEdit.Text = Account;

if (Run\_Environment == "OST\_UAT")

{

UI3800CC\_TPA\_WWSS\_GCCS.WaitForControlExist(10000);

Mouse.Click(UI3800CC\_TPA\_WWSS\_GCCS);

}

//Keyboard.SendKeys("{Tab}");

Assert.AreEqual(uISearchButton.InnerText, "Search");

UIIncludeSubordinatesCheckBox.Checked = true;

UIIncludeGroupsCheckBox.Checked = false;

uIGenerateButton.WaitForControlExist(2000);

Assert.AreEqual((uIGenerateButton.InnerText).Trim(), "Generate Report");

}

/// <summary>

/// Generate\_PST\_Report

/// </summary>

public DateTime Generate\_PST\_Report()

{

#region Variable Declarations

HtmlControl uIPSTReportPane = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIPSTReportPane;

HtmlButton uIGenerateReportButton = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

uIPSTReportPane.WaitForControlExist(5000);

Mouse.Click(uIGenerateReportButton);

DateTime CurrentDateTime = DateTime.Now;

uIReportRequestsforEURPane.WaitForControlExist(5000);

return CurrentDateTime;

}

/// <summary>

/// Generate\_DLP\_Report

/// </summary>

public DateTime Generate\_DLP\_Report()

{

#region Variable Declarations

HtmlControl uIPSTReportPane = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIPSTReportPane;

HtmlButton uIGenerateReportButton = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

uIPSTReportPane.WaitForControlExist(5000);

Mouse.Click(uIGenerateReportButton);

DateTime CurrentDateTime = DateTime.Now;

uIReportRequestsforEURPane.WaitForControlExist(5000);

return CurrentDateTime;

}

/// <summary>

/// Generate\_IDrive\_Report

/// </summary>

public DateTime Generate\_IDrive\_Report()

{

#region Variable Declarations

HtmlControl uIGlobalIDrivePane = this.UIGlobalIDriveOSTOperaWindow.UIGlobalIDriveOSTOperaDocument.UIGlobalIDrivePane;

HtmlInputButton uIGenerateReportButton = this.UIGlobalIDriveOSTOperaWindow.UIGlobalIDriveOSTOperaDocument.UISearchFormCustom.UIGenerateReportButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

uIGlobalIDrivePane.WaitForControlExist(5000);

DateTime CurrentDateTime = DateTime.Now;

Mouse.Click(uIGenerateReportButton);

uIReportRequestsforEURPane.WaitForControlExist(5000);

return CurrentDateTime;

}

/// <summary>

/// Generate\_Exception\_Report

/// </summary>

public void Generate\_Exception\_Report()

{

#region Variable Declarations

HtmlControl uIExceptionReportPane = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UIExceptionReportPane;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

HtmlList uITeamsList = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UITeamsList;

HtmlInputButton uIGenerateReportButton = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UISearchFormCustom.UIGenerateReportButton;

HtmlListItem uI150080 = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UITeamsList.UIItemListItem;

WinButton uISaveButton = this.UIFileDownloadWindow.UISaveWindow.UISaveButton;

WinComboBox uIFilenameComboBox = this.UISaveAsWindow.UIDetailsPanePane.UIFilenameComboBox;

WinEdit uIFilenameEdit = this.UISaveAsWindow.UIItemWindow.UIFilenameEdit;

HtmlListItem uICorporate = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UITeamsList.UIItemListItem\_CorporateClient;

WinControl uIII11NotificationBar = this.UIReportRequestsforEURWindow1.UINotificationToolBar;

WinButton uIIE11SaveClose = this.UIReportRequestsforEURWindow1.UINotificationToolBar.UICloseButton;

WinControl uIIE11SaveDropDown = this.UIReportRequestsforEURWindow1.UISaveSplitButton.UIItemDropDownButton;

WinControl uIIE11SaveAsButton = this.UIItemWindow.UIContextMenu.UISaveasMenuItem;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIExceptionReportPane.WaitForControlExist(5000);

string StrFilepath, StrFileName, StrFileFolderpath;

//Fetch data from application.

StrFileFolderpath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

StrFileName = "Report" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

Console.WriteLine("Saving report as: " + StrFileFolderpath + @"\" + StrFileName + ".xlsx");

StrFilepath = StrFileFolderpath + @"\" + StrFileName + ".xlsx";

uITeamsList.WaitForControlExist(5000);

if (Run\_Environment == "OST\_DEV")

{

Mouse.Click(uI150080);

}

else if (Run\_Environment == "OST\_UAT")

{

Mouse.Click(uICorporate);

}

Mouse.Click(uIGenerateReportButton);

//string ver = (new WebBrowser()).Version.ToString();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

uIII11NotificationBar.WaitForControlExist(5000);

Mouse.Click(uIIE11SaveDropDown);

Mouse.Click(uIIE11SaveAsButton);

}

else

{

// Click '&Save' button

uISaveButton.WaitForControlExist(5000);

Mouse.Click(uISaveButton);

if (uISaveButton.Exists) { Mouse.Click(uISaveButton); }

}

// Enter report name in 'File name:' combo box

uIFilenameComboBox.WaitForControlExist(5000);

uIFilenameComboBox.EditableItem = StrFilepath;

// Type '{Enter}' in 'File name:' text box

Keyboard.SendKeys(uIFilenameEdit, this.Export\_to\_ExcelParams.UIFilenameEditSendKeys, ModifierKeys.None);

}

/// <summary>

/// Get Current Date Time

/// </summary>

public DateTime Get\_Current\_Date\_Time()

{

DateTime CurrentDateTime = DateTime.Now;

return CurrentDateTime;

}

public virtual Verify\_Elements\_Chargeback\_ReportsExpectedValues Verify\_Elements\_Chargeback\_ReportsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Chargeback\_ReportsExpectedValues == null))

{

this.mVerify\_Elements\_Chargeback\_ReportsExpectedValues = new Verify\_Elements\_Chargeback\_ReportsExpectedValues();

}

return this.mVerify\_Elements\_Chargeback\_ReportsExpectedValues;

}

}

private Verify\_Elements\_Chargeback\_ReportsExpectedValues mVerify\_Elements\_Chargeback\_ReportsExpectedValues;

/// <summary>

/// Verify\_Elements\_Chargeback\_Exceptions - Use 'Verify\_Elements\_Chargeback\_ExceptionsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Chargeback\_Exceptions()

{

#region Variable Declarations

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

HtmlLabel uIRegionLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UITeamLabel;

HtmlLabel uIHostLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostLabel;

HtmlLabel uIVolumeLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane.UIVolumeLabel;

HtmlLabel uISharesLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane.UISharesLabel;

HtmlLabel uIDeletedExceptionsLabel = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane.UIDeletedExceptionsLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNameEdit;

HtmlEdit uIVolumeNameBxEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIVolumeNameBxEdit;

HtmlEdit uIShareNameBxEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIShareNameBxEdit;

HtmlCheckBox uIIsDeletedCkbxCheckBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIIsDeletedCkbxCheckBox;

HtmlButton uISearchButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane.UISearchButton;

HtmlButton uIAddExceptionButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane.UIAddExceptionButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIChargebackExceptionsPane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Chargeback Exceptions' pane equals 'Chargeback Exceptions'

StringAssert.StartsWith(uIChargebackExceptionsPane.InnerText, this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIChargebackExceptionsPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides the ability to add chargeback exceptions.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Volume:' label starts with 'Volume:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIVolumeLabelDisplayText, uIVolumeLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Shares:' label starts with 'Shares:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UISharesLabelDisplayText, uISharesLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Deleted Exceptions?' label starts with 'Deleted Exceptions?'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIDeletedExceptionsLabelDisplayText, uIDeletedExceptionsLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'volumeNameBx' text box equals 'volumeNameBx'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIVolumeNameBxEditId, uIVolumeNameBxEdit.Id);

// Verify that the 'Id' property of 'shareNameBx' text box equals 'shareNameBx'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIShareNameBxEditId, uIShareNameBxEdit.Id);

// Verify that the 'Id' property of 'isDeletedCkbx' check box equals 'isDeletedCkbx'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIIsDeletedCkbxCheckBoxId, uIIsDeletedCkbxCheckBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

// Verify that the 'DisplayText' property of 'Add Exception' button equals 'Add Exception'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_ExceptionsExpectedValues.UIAddExceptionButtonDisplayText, uIAddExceptionButton.DisplayText.Trim());

}

public virtual Verify\_Elements\_Chargeback\_ExceptionsExpectedValues Verify\_Elements\_Chargeback\_ExceptionsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Chargeback\_ExceptionsExpectedValues == null))

{

this.mVerify\_Elements\_Chargeback\_ExceptionsExpectedValues = new Verify\_Elements\_Chargeback\_ExceptionsExpectedValues();

}

return this.mVerify\_Elements\_Chargeback\_ExceptionsExpectedValues;

}

}

private Verify\_Elements\_Chargeback\_ExceptionsExpectedValues mVerify\_Elements\_Chargeback\_ExceptionsExpectedValues;

/// <summary>

/// Search\_Elements\_Chargeback\_Cost\_Code

/// </summary>

public void Search\_Elements\_Chargeback\_Cost\_Code(String CostCode)

{

#region Variable Declarations

HtmlControl uIChargebackCostCodeSePane = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UIChargebackCostCodeSePane;

HtmlLabel uICostCodeorSoeIDLabel = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UICostCodeorSoeIDLabel;

HtmlEdit uICostCodeInputBoxEdit = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UICostCodeInputBoxEdit;

HtmlButton uISearchButton = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UISearchFormCustom.UISearchButton;

#endregion

uIChargebackCostCodeSePane.WaitForControlExist(5000);

uICostCodeInputBoxEdit.Text = CostCode;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Datastats\_Overview

/// </summary>

public void Search\_Datastats\_Overview(String Host)

{

#region Variable Declarations

HtmlControl uIDatastatsOverviewPane = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIDatastatsOverviewPane;

HtmlEdit uIHostNameEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIHostNameEdit;

HtmlEdit uILogDatePickerDtPickEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UILogDatePickerDtPickEdit;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

HtmlButton uISearchButton = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchButton;

HtmlCustom apct2sgusrsql2 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apct2sgusrsql2;

HtmlCustom ldnctisapad01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.ldnctisapad01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIDatastatsOverviewPane.WaitForControlExist(5000);

uILogDatePickerDtPickEdit.Text = "";

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

amsgtssf01.WaitForControlExist(5000);

Mouse.Click(amsgtssf01);

uILogDatePickerDtPickEdit.Text = "2/10/2013";

}

else if (Run\_Environment == "OST\_UAT")

{

apct2sgusrsql2.WaitForControlExist(5000);

Mouse.Click(apct2sgusrsql2);

}

else if (Run\_Environment == "OST\_PROD")

{

ldnctisapad01.WaitForControlExist(5000);

Mouse.Click(ldnctisapad01);

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Datastats\_Overview

/// </summary>

public void Search\_Datastats\_Overview\_by\_Host\_LogDate(String Host, String LogDate)

{

#region Variable Declarations

HtmlControl uIDatastatsOverviewPane = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIDatastatsOverviewPane;

HtmlEdit uIHostNameEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIHostNameEdit;

HtmlEdit uILogDatePickerDtPickEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UILogDatePickerDtPickEdit;

HtmlCustom cits390gts001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.cits390gts001;

HtmlButton uISearchButton = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchButton;

HtmlCustom apct2sgusrsql2 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apct2sgusrsql2;

HtmlCustom ldnctisapad01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.ldnctisapad01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIDatastatsOverviewPane.WaitForControlExist(5000);

uILogDatePickerDtPickEdit.Text = "";

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

cits390gts001.WaitForControlExist(5000);

Mouse.Click(cits390gts001);

}

else if (Run\_Environment == "OST\_UAT")

{

apct2sgusrsql2.WaitForControlExist(5000);

Mouse.Click(apct2sgusrsql2);

}

else if (Run\_Environment == "OST\_PROD")

{

ldnctisapad01.WaitForControlExist(10000);

Mouse.Click(ldnctisapad01);

uILogDatePickerDtPickEdit.Text = LogDate;

}

Mouse.Click(uISearchButton);

}

public void Search\_Datastats\_Overview\_by\_Host\_CurrentDate()

{

#region Variable Declarations

HtmlControl uIDatastatsOverviewPane = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIDatastatsOverviewPane;

HtmlEdit uIHostNameEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIHostNameEdit;

HtmlEdit uILogDatePickerDtPickEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UILogDatePickerDtPickEdit;

HtmlCustom cits390gts001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.cits390gts001;

HtmlButton uISearchButton = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchButton;

HtmlCustom apct2sgusrsql2 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apct2sgusrsql2;

HtmlCustom ldnctisapad01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.ldnctisapad01;

#endregion

uIDatastatsOverviewPane.WaitForControlExist(5000);

uILogDatePickerDtPickEdit.Text = "";

cits390gts001.WaitForControlExist(5000);

DateTime DtToday = DateTime.Today;

String LogDate = DtToday.ToString("MM/dd/yyyy");

uILogDatePickerDtPickEdit.Text = LogDate;

Mouse.Click(uISearchButton);

Verify\_search\_finished();

Verify\_Date(10, LogDate);

}

/// <summary>

/// Search\_Datastats\_Find

/// </summary>

public void Search\_Datastats\_Find(String Host, String Extension)

{

#region Variable Declarations

HtmlControl uIDataStatsFindPane = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDataStatsFindPane;

HtmlControl uIAuthDisclaimerPlacehPane = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAuthDisclaimerPlacehPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIHostNameEdit;

HtmlEdit uIVolumeInputBoxEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIVolumeInputBoxEdit;

HtmlEdit uIShareInputBoxEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIShareInputBoxEdit;

HtmlEdit uIFilePickerEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIFilePickerEdit;

HtmlEdit uITxtExtensionEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UITxtExtensionEdit;

HtmlEdit uIDateModifiedDivDtPicEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDateModifiedDivDtPicEdit;

HtmlEdit uIDateAccessedDivDtPicEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDateAccessedDivDtPicEdit;

HtmlEdit uIMinSizeEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIMinSizeEdit;

HtmlButton uISearchButton = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchButton;

HtmlCustom bcnvnascti002 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.bcnvnascti002;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIDataStatsFindPane.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

//dubctisw06dv.WaitForControlExist(5000);

//Mouse.Click(dubctisw06dv);

if (Run\_Environment == "OST\_PROD")

{

bcnvnascti002.WaitForControlExist(5000);

Mouse.Click(bcnvnascti002);

}

uITxtExtensionEdit.Text = Extension;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Datastats\_Find

/// </summary>

public void Search\_Datastats\_File\_Usage\_DLP(String Host)

{

#region Variable Declarations

HtmlCustom nasstl001v2rel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasstl001v2rel;

HtmlControl uIFileUsageDLPPane = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIFileUsageDLPPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIHostNameEdit;

HtmlCheckBox uIShowBySizeCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowBySizeCkCheckBox;

HtmlCheckBox uIShowByExtensionCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowByExtensionCkCheckBox;

HtmlCheckBox uIShowByDateCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowByDateCkCheckBox;

HtmlCheckBox uILargeFilesByExtensioCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UILargeFilesByExtensioCheckBox;

//HtmlCustom nasstl001v2rel = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasstl001v2rel;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIFileUsageDLPPane.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

nasstl001v2rel.WaitForControlExist(5000);

Mouse.Click(nasstl001v2rel);

}

else if (Run\_Environment == "OST\_UAT")

{

nasstl001v2rel.WaitForControlExist(5000);

Mouse.Click(nasstl001v2rel);

}

}

/// <summary>

/// Search\_Datastats\_Find

/// </summary>

public void Search\_Security\_Check(String Host)

{

#region Variable Declarations

HtmlControl uISecurityCheckPane = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UISecurityCheckPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIHostNameEdit;

HtmlEdit uILogDateDivDtPickEdit = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UILogDateDivDtPickEdit;

HtmlButton uISearchButton = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UISearchFormCustom.UISearchButton;

HtmlCustom apct2sgusrsql2 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apct2sgusrsql2;

HtmlCustom apaccitisgap111 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apaccitisgap111;

HtmlCustom apacbddhkmtes01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.apacbddhkmtes01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uISecurityCheckPane.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

apct2sgusrsql2.WaitForControlExist(5000);

Mouse.Click(apct2sgusrsql2);

}

else if (Run\_Environment == "OST\_UAT")

{

apaccitisgap111.WaitForControlExist(5000);

Mouse.Click(apaccitisgap111);

}

else if (Run\_Environment == "OST\_PROD")

{

apacbddhkmtes01.WaitForControlExist(5000);

Mouse.Click(apacbddhkmtes01);

}

uILogDateDivDtPickEdit.Text = "";

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Local\_Users

/// </summary>

public void Search\_Local\_Users(String Host)

{

#region Variable Declarations

HtmlControl uILocalUsersPane = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UILocalUsersPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox uIFilterSelectComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIFilterSelectComboBox;

HtmlButton uISearchButton = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

HtmlCustom dubctisp07 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.dubctisp07;

HtmlCustom mktxinpr01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.mktxinpr01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uILocalUsersPane.WaitForControlExist(10000);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

amsgtssf01.WaitForControlExist(10000);

Mouse.Click(amsgtssf01);

}

else if (Run\_Environment == "OST\_UAT")

{

dubctisp07.WaitForControlExist(5000);

Mouse.Click(dubctisp07);

}

else if (Run\_Environment == "OST\_PROD")

{

mktxinpr01.WaitForControlExist(5000);

Mouse.Click(mktxinpr01);

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Local\_Groups

/// </summary>

public void Search\_Local\_Groups(String Host, String LocalGroup, String Team)

{

#region Variable Declarations

HtmlControl uILocalGroupsPane = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UILocalGroupsPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox uIFilterSelectComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIFilterSelectComboBox;

HtmlComboBox uILocalGroupSelectComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UILocalGroupSelectComboBox;

HtmlButton uISearchButton = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlCustom beycmefs001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.beycmefs001;

HtmlCustom nammwddfs001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nammwddfs001;

HtmlCustom cpna111ap001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.cpna111ap001;

HtmlDiv Column\_MemberAccount = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_MemberAccount;

#endregion

uILocalGroupsPane.WaitForControlExist(60000);

string TeamsLoaded = "No";

for (int i = 0; i < 60; i++ )

{

if (uIAvailableTeamsDdlComboBox.ItemCount > 2)

{

TeamsLoaded = "Yes";

i = 60;

}

Playback.Wait(1000);

}

Assert.AreEqual("Yes", TeamsLoaded, "Team dropdown did not load values within 1 minute.");

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

//Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, Team + "{Enter}");

uIAvailableTeamsDdlComboBox.SelectedItem = Team;

}

else

{

//Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, Team + "ccc");

Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, Team + "{Enter}");

}

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

Keyboard.SendKeys(uIHostNameEdit, strhost);

if (Run\_Environment == "OST\_DEV")

{

beycmefs001.WaitForControlExist(6000);

Mouse.Click(beycmefs001);

}

else if (Run\_Environment == "OST\_UAT")

{

nammwddfs001.WaitForControlExist(6000);

Mouse.Click(nammwddfs001);

}

else if (Run\_Environment == "OST\_PROD")

{

cpna111ap001.WaitForControlExist(6000);

Mouse.Click(cpna111ap001);

}

Keyboard.SendKeys(uILocalGroupSelectComboBox, LocalGroup);

uILocalGroupSelectComboBox.WaitForControlReady(5000);

Mouse.Click(Column\_MemberAccount);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_User\_Rights

/// </summary>

public void Search\_User\_Rights(String Host, String UserRight)

{

#region Variable Declarations

HtmlControl UIUserRightsPane = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIUserRightsPane;

HtmlEdit uIHostNameEdit = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox UIUserRightSelectComboBox = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIUserRightSelectComboBox;

HtmlButton uISearchButton = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlEdit UILogDateDivDtPickEdit = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UILogDateDivDtPickEdit;

HtmlCustom beycmefs001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.beycmefs001;

HtmlCustom namnjwadm05 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.namnjwadm05;

HtmlCustom amsgtssf01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.amsgtssf01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIUserRightsPane.WaitForControlExist(5000);

Host = Host.Remove(Host.Length - 1);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

beycmefs001.WaitForControlExist(5000);

Mouse.Click(beycmefs001);

}

else if (Run\_Environment == "OST\_UAT")

{

namnjwadm05.WaitForControlExist(5000);

Mouse.Click(namnjwadm05);

}

else if (Run\_Environment == "OST\_PROD")

{

amsgtssf01.WaitForControlExist(10000);

Mouse.Click(amsgtssf01);

}

Keyboard.SendKeys(UIUserRightSelectComboBox, UserRight + "{Enter}");

UIUserRightSelectComboBox.WaitForControlReady(5000);

UILogDateDivDtPickEdit.Text = "";

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Orphans

/// </summary>

public void Search\_Orphans(String Host, String OrphanType, String LogDate, String ServerType)

{

#region Variable Declarations

HtmlControl uIOrphansPane = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIOrphansPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIHostNameEdit;

HtmlComboBox uIOrphanTypeSelectComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIOrphanTypeSelectComboBox;

HtmlEdit uILogDateDivDtPickEdit = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UILogDateDivDtPickEdit;

HtmlComboBox uIServerTypeSelectComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIServerTypeSelectComboBox;

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UISearchFormCustom.UISearchButton;

HtmlCustom fravnascrp001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.fravnascrp001;

HtmlCustom degewdrsw01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.degewdrsw01;

HtmlCustom chibmfs02 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.chibmfs02;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIOrphansPane.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

if (Run\_Environment == "OST\_DEV")

{

fravnascrp001.WaitForControlExist(5000);

Mouse.Click(fravnascrp001);

}

else if (Run\_Environment == "OST\_UAT")

{

degewdrsw01.WaitForControlExist(5000);

Mouse.Click(degewdrsw01);

}

else if (Run\_Environment == "OST\_PROD")

{

chibmfs02.WaitForControlExist(5000);

Mouse.Click(chibmfs02);

}

Keyboard.SendKeys(uIOrphanTypeSelectComboBox, OrphanType + "{Enter}");

uIOrphanTypeSelectComboBox.WaitForControlReady(5000);

uILogDateDivDtPickEdit.Text = LogDate;

Keyboard.SendKeys(uIServerTypeSelectComboBox, ServerType + "{Enter}");

uIServerTypeSelectComboBox.WaitForControlReady(5000);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Server\_Scheduled\_Tasks

/// </summary>

public void Search\_Server\_Scheduled\_Tasks(String Host, String LastResult)

{

#region Variable Declarations

HtmlControl uIScheduledTasksPane = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIScheduledTasksPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIHostNameEdit;

HtmlEdit uITaskNameSearchEdit = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UITaskNameSearchEdit;

HtmlComboBox uILastResultComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UILastResultComboBox;

HtmlComboBox uIStateComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIStateComboBox;

HtmlButton uISearchButton = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UISearchButton;

HtmlCustom namdevapp2 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.namdevapp2;

//HtmlCustom secsasvr01 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.secsasvr01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIScheduledTasksPane.WaitForControlExist(5000);

Keyboard.SendKeys(uIHostNameEdit, Host);

//uIHostNameEdit.Text = strhost;

namdevapp2.WaitForControlExist(5000);

Keyboard.SendKeys(uIHostNameEdit, "{Down}" + "{Enter}");

//Mouse.Click(namdevapp2);

Keyboard.SendKeys(uILastResultComboBox, LastResult + "{Enter}");

uILastResultComboBox.WaitForControlReady(5000);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Sharepoint\_Info

/// </summary>

public void Search\_Sharepoint\_Info(String Host)

{

#region Variable Declarations

HtmlControl uiSharepointInformatioPane = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISharepointInformatioPane;

HtmlComboBox uiRegionsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UITeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIHostNameEdit;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

HtmlCustom GMAQLKVAPPPR01 = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.GMAQLKVAPPPR01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uiSharepointInformatioPane.WaitForControlExist(5000);

uiHostNameEdit.Text = Host;

GMAQLKVAPPPR01.WaitForControlExist(5000);

Mouse.Click(GMAQLKVAPPPR01);

Mouse.Click(uiSearchButton);

}

/// <summary>

/// Search\_Sharepoint\_Info

/// </summary>

public void Search\_Sharepoint\_Info\_All(String Region)

{

#region Variable Declarations

HtmlControl uiSharepointInformatioPane = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISharepointInformatioPane;

HtmlComboBox uiRegionsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UITeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIHostNameEdit;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uiSharepointInformatioPane.WaitForControlExist(5000);

uiRegionsDdlComboBox.SelectedItem = Region;

Mouse.Click(uiSearchButton);

}

public void Search\_NAS\_PhysicalDevice\_Info\_Host(String Host)

{

#region Variable Declarations

HtmlControl UINASPhysicalDeviceInfPane = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UINASPhysicalDeviceInfPane;

HtmlComboBox uiRegionsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIHostNameEdit;

//HtmlButton uiSearchButton = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UISearchButton;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

HtmlCustom GMAQLKVAPPPR01 = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.GMAQLKVAPPPR01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UINASPhysicalDeviceInfPane.WaitForControlExist(5000);

uiHostNameEdit.Text = Host;

GMAQLKVAPPPR01.WaitForControlExist(3000);

Mouse.Click(GMAQLKVAPPPR01);

Mouse.Click(uiSearchButton);

}

public void Search\_NAS\_PhysicalDevice\_Info\_Region(String Region)

{

#region Variable Declarations

HtmlControl UINASPhysicalDeviceInfPane = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UINASPhysicalDeviceInfPane;

HtmlComboBox uiRegionsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIHostNameEdit;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

HtmlCustom GMAQLKVAPPPR01 = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.GMAQLKVAPPPR01;

#endregion

UINASPhysicalDeviceInfPane.WaitForControlExist(5000);

uiRegionsDdlComboBox.SelectedItem = Region;

Mouse.Click(uiSearchButton);

}

public void Sort\_by\_PhysicalDevice\_column()

{

#region Variable Declarations

HtmlDiv UIPhysicalDeviceNamePane = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIPhysicalDeviceNamePane;

#endregion

// Click 'Sharename' pane

Mouse.Click(UIPhysicalDeviceNamePane);

Verify\_Sort(2, 0, 0, "String");

}

/// <summary>

/// Search\_Anti\_Virus

/// </summary>

public void Search\_Anti\_Virus(String Team, String LogDate)

{

#region Variable Declarations

HtmlControl uIAntiVirusPane = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UIAntiVirusPane;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uILogDateDivDtPickEdit = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UILogDateDivDtPickEdit;

HtmlButton uISearchButton = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UISearchFormCustom.UISearchButton;

#endregion

uIAntiVirusPane.WaitForControlExist(5000);

//Keyboard.SendKeys(uIAvailableTeamsDdlComboBox, Team + "{Enter}");

uIAvailableTeamsDdlComboBox.SelectedItem = Team;

uILogDateDivDtPickEdit.Text = LogDate;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Datastats\_Details

/// </summary>

public void Search\_Datastats\_Details()

{

#region Variable Declarations

HtmlDiv uIDataStatsDetailsPane = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIDataStatsDetailsPane;

HtmlHyperlink uID531GBsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UID531GBsCustom.UID531GBsHyperlink;

HtmlTable DirectoriesGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable FilesGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

#endregion

uIDataStatsDetailsPane.WaitForControlExist(5000);

uID531GBsHyperlink.WaitForControlReady(5000);

uID531GBsHyperlink.Find();

uID531GBsHyperlink.GetClickablePoint();

Mouse.Click(uID531GBsHyperlink);

DirectoriesGridTable.WaitForControlReady(5000);

FilesGridTable.WaitForControlReady(5000);

}

/// <summary>

/// Open\_Manage\_Monitor\_Account

/// </summary>

public void Open\_Manage\_Monitor\_Account()

{

#region Variable Declarations

HtmlControl uIManageMonitorAccountPane = this.UIPrintQueueHistoryOSTWindow.UIManageMonitorAccountDocument.UIManageBCrumbPane.UIManageMonitorAccountPane;

HtmlHyperlink uIEditHyperlink = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIItem1Row.UIEditHyperlink;

HtmlDiv Column\_Team\_Name = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Team\_Name;

#endregion

uIManageMonitorAccountPane.WaitForControlExist(5000);

Mouse.Click(Column\_Team\_Name);

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

}

/// <summary>

/// Edit\_Manage\_Monitor\_Account

/// </summary>

public void Edit\_Manage\_Monitor\_Account(String Account\_Password)

{

#region Variable Declarations

HtmlEdit uIIAccountNameEdit = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIIAccountNameEdit;

HtmlComboBox uIIAccountDomainComboBox = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIIAccountDomainComboBox;

HtmlEdit uIIAccountPasswordEdit = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIIAccountPasswordEdit;

HtmlEdit uIICAccountPasswordEdit = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIICAccountPasswordEdit;

HtmlInputButton uIValidateAccountButton = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIValidateAccountButton;

HtmlDiv UIValidAccountPane = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIValidAccountPane;

#endregion

Keyboard.SendKeys(uIIAccountPasswordEdit, Account\_Password, true);

Keyboard.SendKeys(uIICAccountPasswordEdit, Account\_Password, true);

Mouse.Click(uIValidateAccountButton);

UIValidAccountPane.WaitForControlExist(5000);

}

/// <summary>

/// Verify\_Elements\_Chargeback\_Cost\_Code - Use 'Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Chargeback\_Cost\_Code()

{

#region Variable Declarations

HtmlControl uIChargebackCostCodeSePane = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UIChargebackCostCodeSePane;

HtmlLabel uICostCodeorSoeIDLabel = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UICostCodeorSoeIDLabel;

HtmlEdit uICostCodeInputBoxEdit = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UICostCodeInputBoxEdit;

HtmlButton uISearchButton = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UISearchFormCustom.UISearchButton;

HtmlControl uIAuthDisclaimerPlacehPane = this.UIChargebackCostCodeSeWindow.UIChargebackCostCodeSeDocument.UIAuthDisclaimerPlacehPane;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIChargebackCostCodeSePane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Chargeback Cost Code Search' pane equals 'Chargeback Cost Code Search'

StringAssert.StartsWith(uIChargebackCostCodeSePane.InnerText, this.Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues.UIChargebackCostCodeSePaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides the ability to search by cost code or SOE ID for chargebacks.");

// Verify that the 'DisplayText' property of 'Cost Code or SoeID:' label equals 'Cost Code or SoeID:'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues.UICostCodeorSoeIDLabelDisplayText, uICostCodeorSoeIDLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'costCodeInputBox' text box equals 'costCodeInputBox'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues.UICostCodeInputBoxEditId, uICostCodeInputBoxEdit.Id);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues.UISearchButtonDisplayText, uISearchButton.InnerText);

// Verify that the 'DisplayText' property of 'authDisclaimerPlaceholder' pane starts with 'Please Note: When searching using an SOEID, the results displayed will be for the SOEID's cost code which may be shared by other users.'

StringAssert.StartsWith(uIAuthDisclaimerPlacehPane.InnerText, this.Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues.UIAuthDisclaimerPlacehPaneDisplayText);

}

public virtual Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Chargeback\_Cost\_CodeExpectedValues == null))

{

this.mVerify\_Elements\_Chargeback\_Cost\_CodeExpectedValues = new Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues();

}

return this.mVerify\_Elements\_Chargeback\_Cost\_CodeExpectedValues;

}

}

private Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues mVerify\_Elements\_Chargeback\_Cost\_CodeExpectedValues;

/// <summary>

/// Verify\_Elements\_Exception\_Report - Use 'Verify\_Elements\_Exception\_ReportExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Exception\_Report()

{

#region Variable Declarations

HtmlControl uIExceptionReportPane = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UIExceptionReportPane;

HtmlLabel uITeamLabel = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UISearchFormCustom.UITeamLabel;

HtmlList uITeamsList = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UITeamsList;

HtmlInputButton uIGenerateReportButton = this.UIExceptionReportOSTOpWindow.UIExceptionReportOSTOpDocument.UISearchFormCustom.UIGenerateReportButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIExceptionReportPane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Exception Report' pane equals 'Exception Report'

StringAssert.StartsWith(uIExceptionReportPane.InnerText, this.Verify\_Elements\_Exception\_ReportExpectedValues.UIExceptionReportPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays team report indicating errors that are preventing the data from scanning. The report details whether the issue is with the permission of the FID on the host or if it’s a queue tree issue.");

// Verify that the 'DisplayText' property of '\* Team:' label starts with '\* Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Exception\_ReportExpectedValues.UITeamLabelDisplayText);

// Verify that the 'Id' property of 'Teams' list box equals 'Teams'

Assert.AreEqual(this.Verify\_Elements\_Exception\_ReportExpectedValues.UITeamsListId, uITeamsList.Id);

// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

Assert.AreEqual(this.Verify\_Elements\_Exception\_ReportExpectedValues.UIGenerateReportButtonDisplayText, uIGenerateReportButton.DisplayText);

}

public virtual Verify\_Elements\_Exception\_ReportExpectedValues Verify\_Elements\_Exception\_ReportExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Exception\_ReportExpectedValues == null))

{

this.mVerify\_Elements\_Exception\_ReportExpectedValues = new Verify\_Elements\_Exception\_ReportExpectedValues();

}

return this.mVerify\_Elements\_Exception\_ReportExpectedValues;

}

}

private Verify\_Elements\_Exception\_ReportExpectedValues mVerify\_Elements\_Exception\_ReportExpectedValues;

/// <summary>

/// Verify\_Elements\_NAS\_Global\_Coverage - Use 'Verify\_Elements\_NAS\_Global\_CoverageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_NAS\_Global\_Coverage()

{

#region Variable Declarations

HtmlControl uINASGlobalCoveragePane = this.UIExceptionReportOSTOpWindow.UINASGlobalCoverageOSTDocument.UINASGlobalCoveragePane;

HtmlInputButton uISubmitQueryButton = this.UIExceptionReportOSTOpWindow.UINASGlobalCoverageOSTDocument.UISubmitQueryButton;

#endregion

uINASGlobalCoveragePane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'NAS Global Coverage' pane equals 'NAS Global Coverage'

StringAssert.StartsWith(uINASGlobalCoveragePane.InnerText, this.Verify\_Elements\_NAS\_Global\_CoverageExpectedValues.UINASGlobalCoveragePaneDisplayText);

// Verify that the 'Id' property of 'Submit Query' button equals 'excelExportSubmit'

Assert.AreEqual(this.Verify\_Elements\_NAS\_Global\_CoverageExpectedValues.UISubmitQueryButtonId, uISubmitQueryButton.Id);

}

public virtual Verify\_Elements\_NAS\_Global\_CoverageExpectedValues Verify\_Elements\_NAS\_Global\_CoverageExpectedValues

{

get

{

if ((this.mVerify\_Elements\_NAS\_Global\_CoverageExpectedValues == null))

{

this.mVerify\_Elements\_NAS\_Global\_CoverageExpectedValues = new Verify\_Elements\_NAS\_Global\_CoverageExpectedValues();

}

return this.mVerify\_Elements\_NAS\_Global\_CoverageExpectedValues;

}

}

private Verify\_Elements\_NAS\_Global\_CoverageExpectedValues mVerify\_Elements\_NAS\_Global\_CoverageExpectedValues;

/// <summary>

/// Verify\_Report\_Generated

/// </summary>

public void Verify\_Report\_Generated(DateTime CurrentDateTime2, String ReportName)

{

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

string GridCellContent = "";

//Modified to convert the current time to UTC -- Check run in DEV once

CurrentDateTime2 = TimeZoneInfo.ConvertTimeToUtc(CurrentDateTime2);

String CurrentDateTime = CurrentDateTime2.ToString("MM/dd/yyyy HH:mm").ToLower();

if (uIJqGridTable.GetCell(1, 1) is HtmlRow)

{

HtmlRow row = (HtmlRow)uIJqGridTable.GetCell(1, 1);

GridCellContent = row.GetCell(2).InnerText.Trim();

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 2).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + row.GetCell(0).InnerText);

Console.WriteLine("Column: " + 2 + " Value: " + row.GetCell(2).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + row.GetCell(4).InnerText);

DateTime GridDateTime = DateTime.Parse(row.GetCell(4).InnerText, System.Globalization.CultureInfo.InvariantCulture);

TimeSpan DifferenceDateTime = (GridDateTime - CurrentDateTime2);

DifferenceDateTime = DifferenceDateTime.Duration();

String ReportDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

StringAssert.StartsWith(row.GetCell(0).InnerText, ReportName, "Report name does not match.");

Console.WriteLine("Report name found: " + ReportName);

Console.WriteLine("DateTime difference between grid and current: " + DifferenceDateTime);

Assert.IsTrue(DifferenceDateTime.TotalSeconds < 30, "DateTime difference between grid and current is greater than 30 seconds.");

//Assert.IsTrue(DifferenceDateTime.TotalSeconds < 60, "DateTime difference between grid and current is greater than 60 seconds.");

Console.WriteLine("Report Date/Time verified successully: " + ReportDateTime);

Console.WriteLine("Measured Click/Report time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

else

{

GridCellContent = uIJqGridTable.GetCell(1, 4).InnerText;

Console.WriteLine("Column count: " + uIJqGridTable.GetRow(1).GetChildren().Count);

Console.WriteLine("Header: " + header.GetCell(0, 4).InnerText.Trim());

Console.WriteLine("Column: " + 0 + " Value: " + uIJqGridTable.GetCell(1, 0).InnerText);

Console.WriteLine("Column: " + 2 + " Value: " + uIJqGridTable.GetCell(1, 2).InnerText);

Console.WriteLine("Column: " + 4 + " Value: " + uIJqGridTable.GetCell(1, 4).InnerText);

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

for (int x = 0; x < GridCellContent.Length; x++)

{

if (Char.IsLetterOrDigit(GridCellContent, x) == true || Char.IsSeparator(GridCellContent, x) == true || Char.IsPunctuation(GridCellContent, x) == true)

{

NewGridCellContent = NewGridCellContent.Append(GridCellContent.Substring(x, 1));

}

}

//Console.WriteLine(NewGridCellContent);

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

TimeSpan DifferenceDateTime = (GridDateTime - CurrentDateTime2);

DifferenceDateTime = DifferenceDateTime.Duration();

String ReportDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

StringAssert.StartsWith(uIJqGridTable.GetCell(1, 0).InnerText, ReportName, "Report name does not match.");

Console.WriteLine("Report name found: " + ReportName);

Console.WriteLine("DateTime difference between grid and current: " + DifferenceDateTime);

Assert.IsTrue(DifferenceDateTime.TotalSeconds < 30, "DateTime difference between grid and current is greater than 30 seconds.");

Console.WriteLine("Report Date/Time verified successully: " + ReportDateTime);

Console.WriteLine("Measured Click/Report time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

else

{

String uIJqGridTableTemp = uIJqGridTable.GetCell(1, 4).InnerText;

System.Text.StringBuilder NewGridCellContent = new System.Text.StringBuilder();

foreach (var ch in uIJqGridTableTemp) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { NewGridCellContent.Append(ch); } }

//String BadChar = uIJqGridTableTemp[uIJqGridTableTemp.Length - 3].ToString(); String NewGridCellContent = uIJqGridTableTemp.Replace(BadChar, "");

DateTime GridDateTime = DateTime.Parse(NewGridCellContent.ToString(), System.Globalization.CultureInfo.InvariantCulture);

TimeSpan DifferenceDateTime = (GridDateTime - CurrentDateTime2);

DifferenceDateTime = DifferenceDateTime.Duration();

String ReportDateTime = GridDateTime.ToString("MM/dd/yyyy HH:mm").ToLower();

StringAssert.StartsWith(uIJqGridTable.GetCell(1, 0).InnerText, ReportName, "Report name does not match.");

Console.WriteLine("Report name found: " + ReportName);

Console.WriteLine("DateTime difference between grid and current: " + DifferenceDateTime);

