Importing ALM Runs using Google Sheets

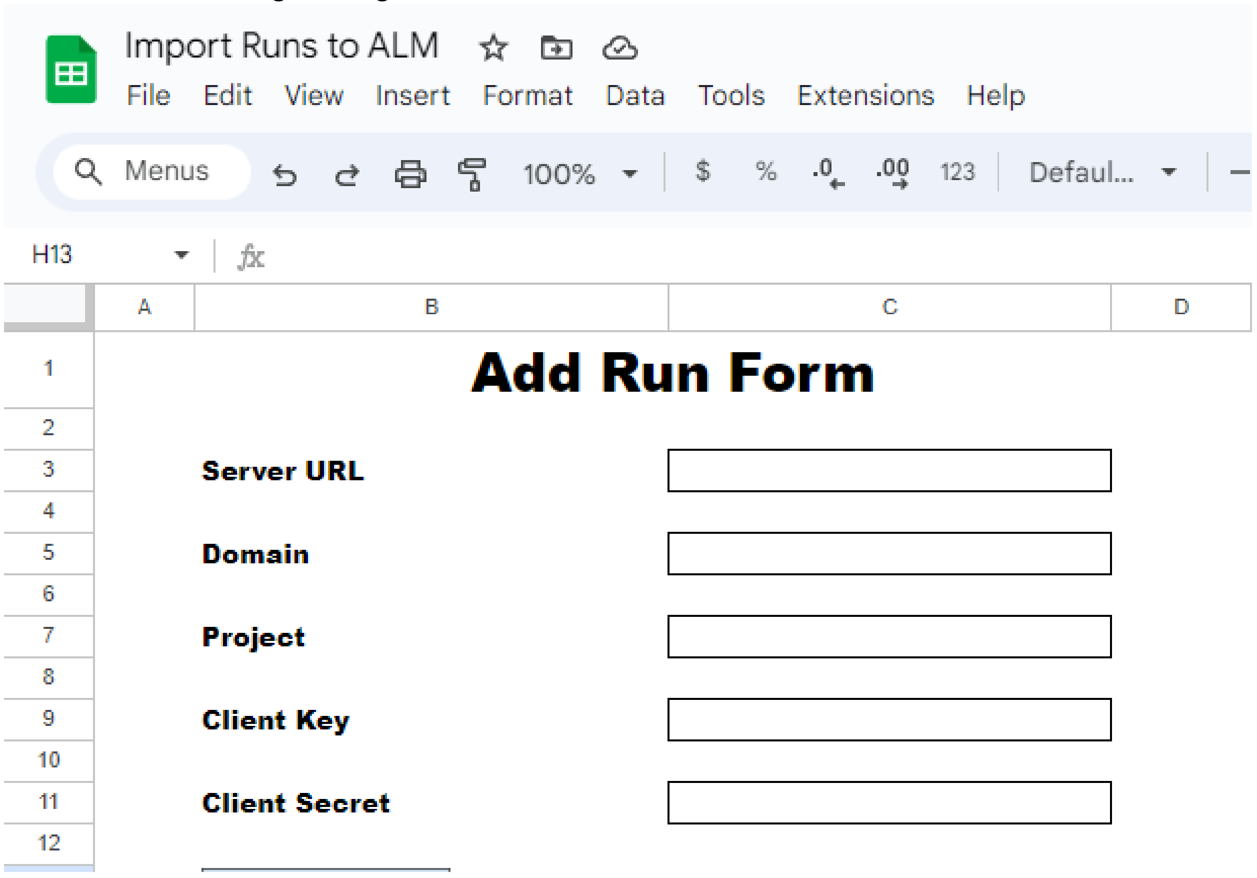
I. Introduction:

While ALM typically allows for runs to be executed within the application, leveraging Google Sheets can offer an additional alternative. With Google Sheets, teams can utilize familiar spreadsheet functionalities to manipulate and organize test run data before importing it into the ALM tool. Moreover, Google Sheets is OS agnostic, ensuring seamless usage across different operating systems without compatibility issues.

In this guide, we'll delve into the step-by-step process of importing ALM runs using Google Sheets in conjunction with the ALM tool's REST API.