Assert.IsTrue(DifferenceDateTime.TotalSeconds < 30, "DateTime difference between grid and current is greater than 30 seconds.");

Console.WriteLine("Report Date/Time verified successully: " + ReportDateTime);

Console.WriteLine("Measured Click/Report time difference in seconds: " + DifferenceDateTime.TotalSeconds);

}

}

}

/// <summary>

/// Verify\_Elements\_Global\_IDrive - Use 'Verify\_Elements\_Global\_IDriveExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Global\_IDrive()

{

#region Variable Declarations

HtmlControl uIGlobalIDrivePane = this.UIGlobalIDriveOSTOperaWindow.UIGlobalIDriveOSTOperaDocument.UIGlobalIDrivePane;

HtmlInputButton uIGenerateReportButton = this.UIGlobalIDriveOSTOperaWindow.UIGlobalIDriveOSTOperaDocument.UISearchFormCustom.UIGenerateReportButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIGlobalIDrivePane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Global IDrive' pane equals 'Global IDrive'

StringAssert.StartsWith(uIGlobalIDrivePane.InnerText, this.Verify\_Elements\_Global\_IDriveExpectedValues.UIGlobalIDrivePaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides the ability to run a report showing all home drives globally.");

// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

Assert.AreEqual(this.Verify\_Elements\_Global\_IDriveExpectedValues.UIGenerateReportButtonDisplayText, uIGenerateReportButton.DisplayText);

}

public virtual Verify\_Elements\_Global\_IDriveExpectedValues Verify\_Elements\_Global\_IDriveExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Global\_IDriveExpectedValues == null))

{

this.mVerify\_Elements\_Global\_IDriveExpectedValues = new Verify\_Elements\_Global\_IDriveExpectedValues();

}

return this.mVerify\_Elements\_Global\_IDriveExpectedValues;

}

}

private Verify\_Elements\_Global\_IDriveExpectedValues mVerify\_Elements\_Global\_IDriveExpectedValues;

/// <summary>

/// Verify\_Elements\_Report\_Requests - Use 'Verify\_Elements\_Report\_RequestsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Report\_Requests()

{

#region Variable Declarations

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

HtmlLabel uIReportNameLabel = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchFormCustom.UIReportNameLabel;

HtmlLabel uIRequestedDateLabel = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchFormCustom.UIRequestedDateLabel;

HtmlLabel uIStatusLabel = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchFormCustom.UIStatusLabel;

HtmlLabel uIStartedDateLabel = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchFormCustom.UIStartedDateLabel;

HtmlLabel uIFinishedDateLabel = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchFormCustom.UIFinishedDateLabel;

HtmlEdit uIReportNameTxtEdit = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportNameTxtEdit;

HtmlEdit uIRequestedDateDivDtPiEdit = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIRequestedDateDivDtPiEdit;

HtmlComboBox uIStatusTxtComboBox = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIStatusTxtComboBox;

HtmlEdit uIStartedDateDivDtPickEdit = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIStartedDateDivDtPickEdit;

HtmlEdit uIFinishedDateDivDtPicEdit = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIFinishedDateDivDtPicEdit;

HtmlButton uISearchButton = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIReportRequestsforEURPane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Report Requests for EUR\kc90246' pane starts with 'Report Requests for'

StringAssert.StartsWith(uIReportRequestsforEURPane.InnerText, this.Verify\_Elements\_Report\_RequestsExpectedValues.UIReportRequestsforEURPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides status of reports currently in your queue as well as the number of reports in total for the queue and allows access to open your own reports when ready.");

// Verify that the 'DisplayText' property of 'Report Name:' label starts with 'Report Name:'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIReportNameLabelDisplayText, uIReportNameLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Requested Date:' label starts with 'Requested Date:'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIRequestedDateLabelDisplayText, uIRequestedDateLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Status:' label starts with 'Status:'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIStatusLabelDisplayText, uIStatusLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Started Date:' label starts with 'Started Date:'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIStartedDateLabelDisplayText, uIStartedDateLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Finished Date:' label starts with 'Finished Date:'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIFinishedDateLabelDisplayText, uIFinishedDateLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'reportNameTxt' text box equals 'reportNameTxt'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIReportNameTxtEditId, uIReportNameTxtEdit.Id);

// Verify that the 'Id' property of 'requestedDateDivDtPick' text box equals 'requestedDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIRequestedDateDivDtPiEditId, uIRequestedDateDivDtPiEdit.Id);

// Verify that the 'Id' property of 'statusTxt' combo box equals 'statusTxt'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIStatusTxtComboBoxId, uIStatusTxtComboBox.Id);

// Verify that the 'Id' property of 'startedDateDivDtPick' text box equals 'startedDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIStartedDateDivDtPickEditId, uIStartedDateDivDtPickEdit.Id);

// Verify that the 'Id' property of 'finishedDateDivDtPick' text box equals 'finishedDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UIFinishedDateDivDtPicEditId, uIFinishedDateDivDtPicEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Report\_RequestsExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Report\_RequestsExpectedValues Verify\_Elements\_Report\_RequestsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Report\_RequestsExpectedValues == null))

{

this.mVerify\_Elements\_Report\_RequestsExpectedValues = new Verify\_Elements\_Report\_RequestsExpectedValues();

}

return this.mVerify\_Elements\_Report\_RequestsExpectedValues;

}

}

private Verify\_Elements\_Report\_RequestsExpectedValues mVerify\_Elements\_Report\_RequestsExpectedValues;

/// <summary>

/// Verify\_Elements\_Datastats\_Overview - Use 'Verify\_Elements\_Datastats\_OverviewExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Datastats\_Overview()

{

#region Variable Declarations

HtmlControl uIDatastatsOverviewPane = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIDatastatsOverviewPane;

HtmlLabel uIRegionLabel = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uILogDateLabel = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchFormCustom.UILogDateLabel;

HtmlLabel uISizeLabel = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchFormCustom.UISizeLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIHostNameEdit;

HtmlEdit uILogDatePickerDtPickEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UILogDatePickerDtPickEdit;

HtmlEdit uIValueOptionInputEdit = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIValueOptionInputEdit;

HtmlComboBox uIOperatorOptionSelectComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIOperatorOptionSelectComboBox;

HtmlComboBox uIUnitOptionSelectComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIUnitOptionSelectComboBox;

HtmlComboBox uITypeOptionSelectComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UITypeOptionSelectComboBox;

HtmlButton uISearchButton = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

#endregion

uIDatastatsOverviewPane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Datastats Overview' pane starts with 'Datastats Overview'

StringAssert.StartsWith(uIDatastatsOverviewPane.InnerText, this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIDatastatsOverviewPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Used by the System Administrators to gauge the host and volume utilization for their team.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText.Trim(), this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

StringAssert.StartsWith(uILogDateLabel.DisplayText.Trim(), this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UILogDateLabelDisplayText);

// Verify that the 'DisplayText' property of 'Size:' label starts with 'Size:'

StringAssert.StartsWith(uISizeLabel.DisplayText.Trim(), this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UISizeLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'logDatePickerDtPick' text box equals 'logDatePickerDtPick'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UILogDatePickerDtPickEditId, uILogDatePickerDtPickEdit.Id);

// Verify that the 'Id' property of 'valueOptionInput' text box equals 'valueOptionInput'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIValueOptionInputEditId, uIValueOptionInputEdit.Id);

// Verify that the 'Id' property of 'operatorOptionSelect' combo box equals 'operatorOptionSelect'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIOperatorOptionSelectComboBoxId, uIOperatorOptionSelectComboBox.Id);

// Verify that the 'Id' property of 'unitOptionSelect' combo box equals 'unitOptionSelect'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UIUnitOptionSelectComboBoxId, uIUnitOptionSelectComboBox.Id);

// Verify that the 'Id' property of 'typeOptionSelect' combo box equals 'typeOptionSelect'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UITypeOptionSelectComboBoxId, uITypeOptionSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_OverviewExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText.Trim());

}

public virtual Verify\_Elements\_Datastats\_OverviewExpectedValues Verify\_Elements\_Datastats\_OverviewExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Datastats\_OverviewExpectedValues == null))

{

this.mVerify\_Elements\_Datastats\_OverviewExpectedValues = new Verify\_Elements\_Datastats\_OverviewExpectedValues();

}

return this.mVerify\_Elements\_Datastats\_OverviewExpectedValues;

}

}

private Verify\_Elements\_Datastats\_OverviewExpectedValues mVerify\_Elements\_Datastats\_OverviewExpectedValues;

/// <summary>

/// Verify\_Elements\_Datastats\_Find - Use 'Verify\_Elements\_Datastats\_FindExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Datastats\_Find()

{

#region Variable Declarations

HtmlControl uIDataStatsFindPane = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDataStatsFindPane;

HtmlLabel uIRegionLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIVolumeLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchFormCustom.UIVolumeLabel;

HtmlLabel uIShareLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchFormCustom.UIShareLabel;

HtmlLabel uIIamSearchingForLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchFormCustom.UIIamSearchingForLabel;

HtmlLabel uISearchCriteriaLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchCriteriaLabel;

HtmlLabel uILastModifiedLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UILastModifiedLabel;

HtmlLabel uILastAccessedLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchFormCustom.UILastAccessedLabel;

HtmlLabel uIFileExtensionLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIFileExtensionLabel;

HtmlLabel uIMinSizeMBLabel = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchFormCustom.UIMinSizeMBLabel;

HtmlControl uIAuthDisclaimerPlacehPane = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAuthDisclaimerPlacehPane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIHostNameEdit;

HtmlEdit uIVolumeInputBoxEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIVolumeInputBoxEdit;

HtmlEdit uIShareInputBoxEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIShareInputBoxEdit;

HtmlEdit uIFilePickerEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIFilePickerEdit;

HtmlEdit uITxtExtensionEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UITxtExtensionEdit;

HtmlEdit uIDateModifiedDivDtPicEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDateModifiedDivDtPicEdit;

HtmlEdit uIDateAccessedDivDtPicEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIDateAccessedDivDtPicEdit;

HtmlEdit uIMinSizeEdit = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UIMinSizeEdit;

HtmlButton uISearchButton = this.UIDataStatsFindOSTOperWindow.UIDataStatsFindOSTOperDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

uIDataStatsFindPane.WaitForControlExist(5000);

// Verify that the 'DisplayText' property of 'Data Stats Find' pane equals 'Data Stats Find'

StringAssert.StartsWith(uIDataStatsFindPane.InnerText, this.Verify\_Elements\_Datastats\_FindExpectedValues.UIDataStatsFindPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides the ability to search for group specific file/directory information");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Volume:' label equals 'Volume:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIVolumeLabelDisplayText, uIVolumeLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Share:' label starts with 'Share:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIShareLabelDisplayText, uIShareLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'I am Searching For:' label starts with 'I am Searching For:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIIamSearchingForLabelDisplayText, uIIamSearchingForLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Search Criteria:' label starts with 'Search Criteria:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UISearchCriteriaLabelDisplayText, uISearchCriteriaLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Last Modified:' label equals 'Last Modified:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UILastModifiedLabelDisplayText, uILastModifiedLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Last Accessed:' label equals 'Last Accessed:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UILastAccessedLabelDisplayText, uILastAccessedLabel.DisplayText);

// Verify that the 'DisplayText' property of 'File Extension:' label starts with 'File Extension:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIFileExtensionLabelDisplayText, uIFileExtensionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Min Size (MB):' label starts with 'Min Size (MB):'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIMinSizeMBLabelDisplayText, uIMinSizeMBLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'authDisclaimerPlaceholder' pane starts with 'Please Note: Only results you are authorized to view will be displayed.'

StringAssert.StartsWith(uIAuthDisclaimerPlacehPane.InnerText, this.Verify\_Elements\_Datastats\_FindExpectedValues.UIAuthDisclaimerPlacehPaneDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'volumeInputBox' text box equals 'volumeInputBox'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIVolumeInputBoxEditId, uIVolumeInputBoxEdit.Id);

// Verify that the 'Id' property of 'shareInputBox' text box equals 'shareInputBox'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIShareInputBoxEditId, uIShareInputBoxEdit.Id);

// Verify that the 'Id' property of 'filePicker' text box equals 'filePicker'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIFilePickerEditId, uIFilePickerEdit.Id);

// Verify that the 'Id' property of 'txtExtension' text box equals 'txtExtension'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UITxtExtensionEditId, uITxtExtensionEdit.Id);

// Verify that the 'Id' property of 'dateModifiedDivDtPick' text box equals 'dateModifiedDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIDateModifiedDivDtPicEditId, uIDateModifiedDivDtPicEdit.Id);

// Verify that the 'Id' property of 'dateAccessedDivDtPick' text box equals 'dateAccessedDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIDateAccessedDivDtPicEditId, uIDateAccessedDivDtPicEdit.Id);

// Verify that the 'Id' property of 'minSize' text box equals 'minSize'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UIMinSizeEditId, uIMinSizeEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_FindExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText.Trim());

}

public virtual Verify\_Elements\_Datastats\_FindExpectedValues Verify\_Elements\_Datastats\_FindExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Datastats\_FindExpectedValues == null))

{

this.mVerify\_Elements\_Datastats\_FindExpectedValues = new Verify\_Elements\_Datastats\_FindExpectedValues();

}

return this.mVerify\_Elements\_Datastats\_FindExpectedValues;

}

}

private Verify\_Elements\_Datastats\_FindExpectedValues mVerify\_Elements\_Datastats\_FindExpectedValues;

/// <summary>

/// Verify\_Elements\_Datastats\_File\_Usage\_DLP - Use 'Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Datastats\_File\_Usage\_DLP()

{

#region Variable Declarations

HtmlControl uIFileUsageDLPPane = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIFileUsageDLPPane;

HtmlLabel uIRegionLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIGroupbySizeLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UISearchFormCustom.UIGroupbySizeLabel;

HtmlLabel uIGroupbyExtensionLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UISearchFormCustom.UIGroupbyExtensionLabel;

HtmlLabel uIGroupbyLastModifiedDLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UISearchFormCustom.UIGroupbyLastModifiedDLabel;

HtmlLabel uIGroupbyExtensionlargLabel = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UISearchFormCustom.UIGroupbyExtensionlargLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIHostNameEdit;

HtmlCheckBox uIShowBySizeCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowBySizeCkCheckBox;

HtmlCheckBox uIShowByExtensionCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowByExtensionCkCheckBox;

HtmlCheckBox uIShowByDateCkCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UIShowByDateCkCheckBox;

HtmlCheckBox uILargeFilesByExtensioCheckBox = this.UIFileUsageDLPOSTOperaWindow.UIFileUsageDLPOSTOperaDocument.UILargeFilesByExtensioCheckBox;

HtmlButton uIGenerateReportButton = this.UIChargebackReportsOSTWindow.UIChargebackReportsOSTDocument.UIGenerateReportButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'File Usage (DLP)' pane equals 'File Usage (DLP)'

StringAssert.StartsWith(uIFileUsageDLPPane.InnerText, this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIFileUsageDLPPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Shows generalized statistics about what type of files are stored on filer servers.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Group by Size' label starts with 'Group by Size'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIGroupbySizeLabelDisplayText, uIGroupbySizeLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Group by Extension' label starts with 'Group by Extension'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIGroupbyExtensionLabelDisplayText, uIGroupbyExtensionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Group by Last Modified Date' label starts with 'Group by Last Modified Date'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIGroupbyLastModifiedDLabelDisplayText, uIGroupbyLastModifiedDLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Group by Extension (large files)' label starts with 'Group by Extension (large files)'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIGroupbyExtensionlargLabelDisplayText, uIGroupbyExtensionlargLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'showBySizeCk' check box equals 'showBySizeCk'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIShowBySizeCkCheckBoxId, uIShowBySizeCkCheckBox.Id);

// Verify that the 'Id' property of 'showByExtensionCk' check box equals 'showByExtensionCk'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIShowByExtensionCkCheckBoxId, uIShowByExtensionCkCheckBox.Id);

// Verify that the 'Id' property of 'showByDateCk' check box equals 'showByDateCk'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UIShowByDateCkCheckBoxId, uIShowByDateCkCheckBox.Id);

// Verify that the 'Id' property of 'largeFilesByExtensionCk' check box equals 'largeFilesByExtensionCk'

Assert.AreEqual(this.Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues.UILargeFilesByExtensioCheckBoxId, uILargeFilesByExtensioCheckBox.Id);

// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIGenerateReportButtonDisplayText, uIGenerateReportButton.DisplayText);

}

public virtual Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues == null))

{

this.mVerify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues = new Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues();

}

return this.mVerify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues;

}

}

private Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues mVerify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues;

/// <summary>

/// Verify\_Elements\_PST\_Report - Use 'Verify\_Elements\_PST\_ReportExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_PST\_Report()

{

#region Variable Declarations

HtmlControl uIPSTReportPane = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIPSTReportPane;

HtmlLabel uIRegionLabel = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIHostNameEdit;

HtmlButton uIGenerateReportButton = this.UIPSTReportOSTOperatioWindow.UIPSTReportOSTOperatioDocument.UIGenerateReportButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'PST Report' pane equals 'PST Report'

StringAssert.StartsWith(uIPSTReportPane.InnerText, this.Verify\_Elements\_PST\_ReportExpectedValues.UIPSTReportPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides .pst file details for specific hosts.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

Assert.AreEqual(this.Verify\_Elements\_PST\_ReportExpectedValues.UIGenerateReportButtonDisplayText, uIGenerateReportButton.DisplayText.Trim());

}

public virtual Verify\_Elements\_PST\_ReportExpectedValues Verify\_Elements\_PST\_ReportExpectedValues

{

get

{

if ((this.mVerify\_Elements\_PST\_ReportExpectedValues == null))

{

this.mVerify\_Elements\_PST\_ReportExpectedValues = new Verify\_Elements\_PST\_ReportExpectedValues();

}

return this.mVerify\_Elements\_PST\_ReportExpectedValues;

}

}

private Verify\_Elements\_PST\_ReportExpectedValues mVerify\_Elements\_PST\_ReportExpectedValues;

/// <summary>

/// Verify\_Elements\_AntiVirus - Use 'Verify\_Elements\_AntiVirusExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_AntiVirus()

{

#region Variable Declarations

HtmlControl uIAntiVirusPane = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UIAntiVirusPane;

HtmlLabel uITeamLabel = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uILogDateLabel = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UISearchFormCustom.UILogDateLabel;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uILogDateDivDtPickEdit = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UILogDateDivDtPickEdit;

HtmlButton uISearchButton = this.UIAntiVirusOSTOperatioWindow.UIAntiVirusOSTOperatioDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'AntiVirus' pane equals 'AntiVirus'

StringAssert.StartsWith(uIAntiVirusPane.InnerText, this.Verify\_Elements\_AntiVirusExpectedValues.UIAntiVirusPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays details for anti-virus software for each host by team.");

// Verify that the 'DisplayText' property of 'Team:' label equals 'Team:'

Assert.AreEqual(this.Verify\_Elements\_AntiVirusExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

StringAssert.StartsWith(uILogDateLabel.DisplayText, this.Verify\_Elements\_AntiVirusExpectedValues.UILogDateLabelDisplayText);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_AntiVirusExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_AntiVirusExpectedValues.UILogDateDivDtPickEditId, uILogDateDivDtPickEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_AntiVirusExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_AntiVirusExpectedValues Verify\_Elements\_AntiVirusExpectedValues

{

get

{

if ((this.mVerify\_Elements\_AntiVirusExpectedValues == null))

{

this.mVerify\_Elements\_AntiVirusExpectedValues = new Verify\_Elements\_AntiVirusExpectedValues();

}

return this.mVerify\_Elements\_AntiVirusExpectedValues;

}

}

private Verify\_Elements\_AntiVirusExpectedValues mVerify\_Elements\_AntiVirusExpectedValues;

/// <summary>

/// Verify\_Elements\_Security\_Check - Use 'Verify\_Elements\_Security\_CheckExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Security\_Check()

{

#region Variable Declarations

HtmlControl uISecurityCheckPane = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UISecurityCheckPane;

HtmlLabel uIRegionLabel = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uILogDateLabel = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UISearchFormCustom.UILogDateLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UIHostNameEdit;

HtmlEdit uILogDateDivDtPickEdit = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UILogDateDivDtPickEdit;

HtmlButton uISearchButton = this.UISecurityCheckOSTOperWindow.UISecurityCheckOSTOperDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Security Check' pane starts with 'Security Check'

StringAssert.StartsWith(uISecurityCheckPane.InnerText, this.Verify\_Elements\_Security\_CheckExpectedValues.UISecurityCheckPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays details about wrap ID for hosts by team.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UITeamLabelDisplayText, uITeamLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UILogDateLabelDisplayText, uILogDateLabel.DisplayText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UILogDateDivDtPickEditId, uILogDateDivDtPickEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Security\_CheckExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Security\_CheckExpectedValues Verify\_Elements\_Security\_CheckExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Security\_CheckExpectedValues == null))

{

this.mVerify\_Elements\_Security\_CheckExpectedValues = new Verify\_Elements\_Security\_CheckExpectedValues();

}

return this.mVerify\_Elements\_Security\_CheckExpectedValues;

}

}

private Verify\_Elements\_Security\_CheckExpectedValues mVerify\_Elements\_Security\_CheckExpectedValues;

/// <summary>

/// Verify\_Elements\_Security\_Groups - Use 'Verify\_Elements\_Security\_GroupsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Security\_Groups()

{

#region Variable Declarations

HtmlControl uISecurityGroupsPane = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIISpanPane.UISecurityGroupsPane;

HtmlInputButton uIAddNewButton = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIAddNewButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Security Groups' pane starts with 'Security Groups'

StringAssert.StartsWith(uISecurityGroupsPane.InnerText, this.Verify\_Elements\_Security\_GroupsExpectedValues.UISecurityGroupsPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays security groups and details around the group. Page provides the ability to add/delete groups.");

// Verify that the 'Id' property of 'Add New' button equals 'btnSecurityAdd'

Assert.AreEqual(this.Verify\_Elements\_Security\_GroupsExpectedValues.UIAddNewButtonId, uIAddNewButton.Id);

}

public virtual Verify\_Elements\_Security\_GroupsExpectedValues Verify\_Elements\_Security\_GroupsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Security\_GroupsExpectedValues == null))

{

this.mVerify\_Elements\_Security\_GroupsExpectedValues = new Verify\_Elements\_Security\_GroupsExpectedValues();

}

return this.mVerify\_Elements\_Security\_GroupsExpectedValues;

}

}

private Verify\_Elements\_Security\_GroupsExpectedValues mVerify\_Elements\_Security\_GroupsExpectedValues;

/// <summary>

/// Verify\_Elements\_Local\_Users - Use 'Verify\_Elements\_Local\_UsersExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Local\_Users()

{

#region Variable Declarations

HtmlControl uILocalUsersPane = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UILocalUsersPane;

HtmlLabel uIRegionLabel = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIFilterLabel = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UISearchFormCustom.UIFilterLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox uIFilterSelectComboBox = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UIFilterSelectComboBox;

HtmlButton uISearchButton = this.UILocalUsersOSTOperatiWindow.UILocalUsersOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Local Users' pane equals 'Local Users'

StringAssert.StartsWith(uILocalUsersPane.InnerText, this.Verify\_Elements\_Local\_UsersExpectedValues.UILocalUsersPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays local user accounts, both enabled and disabled, for hosts by team. You have the ability to filter out Administrator and/or disabled account(s).");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Local\_UsersExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Filter:' label starts with 'Filter:'

StringAssert.StartsWith(uIFilterLabel.DisplayText, this.Verify\_Elements\_Local\_UsersExpectedValues.UIFilterLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'FilterSelect' combo box equals 'FilterSelect'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UIFilterSelectComboBoxId, uIFilterSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Local\_UsersExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Local\_UsersExpectedValues Verify\_Elements\_Local\_UsersExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Local\_UsersExpectedValues == null))

{

this.mVerify\_Elements\_Local\_UsersExpectedValues = new Verify\_Elements\_Local\_UsersExpectedValues();

}

return this.mVerify\_Elements\_Local\_UsersExpectedValues;

}

}

private Verify\_Elements\_Local\_UsersExpectedValues mVerify\_Elements\_Local\_UsersExpectedValues;

/// <summary>

/// Verify\_Elements\_Local\_Groups - Use 'Verify\_Elements\_Local\_GroupsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Local\_Groups()

{

#region Variable Declarations

HtmlControl uILocalGroupsPane = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UILocalGroupsPane;

HtmlLabel uIRegionLabel = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIFilterLabel = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UISearchFormCustom.UIFilterLabel;

HtmlLabel uILocalGroupLabel = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UISearchFormCustom.UILocalGroupLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox uIFilterSelectComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UIFilterSelectComboBox;

HtmlComboBox uILocalGroupSelectComboBox = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UILocalGroupSelectComboBox;

HtmlButton uISearchButton = this.UILocalGroupOSTOperatiWindow.UILocalGroupOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Local Groups' pane equals 'Local Groups'

StringAssert.StartsWith(uILocalGroupsPane.InnerText, this.Verify\_Elements\_Local\_GroupsExpectedValues.UILocalGroupsPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays local groups, both enabled and disabled, for hosts by team. Provides the ability to search by group name as well as filter out Administrator and/or user group(s).");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Local\_GroupsExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Filter:' label starts with 'Filter:'

StringAssert.StartsWith(uIFilterLabel.DisplayText, this.Verify\_Elements\_Local\_GroupsExpectedValues.UIFilterLabelDisplayText);

// Verify that the 'DisplayText' property of 'Local Group :' label starts with 'Local Group:'

StringAssert.StartsWith(uILocalGroupLabel.DisplayText, this.Verify\_Elements\_Local\_GroupsExpectedValues.UILocalGroupLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'FilterSelect' combo box equals 'FilterSelect'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UIFilterSelectComboBoxId, uIFilterSelectComboBox.Id);

// Verify that the 'Id' property of 'LocalGroupSelect' combo box equals 'LocalGroupSelect'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UILocalGroupSelectComboBoxId, uILocalGroupSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Local\_GroupsExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Local\_GroupsExpectedValues Verify\_Elements\_Local\_GroupsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Local\_GroupsExpectedValues == null))

{

this.mVerify\_Elements\_Local\_GroupsExpectedValues = new Verify\_Elements\_Local\_GroupsExpectedValues();

}

return this.mVerify\_Elements\_Local\_GroupsExpectedValues;

}

}

private Verify\_Elements\_Local\_GroupsExpectedValues mVerify\_Elements\_Local\_GroupsExpectedValues;

/// <summary>

/// Verify\_Elements\_User\_Rights - Use 'Verify\_Elements\_User\_RightsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_User\_Rights()

{

#region Variable Declarations

HtmlControl uIUserRightsPane = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIUserRightsPane;

HtmlLabel uIRegionLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIUserRightLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UISearchFormCustom.UIUserRightLabel;

HtmlLabel uILogDateLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UISearchFormCustom.UILogDateLabel;

HtmlLabel uITrusteeLabel = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UISearchFormCustom.UITrusteeLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIHostNameEdit;

HtmlComboBox uIUserRightSelectComboBox = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UIUserRightSelectComboBox;

HtmlEdit uILogDateDivDtPickEdit = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UILogDateDivDtPickEdit;

HtmlComboBox uITrusteeSelectComboBox = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UITrusteeSelectComboBox;

HtmlButton uISearchButton = this.UIUserRightsOSTOperatiWindow.UIUserRightsOSTOperatiDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'User Rights' pane equals 'User Rights'

StringAssert.StartsWith(uIUserRightsPane.InnerText, this.Verify\_Elements\_User\_RightsExpectedValues.UIUserRightsPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays user right privileges for Wintel Hosts for teams. Page provides the ability to search by user right, log date and/or trustee.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_User\_RightsExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'User Right:' label starts with 'User Right:'

StringAssert.StartsWith(uIUserRightLabel.DisplayText, this.Verify\_Elements\_User\_RightsExpectedValues.UIUserRightLabelDisplayText);

// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

StringAssert.StartsWith(uILogDateLabel.DisplayText, this.Verify\_Elements\_User\_RightsExpectedValues.UILogDateLabelDisplayText);

// Verify that the 'DisplayText' property of 'Trustee:' label starts with 'Trustee:'

StringAssert.StartsWith(uITrusteeLabel.DisplayText, this.Verify\_Elements\_User\_RightsExpectedValues.UITrusteeLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'UserRightSelect' combo box equals 'UserRightSelect'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UIUserRightSelectComboBoxId, uIUserRightSelectComboBox.Id);

// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UILogDateDivDtPickEditId, uILogDateDivDtPickEdit.Id);

// Verify that the 'Id' property of 'TrusteeSelect' combo box equals 'TrusteeSelect'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UITrusteeSelectComboBoxId, uITrusteeSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_User\_RightsExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_User\_RightsExpectedValues Verify\_Elements\_User\_RightsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_User\_RightsExpectedValues == null))

{

this.mVerify\_Elements\_User\_RightsExpectedValues = new Verify\_Elements\_User\_RightsExpectedValues();

}

return this.mVerify\_Elements\_User\_RightsExpectedValues;

}

}

/// <summary>

/// Verify\_HostDashboard\_Page

/// </summary>

public void Verify\_HostDashboard\_Page()

{

#region Variable Declarations

HtmlDiv uIHostDetailsPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIHostDetailsPane;

HtmlDiv uINasVirtualDevicesPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UINasVirtualDevicesPane;

HtmlDiv uINasPhysicalDevicePane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UINasPhysicalDevicePane;

HtmlDiv uINasVolumesPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UINasVolumesPane;

HtmlDiv uINasQtreePane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UINasQtreePane;

HtmlDiv uISharesPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UISharesPane;

HtmlDiv uILocalUsersPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UILocalUsersPane;

HtmlDiv uILocalGroupsPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UILocalGroupsPane;

HtmlDiv uIPrinterQueuesPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIPrinterQueuesPane;

HtmlDiv uIOrphansPane = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIOrphansPane;

HtmlCell uIHostnameCell = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIItemTable.UIHostnameCell;

HtmlCell uINastam02ctam1Cell = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIItemTable.UINastam02ctam1Cell;

HtmlCell uIrutvnassec0003Cell = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIItemTable.UIrutvnassec0003Cell;

HtmlCell UIamsgtssf01Cell = this.UIHostDashboardOSTOperWindow.UIHostDashboardOSTOperDocument.UIItemTable.UIamsgtssf01Cell;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Host Details' pane

uINasVirtualDevicesPane.WaitForControlExist(1000);

// Click 'NasVirtualDevices' pane

Assert.IsTrue(uINasVirtualDevicesPane.Exists);

// Click 'NasPhysicalDevice' pane

Assert.IsTrue(uINasVirtualDevicesPane.Exists);

// Verify 'NasVolumes' pane

Assert.IsTrue(uINasVolumesPane.Exists);

// Click 'NasQtree' pane

Assert.IsTrue(uINasVolumesPane.Exists);

// Click 'Shares' pane

Assert.IsTrue(uINasQtreePane.Exists);

// Verify 'LocalUsers' pane

Assert.IsTrue(uILocalUsersPane.Exists);

// Verify 'Local Groups' pane

Assert.IsTrue(uINasVirtualDevicesPane.Exists);

//Verify 'Printer Queues' pane

Assert.IsTrue(uIPrinterQueuesPane.Exists);

// Verify 'Orphans' pane

Assert.IsTrue(uIOrphansPane.Exists);

//Verify 'Hostname' cell

Assert.IsTrue(uIHostnameCell.Exists);

//if (Run\_Environment != "OST\_PROD")

Assert.IsTrue(uINastam02ctam1Cell.Exists);

//else

// Assert.IsTrue(UIamsgtssf01Cell.Exists);

//if (Run\_Environment == "OST\_PROD")

//{

// // Verify 'nastam02ctam1'

// Assert.IsTrue(uIrutvnassec0003Cell.Exists);

//}

//else if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT"))

//{

// //Verify 'nasrut151v1crp'

// Assert.IsTrue(uINastam02ctam1Cell.Exists);

//}

}

private Verify\_Elements\_User\_RightsExpectedValues mVerify\_Elements\_User\_RightsExpectedValues;

/// <summary>

/// Verify\_Elements\_Orphans - Use 'Verify\_Elements\_OrphansExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Orphans()

{

#region Variable Declarations

HtmlControl uIOrphansPane = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIOrphansPane;

HtmlLabel uIRegionLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIOrphanTypeLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UISearchFormCustom.UIOrphanTypeLabel;

HtmlLabel uILogDateLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UISearchFormCustom.UILogDateLabel;

HtmlLabel uIServerTypeLabel = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UISearchFormCustom.UIServerTypeLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIHostNameEdit;

HtmlComboBox uIOrphanTypeSelectComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIOrphanTypeSelectComboBox;

HtmlEdit uILogDateDivDtPickEdit = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UILogDateDivDtPickEdit;

HtmlComboBox uIServerTypeSelectComboBox = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UIServerTypeSelectComboBox;

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIOrphansOSTOperationsDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Orphans' pane equals 'Orphans'

StringAssert.StartsWith(uIOrphansPane.InnerText, this.Verify\_Elements\_OrphansExpectedValues.UIOrphansPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays details around orphan shares. Page provides the ability to search by orphan type, log date and server type. ");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_OrphansExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Orphan Type:' label starts with 'Orphan Type:'

StringAssert.StartsWith(uIOrphanTypeLabel.DisplayText, this.Verify\_Elements\_OrphansExpectedValues.UIOrphanTypeLabelDisplayText);

// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

StringAssert.StartsWith(uILogDateLabel.DisplayText, this.Verify\_Elements\_OrphansExpectedValues.UILogDateLabelDisplayText);

// Verify that the 'DisplayText' property of 'Server Type:' label starts with 'Server Type:'

StringAssert.StartsWith(uIServerTypeLabel.DisplayText, this.Verify\_Elements\_OrphansExpectedValues.UIServerTypeLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'OrphanTypeSelect' combo box equals 'OrphanTypeSelect'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIOrphanTypeSelectComboBoxId, uIOrphanTypeSelectComboBox.Id);

// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UILogDateDivDtPickEditId, uILogDateDivDtPickEdit.Id);

// Verify that the 'Id' property of 'ServerTypeSelect' combo box equals 'ServerTypeSelect'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UIServerTypeSelectComboBoxId, uIServerTypeSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_OrphansExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_OrphansExpectedValues Verify\_Elements\_OrphansExpectedValues

{

get

{

if ((this.mVerify\_Elements\_OrphansExpectedValues == null))

{

this.mVerify\_Elements\_OrphansExpectedValues = new Verify\_Elements\_OrphansExpectedValues();

}

return this.mVerify\_Elements\_OrphansExpectedValues;

}

}

private Verify\_Elements\_OrphansExpectedValues mVerify\_Elements\_OrphansExpectedValues;

/// <summary>

/// Verify\_Elements\_Server\_Scheduled\_Tasks - Use 'Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Server\_Scheduled\_Tasks()

{

#region Variable Declarations

HtmlControl uIScheduledTasksPane = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIScheduledTasksPane;

HtmlLabel uIRegionLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uITaskNameLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UISearchFormCustom.UITaskNameLabel;

HtmlLabel uILastResultLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UISearchFormCustom.UILastResultLabel;

HtmlLabel uIStateLabel = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UISearchFormCustom.UIStateLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIHostNameEdit;

HtmlEdit uITaskNameSearchEdit = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UITaskNameSearchEdit;

HtmlComboBox uILastResultComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UILastResultComboBox;

HtmlComboBox uIStateComboBox = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UIStateComboBox;

HtmlButton uISearchButton = this.UIScheduledTasksOSTOpeWindow.UIScheduledTasksOSTOpeDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Scheduled Tasks' pane equals 'Scheduled Tasks'

StringAssert.StartsWith(uIScheduledTasksPane.InnerText, this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIScheduledTasksPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays task status including next run time and whether last task was a success or failure for Wintel servers.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Task Name:' label starts with 'Task Name:'

StringAssert.StartsWith(uITaskNameLabel.DisplayText, this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UITaskNameLabelDisplayText);

// Verify that the 'DisplayText' property of 'Last Result:' label starts with 'Last Result:'

StringAssert.StartsWith(uILastResultLabel.DisplayText, this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UILastResultLabelDisplayText);

// Verify that the 'DisplayText' property of 'State:' label starts with 'State:'

StringAssert.StartsWith(uIStateLabel.DisplayText, this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIStateLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'taskNameSearch' text box equals 'taskNameSearch'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UITaskNameSearchEditId, uITaskNameSearchEdit.Id);

// Verify that the 'Id' property of 'lastResult' combo box equals 'lastResult'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UILastResultComboBoxId, uILastResultComboBox.Id);

// Verify that the 'Id' property of 'state' combo box equals 'state'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UIStateComboBoxId, uIStateComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Server\_Scheduled\_TasksExpectedValues == null))

{

this.mVerify\_Elements\_Server\_Scheduled\_TasksExpectedValues = new Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues();

}

return this.mVerify\_Elements\_Server\_Scheduled\_TasksExpectedValues;

}

}

private Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues mVerify\_Elements\_Server\_Scheduled\_TasksExpectedValues;

/// <summary>

/// Verify\_Elements\_Share\_Exclusion - Use 'Verify\_Elements\_Share\_ExclusionExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Share\_Exclusion()

{

#region Variable Declarations

HtmlControl uIGlobalExclusionsPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHomeShareExclusionGlPane.UIGlobalExclusionsPane;

HtmlControl uIExclusionsForTeamsPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIISpanPane.UIExclusionsForTeamsPane;

HtmlInputButton uIAddShareExclusionButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddShareExclusionButton;

HtmlInputButton uIAddPathExclusionButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddPathExclusionButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Global Exclusions' pane starts with 'Global Exclusions'

StringAssert.StartsWith(uIGlobalExclusionsPane.InnerText, this.Verify\_Elements\_Share\_ExclusionExpectedValues.UIGlobalExclusionsPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides ability to exclude shares from being picked up by OST Monitors.");

// Verify that the 'DisplayText' property of 'Exclusions For Teams' pane starts with 'Exclusions For Teams'

StringAssert.StartsWith(uIExclusionsForTeamsPane.InnerText, this.Verify\_Elements\_Share\_ExclusionExpectedValues.UIExclusionsForTeamsPaneDisplayText);

// Verify that the 'DisplayText' property of 'Add Share Exclusion' button equals 'Add Share Exclusion'

Assert.AreEqual(this.Verify\_Elements\_Share\_ExclusionExpectedValues.UIAddShareExclusionButtonDisplayText, uIAddShareExclusionButton.DisplayText);

// Verify that the 'DisplayText' property of 'Add Path Exclusion' button equals 'Add Path Exclusion'

Assert.AreEqual(this.Verify\_Elements\_Share\_ExclusionExpectedValues.UIAddPathExclusionButtonDisplayText, uIAddPathExclusionButton.DisplayText);

}

public virtual Verify\_Elements\_Share\_ExclusionExpectedValues Verify\_Elements\_Share\_ExclusionExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Share\_ExclusionExpectedValues == null))

{

this.mVerify\_Elements\_Share\_ExclusionExpectedValues = new Verify\_Elements\_Share\_ExclusionExpectedValues();

}

return this.mVerify\_Elements\_Share\_ExclusionExpectedValues;

}

}

private Verify\_Elements\_Share\_ExclusionExpectedValues mVerify\_Elements\_Share\_ExclusionExpectedValues;

/// <summary>

/// Verify\_Elements\_Manage\_Monitor\_Account - Use 'Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Manage\_Monitor\_Account()

{

#region Variable Declarations

HtmlControl uIManageMonitorAccountPane = this.UIPrintQueueHistoryOSTWindow.UIManageMonitorAccountDocument.UIManageBCrumbPane.UIManageMonitorAccountPane;

HtmlInputButton uIAddAccountButton = this.UIPrintQueueHistoryOSTWindow.UIManageMonitorAccountDocument.UIAddAccountButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'DisplayText' property of 'Manage Monitor Account' pane starts with 'Manage Monitor Account'

StringAssert.StartsWith(uIManageMonitorAccountPane.InnerText, this.Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues.UIManageMonitorAccountPaneDisplayText);

Assert.AreEqual(UIHeaderTooltip.Title, "Allows to manage OST scanning functional account.");

// Verify that the 'DisplayText' property of 'Add Account' button equals 'Add Account'

//Assert.AreEqual(this.Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues.UIAddAccountButtonDisplayText, uIAddAccountButton.DisplayText);

}

public virtual Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Manage\_Monitor\_AccountExpectedValues == null))

{

this.mVerify\_Elements\_Manage\_Monitor\_AccountExpectedValues = new Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues();

}

return this.mVerify\_Elements\_Manage\_Monitor\_AccountExpectedValues;

}

}

private Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues mVerify\_Elements\_Manage\_Monitor\_AccountExpectedValues;

public void Verify\_Elements\_Sharepoint\_Info()

{

#region Variable Declarations

HtmlControl uiSharepointInformatioPane = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISharepointInformatioPane;

HtmlLabel uiRegionLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UIRegionLabel;

HtmlLabel uiTeamLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UITeamLabel;

HtmlLabel uiHostLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UIHostLabel;

HtmlComboBox uiRegionsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UITeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIHostNameEdit;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

#endregion

Assert.AreEqual("Sharepoint Information", uiSharepointInformatioPane.InnerText.Trim(), true);

Assert.AreEqual("Region:", uiRegionLabel.InnerText.Trim(), true);

Assert.AreEqual("Team:", uiTeamLabel.InnerText.Trim(), true);

Assert.AreEqual("Host:", uiHostLabel.InnerText.Trim(), true);

Assert.AreEqual("availableRegionsDdl", uiRegionsDdlComboBox.Id, true);

Assert.AreEqual("availableTeamsDdl", uiTeamsDdlComboBox.Id, true);

Assert.AreEqual("hostName", uiHostNameEdit.Id, true);

Assert.AreEqual("btnSearch", uiSearchButton.Id, true);

}

public void Verify\_Elements\_NAS\_PhysicalDevice\_Info()

{

#region Variable Declarations

HtmlControl UINASPhysicalDevicePane = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UINASPhysicalDeviceInfPane;

HtmlLabel uiRegionLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UIRegionLabel;

HtmlLabel uiTeamLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UITeamLabel;

HtmlLabel uiHostLabel = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UIBaseSearchRegionApPane.UIHostLabel;

HtmlComboBox uiRegionsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uiTeamsDdlComboBox = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uiHostNameEdit = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UIHostNameEdit;

HtmlButton uiSearchButton = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

Assert.AreEqual("NAS Physical Device Information", UINASPhysicalDevicePane.InnerText.Trim(), true);

Assert.AreEqual("Region:", uiRegionLabel.InnerText.Trim(), true);

Assert.AreEqual("Team:", uiTeamLabel.InnerText.Trim(), true);

Assert.AreEqual("Host:", uiHostLabel.InnerText.Trim(), true);

Assert.AreEqual("availableRegionsDdl", uiRegionsDdlComboBox.Id, true);

Assert.AreEqual("availableTeamsDdl", uiTeamsDdlComboBox.Id, true);

Assert.AreEqual("hostName", uiHostNameEdit.Id, true);

Assert.AreEqual("btnSearch", uiSearchButton.Id, true);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays NAS Physical Device information for hosts by team. ");

}

/// <summary>

/// Add\_Volume\_Exception - Use 'Add\_Volume\_ExceptionParams' to pass parameters into this method.

/// </summary>

public void Add\_Volume\_Exception(String Host, String Volume, String CostCode, String CsiID, String Comment)

{

#region Variable Declarations

HtmlEdit uIVirtualDeviceAutoComEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIVirtualDeviceAutoComEdit;

HtmlCustom uIUiid26Custom = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIUiid26Custom;

HtmlCustom nasny34v11cap = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasny34v11cap;

HtmlEdit uIVolumeAutoCompPopupEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIVolumeAutoCompPopupEdit;

HtmlCustom tfi\_nt\_home\_52 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.tfi\_nt\_home\_52;

HtmlEdit uIItemEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIContactsListPane.UIItemEdit;

HtmlEdit uIItemEdit1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIContactsListPane.UIItemEdit1;

HtmlEdit uIItemEdit2 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIContactsListPane.UIItemEdit2;

HtmlTextArea uIItemEdit3 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIContactsListPane.UIItemEdit3;

HtmlButton uISaveButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIAddEditContentPane1.UISaveButton;

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

#endregion

// Type 'nasrut151v1cr' in 'virtualDeviceAutoComp' text box

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIVirtualDeviceAutoComEdit.Text = strhost;

// Click 'ui-id-26' custom control

Mouse.Click(nasny34v11cap);

// Type 'swdcitv001' in 'volumeAutoCompPopup' text box

string strvolume = Volume;

strvolume = strvolume.Remove(strvolume.Length - 1, 1);

uIVolumeAutoCompPopupEdit.Text = strvolume;

// Click 'tfi\_nt\_home\_52' custom control

Mouse.Click(tfi\_nt\_home\_52);

// Type '456' in text box

uIItemEdit.Text = CostCode;

// Type '100' in text box

uIItemEdit1.Text = this.Add\_Volume\_ExceptionParams.UIItemEdit1Text;

// Type '123' in text box

uIItemEdit2.Text = CsiID;

// Type 'Add Volume Exception' in text box

uIItemEdit3.Text = Comment;

// Click 'Save' button

Mouse.Click(uISaveButton);

uIChargebackExceptionsPane.WaitForControlExist(5000);

}

public virtual Add\_Volume\_ExceptionParams Add\_Volume\_ExceptionParams

{

get

{

if ((this.mAdd\_Volume\_ExceptionParams == null))

{

this.mAdd\_Volume\_ExceptionParams = new Add\_Volume\_ExceptionParams();

}

return this.mAdd\_Volume\_ExceptionParams;

}

}

private Add\_Volume\_ExceptionParams mAdd\_Volume\_ExceptionParams;

/// <summary>

/// Open\_Add\_Exception\_Popup

/// </summary>

public void Open\_Add\_Exception\_Popup()

{

#region Variable Declarations

HtmlButton uIAddExceptionButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane1.UIAddExceptionButton;

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

#endregion

uIChargebackExceptionsPane.WaitForControlExist(5000);

// Click 'Add Exception' button

Mouse.Click(uIAddExceptionButton);

}

/// <summary>

/// Edit\_Exception\_Comment - Use 'Edit\_Exception\_CommentParams' to pass parameters into this method.

/// </summary>

public void Edit\_Exception\_Comment(String EditedComment)

{

#region Variable Declarations

HtmlTextArea uICommentEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UICommentEdit;

HtmlButton uISaveButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDetailsContentPane.UISaveButton;

#endregion

// Type 'Edit Volume Exception' in 'comment' text box

uICommentEdit.Text = EditedComment;

// Click 'Save' button

Mouse.Click(uISaveButton);

}

public virtual Edit\_Exception\_CommentParams Edit\_Exception\_CommentParams

{

get

{

if ((this.mEdit\_Exception\_CommentParams == null))

{

this.mEdit\_Exception\_CommentParams = new Edit\_Exception\_CommentParams();

}

return this.mEdit\_Exception\_CommentParams;

}

}

private Edit\_Exception\_CommentParams mEdit\_Exception\_CommentParams;

/// <summary>

/// Search\_Deleted\_Volume\_Exception - Use 'Search\_Deleted\_Volume\_ExceptionParams' to pass parameters into this method.

/// </summary>

public void Search\_Deleted\_Exception()

{

#region Variable Declarations

HtmlDiv uIHostNamePane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNamePane;

HtmlEdit uIHostNameEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNameEdit;

HtmlCustom uIUiid190Custom = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIUiid190Custom;

HtmlEdit uIVolumeNameBxEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIVolumeNameBxEdit;

HtmlCustom uIUiid199Custom = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIUiid199Custom;

HtmlCheckBox uIIsDeletedCkbxCheckBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIIsDeletedCkbxCheckBox;

HtmlButton uISearchButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane2.UISearchButton;

#endregion

// Type 'nasrut151v1cr' in 'hostName' text box

//string strhost = Host;

//strhost = strhost.Remove(strhost.Length - 1, 1);

//uIHostNameEdit.Text = strhost;

// Click 'ui-id-190' custom control

//uIUiid190Custom.WaitForControlReady(3000);

//Mouse.Click(uIUiid190Custom);

// Type '' in 'volumeNameBx' text box

//string strvolume = Volume;

//strvolume = strvolume.Remove(strvolume.Length - 1, 1);

//uIVolumeNameBxEdit.Text = strvolume;

// Click 'ui-id-199' custom control

//uIUiid199Custom.WaitForControlReady(3000);

//Mouse.Click(uIUiid199Custom);

// Select 'isDeletedCkbx' check box

uIIsDeletedCkbxCheckBox.Checked = true;

// Click 'Search' button

Mouse.Click(uISearchButton);

}

public void Verify\_Monitor\_popup\_Daily\_Queue\_Tab\_UAT()

{

#region Variable Declarations

HtmlCustom uIDailyQueueListNoTask = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIDailyQueueListNoTask;

HtmlCustom UIDailyQueueListNoCustom = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIDailyQueueListNoCustom;

#endregion

// Verify if the text 'There are no daily tasks that are queued for execution' or 'This list shows daily tasks that are queued for execution'

//if ((UIDailyQueueListNoCustom.Height > 0) || (uIDailyQueueListNoTask.Height > 0))

//{

// Console.WriteLine("Daily Queue Tab loaded properly");

//}

bool booldailyqueue = (UIDailyQueueListNoCustom.Height > 0) || (uIDailyQueueListNoTask.Height > 0);

if (booldailyqueue == false)

{

Console.WriteLine("Daily Queue Monitor page is not displayed");

Assert.Fail();

}

//Assert.Equals(booldailyqueue, true);

}

public void Verify\_Monitor\_popup\_Weekly\_Queue\_Tab\_PROD()

{

#region Variable Declarations

HtmlCustom uIWeeklyQueueListNoTask = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIWeeklyQueueListNoTask;

HtmlCustom uIWeeklyQueueListNoCustom = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIWeeklyQueueListNoCustom;

#endregion

// Verify if the text 'There are no daily tasks that are queued for execution' or 'This list shows daily tasks that are queued for execution'

//if ((UIDailyQueueListNoCustom.Height > 0) || (uIDailyQueueListNoTask.Height > 0))

//{

// Console.WriteLine("Daily Queue Tab loaded properly");

//}

bool booldailyqueue = (uIWeeklyQueueListNoCustom.Height > 0) || (uIWeeklyQueueListNoTask.Height > 0);

if (booldailyqueue == false)

{

Console.WriteLine("Daily Queue Monitor page is not displayed");

Assert.Fail();

}

//Assert.Equals(booldailyqueue, true);

}

/// <summary>

/// Search\_Added\_Volume\_Exception

/// </summary>

public void Search\_Added\_Volume\_Exception(String Host, String Volume)

{

#region Variable Declarations

HtmlDiv uIHostNamePane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNamePane;

HtmlEdit uIHostNameEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNameEdit;

HtmlCustom uIUiid190Custom = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIUiid190Custom;

HtmlCustom nasny34v11cap = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasny34v11cap;

HtmlCustom tfi\_nt\_home\_52 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.tfi\_nt\_home\_52;

HtmlEdit uIVolumeNameBxEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIVolumeNameBxEdit;

HtmlCustom uIUiid199Custom = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIUiid199Custom;

HtmlCheckBox uIIsDeletedCkbxCheckBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIIsDeletedCkbxCheckBox;

HtmlButton uISearchButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane2.UISearchButton;

#endregion

uIHostNameEdit.WaitForControlReady(3000);

// Type 'nasrut151v1cr' in 'hostName' text box

string strhost = Host;

strhost = strhost.Remove(strhost.Length - 1, 1);

uIHostNameEdit.Text = strhost;

Console.WriteLine("Entered " + strhost + " into host field.");

// Click 'ui-id-190' custom control

nasny34v11cap.WaitForControlExist(3000);

//Mouse.Click(nasny34v11cap);

Keyboard.SendKeys("{Down}" + "{Enter}");

// Type '' in 'volumeNameBx' text box

string strvolume = Volume;

strvolume = strvolume.Remove(strvolume.Length - 1, 1);

uIVolumeNameBxEdit.Text = strvolume;

// Click 'ui-id-199' custom control

tfi\_nt\_home\_52.WaitForControlReady(3000);

Mouse.Click(tfi\_nt\_home\_52);

// Click 'Search' button

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Added\_Share\_Exception

/// </summary>

public void Search\_Added\_Share\_Exception(String Host, String Share)

{

#region Variable Declarations

HtmlDiv uIHostNamePane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNamePane;

HtmlEdit uIHostNameEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIHostNameEdit;

HtmlCustom nasny34v11cap = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasny34v11cap;

HtmlCustom SS81280 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.SS81280;

HtmlEdit uIShareNameBxEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIShareNameBxEdit;

HtmlCheckBox uIIsDeletedCkbxCheckBox = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIIsDeletedCkbxCheckBox;

HtmlButton uISearchButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UISearchSubDivPane2.UISearchButton;

#endregion

uIHostNameEdit.WaitForControlReady(5000);

// Type 'nasny34v11cap' in 'hostName' text box

uIHostNameEdit.Text = Host;

Console.WriteLine("Entered " + Host + " into host field.");

// Click 'nasny34v11cap' custom control

nasny34v11cap.WaitForControlExist(5000);

Keyboard.SendKeys("{DOWN}" + "{ENTER}");

// Type '' in 'ShareNameBx' text box

string strshare = Share;

strshare = strshare.Remove(strshare.Length - 1, 1);

uIShareNameBxEdit.Text = strshare;

Console.WriteLine("Entered " + strshare + " into share field.");

// Click 'dv01382' custom control

SS81280.WaitForControlReady(5000);

if (BrowserWindow.CurrentBrowser == "Chrome")

{

Mouse.Click(SS81280);

}

else

{

Keyboard.SendKeys("{DOWN}" + "{ENTER}");

}

// Click 'Search' button

Mouse.Click(uISearchButton);

}

/// <summary>

/// Verify\_Edit\_Exception\_popup - Use 'Verify\_Edit\_Exception\_popupExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Edit\_Exception\_popup()

{

#region Variable Declarations

HtmlSpan uIExceptionDetailsEditPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIExceptionDetailsEditPane;

HtmlTextArea uICommentEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UICommentEdit;

HtmlButton uISaveButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDetailsContentPane.UISaveButton;

HtmlButton uICloseButton1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UICloseButton1;

#endregion

// Verify that the 'InnerText' property of 'Exception Details - Edit' pane equals 'Exception Details - Edit'

Assert.AreEqual(this.Verify\_Edit\_Exception\_popupExpectedValues.UIExceptionDetailsEditPaneInnerText, uIExceptionDetailsEditPane.InnerText);

// Verify that the 'Id' property of 'comment' text box equals 'comment'

Assert.AreEqual(this.Verify\_Edit\_Exception\_popupExpectedValues.UICommentEditId, uICommentEdit.Id);

// Verify that the 'TagName' property of 'Save' button equals 'BUTTON'

Assert.AreEqual(this.Verify\_Edit\_Exception\_popupExpectedValues.UISaveButtonTagName, uISaveButton.TagName);

// Verify that the 'TagName' property of 'Close' button equals 'BUTTON'

Assert.AreEqual(this.Verify\_Edit\_Exception\_popupExpectedValues.UICloseButton1TagName, uICloseButton1.TagName);

}

public virtual Verify\_Edit\_Exception\_popupExpectedValues Verify\_Edit\_Exception\_popupExpectedValues

{

get

{

if ((this.mVerify\_Edit\_Exception\_popupExpectedValues == null))

{

this.mVerify\_Edit\_Exception\_popupExpectedValues = new Verify\_Edit\_Exception\_popupExpectedValues();

}

return this.mVerify\_Edit\_Exception\_popupExpectedValues;

}

}

private Verify\_Edit\_Exception\_popupExpectedValues mVerify\_Edit\_Exception\_popupExpectedValues;

/// <summary>

/// Verify\_Delete\_Group\_popup

/// </summary>

public void Verify\_Delete\_Group\_popup(String Group, String Name, String Description)

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlTable DeleteTable = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable;

#endregion

uIDeleteButton.WaitForControlExist(5000);

Console.WriteLine("Delete popup loaded.");

Console.WriteLine("Delete popup Row 0 Column 1: " + DeleteTable.GetCell(0, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 1 Column 1: " + DeleteTable.GetCell(1, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 2 Column 1: " + DeleteTable.GetCell(2, 1).InnerText.ToString());

StringAssert.Contains(DeleteTable.GetCell(0, 1).InnerText.ToString(), Group);

StringAssert.Contains(DeleteTable.GetCell(1, 1).InnerText.ToString(), Name);

StringAssert.Contains(DeleteTable.GetCell(2, 1).InnerText.ToString(), Description);

Assert.AreEqual(this.Verify\_Delete\_Exception\_popupExpectedValues.UIDeleteButtonId, uIDeleteButton.Id);

}

/// <summary>

/// Verify\_Delete\_Share\_Exclusion\_popup

/// </summary>

public void Verify\_Delete\_Share\_Exclusion\_popup(String Name, String Description)

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlTable DeleteTable = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable;

#endregion

uIDeleteButton.WaitForControlExist(5000);

Console.WriteLine("Delete popup loaded.");

Console.WriteLine("Delete popup Row 0 Column 1: " + DeleteTable.GetCell(0, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 1 Column 1: " + DeleteTable.GetCell(1, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 2 Column 1: " + DeleteTable.GetCell(2, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 3 Column 1: " + DeleteTable.GetCell(3, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 4 Column 1: " + DeleteTable.GetCell(4, 1).InnerText.ToString());

StringAssert.Contains(DeleteTable.GetCell(2, 1).InnerText.ToString(), Name);

StringAssert.Contains(DeleteTable.GetCell(3, 1).InnerText.ToString(), Description);

StringAssert.Contains(DeleteTable.GetCell(4, 1).InnerText.ToString(), "Share");

Assert.AreEqual(this.Verify\_Delete\_Exception\_popupExpectedValues.UIDeleteButtonId, uIDeleteButton.Id);

}

/// <summary>

/// Verify\_Delete\_Path\_Exclusion\_popup

/// </summary>

public void Verify\_Delete\_Path\_Exclusion\_popup(String Path, String Description)

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlTable DeleteTable = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable;

#endregion

uIDeleteButton.WaitForControlExist(5000);

Console.WriteLine("Delete popup loaded.");

Console.WriteLine("Delete popup Row 0 Column 1: " + DeleteTable.GetCell(0, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 1 Column 1: " + DeleteTable.GetCell(1, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 2 Column 1: " + DeleteTable.GetCell(2, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 3 Column 1: " + DeleteTable.GetCell(3, 1).InnerText.ToString());

StringAssert.Contains(DeleteTable.GetCell(1, 1).InnerText.ToString(), Path);

StringAssert.Contains(DeleteTable.GetCell(2, 1).InnerText.ToString(), Description);

StringAssert.Contains(DeleteTable.GetCell(3, 1).InnerText.ToString(), "Path");

Assert.AreEqual(this.Verify\_Delete\_Exception\_popupExpectedValues.UIDeleteButtonId, uIDeleteButton.Id);

}

/// <summary>

/// Verify\_Delete\_Exception\_popup - Use 'Verify\_Delete\_Exception\_popupExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Delete\_Exception\_popup(String Host, String VolumeOrShare, String EditedComment)

{

#region Variable Declarations

HtmlSpan uIExceptionDeletePane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIExceptionDeletePane;

HtmlCell uINasrut151v1crpCell = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable.UINasrut151v1crpCell;

HtmlCell uISwdcitv0010Cell = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable.UISwdcitv0010Cell;

HtmlCell uIEditVolumeExceptionCell = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable.UIEditVolumeExceptionCell;

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlTable DeleteTable = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteContentPane.UIItemTable;

#endregion

// Verify that the 'InnerText' property of 'Exception - Delete' pane equals 'Exception - Delete'

Assert.AreEqual(this.Verify\_Delete\_Exception\_popupExpectedValues.UIExceptionDeletePaneInnerText, uIExceptionDeletePane.InnerText);

// Verify that the 'InnerText' property of 'nasrut151v1crp' cell starts with 'nasrut151v1crp'

Console.WriteLine("Delete popup Row 0 Column 1: " + DeleteTable.GetCell(0, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 1 Column 1: " + DeleteTable.GetCell(1, 1).InnerText.ToString());

Console.WriteLine("Delete popup Row 2 Column 1: " + DeleteTable.GetCell(2, 1).InnerText.ToString());

StringAssert.Contains(DeleteTable.GetCell(0, 1).InnerText.ToString(), Host);

// Verify that the 'InnerText' property of 'swdcitv0010' cell starts with 'swdcitv0010'

StringAssert.Contains(DeleteTable.GetCell(1, 1).InnerText.ToString(), VolumeOrShare);

// Verify that the 'InnerText' property of 'Edited Volume Exception' cell starts with 'Edited Volume Exception'

StringAssert.Contains(DeleteTable.GetCell(2, 1).InnerText.ToString(), EditedComment);

// Verify that the 'Id' property of 'Delete' button equals 'deleteButton'

Assert.AreEqual(this.Verify\_Delete\_Exception\_popupExpectedValues.UIDeleteButtonId, uIDeleteButton.Id);

}

public virtual Verify\_Delete\_Exception\_popupExpectedValues Verify\_Delete\_Exception\_popupExpectedValues

{

get

{

if ((this.mVerify\_Delete\_Exception\_popupExpectedValues == null))

{

this.mVerify\_Delete\_Exception\_popupExpectedValues = new Verify\_Delete\_Exception\_popupExpectedValues();

}

return this.mVerify\_Delete\_Exception\_popupExpectedValues;

}

}

private Verify\_Delete\_Exception\_popupExpectedValues mVerify\_Delete\_Exception\_popupExpectedValues;

/// <summary>

/// Click\_Delete\_Exception\_button

/// </summary>

public void Click\_Delete\_Exception\_button()

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

#endregion

// Click 'Delete' button

Mouse.Click(uIDeleteButton);

Console.WriteLine("Clicked Delete button");

uIChargebackExceptionsPane.WaitForControlExist(5000);

}

/// <summary>

/// Click\_Delete\_Group\_button

/// </summary>

public void Click\_Delete\_Group\_button()

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlLabel uIProcessingLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIProcessingPane.UIProcessingLabel;

#endregion

// Click 'Delete' button

Mouse.Click(uIDeleteButton);

Console.WriteLine("Clicked Delete button");

uIProcessingLabel.WaitForControlExist(5000);

}

/// <summary>

/// Click\_Delete\_Exclusion\_button

/// </summary>

public void Click\_Delete\_Exclusion\_button()

{

#region Variable Declarations

HtmlButton uIDeleteButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIDeleteButton;

HtmlControl uIGlobalExclusionsPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHomeShareExclusionGlPane.UIGlobalExclusionsPane;

#endregion

// Click 'Delete' button

Mouse.Click(uIDeleteButton);

Console.WriteLine("Clicked Delete button");

//Playback.Wait(3000);

uIGlobalExclusionsPane.WaitForControlExist(5000);

}

/// <summary>

/// Verify\_Exclusion\_deleted

/// </summary>

public void Verify\_Exclusion\_deleted()

{

#region Variable Declarations

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

#endregion

Assert.AreEqual(1, uIJqGridTable.RowCount);

Console.WriteLine("Confirmed that the Exclusion has been deleted.");

}

/// <summary>

/// Navigate\_to\_Chargeback\_Exceptions

/// </summary>

public void Navigate\_to\_Chargeback\_Exceptions()

{

#region Variable Declarations

HtmlHyperlink uIChargebackHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane.UIChargebackHyperlink;

HtmlHyperlink uIChargebackExceptionsHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane.UIChargebackExceptionsHyperlink;

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

#endregion

Playback.Wait(3000);

// Click 'Chargeback' link

Mouse.Click(uIChargebackHyperlink);

// Click 'Chargeback Exceptions' link

Mouse.Click(uIChargebackExceptionsHyperlink);

uIChargebackExceptionsPane.WaitForControlExist(5000);

}

/// <summary>

/// Add\_Share\_Exception - Use 'Add\_Share\_ExceptionParams' to pass parameters into this method.

/// </summary>

public void Add\_Share\_Exception(String Host, String Share, String CostCode, String CsiID, String Comment)

{

#region Variable Declarations

HtmlEdit uIVirtualDeviceAutoComEdit1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIVirtualDeviceAutoComEdit1;

HtmlCustom nasny34v11cap = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.nasny34v11cap;

HtmlCustom SS81280 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.SS81280;

HtmlEdit uIShareAutoCompEdit1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIShareAutoCompEdit1;

HtmlEdit uIItemEdit4 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit4;

HtmlEdit uIItemEdit1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit1;

HtmlEdit uIItemEdit2 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit2;

HtmlTextArea uIItemEdit3 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit3;

HtmlButton uISaveButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIAddEditContentPane1.UISaveButton;

HtmlControl uIChargebackExceptionsPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument.UIChargebackExceptionsPane;

HtmlRadioButton uIExceptionTypeGroupRadioButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIExceptionTypeGroupRadioButton;

#endregion

// Select 'exceptionTypeGroup' radio button

uIExceptionTypeGroupRadioButton.Selected = true;

// Type 'nasrut151v1cr' in 'virtualDeviceAutoComp' text box

uIVirtualDeviceAutoComEdit1.Text = Host;

// Click 'nasny34v11cap' custom control

Mouse.Click(nasny34v11cap);

// Type 'aa93513' in 'shareAutoComp' text box

uIShareAutoCompEdit1.Text = Share;

SS81280.WaitForControlExist(5000);

// Click 'dv01382' custom control

//Mouse.Click(dv01382);

if (BrowserWindow.CurrentBrowser == "Chrome")

{

Mouse.Click(SS81280);

}

else

{

Keyboard.SendKeys("{DOWN}" + "{ENTER}");

}

// Type '123' in text box

uIItemEdit4.Text = CostCode;

// Type '100' in text box

uIItemEdit1.Text = "100";

// Type '456' in text box

uIItemEdit2.Text = CsiID;

// Type 'Added Share Exception' in text box

uIItemEdit3.Text = Comment;

// Click 'Save' button

Mouse.Click(uISaveButton);

uIChargebackExceptionsPane.WaitForControlExist(5000);

}

/// <summary>

/// Open\_Add\_Share\_Exception\_popup - Use 'Open\_Add\_Share\_Exception\_popupParams' to pass parameters into this method.

/// </summary>

public void Open\_Add\_Share\_Exception\_popup()

{

#region Variable Declarations

HtmlButton uIAddExceptionButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UISearchSubDivPane.UIAddExceptionButton;

HtmlSpan uIExceptionDetailsAddPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIExceptionDetailsAddPane;

#endregion

// Click 'Add Exception' button

Mouse.Click(uIAddExceptionButton);

uIExceptionDetailsAddPane.WaitForControlExist(5000);

}

/// <summary>

/// Navigate\_to\_Report\_Requests

/// </summary>

public void Navigate\_to\_Report\_Requests()

{

#region Variable Declarations

HtmlHyperlink uIReportHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane.UIReportHyperlink;

HtmlHyperlink uIReportRequestsHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane.UIReportRequestsHyperlink;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

// Click 'Report' link

Mouse.Click(uIReportHyperlink);

// Click 'Report Requests' link

Mouse.Click(uIReportRequestsHyperlink);

uIReportRequestsforEURPane.WaitForControlExist(5000);

}

/// <summary>

/// Security\_Groups\_wait\_for\_data\_load - Use 'Security\_Groups\_wait\_for\_data\_loadExpectedValues' to pass parameters into this method.

/// </summary>

public void Security\_Groups\_wait\_for\_data\_load()

{

#region Variable Declarations

HtmlLabel uIProcessingLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIProcessingPane.UIProcessingLabel;

HtmlHyperlink uIDeleteHyperlink = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIItem1Row.UIDeleteHyperlink;

HtmlTable uIJqGridTable = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

#endregion

string Loaded = "NotLoaded";

for (int i = 0; i < 120; i++)

{

Playback.Wait(1000);

uIJqGridTable.Find();

if (uIJqGridTable.RowCount - 1 != 0)

{

Loaded = "Loaded";

i = 120;

}

}

if (Loaded == "NotLoaded")

{

Assert.AreEqual("Loaded", Loaded, "Security Groups did not load within 120 seconds.");

}

uIProcessingLabel.WaitForControlNotExist(5000);

Console.WriteLine("Processing finished.");

uIDeleteHyperlink.WaitForControlExist(5000);

Console.WriteLine("Delete button now visible.");

Console.WriteLine("Records in table: " + (uIJqGridTable.RowCount - 1));

Assert.AreNotEqual(0, (uIJqGridTable.RowCount - 1), "No records found in table.");

}

public virtual Security\_Groups\_wait\_for\_data\_loadExpectedValues Security\_Groups\_wait\_for\_data\_loadExpectedValues

{

get

{

if ((this.mSecurity\_Groups\_wait\_for\_data\_loadExpectedValues == null))

{

this.mSecurity\_Groups\_wait\_for\_data\_loadExpectedValues = new Security\_Groups\_wait\_for\_data\_loadExpectedValues();

}

return this.mSecurity\_Groups\_wait\_for\_data\_loadExpectedValues;

}

}

private Security\_Groups\_wait\_for\_data\_loadExpectedValues mSecurity\_Groups\_wait\_for\_data\_loadExpectedValues;

/// <summary>

/// Add\_Security\_Group - Use 'Add\_Security\_GroupParams' to pass parameters into this method.

/// </summary>

public void Add\_Security\_Group(String Group)

{

#region Variable Declarations

HtmlDiv uIInsertedsuccessfullyPane = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIInsertedsuccessfullyPane;

HtmlComboBox uITeamSelectComboBox = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITeamSelectComboBox;

HtmlComboBox uIGroupDomainSelectComboBox = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIGroupDomainSelectComboBox;

HtmlEdit uIIGroupNameEdit = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIIGroupNameEdit;

HtmlEdit uIIDescriptionEdit = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIIDescriptionEdit;

HtmlInputButton uISaveButton = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UISaveButton;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

// Select '150080 - Application Support' in 'TeamSelect' combo box

uITeamSelectComboBox.SelectedItem = this.Add\_Security\_GroupParams.UITeamSelectComboBoxSelectedItem;

}

else if (Run\_Environment != "OST\_DEV")

{

uITeamSelectComboBox.SelectedItem = this.Add\_Security\_GroupParams.UITeamSelectComboBoxSelectedItem\_Prod;

}

// Select 'LACDEV' in 'GroupDomainSelect' combo box

uIGroupDomainSelectComboBox.SelectedItem = Group;

// Type 'AddDeleteTestGroupName' in 'iGroupName' text box

uIIGroupNameEdit.Text = this.Add\_Security\_GroupParams.UIIGroupNameEditText;

// Type 'AddDeleteTestGroupDescription' in 'iDescription' text box

uIIDescriptionEdit.Text = this.Add\_Security\_GroupParams.UIIDescriptionEditText;

// Click 'Save' button

Mouse.Click(uISaveButton);

Console.WriteLine("Save button clicked.");

uIInsertedsuccessfullyPane.WaitForControlExist(5000);

Console.WriteLine("Saved label shown.");

}

public virtual Add\_Security\_GroupParams Add\_Security\_GroupParams

{

get

{

if ((this.mAdd\_Security\_GroupParams == null))

{

this.mAdd\_Security\_GroupParams = new Add\_Security\_GroupParams();

}

return this.mAdd\_Security\_GroupParams;

}

}

private Add\_Security\_GroupParams mAdd\_Security\_GroupParams;

/// <summary>

/// Verify\_Add\_Security\_Group - Use 'Verify\_Add\_Security\_GroupExpectedValues' to pass parameters into this method.

/// </summary>

public void Open\_and\_Verify\_Add\_Security\_Group()

{

#region Variable Declarations

HtmlInputButton uIAddNewButton = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIAddNewButton;

HtmlDiv uIAddSecurityGroupsPane = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIAddBCrumbSecurityPane.UIAddSecurityGroupsPane;

HtmlLabel uITeamNameLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITblSecurityTable.UITeamNameLabel;

HtmlLabel uIGroupDomainLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITblSecurityTable.UIGroupDomainLabel;

HtmlLabel uIGroupNameLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITblSecurityTable.UIGroupNameLabel;

HtmlLabel uIGroupDescriptionLabel = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITblSecurityTable.UIGroupDescriptionLabel;

HtmlComboBox uITeamSelectComboBox = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITeamSelectComboBox;

HtmlComboBox uIGroupDomainSelectComboBox = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIGroupDomainSelectComboBox;

HtmlEdit uIIGroupNameEdit = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIIGroupNameEdit;

HtmlEdit uIIDescriptionEdit = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIIDescriptionEdit;

HtmlHyperlink uIBacktoGridHyperlink = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UITblSaveTable.UIBacktoGridHyperlink;

HtmlInputButton uISaveButton = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UISaveButton;

#endregion

Mouse.Click(uIAddNewButton);

uIAddSecurityGroupsPane.WaitForControlExist(5000);

// Verify that the 'InnerText' property of 'Add Security Groups' pane starts with 'Add Security Groups'

StringAssert.StartsWith(uIAddSecurityGroupsPane.InnerText, this.Verify\_Add\_Security\_GroupExpectedValues.UIAddSecurityGroupsPaneInnerText);

// Verify that the 'InnerText' property of 'Team Name:' label equals 'Team Name:'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UITeamNameLabelInnerText, uITeamNameLabel.InnerText);

// Verify that the 'InnerText' property of 'Group Domain:' label equals 'Group Domain:'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UIGroupDomainLabelInnerText, uIGroupDomainLabel.InnerText);

// Verify that the 'InnerText' property of 'Group Name:' label starts with 'Group Name:'

StringAssert.StartsWith(uIGroupNameLabel.InnerText, this.Verify\_Add\_Security\_GroupExpectedValues.UIGroupNameLabelInnerText);

// Verify that the 'InnerText' property of 'Group Description:' label starts with 'Group Description:'

StringAssert.StartsWith(uIGroupDescriptionLabel.InnerText, this.Verify\_Add\_Security\_GroupExpectedValues.UIGroupDescriptionLabelInnerText);

// Verify that the 'Id' property of 'TeamSelect' combo box equals 'TeamSelect'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UITeamSelectComboBoxId, uITeamSelectComboBox.Id);

// Verify that the 'Id' property of 'GroupDomainSelect' combo box equals 'GroupDomainSelect'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UIGroupDomainSelectComboBoxId, uIGroupDomainSelectComboBox.Id);

// Verify that the 'Id' property of 'iGroupName' text box equals 'iGroupName'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UIIGroupNameEditId, uIIGroupNameEdit.Id);

// Verify that the 'Id' property of 'iDescription' text box equals 'iDescription'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UIIDescriptionEditId, uIIDescriptionEdit.Id);

// Verify that the 'InnerText' property of 'Back to Grid' link equals 'Back to Grid'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UIBacktoGridHyperlinkInnerText, uIBacktoGridHyperlink.InnerText);

// Verify that the 'Id' property of 'Save' button equals 'btnSecurityGroupsSave'

Assert.AreEqual(this.Verify\_Add\_Security\_GroupExpectedValues.UISaveButtonId, uISaveButton.Id);

}

public virtual Verify\_Add\_Security\_GroupExpectedValues Verify\_Add\_Security\_GroupExpectedValues

{

get

{

if ((this.mVerify\_Add\_Security\_GroupExpectedValues == null))

{

this.mVerify\_Add\_Security\_GroupExpectedValues = new Verify\_Add\_Security\_GroupExpectedValues();

}

return this.mVerify\_Add\_Security\_GroupExpectedValues;

}

}

private Verify\_Add\_Security\_GroupExpectedValues mVerify\_Add\_Security\_GroupExpectedValues;

/// <summary>

/// Navigate\_to\_OST - Use 'Navigate\_to\_OSTParams' to pass parameters into this method.