II. Implementation

1. Login to Google Sheets.
2. Add a new sheet and call it "Import Runs to ALM".
3. Add a form for entering ALM login information:



Add Run Form

Server URL

Domain

Project

Client Key

Client Secret

4. Navigate to Extensions -> App Script and add the attached code.

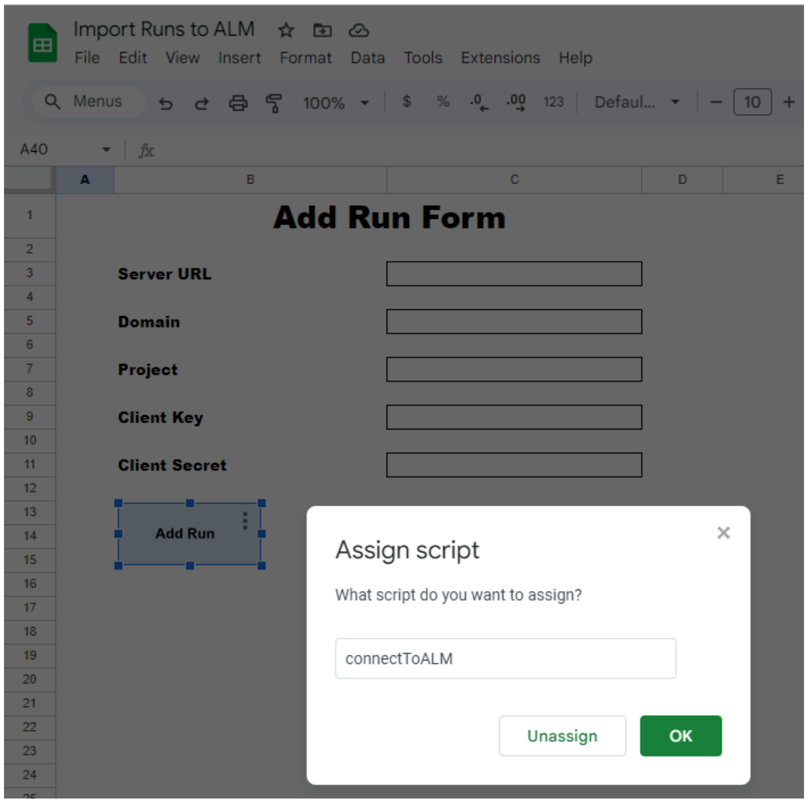
Example code:

```
//-----------------------------------------------------------------------
//Function invoked by the "Add Run" button to login, add run, and logout
//-----------------------------------------------------------------------
function connectToALM() {
  const addrunproj = SpreadsheetApp.getActiveSpreadsheet();
  const formWS = addrunproj.getSheetByName("LoginForm");
  const almURL = formWS.getRange("C3").getValue();
  const domain = formWS.getRange("C5").getValue();
  const project = formWS.getRange("C7").getValue();
  const clientKey = formWS.getRange("C9").getValue();
  const clientSecret = formWS.getRange("C11").getValue();

  var almLoginURL = almURL + "/rest/oauth2/login";
  var almLogoutURL = almURL + "/rest/authentication-point/logout";
  //var ui = SpreadsheetApp.getUi();
  //ui.alert("Server:" + almURL)

  // OAuth2 token request body
  var requestBody = {
    "clientId": clientKey,
    "secret": clientSecret
  };

  // Make a POST request to ALM QC OAuth2 endpoint
  var options = {
    method: "POST",
    contentType: "application/json",
    payload: JSON.stringify(requestBody),
  };

  //Login
  try {
    Logger.log("Logging into ALM.")
    Logger.log("Server: " + almURL)
    var response = UrlFetchApp.fetch(almLoginURL, options);

    if (response.getResponseCode() === 200) {

      Logger.log("Login successful.")

      //Get the cookies
      var oathToken = String(response.getAllHeaders()['Set-Cookie']);
      Logger.log("OathToken:"  + oathToken);

      var xsrfTokenStr = getSRFToken(oathToken);
      Logger.log(xsrfTokenStr);
      if (oathToken !== "" && xsrfTokenStr !== "")
      {

        Logger.log("Login successful.  Adding run data.");

        //Add a run

        var oathTokenFiltered = extractDesiredHeaders(oathToken);
        addRun(almURL,domain,project,oathTokenFiltered,xsrfTokenStr)



        //Logout
        Logger.log("Logging out.")
        options = {
          method: "POST",
          contentType: "application/json",
          headers: {
          "Cookie": oathToken,
          "X-XSRF-TOKEN":  xsrfTokenStr
          }
        };
        response = UrlFetchApp.fetch(almLogoutURL, options);
      }
    }
    else {
      // Handle error - log status code and response content
      Logger.log("Error: " + response.getResponseCode() );
    }
  } catch (error) {
    // Log any exceptions that occurred during the request
    Logger.log("Login Error- " + error );
  }
}
//-----------------------------------------------------------------
//Function to retrieve the session headers
//-----------------------------------------------------------------
function extractDesiredHeaders(cookieString) {
    let cookiesArray = cookieString.split(',');
    let extractedHeaders = '';

    cookiesArray.forEach(cookie => {
        let attributes = cookie.split(';');

        attributes.forEach(attribute => {