/// </summary>

public void Navigate\_to\_OST()

{

#region Variable Declarations

HtmlHyperlink SharesMenu = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UISharesHyperlink;

WinText uIConnectingtovm8b5476Text = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIConnectingtovm8b5476Text;

WinEdit uIUsernameEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIUsernameEdit;

WinEdit uIPasswordEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIPasswordEdit;

WinButton uIOKButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIOKButton;

HtmlHyperlink uIContinuetothiswebsitHyperlink = this.UICertificateErrorNaviWindow.UICertificateErrorNaviDocument1.UIContinuetothiswebsitHyperlink;

WinButton uICloseButton = this.UIChromedriverexeWindow.UIChromedriverexeTitleBar.UICloseButton;

WinButton UICancelButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UICancelButton;

#endregion

foreach (var ie in Process.GetProcessesByName("iexplore"))

{

try

{

ie.Kill();

}

catch { }

}

foreach (var Chrome in Process.GetProcessesByName("Chrome"))

{

try

{

Chrome.Kill();

}

catch { }

}

foreach (var ie in Process.GetProcessesByName("chromedriver"))

{

try

{

ie.Kill();

}

catch { }

}

foreach (var ie in Process.GetProcessesByName("WerFault"))

{

try

{

ie.Kill();

}

catch { }

}

//Playback.PlaybackSettings.ResetToDefault();

// Go to web page 'http://vm-775e-1f0f.nam.nsroot.net:8100/Ostx/' using new browser instance

Playback.PlaybackSettings.ContinueOnError = true;

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OSTParams.Url);

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

else if (Run\_Environment == "OST\_UAT")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OSTParams.UAT\_URL);

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

else if (Run\_Environment == "OST\_PROD")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OSTParams.PROD\_URL);

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

try

{

if (uIContinuetothiswebsitHyperlink.Exists == true)

{

// Click 'Continue to this website (not recommended).' link

Mouse.Click(uIContinuetothiswebsitHyperlink);

}

}

catch { Console.WriteLine("Continue link not found"); }

try

{

if (UIWindowsSecurityWindow.Exists == true)

Mouse.Click(UICancelButton);

}

catch {

Console.WriteLine("Security Pop up not found");

}

SharesMenu.WaitForControlExist(6000);

Playback.PlaybackSettings.ContinueOnError = false;

if (BrowserWindow.CurrentBrowser != "Chrome")

{

if (SharesMenu.Exists == false)

{

if ((uIConnectingtovm8b5476Text.Exists == true) && (Run\_Environment == "OST\_DEV"))

{

uIUsernameEdit.Text = "eur\\kc90246";

Keyboard.SendKeys(uIPasswordEdit, "YPIXzues6ZKEPbC6dU0laP7kIQBE00sp", true);

Mouse.Click(uIOKButton);

}

}

}

}

public void Navigate\_to\_hidden\_IDrive\_page()

{

#region Variable Declarations

HtmlHyperlink SharesMenu = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UISharesHyperlink;

WinText uIConnectingtovm8b5476Text = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIConnectingtovm8b5476Text;

WinEdit uIUsernameEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIUsernameEdit;

WinEdit uIPasswordEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIPasswordEdit;

WinButton uIOKButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIOKButton;

HtmlHyperlink uIContinuetothiswebsitHyperlink = this.UICertificateErrorNaviWindow.UICertificateErrorNaviDocument1.UIContinuetothiswebsitHyperlink;

WinButton uICloseButton = this.UIChromedriverexeWindow.UIChromedriverexeTitleBar.UICloseButton;

WinButton UICancelButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UICancelButton;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

foreach (var ie in Process.GetProcessesByName("iexplore"))

{

try

{

ie.Kill();

}

catch { }

}

if (BrowserWindow.CurrentBrowser == "Chrome")

{

foreach (var Chrome in Process.GetProcessesByName("Chrome"))

{

try

{

Chrome.Kill();

}

catch { }

}

foreach (var ie in Process.GetProcessesByName("chromedriver"))

{

try

{

ie.Kill();

}

catch { }

}

foreach (var ie in Process.GetProcessesByName("WerFault"))

{

try

{

ie.Kill();

}

catch { }

}

}

//Playback.PlaybackSettings.ResetToDefault();

// Go to web page 'https://vm-775e-1f0f.nam.nsroot.net/Ostx/IDrive/' using new browser instance

Playback.PlaybackSettings.ContinueOnError = true;

if (Run\_Environment == "OST\_DEV")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch("https://vm-775e-1f0f.nam.nsroot.net/Ostx/IDrive/");

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

else if (Run\_Environment == "OST\_UAT")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch("https://ostuat.nam.nsroot.net/Ostx/IDrive/");

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

else if (Run\_Environment == "OST\_PROD")

{

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch("https://ost.nam.nsroot.net/Ostx/IDrive/");

vm8b54769dnamnsrootnetBrowser.SetFocus();

vm8b54769dnamnsrootnetBrowser.Maximized = true;

}

try

{

if (uIContinuetothiswebsitHyperlink.Exists == true)

{

// Click 'Continue to this website (not recommended).' link

Mouse.Click(uIContinuetothiswebsitHyperlink);

}

}

catch { Console.WriteLine("Continue link not found"); }

try

{

if (UIWindowsSecurityWindow.Exists == true)

Mouse.Click(UICancelButton);

}

catch

{

Console.WriteLine("Security Pop up not found");

}

SharesMenu.WaitForControlExist(6000);

Playback.PlaybackSettings.ContinueOnError = false;

if (BrowserWindow.CurrentBrowser != "Chrome")

{

if (SharesMenu.Exists == false)

{

if ((uIConnectingtovm8b5476Text.Exists == true) && (Run\_Environment == "OST\_DEV"))

{

uIUsernameEdit.Text = "eur\\kc90246";

Keyboard.SendKeys(uIPasswordEdit, "YPIXzues6ZKEPbC6dU0laP7kIQBE00sp", true);

Mouse.Click(uIOKButton);

}

}

}

HtmlDiv uIIDriveReportsPane = this.UIIDriveWindow.UIIDrivePage.UIIDriveReportsPane;

uIIDriveReportsPane.WaitForControlExist(10000);

}

//public void Navigate\_to\_OST\_UAT()

//{

// #region Variable Declarations

// HtmlHyperlink SharesMenu = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UISharesHyperlink;

// WinText uIConnectingtovm8b5476Text = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIConnectingtovm8b5476Text;

// WinEdit uIUsernameEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIUsernameEdit;

// WinEdit uIPasswordEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIPasswordEdit;

// WinButton uIOKButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIOKButton;

// #endregion

// // Go to web page 'https://operationsstandardtoolset.uat.citigroup.net/ostx' using new browser instance

// Playback.PlaybackSettings.ContinueOnError = true;

// BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OST\_UATParams.Url);

// SharesMenu.WaitForControlExist(5000);

// Playback.PlaybackSettings.ContinueOnError = false;

// if (BrowserWindow.CurrentBrowser != "Chrome")

// {

// if (SharesMenu.Exists == false)

// {

// if (uIConnectingtovm8b5476Text.Exists == true)

// {

// uIUsernameEdit.Text = "sp90104";

// Keyboard.SendKeys(uIPasswordEdit, "Sujee@1234", true);

// Mouse.Click(uIOKButton);

// }

// }

// }

// //System.Environment.SetEnvironmentVariable("OST\_Env", "OST\_UAT");

//}

//public void Navigate\_to\_OST\_PROD()

//{

// #region Variable Declarations

// HtmlHyperlink SharesMenu = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UISharesHyperlink;

// WinText uIConnectingtovm8b5476Text = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIConnectingtovm8b5476Text;

// WinEdit uIUsernameEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIUsernameEdit;

// WinEdit uIPasswordEdit = this.UIWindowsSecurityWindow.UIUseanotheraccountListItem.UIPasswordEdit;

// WinButton uIOKButton = this.UIWindowsSecurityWindow.UIWindowsSecurityPane.UIOKButton;

// #endregion

// // Go to web page 'https://operationsstandardtoolset.uat.citigroup.net/ostx' using new browser instance

// Playback.PlaybackSettings.ContinueOnError = true;

// BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OST\_PRODParams.Url);

// SharesMenu.WaitForControlExist(5000);

// Playback.PlaybackSettings.ContinueOnError = false;

// if (BrowserWindow.CurrentBrowser != "Chrome")

// {

// if (SharesMenu.Exists == false)

// {

// if (uIConnectingtovm8b5476Text.Exists == true)

// {

// uIUsernameEdit.Text = "sp90104";

// Keyboard.SendKeys(uIPasswordEdit, "Sujee@1234", true);

// Mouse.Click(uIOKButton);

// }

// }

// }

// //System.Environment.SetEnvironmentVariable("OST\_Env", "OST\_PROD");

//}

public void Navigate\_to\_OST\_Chrome()

{

//BrowserWindow.CurrentBrowser = "Chrome";

//ChromeOptions options = new ChromeOptions();

//options.AddArguments("auth-server-whitelist=\*nsroot.net");

//options.AddArguments("start-maximized");

//options.AddArguments("user-data-dir=C:\\Users\\kc90246\\Chrome");

//ChromeDriver driver = new ChromeDriver(options);

//driver.Navigate().GoToUrl(this.Navigate\_to\_OSTParams.Url);

//BrowserWindow testvm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(this.Navigate\_to\_OSTParams.Url);

//BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Locate("OST");

//ChromeDriver driver = new ChromeDriver(options);

//capabilities.SetCapability("chrome.binary", @"C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe");

// Go to web page 'http://vm-8b54-769d.nam.nsroot.net:8100/Ostx/' using new browser instance

//Playback.PlaybackSettings.ContinueOnError = true;

//SharesMenu.WaitForControlExist(5000);

//Playback.PlaybackSettings.ContinueOnError = false;

}

public virtual Navigate\_to\_OSTParams Navigate\_to\_OSTParams

{

get

{

if ((this.mNavigate\_to\_OSTParams == null))

{

this.mNavigate\_to\_OSTParams = new Navigate\_to\_OSTParams();

}

return this.mNavigate\_to\_OSTParams;

}

}

//public virtual Navigate\_to\_OST\_UATParams Navigate\_to\_OST\_UATParams

//{

// get

// {

// if ((this.mNavigate\_to\_OST\_UATParams == null))

// {

// this.mNavigate\_to\_OST\_UATParams = new Navigate\_to\_OST\_UATParams();

// }

// return this.mNavigate\_to\_OST\_UATParams;

// }

//}

//public virtual Navigate\_to\_OST\_PRODParams Navigate\_to\_OST\_PRODParams

//{

// get

// {

// if ((this.mNavigate\_to\_OST\_PRODParams == null))

// {

// this.mNavigate\_to\_OST\_PRODParams = new Navigate\_to\_OST\_PRODParams();

// }

// return this.mNavigate\_to\_OST\_PRODParams;

// }

//}

private Navigate\_to\_OSTParams mNavigate\_to\_OSTParams;

//private Navigate\_to\_OST\_UATParams mNavigate\_to\_OST\_UATParams;

//private Navigate\_to\_OST\_PRODParams mNavigate\_to\_OST\_PRODParams;

/// <summary>

/// Command\_Line - Use 'Command\_LineParams' to pass parameters into this method.

/// </summary>

public void Command\_Line()

{

#region Variable Declarations

WinClient uICommandPromptClient = this.UICommandPromptWindow.UICommandPromptClient;

#endregion

// Launch '%windir%\System32\cmd.exe'

ApplicationUnderTest uICommandPromptWindow = ApplicationUnderTest.Launch("C:\\Windows\\System32\\cmd.exe", "%windir%\\System32\\cmd.exe");

Console.WriteLine("Navigating to I: drive");

Keyboard.SendKeys(uICommandPromptClient, "i:{Enter}", ModifierKeys.None);

Console.WriteLine("Creating directory");

Keyboard.SendKeys(uICommandPromptClient, "mkdir{Space}test\_xxx{Enter}", ModifierKeys.None);

Console.WriteLine("Deleting directory");

Keyboard.SendKeys(uICommandPromptClient, "del /Q {Space}test\_xxx{Enter}", ModifierKeys.None);

Console.WriteLine("Exiting Command Line");

Keyboard.SendKeys(uICommandPromptClient, "exit{Enter}", ModifierKeys.None);

}

public void Date\_Parse\_Test()

{

DateTime CurrentDateTime2 = TimeZoneInfo.ConvertTimeToUtc(DateTime.Now);

String CurrentDateTime = CurrentDateTime2.ToString("MM/dd/yyyy HH:mm").ToLower();

var ieVersion = Registry.LocalMachine.OpenSubKey(@"Software\Microsoft\Internet Explorer").GetValue("Version");

if (ieVersion.ToString().StartsWith("9.11") == true)

{

String TestDateTime3 = "04/02/2014 10:16:00 AM";

System.Text.StringBuilder TestDateTime4 = new System.Text.StringBuilder();

foreach (var ch in TestDateTime3)

{

TestDateTime4.Append(ch);

Console.WriteLine(TestDateTime4);

}

//String BadChar = TestDateTime3[TestDateTime3.Length - 3].ToString(); String TestDateTime4 = TestDateTime3.Replace(BadChar, "");

DateTime GridDateTime3 = DateTime.ParseExact(TestDateTime4.ToString(), "MM/dd/yyyy hh:mm:ss", CultureInfo.InvariantCulture);

}

else

{

Console.WriteLine("Not IE11");

String TestDateTime3 = "04/02/2014 10:16:00 AM";

System.Text.StringBuilder TestDateTime4 = new System.Text.StringBuilder();

foreach (var ch in TestDateTime3) { if (char.IsDigit(ch) == true || char.IsLetter(ch) == true || char.IsPunctuation(ch) == true || char.IsWhiteSpace(ch) == true) { TestDateTime4.Append(ch); } }

//DateTime GridDateTime = DateTime.Parse(TestDateTime,CultureInfo.InvariantCulture);

}

}

/// <summary>

/// Add\_Share\_Exclusion - Use 'Add\_Share\_ExclusionParams' to pass parameters into this method.

/// </summary>

public void Add\_Share\_Exclusion(String Team, String Name, String Description, String Host)

{

#region Variable Declarations

HtmlComboBox uIHostNameTeamSelectComboBox = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHostNameTeamSelectComboBox;

HtmlEdit uIIShareExclusionTextEdit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIShareExclusionTextEdit;

HtmlEdit uIIDescriptionEdit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIDescriptionEdit;

HtmlComboBox uIHostNameSelectComboBox = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHostNameSelectComboBox;

HtmlInputButton uISaveButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UISaveButton;

HtmlDiv uIInsertedsuccessfullyPane = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIInsertedsuccessfullyPane;

#endregion

// Select '150080 - Application Support' in 'HostNameTeamSelect' combo box

//Keyboard.SendKeys(uIHostNameTeamSelectComboBox, Team);

uIHostNameTeamSelectComboBox.SelectedItem = Team;

// Type 'kc90246$' in 'iShareExclusionText' text box

uIIShareExclusionTextEdit.Text = Name;

// Type 'zzzzzzDescriptionShareExclusion' in 'iDescription' text box

uIIDescriptionEdit.Text = Description;

if (uIHostNameSelectComboBox.ItemCount < 2)

{

Playback.Wait(2000);

if (uIHostNameSelectComboBox.ItemCount < 2)

{

Playback.Wait(2000);

}

}

// Select 'test kc90246' in 'HostNameSelect' combo box

//Keyboard.SendKeys(uIHostNameSelectComboBox, Host);

uIHostNameSelectComboBox.SelectedItem = Host;

// Click 'Save' button

Mouse.Click(uISaveButton);

Console.WriteLine("Save button clicked.");

uIInsertedsuccessfullyPane.WaitForControlExist(5000);

Console.WriteLine("Saved label shown.");

}

/// <summary>

/// Add\_Path\_Exclusion

/// </summary>

public void Add\_Path\_Exclusion(String Team, String Path, String Description)

{

#region Variable Declarations

HtmlDiv uIAddPathPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddBCrumbPathPane.UIAddPathPane;

HtmlComboBox uIHostNamePathSelectComboBox = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHostNamePathSelectComboBox;

HtmlEdit uIIExclusionPathEdit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIExclusionPathEdit;

HtmlEdit uIIDescription1Edit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIDescription1Edit;

HtmlInputButton uISaveButton1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UISaveButton1;

HtmlDiv uIInsertedsuccessfullyPane = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIInsertedsuccessfullyPane;

#endregion

// Select '150080 - Application Support' in 'HostNameTeamSelect' combo box

//Keyboard.SendKeys(uIHostNamePathSelectComboBox, Team);

uIHostNamePathSelectComboBox.SelectedItem = Team;

// Type 'kc90246$' in 'iShareExclusionText' text box

uIIExclusionPathEdit.Text = Path;

// Type 'zzzzzzDescriptionShareExclusion' in 'iDescription' text box

uIIDescription1Edit.Text = Description;

// Click 'Save' button

Mouse.Click(uISaveButton1);

Console.WriteLine("Save button clicked.");

uIInsertedsuccessfullyPane.WaitForControlExist(5000);

Console.WriteLine("Saved label shown.");

}

/// <summary>

/// Edit\_Share\_Exclusion - Use 'Add\_Share\_ExclusionParams' to pass parameters into this method.

/// </summary>

public void Edit\_Share\_Exclusion(String Name, String Description)

{

#region Variable Declarations

HtmlComboBox uIHostNameTeamSelectComboBox = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHostNameTeamSelectComboBox;

HtmlEdit uIIShareExclusionTextEdit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIShareExclusionTextEdit;

HtmlEdit uIIDescriptionEdit = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIDescriptionEdit;

HtmlComboBox uIHostNameSelectComboBox = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHostNameSelectComboBox;

HtmlInputButton uIUpdateButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIUpdateButton;

HtmlDiv uIUpdatedSuccessfully = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIUpdatedsuccessfully;

#endregion

// Type 'kc90246$' in 'iShareExclusionText' text box

uIIShareExclusionTextEdit.Text = "";

uIIShareExclusionTextEdit.Text = Name + "Edited";

// Type 'zzzzzzDescriptionShareExclusion' in 'iDescription' text box

uIIDescriptionEdit.Text = "";

uIIDescriptionEdit.Text = Description + "Edited";

// Click 'Save' button

Mouse.Click(uIUpdateButton);

Console.WriteLine("Update button clicked.");

uIUpdatedSuccessfully.WaitForControlExist(5000);

Console.WriteLine("Updated label shown.");

}

/// <summary>

/// Edit\_Path\_Exclusion

/// </summary>

public void Edit\_Path\_Exclusion(String Path, String Description)

{

#region Variable Declarations

HtmlDiv uIEditPathPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIEditBCrumbPathPane.UIEditPathPane;

HtmlEdit uIIExclusionPathEdit1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIExclusionPathEdit1;

HtmlEdit uIIDescription1Edit1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIIDescription1Edit1;

HtmlInputButton uIUpdateButton1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIUpdateButton1;

HtmlDiv uIUpdatedSuccessfully = this.UISecurityGroupsOSTOpeWindow.UISecurityGroupsOSTOpeDocument.UIUpdatedsuccessfully;

#endregion

// Type 'kc90246$' in 'iShareExclusionText' text box

uIIExclusionPathEdit1.Text = "";

uIIExclusionPathEdit1.Text = Path + "Edited";

// Type 'zzzzzzDescriptionShareExclusion' in 'iDescription' text box

uIIDescription1Edit1.Text = "";

uIIDescription1Edit1.Text = Description + "Edited";

// Click 'Save' button

Mouse.Click(uIUpdateButton1);

Console.WriteLine("Update button clicked.");

uIUpdatedSuccessfully.WaitForControlExist(5000);

Console.WriteLine("Updated label shown.");

}

public virtual Add\_Share\_ExclusionParams Add\_Share\_ExclusionParams

{

get

{

if ((this.mAdd\_Share\_ExclusionParams == null))

{

this.mAdd\_Share\_ExclusionParams = new Add\_Share\_ExclusionParams();

}

return this.mAdd\_Share\_ExclusionParams;

}

}

private Add\_Share\_ExclusionParams mAdd\_Share\_ExclusionParams;

/// <summary>

/// Open\_Edit\_Path\_Exclusion\_popup

/// </summary>

public void Open\_Edit\_Path\_Exclusion\_popup()

{

#region Variable Declarations

HtmlCell uIEditCell = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIJqGridTable1.UIEditCell;

HtmlHyperlink uIEditHyperlink = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIItem1Row2.UIEditHyperlink;

#endregion

// Mouse hover 'Edit' cell at (40, 19)

Mouse.Hover(uIEditCell);

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

}

/// <summary>

/// UAT\_Edit\_Path\_Exclusion

/// </summary>

public void UAT\_Open\_Edit\_Path\_Exclusion()

{

#region Variable Declarations

HtmlHyperlink uIEditHyperlink = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument1.UIItem1Row1.UIEditHyperlink;

#endregion

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

}

public void UAT\_Open\_Delete\_Path\_Exclusion\_popup1()

{

#region Variable Declarations

HtmlHyperlink uIDeleteHyperlink = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument1.UIItem1Row2.UIDeleteHyperlink;

#endregion

// Click 'Delete' link

Mouse.Click(uIDeleteHyperlink);

}

/// <summary>

/// Open\_Edit\_Share\_Exclusion\_popup

/// </summary>

public void Open\_Edit\_Share\_Exclusion\_popup()

{

#region Variable Declarations

HtmlHyperlink uIEditHyperlink = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIItem1Row.UIEditHyperlink;

#endregion

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

}

public void UAT\_Open\_Edit\_Share\_Exclusion\_popup()

{

#region Variable Declarations

HtmlHyperlink uIEditHyperlink = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument1.UIItem1Row.UIEditHyperlink;

#endregion

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

}

/// <summary>

/// Verify\_Elements\_Data\_Stats\_Details - Use 'Verify\_Elements\_Data\_Stats\_DetailsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Data\_Stats\_Details()

{

#region Variable Declarations

HtmlDiv uIDataStatsDetailsPane = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIDataStatsDetailsPane;

HtmlHyperlink uID531GBsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UID531GBsCustom.UID531GBsHyperlink;

HtmlButton uIExportfoldersButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfoldersButton;

HtmlButton uIExportfilesButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfilesButton;

HtmlButton uIExportfoldersChildreButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfoldersChildreButton;

HtmlButton uIExportfilesChildreniButton = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UIExportfilesChildreniButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

Playback.Wait(5000);

uID531GBsHyperlink.Find();

uID531GBsHyperlink.WaitForControlExist(5000);

// Verify that the 'InnerText' property of 'Data Stats Details' pane equals 'Data Stats Details'

StringAssert.StartsWith(uIDataStatsDetailsPane.InnerText, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UIDataStatsDetailsPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "This page shows file and folder details of a selected volume.");

// Verify that the 'Class' property of 'D$\ (5.31 GBs)' link starts with 'jstree-anchor'

StringAssert.StartsWith(uID531GBsHyperlink.Class, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UID531GBsHyperlinkClass);

// Verify that the 'InnerText' property of 'Export folders' button contains 'Export folders'

StringAssert.Contains(uIExportfoldersButton.InnerText, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UIExportfoldersButtonInnerText);

// Verify that the 'InnerText' property of 'Export files' button contains 'Export files'

StringAssert.Contains(uIExportfilesButton.InnerText, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UIExportfilesButtonInnerText);

// Verify that the 'InnerText' property of 'Export folders (Children included)' button contains 'Export folders (Children included)'

StringAssert.Contains(uIExportfoldersChildreButton.InnerText, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UIExportfoldersChildreButtonInnerText);

// Verify that the 'InnerText' property of 'Export files (Children included)' button contains 'Export files (Children included)'

StringAssert.Contains(uIExportfilesChildreniButton.InnerText, this.Verify\_Elements\_Data\_Stats\_DetailsExpectedValues.UIExportfilesChildreniButtonInnerText);

}

public virtual Verify\_Elements\_Data\_Stats\_DetailsExpectedValues Verify\_Elements\_Data\_Stats\_DetailsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Data\_Stats\_DetailsExpectedValues == null))

{

this.mVerify\_Elements\_Data\_Stats\_DetailsExpectedValues = new Verify\_Elements\_Data\_Stats\_DetailsExpectedValues();

}

return this.mVerify\_Elements\_Data\_Stats\_DetailsExpectedValues;

}

}

private Verify\_Elements\_Data\_Stats\_DetailsExpectedValues mVerify\_Elements\_Data\_Stats\_DetailsExpectedValues;

/// <summary>

/// Verify\_Elements\_Dfs\_Shares - Use 'Verify\_Elements\_Dfs\_SharesExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Dfs\_Shares()

{

#region Variable Declarations

HtmlDiv uIDFSSharesPane = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIDFSSharesPane;

HtmlLabel uIRegionLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uISearchForLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UISearchForLabel;

HtmlLabel uIINLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIINLabel;

HtmlLabel uIShowArchivedLabel = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIShowArchivedLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemEdit;

HtmlComboBox uIItemComboBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemComboBox;

HtmlCheckBox uIItemCheckBox = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UIItemCheckBox;

HtmlButton uISearchButton = this.UIDFSSharesOSTOperatioWindow.UIDFSSharesOSTOperatioDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

#endregion

// Verify that the 'InnerText' property of 'DFS Shares' pane equals 'DFS Shares'

StringAssert.StartsWith(uIDFSSharesPane.InnerText, this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIDFSSharesPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays DFS host/share information along with link name, link host and link share information. You can search by link name, link host, link folder path and DFS shares. You can also include archived information as well if you wish.");

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UITeamLabelInnerText, uITeamLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Search For:' label starts with 'Search For:'

Assert.AreEqual(uISearchForLabel.InnerText.Trim(), this.Verify\_Elements\_Dfs\_SharesExpectedValues.UISearchForLabelInnerText);

// Verify that the 'InnerText' property of 'in' label starts with 'in'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIINLabelInnerText, uIINLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Show Archived:' label starts with 'Show Archived:'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIShowArchivedLabelInnerText, uIShowArchivedLabel.InnerText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIItemEditClass, uIItemEdit.Class);

// Verify that the 'InnerText' property of combo box contains 'Link NameLink HostLink FolderPathDFS Shares'

StringAssert.Contains(uIItemComboBox.InnerText, this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIItemComboBoxInnerText);

// Verify that the 'ControlType' property of check box equals 'CheckBox'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UIItemCheckBoxControlType, uIItemCheckBox.ControlType.ToString());

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Dfs\_SharesExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText.Trim());

}

public virtual Verify\_Elements\_Dfs\_SharesExpectedValues Verify\_Elements\_Dfs\_SharesExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Dfs\_SharesExpectedValues == null))

{

this.mVerify\_Elements\_Dfs\_SharesExpectedValues = new Verify\_Elements\_Dfs\_SharesExpectedValues();

}

return this.mVerify\_Elements\_Dfs\_SharesExpectedValues;

}

}

private Verify\_Elements\_Dfs\_SharesExpectedValues mVerify\_Elements\_Dfs\_SharesExpectedValues;

/// <summary>

/// Verify\_Elements\_Global\_Account\_Search - Use 'Verify\_Elements\_Global\_Account\_SearchExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Global\_Account\_Search()

{

#region Variable Declarations

HtmlDiv uIGlobalAccountSearchPane = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIGlobalAccountSearchPane;

HtmlLabel uIAccountLabel = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchFormCustom.UIAccountLabel;

HtmlEdit uIAccountTxtEdit = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UIAccountTxtEdit;

HtmlButton uISearchButton = this.UIGlobalAccountSearchOWindow.UIGlobalAccountSearchODocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Global Account Search' pane equals 'Global Account Search'

StringAssert.StartsWith(uIGlobalAccountSearchPane.InnerText, this.Verify\_Elements\_Global\_Account\_SearchExpectedValues.UIGlobalAccountSearchPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Provides the ability to search by account name to find details about host/share/permissions/owner information.");

// Verify that the 'InnerText' property of 'Account:' label starts with 'Account:'

StringAssert.StartsWith(uIAccountLabel.InnerText, this.Verify\_Elements\_Global\_Account\_SearchExpectedValues.UIAccountLabelInnerText);

// Verify that the 'Id' property of 'accountTxt' text box equals 'accountTxt'

Assert.AreEqual(this.Verify\_Elements\_Global\_Account\_SearchExpectedValues.UIAccountTxtEditId, uIAccountTxtEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Global\_Account\_SearchExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Global\_Account\_SearchExpectedValues Verify\_Elements\_Global\_Account\_SearchExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Global\_Account\_SearchExpectedValues == null))

{

this.mVerify\_Elements\_Global\_Account\_SearchExpectedValues = new Verify\_Elements\_Global\_Account\_SearchExpectedValues();

}

return this.mVerify\_Elements\_Global\_Account\_SearchExpectedValues;

}

}

private Verify\_Elements\_Global\_Account\_SearchExpectedValues mVerify\_Elements\_Global\_Account\_SearchExpectedValues;

/// <summary>

/// Verify\_Elements\_Global\_Compliance - Use 'Verify\_Elements\_Global\_ComplianceExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Global\_Compliance()

{

#region Variable Declarations

HtmlDiv uIGlobalCompliancePane = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIGlobalCompliancePane;

HtmlLabel uIRegionLabel = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UIAvailableTeamsDdlComboBox;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow.UIGlobalComplianceOSTODocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Global Compliance' pane equals 'Global Compliance'

StringAssert.StartsWith(uIGlobalCompliancePane.InnerText, this.Verify\_Elements\_Global\_ComplianceExpectedValues.UIGlobalCompliancePaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays metrics for teams’ compliance in percentages.");

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Global\_ComplianceExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Global\_ComplianceExpectedValues.UITeamLabelInnerText, uITeamLabel.InnerText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Global\_ComplianceExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Global\_ComplianceExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Global\_ComplianceExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Global\_ComplianceExpectedValues Verify\_Elements\_Global\_ComplianceExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Global\_ComplianceExpectedValues == null))

{

this.mVerify\_Elements\_Global\_ComplianceExpectedValues = new Verify\_Elements\_Global\_ComplianceExpectedValues();

}

return this.mVerify\_Elements\_Global\_ComplianceExpectedValues;

}

}

private Verify\_Elements\_Global\_ComplianceExpectedValues mVerify\_Elements\_Global\_ComplianceExpectedValues;

/// <summary>

/// Verify\_Elements\_Monitor\_Health - Use 'Verify\_Elements\_Monitor\_HealthExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Monitor\_Health()

{

#region Variable Declarations

HtmlCustom uIOstMonitorsOverviewCustom = this.UIPrintQueueHistoryOSTWindow.UIOstMonitorsOverviewODocument.UIMonitorIndexPane.UIOstMonitorsOverviewCustom;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Ost Monitors Overview' custom control equals 'Ost Monitors Overview'

Assert.AreEqual(this.Verify\_Elements\_Monitor\_HealthExpectedValues.UIOstMonitorsOverviewCustomInnerText, uIOstMonitorsOverviewCustom.InnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Shows the overall status and health of the OST Monitors.");

}

public virtual Verify\_Elements\_Monitor\_HealthExpectedValues Verify\_Elements\_Monitor\_HealthExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Monitor\_HealthExpectedValues == null))

{

this.mVerify\_Elements\_Monitor\_HealthExpectedValues = new Verify\_Elements\_Monitor\_HealthExpectedValues();

}

return this.mVerify\_Elements\_Monitor\_HealthExpectedValues;

}

}

private Verify\_Elements\_Monitor\_HealthExpectedValues mVerify\_Elements\_Monitor\_HealthExpectedValues;

/// <summary>

/// Verify\_Elements\_Ntfs\_Permissions - Use 'Verify\_Elements\_Ntfs\_PermissionsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Ntfs\_Permissions()

{

#region Variable Declarations

HtmlDiv uINTFSPermissionsPane = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UINTFSPermissionsPane;

HtmlLabel uIRegionLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIShareNameLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIShareNameLabel;

HtmlLabel uIOwnerLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIOwnerLabel;

HtmlLabel uIPermissionLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIPermissionLabel;

HtmlLabel uIAccountLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIAccountLabel;

HtmlLabel uIInheritanceLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIInheritanceLabel;

HtmlLabel uIPathLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIPathLabel;

HtmlLabel uIInitialScanDateLabel = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIInitialScanDateLabel;

HtmlDiv uIExportwithGroupMembePane = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UIExportwithGroupMembePane;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHostNameEdit;

HtmlEdit uIShareNameTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIShareNameTxtEdit;

HtmlEdit uIOwnerTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIOwnerTxtEdit;

HtmlEdit uIPermissionTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIPermissionTxtEdit;

HtmlEdit uIAccountTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIAccountTxtEdit;

HtmlComboBox uIInheritanceTxtComboBox = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIInheritanceTxtComboBox;

HtmlEdit uIPathTxtEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIPathTxtEdit;

HtmlEdit uIScanDateDivDtPickEdit = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIScanDateDivDtPickEdit;

HtmlCheckBox uIExportGroupMembersCheckBox = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIExportGroupMembersCheckBox;

HtmlButton uISearchButton = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'Title' property of 'Displays NTFS permission details. You can search b...' custom control equals 'Displays NTFS permission details. You can search by host, share name, owner, permission, account, inheritance, path and/or initial scan date.'

Assert.AreEqual(UIHeaderTooltip.Title, "Displays NTFS permission details. You can search by host, share name, owner, permission, account, inheritance, path and/or initial scan date.");

// Verify that the 'InnerText' property of 'NTFS Permissions' pane equals 'NTFS Permissions'

StringAssert.StartsWith(uINTFSPermissionsPane.InnerText, this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UINTFSPermissionsPaneInnerText);

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UITeamLabelInnerText, uITeamLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Share Name:' label starts with 'Share Name:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIShareNameLabelInnerText, uIShareNameLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Owner:' label starts with 'Owner:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIOwnerLabelInnerText, uIOwnerLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Permission:' label starts with 'Permission:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIPermissionLabelInnerText, uIPermissionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Account:' label starts with 'Account:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIAccountLabelInnerText, uIAccountLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Inheritance:' label starts with 'Inheritance:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIInheritanceLabelInnerText, uIInheritanceLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Path:' label starts with 'Path:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIPathLabelInnerText, uIPathLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Initial Scan Date:' label starts with 'Initial Scan Date:'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIInitialScanDateLabelInnerText, uIInitialScanDateLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Export with Group Members' pane starts with 'Export with Group Members'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIExportwithGroupMembePaneInnerText, uIExportwithGroupMembePane.InnerText.Trim());

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'shareNameTxt' text box equals 'shareNameTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIShareNameTxtEditId, uIShareNameTxtEdit.Id);

// Verify that the 'Id' property of 'ownerTxt' text box equals 'ownerTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIOwnerTxtEditId, uIOwnerTxtEdit.Id);

// Verify that the 'Id' property of 'permissionTxt' text box equals 'permissionTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIPermissionTxtEditId, uIPermissionTxtEdit.Id);

// Verify that the 'Id' property of 'accountTxt' text box equals 'accountTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIAccountTxtEditId, uIAccountTxtEdit.Id);

// Verify that the 'Id' property of 'inheritanceTxt' combo box equals 'inheritanceTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIInheritanceTxtComboBoxId, uIInheritanceTxtComboBox.Id);

// Verify that the 'Id' property of 'pathTxt' text box equals 'pathTxt'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIPathTxtEditId, uIPathTxtEdit.Id);

// Verify that the 'Id' property of 'scanDateDivDtPick' text box equals 'scanDateDivDtPick'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIScanDateDivDtPickEditId, uIScanDateDivDtPickEdit.Id);

// Verify that the 'Id' property of 'exportGroupMembers' check box equals 'exportGroupMembers'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UIExportGroupMembersCheckBoxId, uIExportGroupMembersCheckBox.Id);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Ntfs\_PermissionsExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public virtual Verify\_Elements\_Ntfs\_PermissionsExpectedValues Verify\_Elements\_Ntfs\_PermissionsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Ntfs\_PermissionsExpectedValues == null))

{

this.mVerify\_Elements\_Ntfs\_PermissionsExpectedValues = new Verify\_Elements\_Ntfs\_PermissionsExpectedValues();

}

return this.mVerify\_Elements\_Ntfs\_PermissionsExpectedValues;

}

}

private Verify\_Elements\_Ntfs\_PermissionsExpectedValues mVerify\_Elements\_Ntfs\_PermissionsExpectedValues;

/// <summary>

/// Verify\_Elements\_Print\_Queue - Use 'Verify\_Elements\_Print\_QueueExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Print\_Queue()

{

#region Variable Declarations

HtmlControl uICurrentPrintQueuesPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UICurrentPrintQueuesPane;

HtmlLabel uIRegionLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIRegionLabel;

HtmlLabel uITeamLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UITeamLabel;

HtmlLabel uIHostLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIHostLabel;

HtmlLabel uIQueueLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchFormCustom.UIQueueLabel;

HtmlLabel uILocationLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchFormCustom.UILocationLabel;

HtmlLabel uIIPAddressLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchFormCustom.UIIPAddressLabel;

HtmlLabel uIDriverLabel = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchFormCustom.UIDriverLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIHostNameEdit;

HtmlEdit uIQueueInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIQueueInputEdit;

HtmlEdit uILocationInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UILocationInputEdit;

HtmlEdit uIIpAddressInputEdit = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIIpAddressInputEdit;

HtmlComboBox uIDriverTypeSelectComboBox = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDriverTypeSelectComboBox;

HtmlButton uISearchButton = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Current Print Queues' pane starts with 'Current Print Queues'

StringAssert.StartsWith(uICurrentPrintQueuesPane.InnerText, this.Verify\_Elements\_Print\_QueueExpectedValues.UICurrentPrintQueuesPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays all active print queues for the Hosts that are set as Printer Servers");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

StringAssert.StartsWith(uIHostLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UIHostLabelDisplayText);

// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

StringAssert.StartsWith(uIQueueLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UIQueueLabelDisplayText);

// Verify that the 'DisplayText' property of 'Location:' label starts with 'Location:'

StringAssert.StartsWith(uILocationLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UILocationLabelDisplayText);

// Verify that the 'DisplayText' property of 'IP Address:' label starts with 'IP Address:'

StringAssert.StartsWith(uIIPAddressLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UIIPAddressLabelDisplayText);

// Verify that the 'DisplayText' property of 'Driver:' label starts with 'Driver:'

StringAssert.StartsWith(uIDriverLabel.DisplayText, this.Verify\_Elements\_Print\_QueueExpectedValues.UIDriverLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'queueInput' text box equals 'queueInput'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIQueueInputEditId, uIQueueInputEdit.Id);

// Verify that the 'Id' property of 'locationInput' text box equals 'locationInput'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UILocationInputEditId, uILocationInputEdit.Id);

// Verify that the 'Id' property of 'ipAddressInput' text box equals 'ipAddressInput'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIIpAddressInputEditId, uIIpAddressInputEdit.Id);

// Verify that the 'Id' property of 'driverTypeSelect' combo box equals 'driverTypeSelect'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UIDriverTypeSelectComboBoxId, uIDriverTypeSelectComboBox.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Print\_QueueExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Print\_QueueExpectedValues Verify\_Elements\_Print\_QueueExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Print\_QueueExpectedValues == null))

{

this.mVerify\_Elements\_Print\_QueueExpectedValues = new Verify\_Elements\_Print\_QueueExpectedValues();

}

return this.mVerify\_Elements\_Print\_QueueExpectedValues;

}

}

private Verify\_Elements\_Print\_QueueExpectedValues mVerify\_Elements\_Print\_QueueExpectedValues;

/// <summary>

/// Verify\_Elements\_Printer\_popup - Use 'Verify\_Elements\_Printer\_popupExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Printer\_popup()

{

#region Variable Declarations

HtmlDiv uIPrinterDUBCTISP02roxPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIPrinterDUBCTISP02roxPane;

HtmlDiv uIHostPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIHostPane;

HtmlDiv uIQueueNamePane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIQueueNamePane;

HtmlDiv uIShareNamePane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIShareNamePane;

HtmlDiv uIDriverPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIDriverPane;

HtmlDiv uILocationPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UILocationPane;

HtmlDiv uICommentPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UICommentPane;

HtmlDiv uIPortNamePane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIPortNamePane;

HtmlDiv uIIPAddressPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIIPAddressPane;

HtmlDiv uILogDatePane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UILogDatePane;

HtmlDiv uIPrintStatusPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIPrintStatusPane;

HtmlDiv uIPrintLengthPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIPrintLengthPane;

HtmlDiv uIPrintJobsPane = this.UICurrentPrintQueuesOSWindow.UICurrentPrintQueuesOSDocument.UIDetailsContentPane.UIPrintJobsPane;

#endregion

// Verify that the 'InnerText' property of 'Printer \DUBCTISP02\roxNew' pane starts with 'Printer'

StringAssert.StartsWith(uIPrinterDUBCTISP02roxPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIPrinterDUBCTISP02roxPaneInnerText);

// Verify that the 'InnerText' property of 'Host:' pane starts with 'Host:'

StringAssert.StartsWith(uIHostPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIHostPaneInnerText);

// Verify that the 'InnerText' property of 'Queue Name:' pane starts with 'Queue Name:'

StringAssert.StartsWith(uIQueueNamePane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIQueueNamePaneInnerText);

// Verify that the 'InnerText' property of 'Share Name:' pane starts with 'Share Name:'

StringAssert.StartsWith(uIShareNamePane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIShareNamePaneInnerText);

// Verify that the 'InnerText' property of 'Driver:' pane starts with 'Driver:'

StringAssert.StartsWith(uIDriverPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIDriverPaneInnerText);

// Verify that the 'InnerText' property of 'Location:' pane starts with 'Location:'

StringAssert.StartsWith(uILocationPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UILocationPaneInnerText);

// Verify that the 'InnerText' property of 'Comment:' pane starts with 'Comment:'

StringAssert.StartsWith(uICommentPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UICommentPaneInnerText);

// Verify that the 'InnerText' property of 'Port Name:' pane starts with 'Port Name:'

StringAssert.StartsWith(uIPortNamePane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIPortNamePaneInnerText);

// Verify that the 'InnerText' property of 'IP Address:' pane starts with 'IP Address:'

StringAssert.StartsWith(uIIPAddressPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIIPAddressPaneInnerText);

// Verify that the 'InnerText' property of 'LogDate:' pane starts with 'LogDate:'

StringAssert.StartsWith(uILogDatePane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UILogDatePaneInnerText);

// Verify that the 'InnerText' property of 'Print Status:' pane starts with 'Print Status:'

StringAssert.StartsWith(uIPrintStatusPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIPrintStatusPaneInnerText);

// Verify that the 'InnerText' property of 'Print Length:' pane starts with 'Print Length:'

StringAssert.StartsWith(uIPrintLengthPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIPrintLengthPaneInnerText);

// Verify that the 'InnerText' property of 'Print Jobs:' pane starts with 'Print Jobs:'

StringAssert.StartsWith(uIPrintJobsPane.InnerText, this.Verify\_Elements\_Printer\_popupExpectedValues.UIPrintJobsPaneInnerText);

}

public virtual Verify\_Elements\_Printer\_popupExpectedValues Verify\_Elements\_Printer\_popupExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Printer\_popupExpectedValues == null))

{

this.mVerify\_Elements\_Printer\_popupExpectedValues = new Verify\_Elements\_Printer\_popupExpectedValues();

}

return this.mVerify\_Elements\_Printer\_popupExpectedValues;

}

}

private Verify\_Elements\_Printer\_popupExpectedValues mVerify\_Elements\_Printer\_popupExpectedValues;

/// <summary>

/// Verify\_Elements\_Print\_Queue\_History - Use 'Verify\_Elements\_Print\_Queue\_HistoryExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Print\_Queue\_History()

{

#region Variable Declarations

HtmlControl uIPrintQueueHistoryPane = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIPrintQueueHistoryPane;

HtmlLabel uIRegionLabel = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIQueueLabel = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UIQueueLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UIHostNameEdit;

HtmlEdit uIItemEdit = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UIItemEdit;

HtmlButton uISearchButton = this.UIPrintQueueHistoryOSTWindow.UIPrintQueueHistoryOSTDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Print Queue History' pane equals 'Print Queue History'

StringAssert.StartsWith(uIPrintQueueHistoryPane.InnerText, this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIPrintQueueHistoryPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays all print queue history including archived print queues.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

StringAssert.StartsWith(uIQueueLabel.DisplayText, this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIQueueLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UIItemEditClass, uIItemEdit.Class);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_HistoryExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Print\_Queue\_HistoryExpectedValues Verify\_Elements\_Print\_Queue\_HistoryExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Print\_Queue\_HistoryExpectedValues == null))

{

this.mVerify\_Elements\_Print\_Queue\_HistoryExpectedValues = new Verify\_Elements\_Print\_Queue\_HistoryExpectedValues();

}

return this.mVerify\_Elements\_Print\_Queue\_HistoryExpectedValues;

}

}

private Verify\_Elements\_Print\_Queue\_HistoryExpectedValues mVerify\_Elements\_Print\_Queue\_HistoryExpectedValues;

/// <summary>

/// Verify\_Elements\_Print\_Queue\_Statistics - Use 'Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_Print\_Queue\_Statistics()

{

#region Variable Declarations

HtmlControl uIPrintQueueStatisticsPane = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIPrintQueueStatisticsPane;

HtmlLabel uIRegionLabel = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIQueueLabel = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UISearchFormCustom.UIQueueLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIHostNameEdit;

HtmlEdit uIQueueInputEdit = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UIQueueInputEdit;

HtmlButton uISearchButton = this.UIPrintQueueStatisticsWindow.UIPrintQueueStatisticsDocument.UISearchFormCustom.UISearchButton;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Print Queue Statistics' pane equals 'Print Queue Statistics'

StringAssert.StartsWith(uIPrintQueueStatisticsPane.InnerText, this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIPrintQueueStatisticsPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Displays details on print queues such as # printed jobs and queue length.");

// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIRegionLabelDisplayText, uIRegionLabel.DisplayText.Trim());

// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.DisplayText, this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UITeamLabelDisplayText);

// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIHostLabelDisplayText, uIHostLabel.DisplayText);

// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

StringAssert.StartsWith(uIQueueLabel.DisplayText, this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIQueueLabelDisplayText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of 'queueInput' text box equals 'queueInput'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UIQueueInputEditId, uIQueueInputEdit.Id);

// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues.UISearchButtonDisplayText, uISearchButton.DisplayText);

}

public virtual Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues

{

get

{

if ((this.mVerify\_Elements\_Print\_Queue\_StatisticsExpectedValues == null))

{

this.mVerify\_Elements\_Print\_Queue\_StatisticsExpectedValues = new Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues();

}

return this.mVerify\_Elements\_Print\_Queue\_StatisticsExpectedValues;

}

}

private Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues mVerify\_Elements\_Print\_Queue\_StatisticsExpectedValues;

/// <summary>

/// Navigate\_to\_Datastats\_Details

/// </summary>

public void Navigate\_to\_Datastats\_Details()

{

#region Variable Declarations

HtmlHyperlink uIDataStatsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UIDataStatsHyperlink;

HtmlHyperlink uIDetailsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIMainnavPane.UIDetailsHyperlink;

HtmlHyperlink uID531GBsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UID531GBsCustom.UID531GBsHyperlink;

#endregion

// Click 'DataStats' link

Mouse.Click(uIDataStatsHyperlink);

// Click 'Details' link

Mouse.Click(uIDetailsHyperlink);

uID531GBsHyperlink.WaitForControlExist(5000);

}

/// <summary>

/// Navigate\_to\_User\_Roles

/// </summary>

public void Navigate\_to\_User\_Roles()

{

#region Variable Declarations

HtmlHyperlink uISupportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISupportHyperlink;

HtmlHyperlink UIUserRoleslink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIUserRoleslink;

#endregion

// Click 'Support' link

Mouse.Click(uISupportHyperlink);

// Click 'User Roles' link

Mouse.Click(UIUserRoleslink);

}

/// <summary>

/// Navigate\_to\_System\_Information

/// </summary>

public void Navigate\_to\_System\_Information()

{

#region Variable Declarations

HtmlHyperlink uISupportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISupportHyperlink;

HtmlHyperlink UISystemInformationlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UISystemInformationlink;

#endregion

// Click 'Support' link

Mouse.Click(uISupportHyperlink);

// Click 'User Roles' link

Mouse.Click(UISystemInformationlink);

}

public void Navigate\_to\_MongoDB\_Health\_Page()

{

#region Variable Declarations

HtmlHyperlink uISupportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISupportHyperlink;

HtmlHyperlink UIMongoDBHealthlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIMongoDBHealthlink;

#endregion

// Click 'Support' link

Mouse.Click(uISupportHyperlink);

// Click 'User Roles' link

Mouse.Click(UIMongoDBHealthlink);

}

public void Navigate\_to\_Scanner\_Health\_Page()

{

#region Variable Declarations

HtmlHyperlink uISupportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISupportHyperlink;

HtmlHyperlink UIScannerHealthlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane5.UIScannerHealthlink;

#endregion

// Click 'Support' link

Mouse.Click(uISupportHyperlink);

// Click 'User Roles' link

Mouse.Click(UIScannerHealthlink);

}

public void Navigate\_to\_DLP\_Data\_Protection\_Report\_Page()

{

#region Variable Declarations

HtmlHyperlink uIReportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane1.UIReportHyperlink;

HtmlHyperlink uIDLPDataProtectionReportHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIDLPDataProtectionReportHyperlink;

#endregion

Mouse.Click(uIReportHyperlink);

Mouse.Click(uIDLPDataProtectionReportHyperlink);

}

/// <summary>

/// Wait\_for\_Monitor\_Overview\_finish\_loading - Use 'Wait\_for\_Monitor\_Overview\_finish\_loadingExpectedValues' to pass parameters into this method.

/// </summary>

public void Wait\_for\_Monitor\_Overview\_finish\_loading()

{

#region Variable Declarations

HtmlSpan uIObtainedinformationaPane = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument1.UISearchDivPane.UIObtainedinformationaPane;

HtmlImage uIFrankfurtImage = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIFrankfurtImage;

#endregion

uIObtainedinformationaPane.WaitForControlExist(120000);

Console.WriteLine("Monitors page finished loading.");

}

/// <summary>

/// Validate\_Monitor\_Account

/// </summary>

public void Validate\_Monitor\_Account()

{

#region Variable Declarations

HtmlCell uIEditCell = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIJqGridTable1.UIEditCell;

HtmlHyperlink uIValidateHyperlink = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIItem1Row.UIValidateHyperlink;

HtmlDiv uIAccountValid = this.UIManageMonitorAccountWindow.UIManageMonitorAccountDocument.UIAccountisvalidPane;

HtmlDiv Column\_Team\_Name = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Column\_Team\_Name;

#endregion

Mouse.Click(Column\_Team\_Name);

// Click 'Validate' link

Mouse.Click(uIValidateHyperlink);

uIAccountValid.WaitForControlExist(5000);

Console.WriteLine("Account Validated successfully.");

}

/// <summary>

/// Schedule\_Recurring\_Schedule

/// </summary>

public void Schedule\_Recurring\_Schedule()

{

#region Variable Declarations

HtmlSpan uIEditRecurringSchedulPane = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIEditRecurringSchedulPane;

HtmlComboBox uIOutputTypeSlComboBox = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIOutputTypeSlComboBox;

HtmlEdit uIReportNameTxtSchedulEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIReportNameTxtSchedulEdit;

HtmlCheckBox uITimeStampChkCheckBox = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UITimeStampChkCheckBox;

HtmlEdit uIReportShareTxtEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIReportShareTxtEdit;

HtmlEdit uIOccuranceTxtEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIOccuranceTxtEdit;

HtmlButton uISaveButton = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UISaveButton;

HtmlControl uIReportRequestsforEURPane = this.UIReportRequestsforEURWindow.UIReportRequestsforEURDocument.UIReportRequestsforEURPane;

#endregion

String ReportName = "Scheduled\_Report\_Name\_" + DateTime.Now.Year.ToString() + DateTime.Now.Month + DateTime.Now.Day + DateTime.Now.Hour + DateTime.Now.Minute + DateTime.Now.Second;

uIReportNameTxtSchedulEdit.Text = ReportName;

uIOccuranceTxtEdit.Text = "2";

Mouse.Click(uISaveButton);

uIReportRequestsforEURPane.WaitForControlExist(5000);

Console.WriteLine("Report Scheduled Successfully.");

}

/// <summary>

/// Search\_User\_Roles

/// </summary>

public void Search\_User\_Roles(String SoeID)

{

#region Variable Declarations

HtmlDiv uIUserRolesPane = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UIUserRolesPane;

HtmlEdit uIItemEdit = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UISearchFormCustom.UIItemEdit;

HtmlButton uISearchButton = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UISearchFormCustom.UISearchButton;

#endregion

uIUserRolesPane.WaitForControlExist(5000);

uIItemEdit.Text = SoeID;

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Host\_Maintenance

/// </summary>

public void Search\_Host\_Maintenance(String Host)

{

#region Variable Declarations

HtmlDiv uIHostMaintancePane = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIHostMaintancePane;

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIHostNameEdit;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UISearchButton;

HtmlCustom Item125bap001 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Item125bap001;

HtmlCustom Item111wallap002 = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.Item111wallap002;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

uIHostMaintancePane.WaitForControlExist(5000);

uIHostNameEdit.Text = Host;

Item125bap001.WaitForControlExist(5000);

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT"))

{

Mouse.Click(Item125bap001);

}

else if (Run\_Environment == "OST\_PROD")

{

Mouse.Click(Item111wallap002);

}

Mouse.Click(uISearchButton);

}

/// <summary>

/// Search\_Host\_Maintenance

/// </summary>

public void Search\_Host\_Maintenance\_All()

{

#region Variable Declarations

HtmlDiv uIHostMaintancePane = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIHostMaintancePane;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UISearchButton;

#endregion

uIHostMaintancePane.WaitForControlExist(5000);

Mouse.Click(uISearchButton);

}

/// <summary>

/// Quick\_Search\_Cost\_Code\_Routing

/// </summary>

public void Quick\_Search\_Cost\_Code\_Routing(String CostCode)

{

#region Variable Declarations

HtmlEdit UIQuickSearch = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuickSearch;

HtmlInputButton uIGOButton = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search.UIGOButton;

HtmlCustom UIQuick\_Search\_CB036025 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_CB036025;

HtmlCustom UIQuick\_Search\_507G2S = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_507G2S;

HtmlCustom UIQuick\_Search\_27531 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_27531;

HtmlCustom UIQuick\_Search\_GR26714 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_GR26714;

HtmlCustom UIQuick\_Search\_amsgtssf01 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_amsgtssf01;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares;

HtmlCustom UIQuick\_Search\_tr75765\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_tr75765\_in\_NTFS;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIQuickSearch.Text = CostCode;

if (Run\_Environment == "OST\_DEV")

{

UIQuick\_Search\_CB036025.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_CB036025);

}

else if (Run\_Environment == "OST\_PROD")

{

UIQuick\_Search\_507G2S.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_507G2S);

}

else if (Run\_Environment == "OST\_UAT")

{

UIQuick\_Search\_27531.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_27531);

}

}

/// <summary>

/// Quick\_Search\_Cost\_Code\_Routing

/// </summary>

public void Quick\_Search\_User\_Cost\_Code\_Routing(String SoeID)

{

#region Variable Declarations

HtmlEdit UIQuickSearch = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuickSearch;

HtmlInputButton uIGOButton = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search.UIGOButton;

HtmlCustom UIQuick\_Search\_CB036025 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_CB036025;

HtmlCustom UIQuick\_Search\_GR26714 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_GR26714;

HtmlCustom UIQuick\_Search\_AnthonyDaly = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_AnthonyDaly;

HtmlCustom UIQuick\_Search\_Stephen\_Randall = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_Stephen\_Randall; ;

HtmlCustom UIQuick\_Search\_amsgtssf01 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_amsgtssf01;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares;

HtmlCustom UIQuick\_Search\_tr75765\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_tr75765\_in\_NTFS;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIQuickSearch.Text = SoeID;

if (Run\_Environment == "OST\_DEV")

{

UIQuick\_Search\_GR26714.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_GR26714);

}

else if (Run\_Environment == "OST\_PROD")

{

UIQuick\_Search\_AnthonyDaly.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_AnthonyDaly);

}

else if (Run\_Environment == "OST\_UAT")

{

UIQuick\_Search\_Stephen\_Randall.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_Stephen\_Randall);

}

}

/// <summary>

/// Quick\_Search\_Host\_to\_NTFS\_page\_Routing

/// </summary>

public void Quick\_Search\_Host\_to\_NTFS\_page\_Routing(String Host)

{

#region Variable Declarations

HtmlEdit UIQuickSearch = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuickSearch;

HtmlInputButton uIGOButton = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search.UIGOButton;

HtmlCustom UIQuick\_Search\_CB036025 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_CB036025;

HtmlCustom UIQuick\_Search\_GR26714 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_GR26714;

HtmlCustom UIQuick\_Search\_nastam02ctam1 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_nastam02ctam1;

HtmlCustom UIQuick\_Search\_rutvnassec0003 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_rutvnassec0003;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares;

HtmlCustom UIQuick\_Search\_tr75765\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_tr75765\_in\_NTFS;

HtmlCustom UIQuick\_Search\_amsgtssf01 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_amsgtssf01;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIQuickSearch.Text = Host;

UIQuick\_Search\_nastam02ctam1.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_nastam02ctam1);

//if (Run\_Environment != "OST\_PROD")

//{

// UIQuick\_Search\_nastam02ctam1.WaitForControlExist(5000);

// Mouse.Click(UIQuick\_Search\_nastam02ctam1);

//}

//else if (Run\_Environment == "OST\_PROD")

//{

// UIQuick\_Search\_amsgtssf01.WaitForControlExist(5000);

// Mouse.Click(UIQuick\_Search\_amsgtssf01);

//}

}

/// <summary>

/// Quick\_Search\_Host\_to\_NTFS\_page\_Routing

/// </summary>

public void Quick\_Search\_Share\_to\_NTFS\_Permissions\_page\_Routing(String Share)

{

#region Variable Declarations

HtmlEdit UIQuickSearch = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuickSearch;

HtmlInputButton uIGOButton = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search.UIGOButton;

HtmlCustom UIQuick\_Search\_CB036025 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_CB036025;

HtmlCustom UIQuick\_Search\_GR26714 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_GR26714;

HtmlCustom UIQuick\_Search\_amsgtssf01 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_amsgtssf01;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares\_PROD = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares\_PRDO;

HtmlCustom UIQuick\_Search\_tr75765\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_tr75765\_in\_NTFS;

HtmlCustom UIQuick\_Search\_ab10978\_in\_NTFS\_UAT = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_ab10978\_in\_NTFS\_UAT;

HtmlCustom UIQuick\_Search\_ab10978\_in\_NTFS\_UAT\_Chrome = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_ab10978\_in\_NTFS\_UAT\_2;

HtmlCustom UIQuick\_Search\_ad23219\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_ad23219\_in\_NTFS;

HtmlCustom UIQuick\_Search\_AS23966\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_AS23966\_in\_NTFS;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIQuickSearch.Text = Share;

if (Run\_Environment != "OST\_PROD")

{

UIQuick\_Search\_tr75765\_in\_NTFS.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_tr75765\_in\_NTFS);

}

else

{

UIQuick\_Search\_ad23219\_in\_NTFS.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_AS23966\_in\_NTFS);

}

//else if (Run\_Environment == "OST\_UAT")

//{

// if (BrowserWindow.CurrentBrowser == "IE")

// {

// UIQuick\_Search\_ab10978\_in\_NTFS\_UAT.WaitForControlExist(5000);

// //Mouse.Click(UIQuick\_Search\_ab10978\_in\_NTFS\_UAT);

// Mouse.Click(UIQuick\_Search\_ab10978\_in\_NTFS\_UAT);

// }

// else if (BrowserWindow.CurrentBrowser == "Chrome")

// {

// UIQuick\_Search\_ab10978\_in\_NTFS\_UAT\_Chrome.WaitForControlExist(5000);

// //Mouse.Click(UIQuick\_Search\_ab10978\_in\_NTFS\_UAT);

// Mouse.Click(UIQuick\_Search\_ab10978\_in\_NTFS\_UAT\_Chrome);

// }

//}

}

/// <summary>

/// Quick\_Search\_Share\_to\_Shares\_page\_Routing

/// </summary>

public void Quick\_Search\_Share\_to\_Shares\_page\_Routing(String Share)

{

#region Variable Declarations

HtmlEdit UIQuickSearch = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuickSearch;

HtmlInputButton uIGOButton = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search.UIGOButton;

HtmlCustom UIQuick\_Search\_CB036025 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_CB036025;

HtmlCustom UIQuick\_Search\_GR26714 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_GR26714;

HtmlCustom UIQuick\_Search\_amsgtssf01 = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_amsgtssf01;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares;

HtmlCustom UIQuick\_Search\_zy13636\_in\_Shares\_UAT = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_zy13636\_in\_Shares\_UAT;

HtmlCustom UIQuick\_Search\_ad23219\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_ad23219\_in\_Shares;

HtmlCustom UIQuick\_Search\_AS23966\_in\_Shares = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_AS23966\_in\_Shares;

HtmlCustom UIQuick\_Search\_tr75765\_in\_NTFS = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIQuick\_Search\_tr75765\_in\_NTFS;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

UIQuickSearch.Text = Share;

if (Run\_Environment == "OST\_DEV")

{

UIQuick\_Search\_zy13636\_in\_Shares\_UAT.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_zy13636\_in\_Shares\_UAT);

}

else if (Run\_Environment == "OST\_PROD")

{

UIQuick\_Search\_ad23219\_in\_Shares.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_AS23966\_in\_Shares);

}

else if (Run\_Environment == "OST\_UAT")

{

UIQuick\_Search\_ad23219\_in\_Shares.WaitForControlExist(5000);

Mouse.Click(UIQuick\_Search\_zy13636\_in\_Shares\_UAT);

}

}

/// <summary>

/// Verify\_Elements\_User\_Roles - Use 'Verify\_Elements\_User\_RolesExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_User\_Roles()

{

#region Variable Declarations

HtmlDiv uIUserRolesPane = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UIUserRolesPane;

HtmlCustom uIDisplaysausersRoleanCustom = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UIDisplaysausersRoleanCustom;

HtmlLabel uISoeIdLabel = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UISearchFormCustom.UISoeIdLabel;

HtmlEdit uIItemEdit = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UISearchFormCustom.UIItemEdit;

HtmlButton uISearchButton = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UISearchFormCustom.UISearchButton;

#endregion

// Verify that the 'InnerText' property of 'User Roles' pane starts with 'User Roles'

StringAssert.StartsWith(uIUserRolesPane.InnerText, this.Verify\_Elements\_User\_RolesExpectedValues.UIUserRolesPaneInnerText);

// Verify that the 'Title' property of 'Displays a user's Role and Team access.' custom control equals 'Displays a user's Role and Team access.'

Assert.AreEqual(this.Verify\_Elements\_User\_RolesExpectedValues.UIDisplaysausersRoleanCustomTitle, uIDisplaysausersRoleanCustom.Title);

// Verify that the 'InnerText' property of 'Soe Id:' label starts with 'Soe Id:'

Assert.AreEqual(this.Verify\_Elements\_User\_RolesExpectedValues.UISoeIdLabelInnerText, uISoeIdLabel.InnerText.Trim());

// Verify that the 'Id' property of text box equals 'testid'

Assert.AreEqual(this.Verify\_Elements\_User\_RolesExpectedValues.UIItemEditId, uIItemEdit.Id);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Elements\_User\_RolesExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public virtual Verify\_Elements\_User\_RolesExpectedValues Verify\_Elements\_User\_RolesExpectedValues

{

get

{

if ((this.mVerify\_Elements\_User\_RolesExpectedValues == null))

{

this.mVerify\_Elements\_User\_RolesExpectedValues = new Verify\_Elements\_User\_RolesExpectedValues();

}

return this.mVerify\_Elements\_User\_RolesExpectedValues;

}

}

private Verify\_Elements\_User\_RolesExpectedValues mVerify\_Elements\_User\_RolesExpectedValues;

/// <summary>

/// Verify\_Elements\_System\_Information - Use 'Verify\_Elements\_System\_InformationExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Elements\_System\_Information()

{

#region Variable Declarations

HtmlDiv uISystemInfoPane = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UISystemInfoPane;

HtmlCustom uIThispageprovidesinfoCustom = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIThispageprovidesinfoCustom;

HtmlCell uIOstContextCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIOstContextCell;

HtmlCell uIAdSqlContextCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIAdSqlContextCell;

HtmlCell uIEmpContextCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIEmpContextCell;

HtmlCell uISupportContextCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UISupportContextCell;

HtmlCell uIOstDllVersionCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIOstDllVersionCell;

HtmlCell uIOstBuildDateCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIOstBuildDateCell;

HtmlCell uIOstFileVersionCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIOstFileVersionCell;

HtmlCell uIDataSourceNAMDNA390DCell = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIDataSourceNAMDNA390DCell;

HtmlCell uIDatasourceNAMDNA390DCell1 = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIDatasourceNAMDNA390DCell1;

HtmlCell UIDatasourceNAMDNA390DCell1\_Chrome = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIDatasourceNAMDNA390DCell1\_Chrome;

HtmlCell uIDataSourceNAMDNA390DCell2 = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIDataSourceNAMDNA390DCell2;

HtmlCell uIDataSourceNAMDNA390DCell3 = this.UIUserRolesOSTOperatioWindow.UISystemInfoOSTOperatiDocument.UIItemTable.UIDataSourceNAMDNA390DCell3;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'System Info' pane starts with 'System Info'

StringAssert.StartsWith(uISystemInfoPane.InnerText, this.Verify\_Elements\_System\_InformationExpectedValues.UISystemInfoPaneInnerText);

// Verify that the 'Title' property of 'This page provides information about OST version a...' custom control equals 'This page provides information about OST version and overall system information.'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIThispageprovidesinfoCustomTitle, uIThispageprovidesinfoCustom.Title);

// Verify that the 'InnerText' property of 'ostContext:' cell equals 'ostContext:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIOstContextCellInnerText, uIOstContextCell.InnerText);

// Verify that the 'InnerText' property of 'adSqlContext:' cell equals 'adSqlContext:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIAdSqlContextCellInnerText, uIAdSqlContextCell.InnerText);

// Verify that the 'InnerText' property of 'empContext:' cell equals 'empContext:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIEmpContextCellInnerText, uIEmpContextCell.InnerText);

// Verify that the 'InnerText' property of 'supportContext:' cell equals 'supportContext:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UISupportContextCellInnerText, uISupportContextCell.InnerText);

// Verify that the 'InnerText' property of 'ostDllVersion:' cell equals 'ostDllVersion:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIOstDllVersionCellInnerText, uIOstDllVersionCell.InnerText);

// Verify that the 'InnerText' property of 'ostBuildDate:' cell equals 'ostBuildDate:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIOstBuildDateCellInnerText, uIOstBuildDateCell.InnerText);

// Verify that the 'InnerText' property of 'ostFileVersion:' cell equals 'ostFileVersion:'

Assert.AreEqual(this.Verify\_Elements\_System\_InformationExpectedValues.UIOstFileVersionCellInnerText, uIOstFileVersionCell.InnerText);

// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

StringAssert.StartsWith(uIDataSourceNAMDNA390DCell.InnerText, this.Verify\_Elements\_System\_InformationExpectedValues.UIDataSourceNAMDNA390DCellInnerText);

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT") || (Run\_Environment == "OST\_PROD"))

{

// Verify that the 'InnerText' property of 'Data source=NAMDNA390DSQL1.NAMUAT.NSROOT' cell starts with 'Data source='

StringAssert.StartsWith(UIDatasourceNAMDNA390DCell1\_Chrome.InnerText.ToLower(), this.Verify\_Elements\_System\_InformationExpectedValues.UIDatasourceNAMDNA390DCell1InnerText.ToLower());

}

else

{

// Verify that the 'InnerText' property of 'Data source=NAMDNA390DSQL1.NAMUAT.NSROOT' cell starts with 'Data source='

StringAssert.StartsWith(uIDatasourceNAMDNA390DCell1.InnerText.ToLower(), this.Verify\_Elements\_System\_InformationExpectedValues.UIDatasourceNAMDNA390DCell1InnerText.ToLower());

}

// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

StringAssert.StartsWith(uIDataSourceNAMDNA390DCell2.InnerText, this.Verify\_Elements\_System\_InformationExpectedValues.UIDataSourceNAMDNA390DCell2InnerText);

// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

StringAssert.StartsWith(uIDataSourceNAMDNA390DCell3.InnerText, this.Verify\_Elements\_System\_InformationExpectedValues.UIDataSourceNAMDNA390DCell3InnerText);

}

public virtual Verify\_Elements\_System\_InformationExpectedValues Verify\_Elements\_System\_InformationExpectedValues

{

get

{

if ((this.mVerify\_Elements\_System\_InformationExpectedValues == null))

{

this.mVerify\_Elements\_System\_InformationExpectedValues = new Verify\_Elements\_System\_InformationExpectedValues();

}

return this.mVerify\_Elements\_System\_InformationExpectedValues;

}

}

private Verify\_Elements\_System\_InformationExpectedValues mVerify\_Elements\_System\_InformationExpectedValues;

/// <summary>

/// Open\_Monitor\_popup

/// </summary>

public void Open\_Monitor\_popup()

{

#region Variable Declarations

HtmlImage uIFrankfurtImage = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIFrankfurtImage;

HtmlImage uIFFTCTISVOST001UImage = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIFFTCTISVOST001UImage;

HtmlImage uITexasImage = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UITexasImage;

HtmlImage UIOSTMONPV609UImage = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIOSTMONPV609UImage;

HtmlImage FFTCTISVOST001Image = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.FFTCTISVOST001Image;

HtmlSpan uIMonitorOverviewPane = this.UIOstMonitorsOverviewOWindow.UIOstMonitorsOverviewODocument.UIMonitorOverviewPane;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

uIFrankfurtImage.WaitForControlReady(5000);

// Click 'Frankfurt' image

Mouse.Click(uIFrankfurtImage);

}

else if (Run\_Environment == "OST\_UAT")

{

uITexasImage.WaitForControlReady(5000);

// Click 'Frankfurt' image

Mouse.Click(uITexasImage);

UIOSTMONPV609UImage.WaitForControlReady(5000);

// Click 'FFTCTISVOST001U' image

Mouse.Click(UIOSTMONPV609UImage);

}

else if (Run\_Environment == "OST\_PROD")

{

uIFrankfurtImage.WaitForControlReady(5000);

// Click 'Frankfurt' image

Mouse.Click(uIFrankfurtImage);

FFTCTISVOST001Image.WaitForControlReady(5000);

// Click 'FFTCTISVOST001U' image

Mouse.Click(FFTCTISVOST001Image);

}

uIMonitorOverviewPane.WaitForControlReady(5000);

}

/// <summary>

/// Navigate\_to\_Global\_Compliance

/// </summary>

public void Navigate\_to\_Global\_Compliance()

{

#region Variable Declarations

HtmlHyperlink uISharesHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane8.UISharesHyperlink;

HtmlHyperlink uIGlobalComplianceHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane8.UIGlobalComplianceHyperlink;

#endregion

// Click 'Shares' link

Mouse.Click(uISharesHyperlink);

// Click 'Global Compliance' link

Mouse.Click(uIGlobalComplianceHyperlink);

}

/// <summary>

/// Open\_Edit\_Host\_Maintenance\_popup

/// </summary>

public void Open\_Edit\_Host\_Maintenance\_popup()

{

#region Variable Declarations

HtmlHyperlink uIEditHyperlink = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIItem1Row.UIEditHyperlink;

HtmlLabel uIHostNameLabel = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDetailsContentPane.UIHostNameLabel;

#endregion

// Click 'Edit' link

Mouse.Click(uIEditHyperlink);

uIHostNameLabel.WaitForControlExist(5000);

}

/// <summary>

/// Edit\_Host\_on\_Host\_Maintenance\_page

/// </summary>

public void Edit\_Host\_on\_Host\_Maintenance\_page(String TeamName, String Division, String DataCenter, String COBPartner, String MonitoringServer, String OrgTeamName, String OrgDivision, String OrgDataCenter, String OrgCOBPartner, String OrgMonitoringServer)

{

#region Variable Declarations

HtmlDiv uIItem125bap001Pane = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDetailsContentPane.UIItem125bap001Pane;

HtmlComboBox uITeamIDComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UITeamIDComboBox;

HtmlEdit uIDivisionEdit = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDivisionEdit;

HtmlCheckBox uIProductionFileServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIProductionFileServerCheckBox;

HtmlCheckBox uICOBServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICOBServerCheckBox;

HtmlCheckBox uISOEApplicationServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UISOEApplicationServerCheckBox;

HtmlCheckBox uIPrintServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIPrintServerCheckBox;

HtmlCheckBox uIDomainControllerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDomainControllerCheckBox;

HtmlCheckBox uIBranchServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIBranchServerCheckBox;

HtmlCheckBox uIMiscellaneousServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMiscellaneousServerCheckBox;

HtmlCheckBox uINASFilerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UINASFilerCheckBox;

HtmlCheckBox uICompressedFolderChecCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICompressedFolderChecCheckBox;

HtmlCheckBox uIDecommissionfromOSTCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDecommissionfromOSTCheckBox;

HtmlCheckBox uIMigratedCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMigratedCheckBox;

HtmlComboBox uIDataCenterIdComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDataCenterIdComboBox;

HtmlComboBox uICOBPartnerComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICOBPartnerComboBox;

HtmlComboBox uIMonitoringServerIdComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMonitoringServerIdComboBox;

HtmlButton uIUpdateButton = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIUpdateButton;

HtmlDiv uIMessagePlaceholderPane = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMessagePlaceholderPane;

#endregion

Mouse.Click(uITeamIDComboBox);

uITeamIDComboBox.SelectedItem = "--Select One--";

uIDivisionEdit.Text = "";

uIDivisionEdit.Text = Division;

uITeamIDComboBox.SelectedItem = TeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(2000);

uIProductionFileServerCheckBox.Checked = true;

uICOBServerCheckBox.Checked = true;

uISOEApplicationServerCheckBox.Checked = true;

uIPrintServerCheckBox.Checked = true;

uIDomainControllerCheckBox.Checked = true;

uIBranchServerCheckBox.Checked = true;

uIMiscellaneousServerCheckBox.Checked = true;

uINASFilerCheckBox.Checked = true;

uICompressedFolderChecCheckBox.Checked = true;

uIDomainControllerCheckBox.Checked = true;

uIDataCenterIdComboBox.SelectedItem = DataCenter;

if (uICOBPartnerComboBox.ItemCount < 2)

{

uITeamIDComboBox.SelectedItem = "--Select One--";

uITeamIDComboBox.SelectedItem = TeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(3000);

if (uICOBPartnerComboBox.ItemCount < 2)

{

uITeamIDComboBox.SelectedItem = "--Select One--";

uITeamIDComboBox.SelectedItem = TeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(3000);

}

}

uICOBPartnerComboBox.SelectedItem = COBPartner;

uIMonitoringServerIdComboBox.SelectedItem = MonitoringServer;

Mouse.Click(uIUpdateButton);

uIMessagePlaceholderPane.WaitForControlExist(5000);

}

/// <summary>

/// Revert\_Edit\_Host\_on\_Host\_Maintenance\_page

/// </summary>

public void Revert\_Edit\_Host\_on\_Host\_Maintenance\_page(String TeamName, String Division, String DataCenter, String COBPartner, String MonitoringServer, String OrgTeamName, String OrgDivision, String OrgDataCenter, String OrgCOBPartner, String OrgMonitoringServer)

{

#region Variable Declarations

HtmlDiv uIItem125bap001Pane = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDetailsContentPane.UIItem125bap001Pane;

HtmlComboBox uITeamIDComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UITeamIDComboBox;

HtmlEdit uIDivisionEdit = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDivisionEdit;

HtmlCheckBox uIProductionFileServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIProductionFileServerCheckBox;

HtmlCheckBox uICOBServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICOBServerCheckBox;

HtmlCheckBox uISOEApplicationServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UISOEApplicationServerCheckBox;

HtmlCheckBox uIPrintServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIPrintServerCheckBox;

HtmlCheckBox uIDomainControllerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDomainControllerCheckBox;

HtmlCheckBox uIBranchServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIBranchServerCheckBox;

HtmlCheckBox uIMiscellaneousServerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMiscellaneousServerCheckBox;

HtmlCheckBox uINASFilerCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UINASFilerCheckBox;

HtmlCheckBox uICompressedFolderChecCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICompressedFolderChecCheckBox;

HtmlCheckBox uIDecommissionfromOSTCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDecommissionfromOSTCheckBox;

HtmlCheckBox uIMigratedCheckBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMigratedCheckBox;

HtmlComboBox uIDataCenterIdComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIDataCenterIdComboBox;

HtmlComboBox uICOBPartnerComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UICOBPartnerComboBox;

HtmlComboBox uIMonitoringServerIdComboBox = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMonitoringServerIdComboBox;

HtmlButton uIUpdateButton = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIUpdateButton;

HtmlDiv uIMessagePlaceholderPane = this.UIHostMaintenanceOSTOpWindow.UIHostMaintenanceOSTOpDocument.UIMessagePlaceholderPane;

#endregion

Mouse.Click(uITeamIDComboBox);

uITeamIDComboBox.SelectedItem = "--Select One--";

uIDivisionEdit.Text = "";

uIDivisionEdit.Text = OrgDivision;

uITeamIDComboBox.SelectedItem = OrgTeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(2000);

uIProductionFileServerCheckBox.Checked = false;

uICOBServerCheckBox.Checked = false;

uISOEApplicationServerCheckBox.Checked = false;

uIPrintServerCheckBox.Checked = false;

uIDomainControllerCheckBox.Checked = false;

uIBranchServerCheckBox.Checked = false;

uIMiscellaneousServerCheckBox.Checked = false;

uINASFilerCheckBox.Checked = false;

uICompressedFolderChecCheckBox.Checked = false;

uIDomainControllerCheckBox.Checked = false;

uIDataCenterIdComboBox.SelectedItem = OrgDataCenter;

Console.WriteLine("Number of items: " + uICOBPartnerComboBox.ItemCount);

if (uICOBPartnerComboBox.ItemCount < 2)

{

uITeamIDComboBox.SelectedItem = "--Select One--";

uITeamIDComboBox.SelectedItem = OrgTeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(3000);

if (uICOBPartnerComboBox.ItemCount < 2)

{

uITeamIDComboBox.SelectedItem = "--Select One--";

uITeamIDComboBox.SelectedItem = OrgTeamName;

Keyboard.SendKeys(uITeamIDComboBox, "{Down}+{Up}");

Playback.Wait(3000);

}

}

uICOBPartnerComboBox.SelectedItem = OrgCOBPartner;

uIMonitoringServerIdComboBox.SelectedItem = OrgMonitoringServer;

Mouse.Click(uIUpdateButton);

uIMessagePlaceholderPane.WaitForControlExist(5000);

}

/// <summary>

/// Search\_Owner\_Info\_page

/// </summary>

public void Search\_Owner\_Info\_page(String UNCPath)

{

#region Variable Declarations

HtmlLabel uIPrimaryOwnerIDLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane.UIPrimaryOwnerIDLabel;

HtmlLabel uISecondaryOwnerIDLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane.UISecondaryOwnerIDLabel;

HtmlLabel uIStatusCodeLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane.UIStatusCodeLabel;

HtmlLabel uIMessageLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane.UIMessageLabel;

HtmlLabel uIStatusMessageLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane.UIStatusMessageLabel;

HtmlDiv uIPrimaryOwner = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIPrimaryOwner;

HtmlDiv uISecondaryOwner = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UISecondaryOwner;

HtmlDiv uIItem5Pane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIItem5Pane;

HtmlDiv uIMessageDivPane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIMessageDivPane;

HtmlDiv uIStatusMsgDivPane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIStatusMsgDivPane;

HtmlDiv uIResultTablePane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIResultTablePane;

HtmlDiv uIOwnersForUNCPathPane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIOwnersForUNCPathPane;

HtmlEdit uIUncPathInputBoxEdit = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIUncPathInputBoxEdit;

HtmlButton uISearchButton = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UISearchButton;

#endregion

uIOwnersForUNCPathPane.WaitForControlExist(5000);

uIUncPathInputBoxEdit.Text = UNCPath;

Mouse.Click(uISearchButton);

uIMessageDivPane.WaitForControlPropertyNotEqual("InnerText", "", 180000);

Playback.Wait(2000);

try

{

Assert.IsTrue(uIMessageDivPane.InnerText.ToString().Contains("Looking") == true, "Verifying that message is shown.");

Assert.IsTrue(uIMessageDivPane.InnerText.ToString().Contains("Error") == false, "There is Error mentioned in the message.");

Assert.IsTrue(uIPrimaryOwner.InnerText.ToString().Contains("null") == false, "No owner found, null value reported.");

}

catch

{

Assert.AreEqual("y", "x", "Something went wrong, possibly results table did not appear.");

}

}

/// <summary>

/// Verify\_Host\_Maintenance\_page - Use 'Verify\_Host\_Maintenance\_pageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Host\_Maintenance\_page()

{

#region Variable Declarations

HtmlDiv uIHostMaintancePane = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIHostMaintancePane;

HtmlLabel uIRegionLabel = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIBaseSearchRegionPane.UIRegionLabel;

HtmlLabel uITeamLabel = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIBaseSearchRegionPane.UITeamLabel;

HtmlLabel uIHostLabel = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIBaseSearchRegionPane.UIHostLabel;

HtmlLabel uIStatusLabel = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UIStatusLabel;

HtmlLabel uIHostTypeLabel = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UIHostTypeLabel;

HtmlComboBox uIAvailableRegionsDdlComboBox = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIAvailableRegionsDdlComboBox;

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIAvailableTeamsDdlComboBox;

HtmlEdit uIHostNameEdit = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UIHostNameEdit;

HtmlComboBox uIItemComboBox = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UIItemComboBox;

HtmlComboBox uIItemComboBox1 = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UIItemComboBox1;

HtmlButton uISearchButton = this.UIOSTHomeOSTOperationsWindow1.UIHostMaintanceOSTOperDocument.UISearchFormCustom.UISearchButton;

#endregion

// Verify that the 'InnerText' property of 'HostMaintance' pane starts with 'Host Maintenance'

StringAssert.StartsWith(uIHostMaintancePane.InnerText, this.Verify\_Host\_Maintenance\_pageExpectedValues.UIHostMaintancePaneInnerText);

// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIRegionLabelInnerText, uIRegionLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

StringAssert.StartsWith(uITeamLabel.InnerText, this.Verify\_Host\_Maintenance\_pageExpectedValues.UITeamLabelInnerText);

// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIHostLabelInnerText, uIHostLabel.InnerText.Trim());

// Verify that the 'InnerText' property of 'Status:' label equals 'Status:'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIStatusLabelInnerText, uIStatusLabel.InnerText);

// Verify that the 'InnerText' property of 'Host Type:' label equals 'Host Type:'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIHostTypeLabelInnerText, uIHostTypeLabel.InnerText);

// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIAvailableRegionsDdlComboBoxId, uIAvailableRegionsDdlComboBox.Id);

// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIAvailableTeamsDdlComboBoxId, uIAvailableTeamsDdlComboBox.Id);

// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIHostNameEditId, uIHostNameEdit.Id);

// Verify that the 'Id' property of combo box equals 'StatusSelect'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIItemComboBoxId, uIItemComboBox.Id);

// Verify that the 'Id' property of combo box equals 'HostTypeSelect'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UIItemComboBox1Id, uIItemComboBox1.Id);

// Verify that the 'InnerText' property of 'Search' button equals 'Search'

Assert.AreEqual(this.Verify\_Host\_Maintenance\_pageExpectedValues.UISearchButtonInnerText, uISearchButton.InnerText);

}

public virtual Verify\_Host\_Maintenance\_pageExpectedValues Verify\_Host\_Maintenance\_pageExpectedValues

{

get

{

if ((this.mVerify\_Host\_Maintenance\_pageExpectedValues == null))

{

this.mVerify\_Host\_Maintenance\_pageExpectedValues = new Verify\_Host\_Maintenance\_pageExpectedValues();

}

return this.mVerify\_Host\_Maintenance\_pageExpectedValues;

}

}

private Verify\_Host\_Maintenance\_pageExpectedValues mVerify\_Host\_Maintenance\_pageExpectedValues;

/// <summary>

/// Open\_Add\_Share\_Exclusion\_popup

/// </summary>

public void Open\_Add\_Share\_Exclusion\_popup()

{

#region Variable Declarations

HtmlInputButton uIAddShareExclusionButton1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddShareExclusionButton1;

HtmlInputButton uIAddShareExclusionButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddShareExclusionButton;

#endregion

// Click 'Add Share Exclusion' button

Mouse.Click(uIAddShareExclusionButton1);

uIAddShareExclusionButton.WaitForControlExist(5000);

}

/// <summary>

/// Open\_Add\_Path\_Exclusion\_popup

/// </summary>

public void Open\_Add\_Path\_Exclusion\_popup()

{

#region Variable Declarations

HtmlInputButton uIAddPathExclusionButton1 = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddPathExclusionButton1;

HtmlInputButton uIAddPathExclusionButton = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIAddPathExclusionButton;

#endregion

// Click 'Add Path Exclusion' button

Mouse.Click(uIAddPathExclusionButton1);

uIAddPathExclusionButton.WaitForControlExist(5000);

}

/// <summary>

/// Click\_Partition\_Link\_on\_Datastats\_Overview\_page

/// </summary>

public void Click\_Partition\_Link\_on\_Datastats\_Overview\_page()

{

#region Variable Declarations

HtmlHyperlink uIDHyperlink = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIItem1Row.UIDHyperlink;

HtmlHyperlink UIDHyperlink\_Prod = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIItem1Row.UIDHyperlink\_Prod;

HtmlHyperlink uID531GBsHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIDataStatsDetailsOSTODocument.UID531GBsCustom.UID531GBsHyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'D$\' link

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT"))

{

Mouse.Click(uIDHyperlink);

}

else if (Run\_Environment == "OST\_PROD")

{

Mouse.Click(UIDHyperlink\_Prod);

}

uID531GBsHyperlink.WaitForControlExist(5000);

uID531GBsHyperlink.WaitForControlReady(5000);

}

/// <summary>

/// Clear\_Chrome\_Cache - Use 'Clear\_Chrome\_CacheParams' to pass parameters into this method.