            let [name, value] = attribute.trim().split('=');
            if (name === 'JSESSIONID' || name === 'LWSSO_COOKIE_KEY' || name === 'QCSession' || name === 'ALM_USER' ||
name === 'XSRF-TOKEN') {
                extractedHeaders += `${name}=${value}; `;
            }
        });
    });

    // Remove the trailing '; ' if it exists
    if (extractedHeaders.endsWith('; ')) {
        extractedHeaders = extractedHeaders.slice(0, -2);
    }

    return extractedHeaders;
}

//-----------------------------------------------------------------
//Function to add run to test instance
//-----------------------------------------------------------------
function addRun(almURL,domain,project,oathtoken,xsrfTokenStr)
 {
      try {
        //Add Run
      var addRunURL = almURL + "/rest/domains/" + domain + "/projects/" + project + "/runs"
      Logger.log("Run request: " + addRunURL);
      Logger.log("Run oathtoken: " + String(oathtoken));

      //Entity Type=run, test id= 1, test set id= 1, run name=Run_4, Run Status=Passed,
      //Test Type=MANUAL, Tester=saas_tphan,test  instance id=3, duration=6
      var runXML = String(`<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <Entity Type="run"><Fields>
        <Field Name="test-id"><Value>1</Value></Field>
        <Field Name="cycle-id"><Value>1</Value></Field>
        <Field Name="name"><Value>Run_4</Value></Field>
        <Field Name="status"><Value>Passed</Value></Field>
      <Field Name="subtype-id"><Value>hp.qc.run.MANUAL</Value></Field>
        <Field Name="owner"><Value>saas_tphan</Value></Field>
        <Field Name="testcycl-id"><Value>3</Value></Field>
        <Field Name="duration"><Value>6</Value></Field>
        </Fields></Entity>`);


      var runOptions = {
        method: "POST",
        headers: {
        "Content-Type": "application/xml",
        "Accept": "application/xml",
        "Cookie": oathtoken,
        "X-XSRF-TOKEN":  xsrfTokenStr
        },
        payload: runXML,
        muteHttpExceptions: true
      };
       Logger.log("Run body: " + runXML);


      var response = UrlFetchApp.fetch(addRunURL, runOptions);

      Logger.log("Run request response:"  + response.getResponseCode());

      if (response.getResponseCode() === 201) {
        Logger.log("Run added.");
      }
      else{
         Logger.log("Failed to add Run.");
         Logger.log("Run request response:"  + response.getContentText());
      }
    } catch (error){
      // Log any exceptions that occurred during the request
      Logger.log("Add Run Error- " + error );
    }
}
//------------------------------------------------------------------
//Function to retrieve SRFToken
//------------------------------------------------------------------
function getSRFToken(oathToken) {
    try {
        var oathTokenStr = String(oathToken);
        var positionSRFToken = oathTokenStr.indexOf("XSRF-TOKEN=") + "XSRF-TOKEN=".length;
        var tokenstr = oathTokenStr.substring(positionSRFToken);
        var positionSRFTokenEnd = tokenstr.indexOf(";");
        tokenstr = tokenstr.substring(0, positionSRFTokenEnd);
        return tokenstr;
    } catch (error) {
        console.error("Error in getSRFToken: " + error.message);
        return "";
    }
}
```