/// </summary>

public void Clear\_Chrome\_Cache()

{

#region Variable Declarations

WinEdit uIAddressandsearchbarEdit = this.UIGoogleChromeWindow.UIAddressandsearchbarGroup.UIAddressandsearchbarEdit;

HtmlButton UIClear\_Cache\_Button = this.UIGoogleChromeWindow.UIChromeWindow.UIClear\_Cache\_Button;

#endregion

BrowserWindow vm8b54769dnamnsrootnetBrowser = BrowserWindow.Launch(new System.Uri("chrome://settings/clearBrowserData"));

//Mouse.Click(UIClear\_Cache\_Button);

Keyboard.SendKeys("{Tab}" + "{Enter}");

//uIAddressandsearchbarEdit.WaitForControlPropertyEqual(uIAddressandsearchbarEdit.Text, "chrome://settings/", 10000);

Playback.Wait(8000);

}

public void Wait\_for\_Chrome\_Cache\_clear()

{

#region Variable Declarations

WinEdit uIAddressandsearchbarEdit = this.UIGoogleChromeWindow.UIAddressandsearchbarGroup.UIAddressandsearchbarEdit;

#endregion

uIAddressandsearchbarEdit.WaitForControlPropertyEqual(uIAddressandsearchbarEdit.Text, "chrome://settings", 10000);

}

/// <summary>

/// Open\_Edit\_Recurring\_Schedule\_page

/// </summary>

public void Open\_Edit\_Recurring\_Schedule\_page()

{

#region Variable Declarations

HtmlHyperlink uIAddRecurrenceHyperlink = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIItem1799Row.UIAddRecurrenceHyperlink;

HtmlSpan uIEditRecurringSchedulPane = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIEditRecurringSchedulPane;

#endregion

// Click 'Add Recurrence' link

Mouse.Click(uIAddRecurrenceHyperlink);

uIEditRecurringSchedulPane.WaitForControlExist(5000);

}

public void Search\_Home\_Global\_Search(String Text)

{

#region Variable Declarations

HtmlEdit uIHomeSearchTextEdit = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHomeSearchTextEdit;

HtmlButton uIGOButton = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIOstGlobalSearchGOPane.UIGOButton;

#endregion

uIHomeSearchTextEdit.Text = Text;

Playback.Wait(5000);

}

public void Search\_Home\_Global\_Search\_for\_Host(String Host)

{

#region Variable Declarations

HtmlEdit uIHomeSearchTextEdit = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHomeSearchTextEdit;

HtmlButton uIGOButton = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIOstGlobalSearchGOPane.UIGOButton;

#endregion

//uIHomeSearchTextEdit.Text = Host;

Mouse.Click(uIHomeSearchTextEdit);

Keyboard.SendKeys(uIHomeSearchTextEdit, Host);

Playback.Wait(500);

SendKeys.SendWait("{BACKSPACE}");

Playback.Wait(2000);

SendKeys.SendWait("{BACKSPACE}");

Playback.Wait(5000);

}

public virtual Search\_Home\_Global\_Search\_by\_SoeidParams Search\_Home\_Global\_Search\_by\_SoeidParams

{

get

{

if ((this.mSearch\_Home\_Global\_Search\_by\_SoeidParams == null))

{

this.mSearch\_Home\_Global\_Search\_by\_SoeidParams = new Search\_Home\_Global\_Search\_by\_SoeidParams();

}

return this.mSearch\_Home\_Global\_Search\_by\_SoeidParams;

}

}

private Search\_Home\_Global\_Search\_by\_SoeidParams mSearch\_Home\_Global\_Search\_by\_SoeidParams;

/// <summary>

/// Verify\_Home\_Global\_Search\_results - Use 'Verify\_Home\_Global\_Search\_resultsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Home\_Global\_Search\_results\_for\_User\_Cost\_Code(String CostCode)

{

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

#region Variable Declarations

HtmlDiv uIHostPane = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHosttestkc90246Pane.UIHostPane;

HtmlDiv uISoeIdPane = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UISoeIdKrzysztofCyganPane.UISoeIdPane;

HtmlHyperlink uITestkc90246Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UITestkc90246Hyperlink;

HtmlHyperlink uIKrzysztofCyganHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentSoeIdCustom.UIKrzysztofCyganHyperlink;

HtmlHyperlink uICostCodePaneHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UICostCode\_Hyperlink;

HtmlHyperlink uIGloriaRosarioHyperlink = this.UIOSTHomeOSTOperationsWindow3.UIOSTHomeOSTOperationsDocument.UIContentSoeIdCustom.UIGloriaRosarioHyperlink;

#endregion

if (Run\_Environment == "OST\_DEV")

{

StringAssert.Contains(uIGloriaRosarioHyperlink.Href.ToString().ToLower(), CostCode.ToLower());

}

//else if (Run\_Environment == "OST\_UAT")

//{

// //Assert.AreEqual(this.Verify\_Home\_Global\_Search\_resultsExpectedValues.UICostCodePaneInnerText, uICostCodePane.InnerText);

// StringAssert.Contains(uICostCodePaneHyperlink.Href.ToString().ToLower(), CostCode.ToLower());

//}

else if ((Run\_Environment == "OST\_PROD")|| (Run\_Environment == "OST\_UAT"))

{

//Assert.AreEqual(this.Verify\_Home\_Global\_Search\_resultsExpectedValues.UICostCodePaneInnerText, uICostCodePane.InnerText);

StringAssert.Contains(uICostCodePaneHyperlink.Href.ToString().ToLower(), CostCode.ToLower());

}

}

public virtual Verify\_Home\_Global\_Search\_resultsExpectedValues Verify\_Home\_Global\_Search\_resultsExpectedValues

{

get

{

if ((this.mVerify\_Home\_Global\_Search\_resultsExpectedValues == null))

{

this.mVerify\_Home\_Global\_Search\_resultsExpectedValues = new Verify\_Home\_Global\_Search\_resultsExpectedValues();

}

return this.mVerify\_Home\_Global\_Search\_resultsExpectedValues;

}

}

private Verify\_Home\_Global\_Search\_resultsExpectedValues mVerify\_Home\_Global\_Search\_resultsExpectedValues;

public void Verify\_Home\_Global\_Search\_results\_for\_Host(String Host)

{

#region Variable Declarations

HtmlDiv uIHostPane = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHosttestkc90246Pane.UIHostPane;

HtmlHyperlink UInasrut151v1crpHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UInasrut151v1crpHyperlink;

HtmlHyperlink UInastam02ctam1Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UInastam02ctam1Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Host' pane equals 'Host'

Assert.AreEqual(this.Verify\_Home\_Global\_Search\_resultsExpectedValues.UIHostPaneInnerText, uIHostPane.InnerText);

if (Run\_Environment == "OST\_DEV")

{

Assert.AreEqual(UInastam02ctam1Hyperlink.InnerText.ToString().ToLower(), Host);

}

else if ((Run\_Environment == "OST\_PROD") || (Run\_Environment == "OST\_UAT"))

{

Assert.AreEqual(UInastam02ctam1Hyperlink.InnerText.ToString().ToLower(), Host);

}

}

public void Verify\_Home\_Global\_Search\_results\_for\_Share\_NTFS(String Share)

{

#region Variable Declarations

HtmlDiv uIHostPane = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHosttestkc90246Pane.UIHostPane;

HtmlHyperlink uINastam02ctam1tr75765Hyperlink = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIContentShareCustom.UINastam02ctam1tr75765Hyperlink;

HtmlHyperlink UIautoaudit\_archives\_NTFS\_Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIautoaudit\_archives\_NTFS\_Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Host' pane equals 'Host'

//Assert.AreEqual(this.Verify\_Home\_Global\_Search\_resultsExpectedValues.UIHostPaneInnerText, uIHostPane.InnerText);

//if(Run\_Environment != "OST\_PROD")

StringAssert.Contains(uINastam02ctam1tr75765Hyperlink.InnerText.ToString().ToLower(), Share);

//else

// StringAssert.Contains(UIautoaudit\_archives\_NTFS\_Hyperlink.InnerText.ToString().ToLower(), Share);

}

public void Verify\_Home\_Global\_Search\_results\_for\_Share\_Shares(String Share)

{

#region Variable Declarations

HtmlDiv uIHostPane = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIHosttestkc90246Pane.UIHostPane;

HtmlHyperlink uINastam02ctam1zy13636ShareHyperlink = this.UIOSTHomeOSTOperationsWindow3.UIOSTHomeOSTOperationsDocument.UIContentShareCustom.UINastam02ctam1zy13636ShareHyperlink;

HtmlHyperlink UIautoaudit\_archives\_Shares\_Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIautoaudit\_archives\_Shares\_Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Verify that the 'InnerText' property of 'Host' pane equals 'Host'

//Assert.AreEqual(this.Verify\_Home\_Global\_Search\_resultsExpectedValues.UIHostPaneInnerText, uIHostPane.InnerText);

// Verify that the 'InnerText' property of 'test kc90246' link equals 'TEST KC90246'

if((Run\_Environment == "OST\_DEV")||(Run\_Environment == "OST\_PROD"))

StringAssert.Contains(uINastam02ctam1zy13636ShareHyperlink.InnerText.ToString().ToLower(), Share.ToLower());

else

StringAssert.Contains(UIautoaudit\_archives\_Shares\_Hyperlink.InnerText.ToString().ToLower(), Share);

}

public void Click\_CostCode\_link\_on\_Home\_Global\_Search\_results()

{

#region Variable Declarations

HtmlHyperlink UICostCodeLink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIGB5300Hyperlink;

HtmlHyperlink UICostCodeLinkPROD = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UI507G2SHyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Krzysztof Cygan' link

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_UAT"))

{

Mouse.Click(UICostCodeLink);

}

else if ((Run\_Environment == "OST\_PROD"))

{

Mouse.Click(UICostCodeLinkPROD);

}

}

public void Click\_Host\_link\_on\_Home\_Global\_Search\_results()

{

#region Variable Declarations

HtmlHyperlink UInasrut151v1crpHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UInasrut151v1crpHyperlink;

HtmlHyperlink UInastam02ctam1Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UInastam02ctam1Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if (Run\_Environment == "OST\_DEV")

{

// Click 'Krzysztof Cygan' link

Mouse.Click(UInastam02ctam1Hyperlink);

}

else if ((Run\_Environment == "OST\_PROD") || (Run\_Environment == "OST\_UAT"))

{

// Click 'Krzysztof Cygan' link

Mouse.Click(UInastam02ctam1Hyperlink);

}

}

public void Click\_Share\_NTFS\_link\_on\_Home\_Global\_Search\_results()

{

#region Variable Declarations

//HtmlHyperlink UIautoaudit\_archives\_NTFS\_Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIautoaudit\_archives\_NTFS\_Hyperlink;

HtmlHyperlink uINastam02ctam1tr75765Hyperlink = this.UIOSTHomeOSTOperationsWindow2.UIOSTHomeOSTOperationsDocument.UIContentShareCustom.UINastam02ctam1tr75765Hyperlink;

HtmlHyperlink UIautoaudit\_archives\_NTFS\_Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIautoaudit\_archives\_NTFS\_Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

// Click 'Krzysztof Cygan' link

//if(Run\_Environment != "OST\_PROD")

Mouse.Click(uINastam02ctam1tr75765Hyperlink);

//else

// Mouse.Click(UIautoaudit\_archives\_NTFS\_Hyperlink);

}

public void Click\_Share\_Shares\_link\_on\_Home\_Global\_Search\_results()

{

#region Variable Declarations

HtmlHyperlink uINastam02ctam1zy13636ShareHyperlink = this.UIOSTHomeOSTOperationsWindow3.UIOSTHomeOSTOperationsDocument.UIContentShareCustom.UINastam02ctam1zy13636ShareHyperlink;

HtmlHyperlink UIautoaudit\_archives\_Shares\_Hyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentHostCustom.UIautoaudit\_archives\_Shares\_Hyperlink;

#endregion

string Run\_Environment = System.Environment.GetEnvironmentVariable("OST\_Env");

if ((Run\_Environment == "OST\_DEV") || (Run\_Environment == "OST\_PROD"))

Mouse.Click(uINastam02ctam1zy13636ShareHyperlink);

else

Mouse.Click(UIautoaudit\_archives\_Shares\_Hyperlink);

}

/// <summary>

/// Navigate\_to\_Share\_Exclusion

/// </summary>

public void Navigate\_to\_Share\_Exclusion()

{

#region Variable Declarations

HtmlHyperlink uIComplianceHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIComplianceHyperlink;

HtmlHyperlink uIShareExclusionHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIShareExclusionHyperlink;

HtmlControl uIGlobalExclusionsPane = this.UIShareExclusionOSTOpeWindow.UIShareExclusionOSTOpeDocument.UIHomeShareExclusionGlPane.UIGlobalExclusionsPane;

#endregion

// Click 'Compliance' link

Mouse.Click(uIComplianceHyperlink);

// Click 'Share Exclusion' link

Mouse.Click(uIShareExclusionHyperlink);

uIGlobalExclusionsPane.WaitForControlExist(3000);

}

/// <summary>

/// Navigate\_to\_Sharepoint\_Info

/// </summary>

public void Navigate\_to\_Sharepoint\_Info()

{

#region Variable Declarations

HtmlHyperlink uIComplianceHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIComplianceHyperlink;

HtmlHyperlink uiSharepointInfoHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UISharepointInfoHyperlink;

HtmlControl uiSharepointInformatioPane = this.UISharepointInformatioWindow.UISharepointInformatioDocument.UISharepointInformatioPane;

#endregion

// Click 'Compliance' link

Mouse.Click(uIComplianceHyperlink);

// Click 'Sharepoint Info' link

Mouse.Click(uiSharepointInfoHyperlink);

uiSharepointInformatioPane.WaitForControlExist(3000);

}

public void Navigate\_to\_NAS\_PhysicaDevice\_Info()

{

#region Variable Declarations

HtmlHyperlink uIComplianceHyperlink = this.UIPrintQueueHistoryOSTWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane2.UIComplianceHyperlink;

HtmlHyperlink UINasPhysicalDeviceHyperlink = this.UIOSTHomeOSTOperationsWindow.UIOSTHomeOSTOperationsDocument1.UIMainnavPane4.UINasPhysicalDeviceHyperlink;

HtmlControl UINasDeviceHeaderPane = this.UINASPhysicalDeviceInfWindow.UINASPhysicalDeviceInfDocument.UINASPhysicalDeviceInfPane;

#endregion

// Click 'Compliance' link

Mouse.Click(uIComplianceHyperlink);

// Click 'Sharepoint Info' link

Mouse.Click(UINasPhysicalDeviceHyperlink);

UINasDeviceHeaderPane.WaitForControlExist(3000);

}

/// <summary>

/// Verify\_MongoDB\_Health\_Page - Use 'Verify\_MongoDB\_Health\_PageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_MongoDB\_Health\_Page()

{

#region Variable Declarations

HtmlCell uIHostNameCell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIHostNameCell;

HtmlCell uIDisconnectedCell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIDisconnectedCell;

HtmlCell uIVm1b98f53fCell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIVm1b98f53fCell;

HtmlCell uIPortCell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIPortCell;

HtmlCell uIItem32017Cell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIItem32017Cell;

HtmlCell uIStatusCell = this.UIMongoDBHealthOSTOperWindow.UIMongoDBHealthOSTOperDocument.UIResultTablePane.UIItemTable.UIStatusCell;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

// Verify that the 'InnerText' property of 'Share Information' pane equals 'Share Information'

//StringAssert.StartsWith(uIShareInformationPane.InnerText, this.Verify\_Elements\_SharesExpectedValues.UIShareInformationPaneInnerText);

Assert.AreEqual(UIHeaderTooltip.Title, "Shows the overall status and health of the Mongodb");

// Verify that the 'InnerText' property of 'Host Name :' cell equals 'Host Name :'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIHostNameCellInnerText, uIHostNameCell.InnerText);

// Verify that the 'InnerText' property of 'Disconnected' cell equals 'Connected'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIDisconnectedCellInnerText, uIDisconnectedCell.InnerText);

// Verify that the 'InnerText' property of 'vm-1b98-f53f' cell equals 'vm-1b98-f53f'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIVm1b98f53fCellInnerText, uIVm1b98f53fCell.InnerText);

// Verify that the 'InnerText' property of 'Port :' cell equals 'Port :'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIPortCellInnerText, uIPortCell.InnerText);

// Verify that the 'InnerText' property of '32017' cell equals '32017'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIItem32017CellInnerText, uIItem32017Cell.InnerText);

// Verify that the 'InnerText' property of 'Status :' cell equals 'Status :'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIStatusCellInnerText, uIStatusCell.InnerText);

// Verify that the 'InnerText' property of 'Disconnected' cell equals 'Connected'

Assert.AreEqual(this.Verify\_MongoDB\_Health\_PageExpectedValues.UIDisconnectedCellInnerText1, uIDisconnectedCell.InnerText);

}

public virtual Verify\_MongoDB\_Health\_PageExpectedValues Verify\_MongoDB\_Health\_PageExpectedValues

{

get

{

if ((this.mVerify\_MongoDB\_Health\_PageExpectedValues == null))

{

this.mVerify\_MongoDB\_Health\_PageExpectedValues = new Verify\_MongoDB\_Health\_PageExpectedValues();

}

return this.mVerify\_MongoDB\_Health\_PageExpectedValues;

}

}

private Verify\_MongoDB\_Health\_PageExpectedValues mVerify\_MongoDB\_Health\_PageExpectedValues;

/// <summary>

/// Verify\_Scanner\_Health\_Page - Use 'Verify\_Scanner\_Health\_PageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Scanner\_Health\_Page()

{

#region Variable Declarations

HtmlCell uIRabbitVM03EF8075Cell = this.UIScannerHealthOSTOperWindow.UIScannerHealthOSTOperDocument.UIJqGridTable.UIRabbitVM03EF8075Cell;

HtmlDiv uIScannerHealthPane = this.UIScannerHealthOSTOperWindow.UIScannerHealthOSTOperDocument.UIManageBCrumbPane.UIScannerHealthPane;

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

#endregion

//Assert.AreEqual(UIHeaderTooltip.Title, "To check the health of scanner.");

//// Verify that the 'InnerText' property of 'rabbit@VM-03EF-8075' cell equals 'rabbit@VM-03EF-8075'

//Assert.AreEqual(this.Verify\_Scanner\_Health\_PageExpectedValues.UIRabbitVM03EF8075CellInnerText, uIRabbitVM03EF8075Cell.InnerText);

// Verify that the 'TagName' property of 'Scanner Health' pane equals 'Scanner Health'

//Assert.AreEqual(this.Verify\_Scanner\_Health\_PageExpectedValues.UIScannerHealthPaneTagName, uIScannerHealthPane.InnerText.Trim(), true);

String[] DataTableArray = Flow\_GetTableData();

int ArraySize = DataTableArray.Length;

String[] StringSeperators = new string[] { "^#$@~" };

String[] NodeNames = new string[] { "rabbit@VM-03EF-8075", "rabbit@VM-0F28-9FED", "rabbit@VM-859F-4B3F", "rabbit@VM-B743-D8AD"};

for(int i=1; i<ArraySize; i++)

{

string[] TableData = DataTableArray[i].Split(StringSeperators, StringSplitOptions.None);

TableData = TableData.Where(x => !string.IsNullOrEmpty(x)).ToArray();

Assert.AreEqual(TableData[0], NodeNames[i - 1]);

Console.WriteLine("Comparing " + TableData[0] + " and " + NodeNames[i - 1]);

if ((TableData[1] == "Service running") || (TableData[1] == "RabbitMQ down"))

{

Console.WriteLine(NodeNames[i - 1] + " is in Running state");

//check for Description value

Assert.AreEqual(TableData[2], "RabbitMQ Management");

Console.WriteLine("Description of the Node " + NodeNames[i - 1] + " is " + TableData[2]);

//Check for the path of the Node

Assert.AreEqual(TableData[3], "/");

Console.WriteLine("Path of the Node " + NodeNames[i - 1] + " is " + TableData[3]);

//check for the Port of the Node

Assert.AreEqual(TableData[4], "15672");

Console.WriteLine("Port of the Node " + NodeNames[i - 1] + " is " + TableData[4]);

}

else

{

Assert.AreEqual(TableData[1],"VM Down");

Console.WriteLine(NodeNames[i - 1] + " is not Running");

Assert.AreEqual(TableData[2], "");

Assert.AreEqual(TableData[3], "");

Assert.AreEqual(TableData[4], "");

}

}

}

public void Verify\_MongoDB\_Health()

{

#region Variable Declarations

HtmlCustom UIHeaderTooltip = this.UINTFSPermissionsOSTOpWindow.UINTFSPermissionsOSTOpDocument.UIHeaderTooltip;

HtmlTable UIMongoDBHealthPage = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.UIJqGridTable;

#endregion

//Assert.AreEqual(UIHeaderTooltip.Title, "To check the health of scanner.");

//// Verify that the 'InnerText' property of 'rabbit@VM-03EF-8075' cell equals 'rabbit@VM-03EF-8075'

//Assert.AreEqual(this.Verify\_Scanner\_Health\_PageExpectedValues.UIRabbitVM03EF8075CellInnerText, uIRabbitVM03EF8075Cell.InnerText);

// Verify that the 'TagName' property of 'Scanner Health' pane equals 'Scanner Health'

//Assert.AreEqual(this.Verify\_Scanner\_Health\_PageExpectedValues.UIScannerHealthPaneTagName, uIScannerHealthPane.InnerText.Trim(), true);

String[] DataTableArray = Flow\_GetTableData\_Second();

int ArraySize = DataTableArray.Length;

String[] StringSeperators = new string[] { "^#$@~" };

String[] NodeNames = new string[] { "vm-1b98-f53f" };

for (int i = 1; i < ArraySize; i++)

{

string[] TableData = DataTableArray[i].Split(StringSeperators, StringSplitOptions.None);

TableData = TableData.Where(x => !string.IsNullOrEmpty(x)).ToArray();

//Checking the Host Names

Assert.AreEqual(TableData[0], NodeNames[i - 1]);

Console.WriteLine("Comparing " + TableData[0] + " and " + NodeNames[i - 1]);

//checking the Port details

Assert.AreEqual(TableData[1], "32017");

Console.WriteLine("Port of the Node " + NodeNames[i - 1] + " is " + TableData[1]);

//Checking the connection status

Assert.AreEqual(TableData[2], "Connected");

Console.WriteLine(NodeNames[i - 1] + " is in " + TableData[2] + " State");

}

}

public virtual Verify\_Scanner\_Health\_PageExpectedValues Verify\_Scanner\_Health\_PageExpectedValues

{

get

{

if ((this.mVerify\_Scanner\_Health\_PageExpectedValues == null))

{

this.mVerify\_Scanner\_Health\_PageExpectedValues = new Verify\_Scanner\_Health\_PageExpectedValues();

}

return this.mVerify\_Scanner\_Health\_PageExpectedValues;

}

}

private Verify\_Scanner\_Health\_PageExpectedValues mVerify\_Scanner\_Health\_PageExpectedValues;

/// <summary>

/// Verify\_Owners\_Info\_page - Use 'Verify\_Owners\_Info\_pageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Owners\_Info\_page()

{

#region Variable Declarations

HtmlDiv uIOwnersForUNCPathPane = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIOwnersForUNCPathPane;

HtmlCustom uIDisplaysOwnerinformaCustom = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIDisplaysOwnerinformaCustom;

HtmlLabel uIUNCPathLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIUNCPathLabel;

HtmlEdit uIUncPathInputBoxEdit = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIUncPathInputBoxEdit;

HtmlButton uISearchButton = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UISearchButton;

HtmlLabel uISelectUNCfiletoproceLabel = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UISearchFormCustom.UISelectUNCfiletoproceLabel;

HtmlFileInput uIFileUploadControlFileInput = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIFileUploadControlFileInput;

HtmlButton uIUploadButton = this.UIOwnersForUNCPathOSTOWindow.UIOwnersForUNCPathOSTODocument.UIUploadButton;

#endregion

// Verify that the 'InnerText' property of 'Owners For UNC Path' pane starts with 'Owners For UNC Path'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UIOwnersForUNCPathPaneInnerText, uIOwnersForUNCPathPane.InnerText.Trim(), true);

// Verify that the 'Title' property of 'Displays Owner information for a given UNC Path' custom control equals 'Displays Owner information for a given UNC Path'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UIDisplaysOwnerinformaCustomTitle.Trim(), uIDisplaysOwnerinformaCustom.Title);

// Verify that the 'InnerText' property of 'UNC Path:' label equals 'UNC Path:'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UIUNCPathLabelInnerText, uIUNCPathLabel.InnerText.Trim());

// Verify that the 'Id' property of 'uncPathInputBox' text box equals 'uncPathInputBox'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UIUncPathInputBoxEditId, uIUncPathInputBoxEdit.Id);

// Verify that the 'Id' property of 'Search' button equals 'searchButton'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UISearchButtonId, uISearchButton.Id);

// Verify that the 'InnerText' property of 'Select UNC file to process:' label starts with 'Select UNC file to process:'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UISelectUNCfiletoproceLabelInnerText, uISelectUNCfiletoproceLabel.InnerText.Trim());

// Verify that the 'Id' property of 'fileUploadControl' file input equals 'fileUpload'

Assert.AreEqual(this.Verify\_Owners\_Info\_pageExpectedValues.UIFileUploadControlFileInputId, uIFileUploadControlFileInput.Id);

// Verify that the 'InnerText' property of 'Upload' button contains 'Upload'

StringAssert.Contains(uIUploadButton.InnerText, this.Verify\_Owners\_Info\_pageExpectedValues.UIUploadButtonInnerText);

}

public virtual Verify\_Owners\_Info\_pageExpectedValues Verify\_Owners\_Info\_pageExpectedValues

{

get

{

if ((this.mVerify\_Owners\_Info\_pageExpectedValues == null))

{

this.mVerify\_Owners\_Info\_pageExpectedValues = new Verify\_Owners\_Info\_pageExpectedValues();

}

return this.mVerify\_Owners\_Info\_pageExpectedValues;

}

}

private Verify\_Owners\_Info\_pageExpectedValues mVerify\_Owners\_Info\_pageExpectedValues;

public void Verify\_DLP\_Data\_Protection\_Report\_Page()

{

#region Variable Declarations

HtmlDiv UIDLPDataProtectionRepPane = this.UIDLPDataProtectionReportWindow.UIDLPDataProtectionReportDocument.UIDLPDataProtectionRepPane;

HtmlCustom UIHeaderTooltip = this.UIDLPDataProtectionReportWindow.UIDLPDataProtectionReportDocument.UIHeaderTooltip;

HtmlFileInput UIChoosefileordraghereFileInput = this.UIDLPDataProtectionReportWindow.UIDLPDataProtectionReportDocument.UIChoosefileordraghereFileInput;

HtmlLabel UISelectafileLabel = this.UIDLPDataProtectionReportWindow.UIDLPDataProtectionReportDocument.UISearchDivPane.UISelectafileLabel;

HtmlButton UIUploadandgeneraterepButton = this.UIDLPDataProtectionReportWindow.UIDLPDataProtectionReportDocument.UIUploadandgeneraterepButton;

#endregion

Assert.AreEqual(UIDLPDataProtectionRepPane.InnerText.Trim(), "DLP/Data Protection Report");

Assert.AreEqual(UIHeaderTooltip.Title.Trim(), "Displays Owner information for a given UNC Path or list of UNC Paths provided in a file");

Assert.AreEqual(UIChoosefileordraghereFileInput.Id, "csvFileUpload");

Assert.AreEqual(UISelectafileLabel.InnerText.Trim(), "Select a file:");

Assert.AreEqual(UIUploadandgeneraterepButton.InnerText.Trim(), "Upload and generate report");

}

/// <summary>

/// Verify\_User\_Roles\_Search\_Results - Use 'Verify\_User\_Roles\_Search\_ResultsExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_User\_Roles\_Search\_Results()

{

#region Variable Declarations

HtmlSpan uIItemDropdown = this.UIUserRolesOSTOperatioWindow.UIUserRolesOSTOperatioDocument.UIUiaccordionteamAccorCustom.UIItemDropdown;

#endregion

uIItemDropdown.WaitForControlExist(60000);

// Verify that the 'InnerText' property of 'UIItemDropdown' pane equals 'Teams'

Assert.AreEqual(this.Verify\_User\_Roles\_Search\_ResultsExpectedValues.UIItemDropdownInnerText, uIItemDropdown.InnerText, "Most likely the results did not load.");

}

public virtual Verify\_User\_Roles\_Search\_ResultsExpectedValues Verify\_User\_Roles\_Search\_ResultsExpectedValues

{

get

{

if ((this.mVerify\_User\_Roles\_Search\_ResultsExpectedValues == null))

{

this.mVerify\_User\_Roles\_Search\_ResultsExpectedValues = new Verify\_User\_Roles\_Search\_ResultsExpectedValues();

}

return this.mVerify\_User\_Roles\_Search\_ResultsExpectedValues;

}

}

private Verify\_User\_Roles\_Search\_ResultsExpectedValues mVerify\_User\_Roles\_Search\_ResultsExpectedValues;

/// <summary>

/// Verify\_Add\_Share\_Exception\_popup - Use 'Verify\_Add\_Share\_Exception\_popupExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Add\_Share\_Exception\_popup()

{

#region Variable Declarations

HtmlSpan uIExceptionDetailsAddPane = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIExceptionDetailsAddPane;

HtmlEdit uIVirtualDeviceAutoComEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIVirtualDeviceAutoComEdit;

HtmlEdit uIShareAutoCompEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIShareAutoCompEdit;

HtmlEdit uIItemEdit = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit;

HtmlEdit uIItemEdit1 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit1;

HtmlEdit uIItemEdit2 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit2;

HtmlTextArea uIItemEdit3 = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIContactsListPane.UIItemEdit3;

HtmlButton uISaveButton = this.UIChargebackReportsOSTWindow.UIChargebackExceptionsDocument1.UIAddEditContentPane.UISaveButton;

#endregion

// Verify that the 'InnerText' property of 'Exception Details - Add' pane equals 'Exception Details - Add'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIExceptionDetailsAddPaneInnerText, uIExceptionDetailsAddPane.InnerText);

// Verify that the 'Id' property of 'virtualDeviceAutoComp' text box equals 'virtualDeviceAutoComp'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIVirtualDeviceAutoComEditId, uIVirtualDeviceAutoComEdit.Id);

// Verify that the 'Id' property of 'shareAutoComp' text box equals 'shareAutoComp'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIShareAutoCompEditId, uIShareAutoCompEdit.Id);

// Verify that the 'TagName' property of text box equals 'INPUT'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIItemEditTagName, uIItemEdit.TagName);

// Verify that the 'TagName' property of text box equals 'INPUT'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIItemEdit1TagName, uIItemEdit1.TagName);

// Verify that the 'TagName' property of text box equals 'INPUT'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIItemEdit2TagName, uIItemEdit2.TagName);

// Verify that the 'TagName' property of text box equals 'TEXTAREA'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UIItemEdit3TagName, uIItemEdit3.TagName);

// Verify that the 'InnerText' property of 'Save' button equals 'Save'

Assert.AreEqual(this.Verify\_Add\_Share\_Exception\_popupExpectedValues.UISaveButtonInnerText, uISaveButton.InnerText.Trim());

}

public virtual Verify\_Add\_Share\_Exception\_popupExpectedValues Verify\_Add\_Share\_Exception\_popupExpectedValues

{

get

{

if ((this.mVerify\_Add\_Share\_Exception\_popupExpectedValues == null))

{

this.mVerify\_Add\_Share\_Exception\_popupExpectedValues = new Verify\_Add\_Share\_Exception\_popupExpectedValues();

}

return this.mVerify\_Add\_Share\_Exception\_popupExpectedValues;

}

}

private Verify\_Add\_Share\_Exception\_popupExpectedValues mVerify\_Add\_Share\_Exception\_popupExpectedValues;

/// <summary>

/// Verify\_Edit\_Recurring\_Schedule\_page - Use 'Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues' to pass parameters into this method.

/// </summary>

public void Verify\_Edit\_Recurring\_Schedule\_page()

{

#region Variable Declarations

HtmlSpan uIEditRecurringSchedulPane = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIEditRecurringSchedulPane;

HtmlComboBox uIOutputTypeSlComboBox = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIOutputTypeSlComboBox;

HtmlEdit uIReportNameTxtSchedulEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIReportNameTxtSchedulEdit;

HtmlCheckBox uITimeStampChkCheckBox = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UITimeStampChkCheckBox;

HtmlEdit uIReportShareTxtEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIReportShareTxtEdit;

HtmlEdit uIOccuranceTxtEdit = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UIOccuranceTxtEdit;

HtmlButton uISaveButton = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UISaveButton;

HtmlButton uICloseButton = this.UIReportRequestsforEURWindow1.UIReportRequestsforEURDocument.UICloseButton;

#endregion

// Verify that the 'InnerText' property of 'Edit Recurring Schedule' pane equals 'Add Recurring Schedule'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UIEditRecurringSchedulPaneInnerText, uIEditRecurringSchedulPane.InnerText);

// Verify that the 'Id' property of 'outputTypeSl' combo box equals 'outputTypeSl'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UIOutputTypeSlComboBoxId, uIOutputTypeSlComboBox.Id);

// Verify that the 'Id' property of 'reportNameTxtScheduler' text box equals 'reportNameTxtScheduler'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UIReportNameTxtSchedulEditId, uIReportNameTxtSchedulEdit.Id);

// Verify that the 'Id' property of 'timeStampChk' check box equals 'timeStampChk'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UITimeStampChkCheckBoxId, uITimeStampChkCheckBox.Id);

// Verify that the 'Id' property of 'reportShareTxt' text box equals 'reportShareTxt'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UIReportShareTxtEditId, uIReportShareTxtEdit.Id);

// Verify that the 'Id' property of 'occuranceTxt' text box equals 'occuranceTxt'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UIOccuranceTxtEditId, uIOccuranceTxtEdit.Id);

// Verify that the 'InnerText' property of 'Save' button equals 'Save'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UISaveButtonInnerText, uISaveButton.InnerText.Trim());

// Verify that the 'InnerText' property of 'Close' button equals 'Close'

Assert.AreEqual(this.Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues.UICloseButtonInnerText, uICloseButton.InnerText.Trim());

}

public virtual Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues

{

get

{

if ((this.mVerify\_Edit\_Recurring\_Schedule\_pageExpectedValues == null))

{

this.mVerify\_Edit\_Recurring\_Schedule\_pageExpectedValues = new Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues();

}

return this.mVerify\_Edit\_Recurring\_Schedule\_pageExpectedValues;

}

}

private Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues mVerify\_Edit\_Recurring\_Schedule\_pageExpectedValues;

public void Kill\_all\_iexplore\_and\_chromedriver()

{

foreach (var ie in Process.GetProcessesByName("iexplore"))

{

try

{

ie.Kill();

}

catch { }

}

foreach (var ie in Process.GetProcessesByName("chromedriver"))

{

try

{

ie.Kill();

}

catch { }

}

}

public void Restart\_Volume\_Info\_Job(string Team)

{

HtmlComboBox uIAvailableTeamsDdlComboBox = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIAvailableTeamsDdlComboBox;

HtmlButton uIRestartButton = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIRestartButton;

HtmlDiv uIVolumeInfoJobRestartPane = this.UIOverviewOSTOperationWindow.UIOverviewOSTOperationDocument.UIVolumeInfoJobRestartPane;

Mouse.Click(uIAvailableTeamsDdlComboBox);

Keyboard.SendKeys(Team + "{ENTER}");

Mouse.Click(uIRestartButton);

uIVolumeInfoJobRestartPane.WaitForControlPropertyEqual("InnerText", "Volume Info Job Restarted", 60000);

}

public virtual ObjectsExpectedValues ObjectsExpectedValues

{

get

{

if ((this.mObjectsExpectedValues == null))

{

this.mObjectsExpectedValues = new ObjectsExpectedValues();

}

return this.mObjectsExpectedValues;

}

}

private ObjectsExpectedValues mObjectsExpectedValues;

/// <summary>

/// Click\_Username\_link\_on\_Home\_Global\_Search\_results

/// </summary>

public void Click\_Username\_link\_on\_Home\_Global\_Search\_results()

{

#region Variable Declarations

HtmlHyperlink uIKrzysztofCyganHyperlink = this.UIOSTHomeOSTOperationsWindow1.UIOSTHomeOSTOperationsDocument.UIContentSoeIdCustom.UIKrzysztofCyganHyperlink;

HtmlHyperlink uIGloriaRosarioHyperlink = this.UIOSTHomeOSTOperationsWindow3.UIOSTHomeOSTOperationsDocument.UIContentSoeIdCustom.UIGloriaRosarioHyperlink;

#endregion

// Click 'Krzysztof Cygan' link

Mouse.Click(uIGloriaRosarioHyperlink);

}

public void Verify\_Column\_Headers(string ColumnsDetailsJSON)

{

HtmlTable header = this.UIOSTHomeOSTOperationsWindow.UIShareInformationOSTODocument.JqGridTableColumnHeaders;

//JSON Column Details generator

//http://www.json-generator.com/

//[{

//Column\_Number: '{{index()}}',

//Column\_Name: 'Details'

//},

//{

//Column\_Number: '{{index()}}',

//Column\_Name: 'NTFS'

//}]

//List all Column Headers names

Console.WriteLine("Header cell count: " + header.CellCount);

//Console.WriteLine("Test: " + header.GetCell(0, 1).InnerText);

Console.WriteLine();

Console.WriteLine("Column Names list:");

if (BrowserWindow.CurrentBrowser == "IE")

{

String[] HeaderArray = new String[header.CellCount];

for (int i = 0; i < header.CellCount; i++)

{

if (header.GetCell(0,i).Height > 0)

{

HeaderArray[i] = header.GetCell(0, i).InnerText.Trim();

Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

}

}

HeaderArray = HeaderArray.Where(str => str != null).ToArray();

}

else if (BrowserWindow.CurrentBrowser == "Chrome")

{

string[] stringSeparators = new string[] { "\r\n" };

string[] header1 = header.GetCell(0, 1).InnerText.Trim().Split(stringSeparators, StringSplitOptions.None);

String[] HeaderArray = new String[header1.Count()];

for (int i = 0; i < header1.Count(); i++)

{

if (header.GetCell(0, i).Height > 0)

{

HeaderArray[i] = header1[i];

Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

}

}

}

Console.WriteLine();

Console.WriteLine("Begin Column Name verification:");

dynamic DynamicResult = JsonConvert.DeserializeObject(ColumnsDetailsJSON);

//Console.WriteLine("JSON: " + DynamicResult);

if (BrowserWindow.CurrentBrowser == "IE")

{

String[] HeaderArray = new String[header.CellCount];

for (int i = 0; i < header.CellCount; i++)

{

if (header.GetCell(0, i).Height > 0)

{

HeaderArray[i] = header.GetCell(0, i).InnerText.Trim();

Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

}

}

HeaderArray = HeaderArray.Where(str => str != null).ToArray();

for (int i = 0; i < HeaderArray.Count(); i++)

{

Assert.AreEqual(DynamicResult[i].Column\_Name.ToString().Trim(), HeaderArray[i].ToString().Trim(), "Expected column name: " + DynamicResult[i].Column\_Name.ToString().Trim() + " does not match result: " + HeaderArray[i].ToString().Trim());

}

}

else if (BrowserWindow.CurrentBrowser == "Chrome")

{

string[] stringSeparators = new string[] { "\r\n" };

string[] header1 = header.GetCell(0, 1).InnerText.Trim().Split(stringSeparators, StringSplitOptions.None);

String[] HeaderArray = new String[header1.Count()];

for (int i = 0; i < header1.Count(); i++)

{

if (header.GetCell(0, i).Height > 0)

{

HeaderArray[i] = header1[i];

Console.WriteLine("Header " + i + ": " + HeaderArray[i].ToString());

}

Assert.AreEqual(DynamicResult[i].Column\_Name.ToString().Trim(), HeaderArray[i].ToString().Trim(), "Expected column name: " + DynamicResult[i].Column\_Name.ToString().Trim() + " does not match result: " + HeaderArray[i].ToString().Trim());

}

}

else

Assert.AreEqual("x", "y", "Neither Chrome nor IE code executed?");

}

public void Verify\_IDrive\_page()

{

#region Variable Declarations

HtmlDiv uIIDriveReportsPane = this.UIIDriveWindow.UIIDrivePage.UIIDriveReportsPane;

HtmlDiv uIIDrivePane = this.UIIDriveWindow.UIIDrivePage.UIReportWellPane.UIIDrivePane;

HtmlCustom uIFailureForChrome = this.UIIDriveWindow.UIIDriveReportsOSTOperDocument.UIHighcharts0Pane.uIFailureChrome;

HtmlCustom UIFailureForIE = this.UIIDriveWindow.UIIDriveReportsOSTOperDocument.UIHighcharts0Pane.uIFailureIE;

HtmlDiv uIIDriveErrorPane = this.UIIDriveWindow.UIIDrivePage.UIErrorWellPane.UIIDriveErrorPane;

HtmlCustom UIErrorBreakdownChrome = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UIErrorBreakdownChrome;

HtmlCustom UITotalErrorsChrome = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UITotalErrorsChrome;

HtmlCustom UISoeidIsNullChrome = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UISoeidIsNullChrome;

HtmlCustom UIErrorBreakdownIE = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UIErrorBreakdownIE;

HtmlCustom UITotalErrorsIE = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UITotalErrorsIE;

HtmlCustom UISoeidIsNullIE = this.UIIDriveWindow.UIIDrivePage.UIErrorBreakdownPane.UISoeidIsNullIE;

HtmlButton UIBackButton = this.UIIDriveWindow.UIIDriveReportsOSTOperDocument.UIErrorWellPane.UIBackButton;

HtmlCustom uIFailure\_Area = this.UIIDriveWindow.UIIDriveReportsOSTOperDocument1.UIHighcharts2Pane.UIItemCustom;

#endregion

string Browser\_Type = System.Environment.GetEnvironmentVariable("Browser\_Type");

uIIDriveReportsPane.WaitForControlExist(15000);

uIIDrivePane.WaitForControlExist(15000);

if (Browser\_Type == "IE")

{

uIFailure\_Area.WaitForControlReady(15000);

Mouse.Click(uIFailure\_Area, new System.Drawing.Point(66, 103));

}

if (Browser\_Type == "Chrome")

{

uIFailureForChrome.WaitForControlExist(15000);

Mouse.Click(uIFailureForChrome);

}

uIIDriveErrorPane.WaitForControlExist(15000);

if (Browser\_Type == "IE")

{

UIErrorBreakdownIE.WaitForControlExist(15000);

UITotalErrorsIE.WaitForControlExist(15000);

}

if (Browser\_Type == "Chrome")

{

UIErrorBreakdownChrome.WaitForControlExist(15000);

UITotalErrorsChrome.WaitForControlExist(15000);

}

UIBackButton.WaitForControlExist(15000);

//uISoeidIsNull.WaitForControlExist(15000);

//Assert.IsTrue(uISoeidIsNull.Exists, "");

if (Browser\_Type == "IE")

{

Assert.IsTrue(UIErrorBreakdownIE.Exists, "Error Breakdown is not shown");

Console.WriteLine("Error Breakdown was found.");

Assert.IsTrue(UITotalErrorsIE.Exists, "Total Errors is not shown");

Console.WriteLine("Total Errors was found.");

}

if (Browser\_Type == "Chrome")

{

Assert.IsTrue(UIErrorBreakdownChrome.Exists, "Error Breakdown is not shown");

Console.WriteLine("Error Breakdown was found.");

Assert.IsTrue(UITotalErrorsChrome.Exists, "Total Errors is not shown");

Console.WriteLine("Total Errors was found.");

}

Mouse.Click(UIBackButton);

uIIDriveReportsPane.WaitForControlExist(15000);

uIIDrivePane.WaitForControlExist(15000);

Assert.IsTrue(uIIDriveReportsPane.Exists, "uIIDriveReportsPane is not shown");

Assert.IsTrue(uIIDrivePane.Exists, "uIIDrivePane is not shown");

Console.WriteLine("Back button worked as expected.");

}

public void Test()

{

string Date = "6/20/2016 12:34:58 PM";

DateTime DateEntered;

if (DateTime.TryParseExact(Date, "M/d/yyyy h:mm:ss tt", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateEntered));

else if (DateTime.TryParseExact(Date, "dd/MM/yyyy HH:mm:ss", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateEntered));

else

{

Console.WriteLine("Unable to parse '{0}'.", Date.ToString());

}

}

}

/// <summary>

/// Parameters to be passed into 'Verify\_Shares\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Shares\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Shares' link equals 'Shares'

/// </summary>

public string UISharesHyperlinkInnerText = "Shares";

/// <summary>

/// Verify that the 'InnerText' property of 'Dfs Shares' link equals 'Dfs Shares'

/// </summary>

public string UIDfsSharesHyperlinkInnerText = "Dfs Shares";

/// <summary>

/// Verify that the 'InnerText' property of 'Ntfs Permissions' link equals 'Ntfs Permissions'

/// </summary>

public string UINtfsPermissionsHyperlinkInnerText = "Ntfs Permissions";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Compliance' link equals 'Share Compliance'

/// </summary>

public string UIShareComplianceHyperlinkInnerText = "Share Compliance";

/// <summary>

/// Verify that the 'InnerText' property of 'Owners' link equals 'Owners'

/// </summary>

public string UIOwnersHyperlinkInnerText = "Owners";

/// <summary>

/// Verify that the 'InnerText' property of 'Global Compliance' link equals 'Global Compliance'

/// </summary>

public string UIGlobalComplianceHyperlinkInnerText = "Global Compliance";

/// <summary>

/// Verify that the 'InnerText' property of 'Global Account Search' link equals 'Global Account Search'

/// </summary>

public string UIGlobalAccountSearchHyperlinkInnerText = "Global Account Entitlement Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Printers\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Printers\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Print Queue' link equals 'Print Queue'

/// </summary>

public string UIPrintQueueHyperlinkInnerText = "Print Queue";

/// <summary>

/// Verify that the 'InnerText' property of 'Print Queue History' link equals 'Print Queue History'

/// </summary>

public string UIPrintQueueHistoryHyperlinkInnerText = "Print Queue History";

/// <summary>

/// Verify that the 'InnerText' property of 'Print Queue Statistics' link equals 'Print Queue Statistics'

/// </summary>

public string UIPrintQueueStatisticsHyperlinkInnerText = "Print Queue Statistics";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Report\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Report\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Exception Reports' link equals 'Exception Reports'

/// </summary>

public string UIExceptionReportsHyperlinkInnerText = "Exception Reports";

/// <summary>

/// Verify that the 'InnerText' property of 'NAS Global Coverage' link equals 'NAS Global Coverage'

/// </summary>

public string UINASGlobalCoverageHyperlinkInnerText = "NAS Global Coverage";

/// <summary>

/// Verify that the 'InnerText' property of 'Global IDrive' link equals 'Global IDrive'

/// </summary>

public string UIGlobalIDriveHyperlinkInnerText = "Global IDrive";

/// <summary>

/// Verify that the 'InnerText' property of 'Report Requests' link equals 'Report Requests'

/// </summary>

public string UIReportRequestsHyperlinkInnerText = "Report Requests";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Support\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Support\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Monitor Health' link equals 'Monitor Health'

/// </summary>

public string UIMonitorHealthHyperlinkInnerText = "Monitor Health";

/// <summary>

/// Verify that the 'InnerText' property of 'Manage Monitor Account' link equals 'Manage Monitor Account'

/// </summary>

public string UIManageMonitorAccountHyperlinkInnerText = "Manage Monitor Account";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Chargeback\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Chargeback\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Chargeback Reports' link equals 'Chargeback Reports'

/// </summary>

public string UIChargebackReportsHyperlinkInnerText = "Chargeback Reports";

/// <summary>

/// Verify that the 'InnerText' property of 'Chargeback Exceptions' link equals 'Chargeback Exceptions'

/// </summary>

public string UIChargebackExceptionsHyperlinkInnerText = "Chargeback Exceptions";

/// <summary>

/// Verify that the 'InnerText' property of 'Chargeback Cost Code' link equals 'Chargeback Cost Code'

/// </summary>

public string UIChargebackCostCodeHyperlinkInnerText = "Chargeback Cost Code";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Compliance\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Compliance\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Anti Virus' link equals 'Anti Virus'

/// </summary>

public string UIAntiVirusHyperlinkInnerText = "Anti Virus";

/// <summary>

/// Verify that the 'InnerText' property of 'Security Checks' link equals 'Security Checks'

/// </summary>

public string UISecurityChecksHyperlinkInnerText = "Security Checks";

/// <summary>

/// Verify that the 'InnerText' property of 'Local Users' link equals 'Local Users'

/// </summary>

public string UILocalUsersHyperlinkInnerText = "Local Users";

/// <summary>

/// Verify that the 'InnerText' property of 'Local Groups' link equals 'Local Groups'

/// </summary>

public string UILocalGroupsHyperlinkInnerText = "Local Groups";

/// <summary>

/// Verify that the 'InnerText' property of 'User Rights' link equals 'User Rights'

/// </summary>

public string UIUserRightsHyperlinkInnerText = "User Rights";

/// <summary>

/// Verify that the 'InnerText' property of 'Orphans' link equals 'Orphans'

/// </summary>

public string UIOrphansHyperlinkInnerText = "Orphans";

/// <summary>

/// Verify that the 'InnerText' property of 'Server Scheduled Tasks' link equals 'Server Scheduled Tasks'

/// </summary>

public string UIServerScheduledTasksHyperlinkInnerText = "Server Scheduled Tasks";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Exclusion' link equals 'Share Exclusion'

/// </summary>

public string UIShareExclusionHyperlinkInnerText = "Share Exclusion";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_DataStats\_Dropdown'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_DataStats\_DropdownExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Overview' link equals 'Overview'

/// </summary>

public string UIOverviewHyperlinkInnerText = "Overview";

/// <summary>

/// Verify that the 'InnerText' property of 'Find' link equals 'Find'

/// </summary>

public string UIFindHyperlinkInnerText = "Find";

/// <summary>

/// Verify that the 'InnerText' property of 'File Usage (DLP)' link equals 'File Usage (DLP)'

/// </summary>

public string UIFileUsageDLPHyperlinkInnerText = "File Usage (DLP)";

/// <summary>

/// Verify that the 'InnerText' property of 'PST Report' link equals 'PST Report'

/// </summary>

public string UIPSTReportHyperlinkInnerText = "PST Report";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_MongoDB\_Health\_Page'

/// </summary>

//[GeneratedCode("Coded UITest Builder", "12.0.31101.0")]

public class Verify\_MongoDB\_Health\_PageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Host Name :' cell equals 'Host Name :'

/// </summary>

public string UIHostNameCellInnerText = "Host Name :";

/// <summary>

/// Verify that the 'InnerText' property of 'Disconnected' cell equals 'Connected'

/// </summary>

public string UIDisconnectedCellInnerText = "Connected";

/// <summary>

/// Verify that the 'InnerText' property of 'vm-1b98-f53f' cell equals 'vm-1b98-f53f'

/// </summary>

public string UIVm1b98f53fCellInnerText = "vm-1b98-f53f";

/// <summary>

/// Verify that the 'InnerText' property of 'Port :' cell equals 'Port :'

/// </summary>

public string UIPortCellInnerText = "Port :";

/// <summary>

/// Verify that the 'InnerText' property of '32017' cell equals '32017'

/// </summary>

public string UIItem32017CellInnerText = "32017";

/// <summary>

/// Verify that the 'InnerText' property of 'Status :' cell equals 'Status :'

/// </summary>

public string UIStatusCellInnerText = "Status :";

/// <summary>

/// Verify that the 'InnerText' property of 'Disconnected' cell equals 'Connected'

/// </summary>

public string UIDisconnectedCellInnerText1 = "Connected";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Share\_Compliance'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Share\_ComplianceExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Share Compliance' pane starts with 'Share Compliance'

/// </summary>

public string UIShareCompliancePaneInnerText = "Share Compliance";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Filter:' label starts with 'Filter:'

/// </summary>

public string UIFilterLabelInnerText = "Filter:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'InnerText' property of combo box equals 'AllCompliantNon-Compliant'

/// </summary>

public string UIItemComboBoxInnerText = "Compliant";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Owners'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_OwnersExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Owners' pane equals 'Owners'

/// </summary>

public string UIOwnersPaneInnerText = "Owners";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Name:' label starts with 'Share Name:'

/// </summary>

public string UIShareNameLabelInnerText = "Share Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Owner Name:' label starts with 'Owner Name:'

/// </summary>

public string UIOwnerNameLabelInnerText = "Owner Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'SOE Id:' label starts with 'SOE Id:'

/// </summary>

public string UISOEIdLabelInnerText = "SOE Id:";

/// <summary>

/// Verify that the 'InnerText' property of 'Folder Path:' label starts with 'Folder Path:'

/// </summary>

public string UIFolderPathLabelInnerText = "Folder Path:";

/// <summary>

/// Verify that the 'InnerText' property of 'This report's complexity makes it slow t' cell equals 'This report's complexity makes it slow to retrieve. Please be patient after submitting a query.'

/// </summary>

public string UIThisreportscomplexitCellInnerText = "This report\'s complexity makes it slow to retrieve. Please be patient after submi" +

"tting a query.";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

/// </summary>

public string UIItemEditClass = "ui-autocomplete-input";

/// <summary>

/// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

/// </summary>

public string UIItemEdit1TagName = "INPUT";

/// <summary>

/// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

/// </summary>

public string UIItemEdit2TagName = "INPUT";

/// <summary>

/// Verify that the 'ClassName' property of text box equals 'HtmlTextBox'

/// </summary>

public string UIItemEdit3TagName = "INPUT";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Shares'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_SharesExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Share Information' pane equals 'Share Information'

/// </summary>

public string UIShareInformationPaneInnerText = "Share Information";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Partition:' label equals 'Partition:'

/// </summary>

public string UIPartitionLabelInnerText = "Partition:";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Name:' label equals 'Share Name:'

/// </summary>

public string UIShareNameLabelInnerText = "Share Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label equals 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Type:' label equals 'Share Type:'

/// </summary>

public string UIShareTypeLabelInnerText = "Share Type:";

/// <summary>

/// Verify that the 'InnerText' property of 'Include Archived:' label equals 'Include Archived: '

/// </summary>

public string UIIncludeArchivedLabelInnerText = "Include Archived:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

/// </summary>

public string UIItemEditClass = "ui-autocomplete-input";

/// <summary>

/// Verify that the 'TagName' property of text box equals 'INPUT'

/// </summary>

public string UIItemEditTagName = "INPUT";

/// <summary>

/// Verify that the 'TagInstance' property of text box equals '3'

/// </summary>

public int UIItemEditTagInstance = 3;

/// <summary>

/// Verify that the 'Id' property of 'shareNameTxt' text box equals 'shareNameTxt'

/// </summary>

public string UIShareNameTxtEditId = "shareNameTxt";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'TagName' property of combo box equals 'SELECT'

/// </summary>

public string UIItemComboBoxTagName = "SELECT";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'TagName' property of check box equals 'INPUT'

/// </summary>

public string UIItemCheckBoxTagName = "INPUT";

/// <summary>

/// Verify that the 'ControlType' property of check box equals 'CheckBox'

/// </summary>

public string UIItemCheckBoxControlType = "CheckBox";

/// <summary>

/// Verify that the 'TagInstance' property of check box equals '5'

/// </summary>

public int UIItemCheckBoxTagInstance = 5;

/// <summary>

/// Verify that the 'ControlType' property of 'Search' button equals 'Button'

/// </summary>

public string UISearchButtonControlType = "Button";

/// <summary>

/// Verify that the 'TagName' property of 'Search' button equals 'BUTTON'

/// </summary>

public string UISearchButtonTagName = "BUTTON";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Perform\_search\_on\_Shares\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Perform\_search\_on\_Shares\_pageParams

{

#region Fields

/// <summary>

/// Type 'nasspr904rv1cr' in 'hostName' text box

/// </summary>

//public string UIHostNameEditText = "nasspr904rv1cr";

/// <summary>

/// Type 'dat' in text box

/// </summary>

public string UIItemEditText = "dat";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_search\_results\_on\_Shares\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_search\_results\_on\_Shares\_pageExpectedValues

{

#region Fields

/// <summary>

/// Wait for 2 seconds for user delay between actions; Verify that the 'InnerText' property of 'gbox\_jqGrid' pane contains 'View 1 - '

/// </summary>

public string UIGbox\_jqGridPaneInnerText = "View 1 - ";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Shares\_search\_finished'

/// </summary>

[GeneratedCode("Coded UITest Builder", "11.0.60315.1")]

public class Verify\_Shares\_search\_finishedExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'Enabled' property of 'Processing...' label equals 'True'

/// </summary>

public bool UIProcessingLabelEnabled = true;

public bool UIProcessingLabelDisabled = false;

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Shares\_Table'

/// </summary>

[GeneratedCode("Coded UITest Builder", "11.0.60315.1")]

public class Verify\_Shares\_TableExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'Id' property of 'jqGrid' table equals 'jqGrid'

/// </summary>

public string UIJqGridTableId = "jqGrid";

#endregion

}

// /// <summary>

// /// Parameters to be passed into 'Filter\_Shares'

// /// </summary>

// [GeneratedCode("Coded UITest Builder", "11.0.60315.1")]

// public class Filter\_SharesParams

// {

// #region Fields

// /// <summary>

// /// Type 'autoaudit\_archives' in 'jqg1' text box

// /// </summary>

// public string UIJqg1EditText = "autoaudit\_archives";

// #endregion

//}

/// <summary>

/// Parameters to be passed into 'Filter\_Shares\_verify\_filter\_label'

/// </summary>

[GeneratedCode("Coded UITest Builder", "11.0.60315.1")]

public class Filter\_Shares\_verify\_filter\_labelExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Filter: [ShareName is equal to 'autoaudit\_archives...' label equals 'Filter: [ShareName is equal to 'autoaudit\_archives']'

/// </summary>

public string UIFilterShareNameisequLabelInnerText = "Filter: [ShareName is equal to \'autoaudit\_archives\']";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Perform\_search\_on\_Dfs\_Shares\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "11.0.60315.1")]

public class Perform\_search\_on\_Dfs\_Shares\_pageParams

{

#region Fields

/// <summary>

/// Type 'namdevapp3' in 'hostName' text box

/// </summary>

public string UIHostNameEditText = "namdevapp3";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Export\_to\_Excel'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Export\_to\_ExcelParams

{

#region Fields

/// <summary>

/// Type '{Enter}' in 'File name:' text box

/// </summary>

public string UIFilenameEditSendKeys = "{Enter}";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Chargeback\_Reports'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Chargeback\_ReportsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Chargeback Reports' pane equals 'Chargeback Reports'

/// </summary>

public string UIChargebackReportsPaneInnerText = "Chargeback Reports";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Volume:' label equals 'Volume:'

/// </summary>

public string UIVolumeLabelDisplayText = "Volume:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Share:' label equals 'Share:'

/// </summary>

public string UIShareLabelDisplayText = "Share:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostInputBox' text box equals 'hostInputBox'

/// </summary>

public string UIHostInputBoxEditId = "hostInputBox";

/// <summary>

/// Verify that the 'Id' property of 'volumeInputBox' text box equals 'volumeInputBox'

/// </summary>

public string UIVolumeInputBoxEditId = "volumeInputBox";

/// <summary>

/// Verify that the 'Id' property of 'shareInputBox' text box equals 'shareInputBox'

/// </summary>

public string UIShareInputBoxEditId = "shareInputBox";

/// <summary>

/// Verify that the 'Id' property of 'Generate Report' button equals 'generateReportButton'

/// </summary>

public string UIGenerateReportButtonId = "generateReportButton";

/// <summary>

/// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

/// </summary>

public string UIGenerateReportButtonDisplayText = "Generate Report";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Chargeback\_Exceptions'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Chargeback\_ExceptionsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Chargeback Exceptions' pane equals 'Chargeback Exceptions'

/// </summary>

public string UIChargebackExceptionsPaneDisplayText = "Chargeback Exceptions";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Volume:' label starts with 'Volume:'

/// </summary>

public string UIVolumeLabelDisplayText = "Volume:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Shares:' label starts with 'Shares:'

/// </summary>

public string UISharesLabelDisplayText = "Shares:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Deleted Exceptions?' label starts with 'Deleted Exceptions?'

/// </summary>

public string UIDeletedExceptionsLabelDisplayText = "Deleted Exceptions?";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'volumeNameBx' text box equals 'volumeNameBx'

/// </summary>

public string UIVolumeNameBxEditId = "volumeNameBx";

/// <summary>

/// Verify that the 'Id' property of 'shareNameBx' text box equals 'shareNameBx'

/// </summary>

public string UIShareNameBxEditId = "shareNameBx";

/// <summary>

/// Verify that the 'Id' property of 'isDeletedCkbx' check box equals 'isDeletedCkbx'

/// </summary>

public string UIIsDeletedCkbxCheckBoxId = "isDeletedCkbx";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

/// <summary>

/// Verify that the 'DisplayText' property of 'Add Exception' button equals 'Add Exception'

/// </summary>

public string UIAddExceptionButtonDisplayText = "Add Exception";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Chargeback\_Cost\_Code'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Chargeback\_Cost\_CodeExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Chargeback Cost Code Search' pane equals 'Chargeback Cost Code Search'

/// </summary>

public string UIChargebackCostCodeSePaneDisplayText = "Chargeback Cost Code Search";

/// <summary>

/// Verify that the 'DisplayText' property of 'Cost Code or SoeID:' label equals 'Cost Code or SoeID:'

/// </summary>

public string UICostCodeorSoeIDLabelDisplayText = "Cost Code or SoeID:";

/// <summary>

/// Verify that the 'Id' property of 'costCodeInputBox' text box equals 'costCodeInputBox'

/// </summary>

public string UICostCodeInputBoxEditId = "costCodeInputBox";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

/// <summary>

/// Verify that the 'DisplayText' property of 'authDisclaimerPlaceholder' pane starts with 'Please Note: When searching using an SOEID, the results displayed will be for the SOEID's cost code which may be shared by other users.'

/// </summary>

public string UIAuthDisclaimerPlacehPaneDisplayText = "Please Note: When searching using an SOEID, the results displayed will be for the" +

" SOEID\'s cost code which may be shared by other users.";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Exception\_Report'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Exception\_ReportExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Exception Report' pane equals 'Exception Report'

/// </summary>

public string UIExceptionReportPaneDisplayText = "Exception Report";

/// <summary>

/// Verify that the 'DisplayText' property of '\* Team:' label starts with '\* Team:'

/// </summary>

public string UITeamLabelDisplayText = "\* Team:";

/// <summary>

/// Verify that the 'Id' property of 'Teams' list box equals 'Teams'

/// </summary>

public string UITeamsListId = "Teams";

/// <summary>

/// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

/// </summary>

public string UIGenerateReportButtonDisplayText = "Generate Report";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_NAS\_Global\_Coverage'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_NAS\_Global\_CoverageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'NAS Global Coverage' pane equals 'NAS Global Coverage'

/// </summary>

public string UINASGlobalCoveragePaneDisplayText = "NAS Global Coverage";

/// <summary>

/// Verify that the 'Id' property of 'Submit Query' button equals 'excelExportSubmit'

/// </summary>

public string UISubmitQueryButtonId = "excelExportSubmit";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Global\_IDrive'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Global\_IDriveExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Global IDrive' pane equals 'Global IDrive'

/// </summary>

public string UIGlobalIDrivePaneDisplayText = "Global IDrive";

/// <summary>

/// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

/// </summary>

public string UIGenerateReportButtonDisplayText = "Generate Report";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Report\_Requests'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Report\_RequestsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Report Requests for EUR\kc90246' pane starts with 'Report Requests for'

/// </summary>

public string UIReportRequestsforEURPaneDisplayText = "Report Requests";

/// <summary>

/// Verify that the 'DisplayText' property of 'Report Name:' label starts with 'Report Name:'

/// </summary>

public string UIReportNameLabelDisplayText = "Report Name:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Requested Date:' label starts with 'Requested Date:'

/// </summary>

public string UIRequestedDateLabelDisplayText = "Requested Date:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Status:' label starts with 'Status:'

/// </summary>

public string UIStatusLabelDisplayText = "Status:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Started Date:' label starts with 'Started Date:'

/// </summary>

public string UIStartedDateLabelDisplayText = "Started Date:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Finished Date:' label starts with 'Finished Date:'

/// </summary>

public string UIFinishedDateLabelDisplayText = "Finished Date:";

/// <summary>

/// Verify that the 'Id' property of 'reportNameTxt' text box equals 'reportNameTxt'

/// </summary>

public string UIReportNameTxtEditId = "reportNameTxt";

/// <summary>

/// Verify that the 'Id' property of 'requestedDateDivDtPick' text box equals 'requestedDateDivDtPick'

/// </summary>

public string UIRequestedDateDivDtPiEditId = "requestedDateDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'statusTxt' combo box equals 'statusTxt'

/// </summary>

public string UIStatusTxtComboBoxId = "statusTxt";

/// <summary>

/// Verify that the 'Id' property of 'startedDateDivDtPick' text box equals 'startedDateDivDtPick'

/// </summary>

public string UIStartedDateDivDtPickEditId = "startedDateDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'finishedDateDivDtPick' text box equals 'finishedDateDivDtPick'

/// </summary>

public string UIFinishedDateDivDtPicEditId = "finishedDateDivDtPick";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Datastats\_Overview'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Datastats\_OverviewExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Datastats Overview' pane starts with 'Datastats Overview'

/// </summary>

public string UIDatastatsOverviewPaneDisplayText = "Datastats Overview";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

/// </summary>

public string UILogDateLabelDisplayText = "Log Date:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Size:' label starts with 'Size:'

/// </summary>

public string UISizeLabelDisplayText = "Size:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'logDatePickerDtPick' text box equals 'logDatePickerDtPick'

/// </summary>

public string UILogDatePickerDtPickEditId = "logDatePickerDtPick";

/// <summary>

/// Verify that the 'Id' property of 'valueOptionInput' text box equals 'valueOptionInput'

/// </summary>

public string UIValueOptionInputEditId = "valueOptionInput";

/// <summary>

/// Verify that the 'Id' property of 'operatorOptionSelect' combo box equals 'operatorOptionSelect'

/// </summary>

public string UIOperatorOptionSelectComboBoxId = "operatorOptionSelect";

/// <summary>

/// Verify that the 'Id' property of 'unitOptionSelect' combo box equals 'unitOptionSelect'

/// </summary>

public string UIUnitOptionSelectComboBoxId = "unitOptionSelect";

/// <summary>

/// Verify that the 'Id' property of 'typeOptionSelect' combo box equals 'typeOptionSelect'

/// </summary>

public string UITypeOptionSelectComboBoxId = "typeOptionSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Datastats\_Find'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Datastats\_FindExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Data Stats Find' pane equals 'Data Stats Find'

/// </summary>

public string UIDataStatsFindPaneDisplayText = "Data Stats Find";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Volume:' label equals 'Volume:'

/// </summary>

public string UIVolumeLabelDisplayText = "Volume:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Share:' label starts with 'Share:'

/// </summary>

public string UIShareLabelDisplayText = "Share:";

/// <summary>

/// Verify that the 'DisplayText' property of 'I am Searching For:' label starts with 'I am Searching For:'

/// </summary>

public string UIIamSearchingForLabelDisplayText = "I am Searching For:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search Criteria:' label starts with 'Search Criteria:'

/// </summary>

public string UISearchCriteriaLabelDisplayText = "Search Criteria:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Last Modified:' label equals 'Last Modified:'

/// </summary>

public string UILastModifiedLabelDisplayText = "Last Modified:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Last Accessed:' label equals 'Last Accessed:'

/// </summary>

public string UILastAccessedLabelDisplayText = "Last Accessed:";

/// <summary>

/// Verify that the 'DisplayText' property of 'File Extension:' label starts with 'File Extension:'

/// </summary>

public string UIFileExtensionLabelDisplayText = "File Extension:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Min Size (MB):' label starts with 'Min Size (MB):'

/// </summary>

public string UIMinSizeMBLabelDisplayText = "Min Size (MB):";

/// <summary>

/// Verify that the 'DisplayText' property of 'authDisclaimerPlaceholder' pane starts with 'Please Note: Only results you are authorized to view will be displayed.'

/// </summary>

public string UIAuthDisclaimerPlacehPaneDisplayText = "Please Note: Only results you are authorized to view will be displayed.";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'volumeInputBox' text box equals 'volumeInputBox'

/// </summary>

public string UIVolumeInputBoxEditId = "volumeInputBox";

/// <summary>

/// Verify that the 'Id' property of 'shareInputBox' text box equals 'shareInputBox'

/// </summary>

public string UIShareInputBoxEditId = "shareInputBox";

/// <summary>

/// Verify that the 'Id' property of 'filePicker' text box equals 'filePicker'

/// </summary>

public string UIFilePickerEditId = "filePicker";

/// <summary>

/// Verify that the 'Id' property of 'txtExtension' text box equals 'txtExtension'

/// </summary>

public string UITxtExtensionEditId = "txtExtension";

/// <summary>

/// Verify that the 'Id' property of 'dateModifiedDivDtPick' text box equals 'dateModifiedDivDtPick'

/// </summary>

public string UIDateModifiedDivDtPicEditId = "dateModifiedDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'dateAccessedDivDtPick' text box equals 'dateAccessedDivDtPick'

/// </summary>

public string UIDateAccessedDivDtPicEditId = "dateAccessedDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'minSize' text box equals 'minSize'

/// </summary>

public string UIMinSizeEditId = "minSize";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Datastats\_File\_Usage\_DLP'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Datastats\_File\_Usage\_DLPExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'File Usage (DLP)' pane equals 'File Usage (DLP)'

/// </summary>

public string UIFileUsageDLPPaneDisplayText = "File Usage (DLP)";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Group by Size' label starts with 'Group by Size'

/// </summary>

public string UIGroupbySizeLabelDisplayText = "Group by Size";

/// <summary>

/// Verify that the 'DisplayText' property of 'Group by Extension' label starts with 'Group by Extension'

/// </summary>

public string UIGroupbyExtensionLabelDisplayText = "Group by Extension";

/// <summary>

/// Verify that the 'DisplayText' property of 'Group by Last Modified Date' label starts with 'Group by Last Modified Date'

/// </summary>

public string UIGroupbyLastModifiedDLabelDisplayText = "Group by Last Modified Date";

/// <summary>

/// Verify that the 'DisplayText' property of 'Group by Extension (large files)' label starts with 'Group by Extension (large files)'

/// </summary>

public string UIGroupbyExtensionlargLabelDisplayText = "Group by Extension (large files)";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'showBySizeCk' check box equals 'showBySizeCk'

/// </summary>

public string UIShowBySizeCkCheckBoxId = "showBySizeCk";

/// <summary>

/// Verify that the 'Id' property of 'showByExtensionCk' check box equals 'showByExtensionCk'

/// </summary>

public string UIShowByExtensionCkCheckBoxId = "showByExtensionCk";

/// <summary>

/// Verify that the 'Id' property of 'showByDateCk' check box equals 'showByDateCk'

/// </summary>

public string UIShowByDateCkCheckBoxId = "showByDateCk";

/// <summary>

/// Verify that the 'Id' property of 'largeFilesByExtensionCk' check box equals 'largeFilesByExtensionCk'

/// </summary>

public string UILargeFilesByExtensioCheckBoxId = "largeFilesByExtensionCk";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_PST\_Report'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_PST\_ReportExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'PST Report' pane equals 'PST Report'

/// </summary>

public string UIPSTReportPaneDisplayText = "PST Report";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'DisplayText' property of 'Generate Report' button equals 'Generate Report'

/// </summary>

public string UIGenerateReportButtonDisplayText = "Generate Report";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_AntiVirus'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_AntiVirusExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'AntiVirus' pane equals 'AntiVirus'

/// </summary>

public string UIAntiVirusPaneDisplayText = "AntiVirus";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label equals 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

/// </summary>

public string UILogDateLabelDisplayText = "Log Date:";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

/// </summary>

public string UILogDateDivDtPickEditId = "logDateDivDtPick";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Security\_Check'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Security\_CheckExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Security Check' pane starts with 'Security Check'

/// </summary>

public string UISecurityCheckPaneDisplayText = "Security Check";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

/// </summary>

public string UILogDateLabelDisplayText = "Log Date:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

/// </summary>

public string UILogDateDivDtPickEditId = "logDateDivDtPick";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Security\_Groups'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Security\_GroupsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Security Groups' pane starts with 'Security Groups'

/// </summary>

public string UISecurityGroupsPaneDisplayText = "Security Groups";

/// <summary>

/// Verify that the 'Id' property of 'Add New' button equals 'btnSecurityAdd'

/// </summary>

public string UIAddNewButtonId = "btnSecurityAdd";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Local\_Users'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Local\_UsersExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Local Users' pane equals 'Local Users'

/// </summary>

public string UILocalUsersPaneDisplayText = "Local Users";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Filter:' label starts with 'Filter:'

/// </summary>

public string UIFilterLabelDisplayText = "Filter:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'FilterSelect' combo box equals 'FilterSelect'

/// </summary>

public string UIFilterSelectComboBoxId = "FilterSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Local\_Groups'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Local\_GroupsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Local Groups' pane equals 'Local Groups'

/// </summary>

public string UILocalGroupsPaneDisplayText = "Local Groups";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Filter:' label starts with 'Filter:'

/// </summary>

public string UIFilterLabelDisplayText = "Filter:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Local Group :' label starts with 'Local Group:'

/// </summary>

public string UILocalGroupLabelDisplayText = "Local Group:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'FilterSelect' combo box equals 'FilterSelect'

/// </summary>

public string UIFilterSelectComboBoxId = "FilterSelect";

/// <summary>

/// Verify that the 'Id' property of 'LocalGroupSelect' combo box equals 'LocalGroupSelect'

/// </summary>

public string UILocalGroupSelectComboBoxId = "LocalGroupSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_User\_Rights'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_User\_RightsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'User Rights' pane equals 'User Rights'

/// </summary>

public string UIUserRightsPaneDisplayText = "User Rights";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'User Right:' label starts with 'User Right:'

/// </summary>

public string UIUserRightLabelDisplayText = "User Right:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

/// </summary>

public string UILogDateLabelDisplayText = "Log Date:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Trustee:' label starts with 'Trustee:'

/// </summary>

public string UITrusteeLabelDisplayText = "Trustee:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'UserRightSelect' combo box equals 'UserRightSelect'

/// </summary>

public string UIUserRightSelectComboBoxId = "UserRightSelect";

/// <summary>

/// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

/// </summary>

public string UILogDateDivDtPickEditId = "logDateDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'TrusteeSelect' combo box equals 'TrusteeSelect'

/// </summary>

public string UITrusteeSelectComboBoxId = "TrusteeSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Orphans'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_OrphansExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Orphans' pane equals 'Orphans'

/// </summary>

public string UIOrphansPaneDisplayText = "Orphans";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Orphan Type:' label starts with 'Orphan Type:'

/// </summary>

public string UIOrphanTypeLabelDisplayText = "Orphan Type:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Log Date:' label starts with 'Log Date:'

/// </summary>

public string UILogDateLabelDisplayText = "Log Date:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Server Type:' label starts with 'Server Type:'

/// </summary>

public string UIServerTypeLabelDisplayText = "Server Type:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'OrphanTypeSelect' combo box equals 'OrphanTypeSelect'

/// </summary>

public string UIOrphanTypeSelectComboBoxId = "OrphanTypeSelect";

/// <summary>

/// Verify that the 'Id' property of 'logDateDivDtPick' text box equals 'logDateDivDtPick'

/// </summary>

public string UILogDateDivDtPickEditId = "logDateDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'ServerTypeSelect' combo box equals 'ServerTypeSelect'

/// </summary>

public string UIServerTypeSelectComboBoxId = "ServerTypeSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Server\_Scheduled\_Tasks'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Server\_Scheduled\_TasksExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Scheduled Tasks' pane equals 'Scheduled Tasks'

/// </summary>

public string UIScheduledTasksPaneDisplayText = "Scheduled Tasks";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Task Name:' label starts with 'Task Name:'

/// </summary>

public string UITaskNameLabelDisplayText = "Task Name:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Last Result:' label starts with 'Last Result:'

/// </summary>

public string UILastResultLabelDisplayText = "Last Result:";

/// <summary>

/// Verify that the 'DisplayText' property of 'State:' label starts with 'State:'

/// </summary>

public string UIStateLabelDisplayText = "State:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'taskNameSearch' text box equals 'taskNameSearch'

/// </summary>

public string UITaskNameSearchEditId = "taskNameSearch";

/// <summary>

/// Verify that the 'Id' property of 'lastResult' combo box equals 'lastResult'

/// </summary>

public string UILastResultComboBoxId = "lastResult";

/// <summary>

/// Verify that the 'Id' property of 'state' combo box equals 'state'

/// </summary>

public string UIStateComboBoxId = "state";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Share\_Exclusion'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Share\_ExclusionExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Global Exclusions' pane starts with 'Global Exclusions'

/// </summary>

public string UIGlobalExclusionsPaneDisplayText = "Global Exclusions";

/// <summary>

/// Verify that the 'DisplayText' property of 'Exclusions For Teams' pane starts with 'Exclusions For Teams'

/// </summary>

public string UIExclusionsForTeamsPaneDisplayText = "Exclusions For Teams";

/// <summary>

/// Verify that the 'DisplayText' property of 'Add Share Exclusion' button equals 'Add Share Exclusion'

/// </summary>

public string UIAddShareExclusionButtonDisplayText = "Add Share Exclusion";

/// <summary>

/// Verify that the 'DisplayText' property of 'Add Path Exclusion' button equals 'Add Path Exclusion'

/// </summary>

public string UIAddPathExclusionButtonDisplayText = "Add Path Exclusion";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Manage\_Monitor\_Account'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Manage\_Monitor\_AccountExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'DisplayText' property of 'Manage Monitor Account' pane starts with 'Manage Monitor Account'

/// </summary>

public string UIManageMonitorAccountPaneDisplayText = "Manage Monitor Account";

/// <summary>

/// Verify that the 'DisplayText' property of 'Add Account' button equals 'Add Account'

/// </summary>

public string UIAddAccountButtonDisplayText = "Add Account";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Add\_Volume\_Exception'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Add\_Volume\_ExceptionParams

{

#region Fields

/// <summary>

/// Type 'nasrut151v1cr' in 'virtualDeviceAutoComp' text box

/// </summary>

public string UIVirtualDeviceAutoComEditText = "nasrut151v1cr";

/// <summary>

/// Type 'swdcitv001' in 'volumeAutoCompPopup' text box

/// </summary>

public string UIVolumeAutoCompPopupEditText = "swdcitv001";

/// <summary>

/// Type '456' in text box

/// </summary>

public string UIItemEditText = "456";

/// <summary>

/// Type '100' in text box

/// </summary>

public string UIItemEdit1Text = "100";

/// <summary>

/// Type '123' in text box

/// </summary>

public string UIItemEdit2Text = "123";

/// <summary>

/// Type 'Add Volume Exception' in text box

/// </summary>

public string UIItemEdit3Text = "Add Volume Exception";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Edit\_Exception\_Comment'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Edit\_Exception\_CommentParams

{

#region Fields

/// <summary>

/// Type 'Edit Volume Exception' in 'comment' text box

/// </summary>

public string UICommentEditText = "Edit Volume Exception";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Edit\_Exception\_popup'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Edit\_Exception\_popupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Exception Details - Edit' pane equals 'Exception Details - Edit'

/// </summary>

public string UIExceptionDetailsEditPaneInnerText = "Exception Details - Edit";

/// <summary>

/// Verify that the 'Id' property of 'comment' text box equals 'comment'

/// </summary>

public string UICommentEditId = "comment";

/// <summary>

/// Verify that the 'TagName' property of 'Save' button equals 'BUTTON'

/// </summary>

public string UISaveButtonTagName = "BUTTON";

/// <summary>

/// Verify that the 'TagName' property of 'Close' button equals 'BUTTON'

/// </summary>

public string UICloseButton1TagName = "BUTTON";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Delete\_Exception\_popup'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Delete\_Exception\_popupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Exception - Delete' pane equals 'Exception - Delete'

/// </summary>

public string UIExceptionDeletePaneInnerText = "Exception - Delete";

/// <summary>

/// Verify that the 'InnerText' property of 'nasrut151v1crp' cell starts with 'nasrut151v1crp'

/// </summary>

public string UINasrut151v1crpCellInnerText = "nasrut151v1crp";

/// <summary>

/// Verify that the 'InnerText' property of 'swdcitv0010' cell starts with 'swdcitv0010'

/// </summary>

public string UISwdcitv0010CellInnerText = "swdcitv0010";

/// <summary>

/// Verify that the 'InnerText' property of 'Edited Volume Exception' cell starts with 'Edited Volume Exception'

/// </summary>

public string UIEditVolumeExceptionCellInnerText = "Edited Volume Exception";

/// <summary>

/// Verify that the 'Id' property of 'Delete' button equals 'deleteButton'

/// </summary>

public string UIDeleteButtonId = "deleteButton";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Select\_value\_from\_list\_box'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Select\_value\_from\_list\_boxParams

{

#region Fields

/// <summary>

/// Select '150080 - Application Support' in 'Teams' list box

/// </summary>

public string UITeamsListSelectedItemsAsString = "150080 - Application Support";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Select\_from\_list\_box'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Select\_from\_list\_boxParams

{

#region Fields

/// <summary>

/// Select '150080 - Application Support' in 'Teams' list box

/// </summary>

public string UITeamsListSelectedItemsAsString = "150080 - Application Support";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Item\_List\_Box'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Item\_List\_BoxExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of list item equals '150080 - Application Support'

/// </summary>

public string UIItemListItemInnerText = "150080 - Application Support";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Security\_Groups\_wait\_for\_data\_load'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Security\_Groups\_wait\_for\_data\_loadExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Processing...' label contains 'Processing...'