5. Add a button and link it to the “connectToALM” function in the App Script.

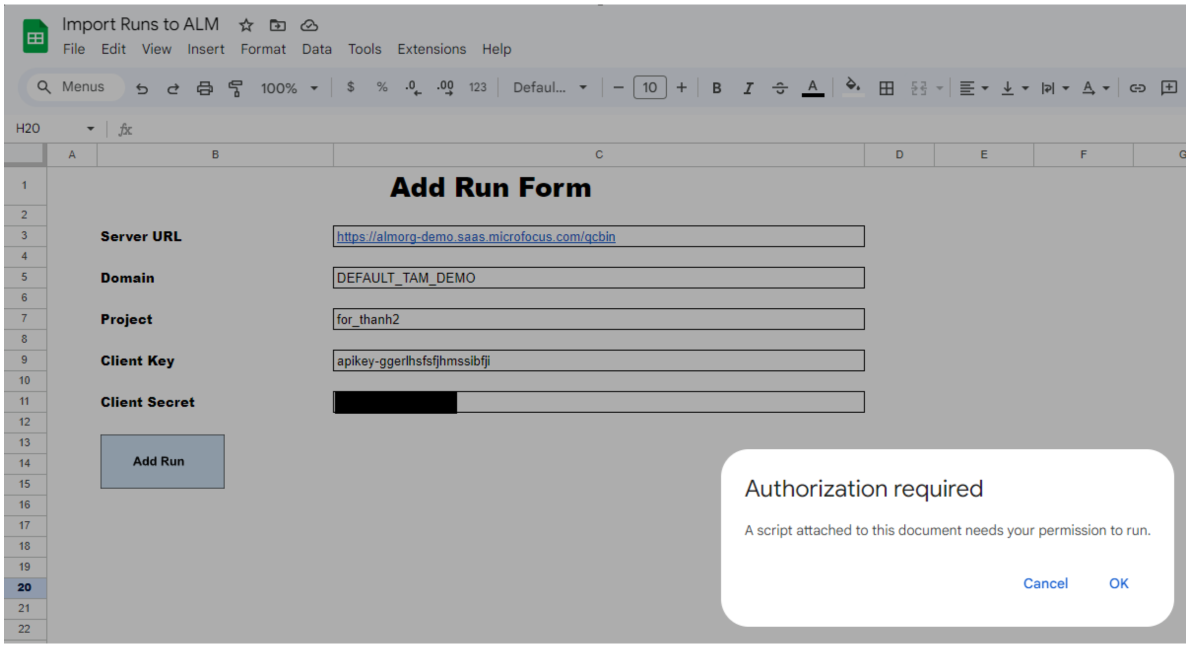
- **Open your Google Sheets document**.
- Click on `Insert` in the menu at the top.
- Choose `Drawing` and then `+ New`.

- In the drawing dialog box, click on `Shapes` and select the shape you want for your button (for example, a rectangle).
- Draw the shape for your button on the canvas that appears.
- Click on the Save form as "LoginForm".
- Click on the newly created shape/button in your sheet.
- You should see a small toolbar appear near the shape. Click on the `Assign script` icon (looks like a pencil).
- A dialog box will appear where you can enter the name of the function you want to run when the button is clicked. Enter the name of your Google Apps Script function (e.g., `connectToALM`) and click `OK`.



- Now, whenever you click the button in your Google Sheets document, it will execute the function `connectToALM` in your Google Apps Script project.

6. Execute the form.



Add Run Form

Server URL	https://almorg-demo.saas.microfocus.com/qcbin
Domain	DEFAULT_TAM_DEMO
Project	for_thanh2
Client Key	apikey-ggerlhsfsfjhmssibfji
Client Secret	[redacted]

Add Run

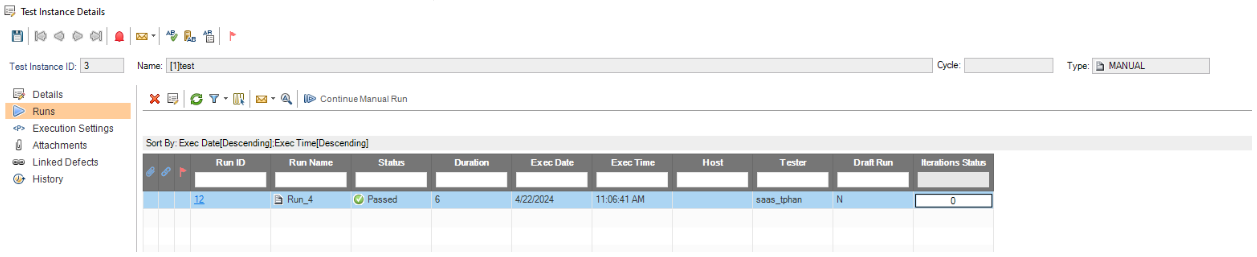
Authorization required

A script attached to this document needs your permission to run.

Cancel OK

Notes:
You will be prompted for authorization. Click on OK and follow the instructions.

7. A new run is created in the ALM Project.



Test Instance ID: 3 Name: [1]test Cycle: Type: MANUAL

Sort By: Exec Date[Descending];Exec Time[Descending]

Run ID	Run Name	Status	Duration	Exec Date	Exec Time	Host	Tester	Draft Run	Iterations Status
12	Run_4	Passed	6	4/22/2024	11:06:41 AM		saas_tphan	N	0