/// </summary>

public string UIProcessingLabelInnerText = "Processing...";

/// <summary>

/// Verify that the 'InnerText' property of 'Delete' link equals 'Delete'

/// </summary>

public string UIDeleteHyperlinkInnerText = "Delete";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Add\_Security\_Group'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Add\_Security\_GroupParams

{

#region Fields

/// <summary>

/// Select '150080 - Application Support' in 'TeamSelect' combo box

/// </summary>

public string UITeamSelectComboBoxSelectedItem = "150080 - Application Support";

public string UITeamSelectComboBoxSelectedItem\_Prod = "APPC Wintel Cluster A";

/// <summary>

/// Select 'LAC' in 'GroupDomainSelect' combo box

/// </summary>

public string UIGroupDomainSelectComboBoxSelectedItem = "LACDEV";

/// <summary>

/// Type 'AddDeleteTestGroupName' in 'iGroupName' text box

/// </summary>

public string UIIGroupNameEditText = "AddDeleteTestGroupName";

/// <summary>

/// Type 'AddDeleteTestGroupDescription' in 'iDescription' text box

/// </summary>

public string UIIDescriptionEditText = "AddDeleteTestGroupDescription";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Add\_Security\_Group'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Add\_Security\_GroupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Add Security Groups' pane starts with 'Add Security Groups'

/// </summary>

public string UIAddSecurityGroupsPaneInnerText = "Add Security Groups";

/// <summary>

/// Verify that the 'InnerText' property of 'Team Name:' label equals 'Team Name:'

/// </summary>

public string UITeamNameLabelInnerText = "Team Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Group Domain:' label equals 'Group Domain:'

/// </summary>

public string UIGroupDomainLabelInnerText = "Group Domain:";

/// <summary>

/// Verify that the 'InnerText' property of 'Group Name:' label starts with 'Group Name:'

/// </summary>

public string UIGroupNameLabelInnerText = "Group Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Group Description:' label starts with 'Group Description:'

/// </summary>

public string UIGroupDescriptionLabelInnerText = "Group Description:";

/// <summary>

/// Verify that the 'Id' property of 'TeamSelect' combo box equals 'TeamSelect'

/// </summary>

public string UITeamSelectComboBoxId = "TeamSelect";

/// <summary>

/// Verify that the 'Id' property of 'GroupDomainSelect' combo box equals 'GroupDomainSelect'

/// </summary>

public string UIGroupDomainSelectComboBoxId = "GroupDomainSelect";

/// <summary>

/// Verify that the 'Id' property of 'iGroupName' text box equals 'iGroupName'

/// </summary>

public string UIIGroupNameEditId = "iGroupName";

/// <summary>

/// Verify that the 'Id' property of 'iDescription' text box equals 'iDescription'

/// </summary>

public string UIIDescriptionEditId = "iDescription";

/// <summary>

/// Verify that the 'InnerText' property of 'Back to Grid' link equals 'Back to Grid'

/// </summary>

public string UIBacktoGridHyperlinkInnerText = "Back to Grid";

/// <summary>

/// Verify that the 'Id' property of 'Save' button equals 'btnSecurityGroupsSave'

/// </summary>

public string UISaveButtonId = "btnSecurityGroupsSave";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Handle\_IE\_OST\_Login\_Popup'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Handle\_IE\_OST\_Login\_PopupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'Name' property of 'Connecting to vm-8b54-769d.nam.nsroot.net.' label equals 'Connecting to vm-8b54-769d.nam.nsroot.net.'

/// </summary>

public string UIConnectingtovm8b5476TextName = "Connecting to vm-8b54-769d.nam.nsroot.net.";

/// <summary>

/// Verify that the 'Name' property of 'User name' text box equals 'User name'

/// </summary>

public string UIUsernameEditName = "User name";

/// <summary>

/// Verify that the 'Name' property of 'Password' text box equals 'Password'

/// </summary>

public string UIPasswordEditName = "Password";

/// <summary>

/// Verify that the 'Name' property of 'OK' button equals 'OK'

/// </summary>

public string UIOKButtonName = "OK";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Navigate\_to\_OST'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Navigate\_to\_OSTParams

{

#region Fields

/// <summary>

/// Go to web page 'http://vm-775e-1f0f.nam.nsroot.net:8100/Ostx/' using new browser instance

/// </summary>

public string Url = "http://vm-775e-1f0f.nam.nsroot.net:8100/Ostx/";

public string UAT\_URL = "https://operationsstandardtoolset.uat.citigroup.net/ostx";

public string PROD\_URL = "https://operationsstandardtoolset.citigroup.net/Ostx/";

#endregion

}

//public class Navigate\_to\_OST\_UATParams

//{

// #region Fields

// /// <summary>

// /// Go to web page 'http://ostuat.nam.nsroot.net/Ostx' using new browser instance

// /// </summary>

// public string Url = "http://ostuat.nam.nsroot.net/Ostx";

// #endregion

//}

//public class Navigate\_to\_OST\_PRODParams

//{

// #region Fields

// /// <summary>

// /// Go to web page 'http://ost.nam.nsroot.net/Ostx' using new browser instance

// /// </summary>

// public string Url = "http://ost.nam.nsroot.net/Ostx";

// #endregion

//}

/// <summary>

/// Parameters to be passed into 'Add\_Share\_Exclusion'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Add\_Share\_ExclusionParams

{

#region Fields

/// <summary>

/// Select '150080 - Application Support' in 'HostNameTeamSelect' combo box

/// </summary>

public string UIHostNameTeamSelectComboBoxSelectedItem = "150080 - Application Support";

/// <summary>

/// Type 'kc90246$' in 'iShareExclusionText' text box

/// </summary>

public string UIIShareExclusionTextEditText = "kc90246$";

/// <summary>

/// Type 'zzzzzzDescriptionShareExclusion' in 'iDescription' text box

/// </summary>

public string UIIDescriptionEditText = "zzzzzzDescriptionShareExclusion";

/// <summary>

/// Select 'test kc90246' in 'HostNameSelect' combo box

/// </summary>

public string UIHostNameSelectComboBoxSelectedItem = "test kc90246";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Data\_Stats\_Details'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Data\_Stats\_DetailsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Data Stats Details' pane equals 'Data Stats Details'

/// </summary>

public string UIDataStatsDetailsPaneInnerText = "Data Stats Details";

/// <summary>

/// Verify that the 'Class' property of 'D$\ (5.31 GBs)' link starts with 'jstree-anchor'

/// </summary>

public string UID531GBsHyperlinkClass = "jstree-anchor";

/// <summary>

/// Verify that the 'InnerText' property of 'Export folders' button contains 'Export folders'

/// </summary>

public string UIExportfoldersButtonInnerText = "Export folders";

/// <summary>

/// Verify that the 'InnerText' property of 'Export files' button contains 'Export files'

/// </summary>

public string UIExportfilesButtonInnerText = "Export files";

/// <summary>

/// Verify that the 'InnerText' property of 'Export folders (Children included)' button contains 'Export folders (Children included)'

/// </summary>

public string UIExportfoldersChildreButtonInnerText = "Export folders (Children included)";

/// <summary>

/// Verify that the 'InnerText' property of 'Export files (Children included)' button contains 'Export files (Children included)'

/// </summary>

public string UIExportfilesChildreniButtonInnerText = "Export files (Children included)";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Dfs\_Shares'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Dfs\_SharesExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'DFS Shares' pane equals 'DFS Shares'

/// </summary>

public string UIDFSSharesPaneInnerText = "DFS Shares";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Search For:' label starts with 'Search For:'

/// </summary>

public string UISearchForLabelInnerText = "Search For:";

/// <summary>

/// Verify that the 'InnerText' property of 'in' label starts with 'in'

/// </summary>

public string UIINLabelInnerText = "in";

/// <summary>

/// Verify that the 'InnerText' property of 'Show Archived:' label starts with 'Show Archived:'

/// </summary>

public string UIShowArchivedLabelInnerText = "Show Archived:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

/// </summary>

public string UIItemEditClass = "ui-autocomplete-input";

/// <summary>

/// Verify that the 'InnerText' property of combo box contains 'Link NameLink HostLink FolderPathDFS Shares'

/// </summary>

public string UIItemComboBoxInnerText = "Link Name";

/// <summary>

/// Verify that the 'ControlType' property of check box equals 'CheckBox'

/// </summary>

public string UIItemCheckBoxControlType = "CheckBox";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Global\_Account\_Search'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Global\_Account\_SearchExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Global Account Search' pane equals 'Global Account Search'

/// </summary>

public string UIGlobalAccountSearchPaneInnerText = "Global Account Entitlement Search";

/// <summary>

/// Verify that the 'InnerText' property of 'Account:' label starts with 'Account:'

/// </summary>

public string UIAccountLabelInnerText = "User/FID/Group:";

/// <summary>

/// Verify that the 'Id' property of 'accountTxt' text box equals 'accountTxt'

/// </summary>

public string UIAccountTxtEditId = "accountTxt";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Global\_Compliance'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Global\_ComplianceExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Global Compliance' pane equals 'Global Compliance'

/// </summary>

public string UIGlobalCompliancePaneInnerText = "Global Compliance";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Monitor\_Health'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Monitor\_HealthExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Ost Monitors Overview' custom control equals 'Ost Monitors Overview'

/// </summary>

public string UIOstMonitorsOverviewCustomInnerText = "Ost Monitors Overview";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Ntfs\_Permissions'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Ntfs\_PermissionsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'NTFS Permissions' pane equals 'NTFS Permissions'

/// </summary>

public string UINTFSPermissionsPaneInnerText = "NTFS Permissions";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Name:' label starts with 'Share Name:'

/// </summary>

public string UIShareNameLabelInnerText = "Share Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Owner:' label starts with 'Owner:'

/// </summary>

public string UIOwnerLabelInnerText = "Owner:";

/// <summary>

/// Verify that the 'InnerText' property of 'Permission:' label starts with 'Permission:'

/// </summary>

public string UIPermissionLabelInnerText = "Permission:";

/// <summary>

/// Verify that the 'InnerText' property of 'Account:' label starts with 'Account:'

/// </summary>

public string UIAccountLabelInnerText = "Account:";

/// <summary>

/// Verify that the 'InnerText' property of 'Inheritance:' label starts with 'Inheritance:'

/// </summary>

public string UIInheritanceLabelInnerText = "Inheritance:";

/// <summary>

/// Verify that the 'InnerText' property of 'Path:' label starts with 'Path:'

/// </summary>

public string UIPathLabelInnerText = "Path:";

/// <summary>

/// Verify that the 'InnerText' property of 'Initial Scan Date:' label starts with 'Initial Scan Date:'

/// </summary>

public string UIInitialScanDateLabelInnerText = "Initial Scan Date:";

/// <summary>

/// Verify that the 'InnerText' property of 'Export with Group Members' pane starts with 'Export with Group Members'

/// </summary>

public string UIExportwithGroupMembePaneInnerText = "Export with Group Members";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'shareNameTxt' text box equals 'shareNameTxt'

/// </summary>

public string UIShareNameTxtEditId = "shareNameTxt";

/// <summary>

/// Verify that the 'Id' property of 'ownerTxt' text box equals 'ownerTxt'

/// </summary>

public string UIOwnerTxtEditId = "ownerTxt";

/// <summary>

/// Verify that the 'Id' property of 'permissionTxt' text box equals 'permissionTxt'

/// </summary>

public string UIPermissionTxtEditId = "permissionTxt";

/// <summary>

/// Verify that the 'Id' property of 'accountTxt' text box equals 'accountTxt'

/// </summary>

public string UIAccountTxtEditId = "accountTxt";

/// <summary>

/// Verify that the 'Id' property of 'inheritanceTxt' combo box equals 'inheritanceTxt'

/// </summary>

public string UIInheritanceTxtComboBoxId = "inheritanceTxt";

/// <summary>

/// Verify that the 'Id' property of 'pathTxt' text box equals 'pathTxt'

/// </summary>

public string UIPathTxtEditId = "pathTxt";

/// <summary>

/// Verify that the 'Id' property of 'scanDateDivDtPick' text box equals 'scanDateDivDtPick'

/// </summary>

public string UIScanDateDivDtPickEditId = "scanDateDivDtPick";

/// <summary>

/// Verify that the 'Id' property of 'exportGroupMembers' check box equals 'exportGroupMembers'

/// </summary>

public string UIExportGroupMembersCheckBoxId = "exportGroupMembers";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Print\_Queue'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Print\_QueueExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Current Print Queues' pane starts with 'Current Print Queues'

/// </summary>

public string UICurrentPrintQueuesPaneInnerText = "Current Print Queues";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label starts with 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

/// </summary>

public string UIQueueLabelDisplayText = "Queue:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Location:' label starts with 'Location:'

/// </summary>

public string UILocationLabelDisplayText = "Location:";

/// <summary>

/// Verify that the 'DisplayText' property of 'IP Address:' label starts with 'IP Address:'

/// </summary>

public string UIIPAddressLabelDisplayText = "IP Address:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Driver:' label starts with 'Driver:'

/// </summary>

public string UIDriverLabelDisplayText = "Driver:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'queueInput' text box equals 'queueInput'

/// </summary>

public string UIQueueInputEditId = "queueInput";

/// <summary>

/// Verify that the 'Id' property of 'locationInput' text box equals 'locationInput'

/// </summary>

public string UILocationInputEditId = "locationInput";

/// <summary>

/// Verify that the 'Id' property of 'ipAddressInput' text box equals 'ipAddressInput'

/// </summary>

public string UIIpAddressInputEditId = "ipAddressInput";

/// <summary>

/// Verify that the 'Id' property of 'driverTypeSelect' combo box equals 'driverTypeSelect'

/// </summary>

public string UIDriverTypeSelectComboBoxId = "driverTypeSelect";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Printer\_popup'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Printer\_popupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Printer \DUBCTISP02\roxNew' pane starts with 'Printer'

/// </summary>

public string UIPrinterDUBCTISP02roxPaneInnerText = "Printer";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' pane starts with 'Host:'

/// </summary>

public string UIHostPaneInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Queue Name:' pane starts with 'Queue Name:'

/// </summary>

public string UIQueueNamePaneInnerText = "Queue Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Share Name:' pane starts with 'Share Name:'

/// </summary>

public string UIShareNamePaneInnerText = "Share Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'Driver:' pane starts with 'Driver:'

/// </summary>

public string UIDriverPaneInnerText = "Driver:";

/// <summary>

/// Verify that the 'InnerText' property of 'Location:' pane starts with 'Location:'

/// </summary>

public string UILocationPaneInnerText = "Location:";

/// <summary>

/// Verify that the 'InnerText' property of 'Comment:' pane starts with 'Comment:'

/// </summary>

public string UICommentPaneInnerText = "Comment:";

/// <summary>

/// Verify that the 'InnerText' property of 'Port Name:' pane starts with 'Port Name:'

/// </summary>

public string UIPortNamePaneInnerText = "Port Name:";

/// <summary>

/// Verify that the 'InnerText' property of 'IP Address:' pane starts with 'IP Address:'

/// </summary>

public string UIIPAddressPaneInnerText = "IP Address:";

/// <summary>

/// Verify that the 'InnerText' property of 'LogDate:' pane starts with 'LogDate:'

/// </summary>

public string UILogDatePaneInnerText = "LogDate:";

/// <summary>

/// Verify that the 'InnerText' property of 'Print Status:' pane starts with 'Print Status:'

/// </summary>

public string UIPrintStatusPaneInnerText = "Print Status:";

/// <summary>

/// Verify that the 'InnerText' property of 'Print Length:' pane starts with 'Print Length:'

/// </summary>

public string UIPrintLengthPaneInnerText = "Print Length:";

/// <summary>

/// Verify that the 'InnerText' property of 'Print Jobs:' pane starts with 'Print Jobs:'

/// </summary>

public string UIPrintJobsPaneInnerText = "Print Jobs:";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Print\_Queue\_History'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Print\_Queue\_HistoryExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Print Queue History' pane equals 'Print Queue History'

/// </summary>

public string UIPrintQueueHistoryPaneInnerText = "Print Queue History";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

/// </summary>

public string UIQueueLabelDisplayText = "Queue:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Class' property of text box equals 'ui-autocomplete-input'

/// </summary>

public string UIItemEditClass = "ui-autocomplete-input";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_Print\_Queue\_Statistics'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_Print\_Queue\_StatisticsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Print Queue Statistics' pane equals 'Print Queue Statistics'

/// </summary>

public string UIPrintQueueStatisticsPaneInnerText = "Print Queue Statistics";

/// <summary>

/// Verify that the 'DisplayText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelDisplayText = "Region:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelDisplayText = "Team:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelDisplayText = "Host:";

/// <summary>

/// Verify that the 'DisplayText' property of 'Queue:' label starts with 'Queue:'

/// </summary>

public string UIQueueLabelDisplayText = "Queue:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of 'queueInput' text box equals 'queueInput'

/// </summary>

public string UIQueueInputEditId = "queueInput";

/// <summary>

/// Verify that the 'DisplayText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonDisplayText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_User\_Roles'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_User\_RolesExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'User Roles' pane starts with 'User Roles'

/// </summary>

public string UIUserRolesPaneInnerText = "User Roles";

/// <summary>

/// Verify that the 'Title' property of 'Displays a user's Role and Team access.' custom control equals 'Displays a user's Role and Team access.'

/// </summary>

public string UIDisplaysausersRoleanCustomTitle = "Displays users Role and Team access.";

/// <summary>

/// Verify that the 'InnerText' property of 'Soe Id:' label starts with 'Soe Id:'

/// </summary>

public string UISoeIdLabelInnerText = "Soe Id:";

/// <summary>

/// Verify that the 'Id' property of text box equals 'testid'

/// </summary>

public string UIItemEditId = "soeIdInput";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Elements\_System\_Information'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Elements\_System\_InformationExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'System Info' pane starts with 'System Info'

/// </summary>

public string UISystemInfoPaneInnerText = "System Info";

/// <summary>

/// Verify that the 'Title' property of 'This page provides information about OST version a...' custom control equals 'This page provides information about OST version and overall system information.'

/// </summary>

public string UIThispageprovidesinfoCustomTitle = "This page provides information about OST version and overall system information.";

/// <summary>

/// Verify that the 'InnerText' property of 'ostContext:' cell equals 'ostContext:'

/// </summary>

public string UIOstContextCellInnerText = "ostContext:";

/// <summary>

/// Verify that the 'InnerText' property of 'adSqlContext:' cell equals 'adSqlContext:'

/// </summary>

public string UIAdSqlContextCellInnerText = "adSqlContext:";

/// <summary>

/// Verify that the 'InnerText' property of 'empContext:' cell equals 'empContext:'

/// </summary>

public string UIEmpContextCellInnerText = "empContext:";

/// <summary>

/// Verify that the 'InnerText' property of 'supportContext:' cell equals 'supportContext:'

/// </summary>

public string UISupportContextCellInnerText = "supportContext:";

/// <summary>

/// Verify that the 'InnerText' property of 'ostDllVersion:' cell equals 'ostDllVersion:'

/// </summary>

public string UIOstDllVersionCellInnerText = "ostDllVersion:";

/// <summary>

/// Verify that the 'InnerText' property of 'ostBuildDate:' cell equals 'ostBuildDate:'

/// </summary>

public string UIOstBuildDateCellInnerText = "ostBuildDate:";

/// <summary>

/// Verify that the 'InnerText' property of 'ostFileVersion:' cell equals 'ostFileVersion:'

/// </summary>

public string UIOstFileVersionCellInnerText = "ostFileVersion:";

/// <summary>

/// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

/// </summary>

public string UIDataSourceNAMDNA390DCellInnerText = "Data Source=";

/// <summary>

/// Verify that the 'InnerText' property of 'Data source=NAMDNA390DSQL1.NAMUAT.NSROOT' cell starts with 'Data source='

/// </summary>

public string UIDatasourceNAMDNA390DCell1InnerText = "Data source=";

/// <summary>

/// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

/// </summary>

public string UIDataSourceNAMDNA390DCell2InnerText = "Data Source=";

/// <summary>

/// Verify that the 'InnerText' property of 'Data Source=NAMDNA390DSQL3.namuat.nsroot' cell starts with 'Data Source='

/// </summary>

public string UIDataSourceNAMDNA390DCell3InnerText = "Data Source=";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Host\_Maintenance\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Host\_Maintenance\_pageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'HostMaintance' pane equals 'Host Maintenance '

/// </summary>

public string UIHostMaintancePaneInnerText = "Host Maintenance";

/// <summary>

/// Verify that the 'InnerText' property of 'Region:' label equals 'Region:'

/// </summary>

public string UIRegionLabelInnerText = "Region:";

/// <summary>

/// Verify that the 'InnerText' property of 'Team:' label starts with 'Team:'

/// </summary>

public string UITeamLabelInnerText = "Team:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host:' label equals 'Host:'

/// </summary>

public string UIHostLabelInnerText = "Host:";

/// <summary>

/// Verify that the 'InnerText' property of 'Status:' label equals 'Status:'

/// </summary>

public string UIStatusLabelInnerText = "Status:";

/// <summary>

/// Verify that the 'InnerText' property of 'Host Type:' label equals 'Host Type:'

/// </summary>

public string UIHostTypeLabelInnerText = "Host Type:";

/// <summary>

/// Verify that the 'Id' property of 'availableRegionsDdl' combo box equals 'availableRegionsDdl'

/// </summary>

public string UIAvailableRegionsDdlComboBoxId = "availableRegionsDdl";

/// <summary>

/// Verify that the 'Id' property of 'availableTeamsDdl' combo box equals 'availableTeamsDdl'

/// </summary>

public string UIAvailableTeamsDdlComboBoxId = "availableTeamsDdl";

/// <summary>

/// Verify that the 'Id' property of 'hostName' text box equals 'hostName'

/// </summary>

public string UIHostNameEditId = "hostName";

/// <summary>

/// Verify that the 'Id' property of combo box equals 'StatusSelect'

/// </summary>

public string UIItemComboBoxId = "StatusSelect";

/// <summary>

/// Verify that the 'Id' property of combo box equals 'HostTypeSelect'

/// </summary>

public string UIItemComboBox1Id = "HostTypeSelect";

/// <summary>

/// Verify that the 'InnerText' property of 'Search' button equals 'Search'

/// </summary>

public string UISearchButtonInnerText = "Search";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Clear\_Chrome\_Cache'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Clear\_Chrome\_CacheParams

{

#region Fields

/// <summary>

/// Type '\*\*\*\*\*\*\*\*' in 'Address and search bar' text box

/// </summary>

public string UIAddressandsearchbarEditSendKeys = "icqE0xhGooZu5O43pVCFcE299lf3EiwA";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Search\_Home\_Global\_Search\_by\_Soeid'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Search\_Home\_Global\_Search\_by\_SoeidParams

{

#region Fields

/// <summary>

/// Type 'kc90246' in 'homeSearchText' text box

/// </summary>

public string UIHomeSearchTextEditText = "kc90246";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Home\_Global\_Search\_results'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Home\_Global\_Search\_resultsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Host' pane equals 'Host'

/// </summary>

public string UIHostPaneInnerText = "Host";

/// <summary>

/// Verify that the 'InnerText' property of 'SoeId' pane equals 'SoeId'

/// </summary>

public string UISoeIdPaneInnerText = "SoeId";

/// <summary>

/// Verify that the 'InnerText' property of 'test kc90246' link equals 'TEST KC90246'

/// </summary>

public string UITestkc90246HyperlinkInnerText = "test kc90246";

/// <summary>

/// Verify that the 'InnerText' property of 'Krzysztof Cygan' link equals 'KRZYSZTOF CYGAN'

/// </summary>

public string UIKrzysztofCyganHyperlinkInnerText = "krzysztof cygan";

public string UICostCodePaneInnerText = "CostCode";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Owners\_Info\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Owners\_Info\_pageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'OwnersForUNCPath' pane starts with 'OwnersForUNCPath'

/// </summary>

public string UIOwnersForUNCPathPaneInnerText = "Owners for UNC Path";

/// <summary>

/// Verify that the 'Title' property of 'Displays Owner information for a given UNC Path' custom control equals 'Displays Owner information for a given UNC Path'

/// </summary>

public string UIDisplaysOwnerinformaCustomTitle = "Displays Owner information for a given UNC Path or list of UNC Paths provided in a file";

/// <summary>

/// Verify that the 'InnerText' property of 'UNC Path:' label equals 'UNC Path:'

/// </summary>

public string UIUNCPathLabelInnerText = "UNC Path:";

/// <summary>

/// Verify that the 'Id' property of 'uncPathInputBox' text box equals 'uncPathInputBox'

/// </summary>

public string UIUncPathInputBoxEditId = "uncPathInputBox";

/// <summary>

/// Verify that the 'Id' property of 'Search' button equals 'searchButton'

/// </summary>

public string UISearchButtonId = "searchButton";

/// <summary>

/// Verify that the 'InnerText' property of 'Select UNC file to process:' label starts with 'Select UNC file to process:'

/// </summary>

public string UISelectUNCfiletoproceLabelInnerText = "Select UNC file to process:";

/// <summary>

/// Verify that the 'Id' property of 'fileUploadControl' file input equals 'fileUpload'

/// </summary>

public string UIFileUploadControlFileInputId = "fileUpload";

/// <summary>

/// Verify that the 'InnerText' property of 'Upload' button contains 'Upload'

/// </summary>

public string UIUploadButtonInnerText = "Upload";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_User\_Roles\_Search\_Results'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_User\_Roles\_Search\_ResultsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'UIItemDropdown' pane equals 'Teams'

/// </summary>

public string UIItemDropdownInnerText = "Teams";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Add\_Share\_Exception\_popup'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Add\_Share\_Exception\_popupExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Exception Details - Add' pane equals 'Exception Details - Add'

/// </summary>

public string UIExceptionDetailsAddPaneInnerText = "Exception Details - Add";

/// <summary>

/// Verify that the 'Id' property of 'virtualDeviceAutoComp' text box equals 'virtualDeviceAutoComp'

/// </summary>

public string UIVirtualDeviceAutoComEditId = "virtualDeviceAutoComp";

/// <summary>

/// Verify that the 'Id' property of 'shareAutoComp' text box equals 'shareAutoComp'

/// </summary>

public string UIShareAutoCompEditId = "shareAutoComp";

/// <summary>

/// Verify that the 'TagName' property of text box equals 'INPUT'

/// </summary>

public string UIItemEditTagName = "INPUT";

/// <summary>

/// Verify that the 'TagName' property of text box equals 'INPUT'

/// </summary>

public string UIItemEdit1TagName = "INPUT";

/// <summary>

/// Verify that the 'TagName' property of text box equals 'INPUT'

/// </summary>

public string UIItemEdit2TagName = "INPUT";

/// <summary>

/// Verify that the 'TagName' property of text box equals 'TEXTAREA'

/// </summary>

public string UIItemEdit3TagName = "TEXTAREA";

/// <summary>

/// Verify that the 'InnerText' property of 'Save' button equals 'Save'

/// </summary>

public string UISaveButtonInnerText = "Save";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Edit\_Recurring\_Schedule\_page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.21005.1")]

public class Verify\_Edit\_Recurring\_Schedule\_pageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'Edit Recurring Schedule' pane equals 'Add Recurring Schedule'

/// </summary>

public string UIEditRecurringSchedulPaneInnerText = "Add Recurring Schedule";

/// <summary>

/// Verify that the 'Id' property of 'outputTypeSl' combo box equals 'outputTypeSl'

/// </summary>

public string UIOutputTypeSlComboBoxId = "outputTypeSl";

/// <summary>

/// Verify that the 'Id' property of 'reportNameTxtScheduler' text box equals 'reportNameTxtScheduler'

/// </summary>

public string UIReportNameTxtSchedulEditId = "reportNameTxtScheduler";

/// <summary>

/// Verify that the 'Id' property of 'timeStampChk' check box equals 'timeStampChk'

/// </summary>

public string UITimeStampChkCheckBoxId = "timeStampChk";

/// <summary>

/// Verify that the 'Id' property of 'reportShareTxt' text box equals 'reportShareTxt'

/// </summary>

public string UIReportShareTxtEditId = "reportShareTxt";

/// <summary>

/// Verify that the 'Id' property of 'occuranceTxt' text box equals 'occuranceTxt'

/// </summary>

public string UIOccuranceTxtEditId = "occuranceTxt";

/// <summary>

/// Verify that the 'InnerText' property of 'Save' button equals 'Save'

/// </summary>

public string UISaveButtonInnerText = "Save";

/// <summary>

/// Verify that the 'InnerText' property of 'Close' button equals 'Close'

/// </summary>

public string UICloseButtonInnerText = "Close";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Objects'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.31101.0")]

public class ObjectsExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'prodName' custom control equals 'Terminated Staff Access Removal'

/// </summary>

public string UIProdNameCustomInnerText = "Terminated Staff Access Removal";

/// <summary>

/// Verify that the 'Id' property of 'List View' button equals 'listtoggle'

/// </summary>

public string UIListViewButtonId = "listtoggle";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_263895'

/// </summary>

public string UIBehalfOfLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_263895";

/// <summary>

/// Verify that the 'Id' property of 'Update' link equals 'link\_POD\_0\_DK\_47361\_21119\_9259'

/// </summary>

public string UIUpdateButtonId = "link\_POD\_0\_DK\_47361\_21119\_9259";

/// <summary>

/// Verify that the 'Id' property of '9259' text box equals 'query\_POD\_0\_DK\_47361\_21119\_9259'

/// </summary>

public string UITerminatedTextBoxId = "query\_POD\_0\_DK\_47361\_21119\_9259";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_9259'

/// </summary>

public string UITerminatedLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_9259";

/// <summary>

/// Verify that the 'Id' property of '(Please select) Execute Immediately Schedule to O...' combo box equals 'POD\_0\_DK\_47361\_21119\_294320'

/// </summary>

public string UIPleaseselectExecuteIComboBoxId = "POD\_0\_DK\_47361\_21119\_294320";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_173863'

/// </summary>

public string UIAuthorityLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_173863";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_622584'

/// </summary>

public string UINotAuthorisedLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_622584";

/// <summary>

/// Verify that the 'InnerText' property of 'Cygan, Krzysztof - 0004316355' pane equals 'Cygan, Krzysztof - 0004316355'

/// </summary>

public string UIBehalfOfCyganKrzysztofInnerText = "Cygan, Krzysztof - 0004316355";

/// <summary>

/// Verify that the 'InnerText' property of 'No Value Set' pane equals 'Cygan, Krzysztof - 0004316355'

/// </summary>

public string UITerminatedCyganKrzysztofInnerText = "Cygan, Krzysztof - 0004316355";

/// <summary>

/// Verify that the 'InnerText' property of 'No access - 0' pane equals 'No access - 0'

/// </summary>

public string UIAuthorityNoAccessInnerText = "No access - 0";

/// <summary>

/// Verify that the 'InnerText' property of 'MAZZUKI, MICHAEL' pane equals 'MAZZUKI, MICHAEL'

/// </summary>

public string UINotAuthorizedMAZZUKIMICHAELInnerText = "MAZZUKI, MICHAEL";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_585516'

/// </summary>

public string UIServicesLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_585516";

/// <summary>

/// Verify that the 'InnerText' property of 'No Value Set' pane equals 'Ireland'

/// </summary>

public string UIServicesIrelandInnerText = "Ireland";

/// <summary>

/// Verify that the 'Id' property of '(Please select) Yes No' combo box equals 'POD\_0\_DK\_47361\_21119\_711548'

/// </summary>

public string UIWorkingRemotelyYesNoComboBoxId = "POD\_0\_DK\_47361\_21119\_711548";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_276705'

/// </summary>

public string UITelephoneLookUpButtonId = "lookup\_POD\_0\_DK\_47361\_21119\_276705";

/// <summary>

/// Verify that the 'InnerText' property of 'None' pane equals 'None'

/// </summary>

public string UITelephoneNoneInnerText = "None";

/// <summary>

/// Verify that the 'Id' property of 'Look Up' link equals 'lookup\_POD\_0\_DK\_47361\_21119\_282064'

/// </summary>

public string UILookUpHyperlinkId = "lookup\_POD\_0\_DK\_47361\_21119\_282064";

/// <summary>

/// Verify that the 'Id' property of 'Yes' radio button equals 'POD\_0\_DK\_47361\_21119\_235413\_V\_Yes'

/// </summary>

public string UIYesRadioButtonId = "POD\_0\_DK\_47361\_21119\_235413\_V\_Yes";

/// <summary>

/// Verify that the 'InnerText' property of 'Quick Order' link contains 'Quick Order'

/// </summary>

public string UIQuickOrderHyperlinkInnerText = "Quick Order";

/// <summary>

/// Verify that the 'InnerText' property of 'Submit Order' link equals 'Submit Order'

/// </summary>

public string UISubmitOrderHyperlinkInnerText = "Submit Order";

/// <summary>

/// Verify that the 'InnerText' property of 'Thank You for Your Order' pane equals 'Thank You for Your Order'

/// </summary>

public string UIThankYouforYourOrderPaneInnerText = "Thank You for Your Order";

/// <summary>

/// Verify that the 'InnerText' property of 'Order #: WS239348' pane starts with 'Order #: WS'

/// </summary>

public string UIOrderWS239348PaneInnerText = "Order #: WS";

#endregion

}

/// <summary>

/// Parameters to be passed into 'Verify\_Scanner\_Health\_Page'

/// </summary>

[GeneratedCode("Coded UITest Builder", "12.0.31101.0")]

public class Verify\_Scanner\_Health\_PageExpectedValues

{

#region Fields

/// <summary>

/// Verify that the 'InnerText' property of 'rabbit@VM-03EF-8075' cell equals 'rabbit@VM-03EF-8075'

/// </summary>

public string UIRabbitVM03EF8075CellInnerText = "rabbit@VM-03EF-8075";

/// <summary>

/// Verify that the 'TagName' property of 'Scanner Health' pane equals 'Scanner Health'

/// </summary>

public string UIScannerHealthPaneTagName = "Scanner Health";

#endregion

}

}