Use Webhook Action to Prepopulate Questions

Scripting Solutions

Additional scripting solutions will be added in the future. Please reach out to Alchemer with comments and suggestions on solutions you'd like to see via the link here.

Scripting and Other Custom Solutions

We're always happy to help you debug any documented script that is used as is. That said, we do not have the resources to write scripts on demand or to debug a customized script.

If you have customization ideas that you haven't figured out how to tackle, we're happy to be a sounding board for Alchemer features and functionality ideas that might meet your needs. Beyond this, check out our Professional Services; these folks have the scripting chops to help you to achieve what you are looking for!

Goal

Pre-populate Hidden Value Actions from a Webhook Action where the call returns JSON. While the Webhook Action has the option to "Use it to pre-populate the following questions" it doesn't recognize JSON, a popular format returned by APIs.

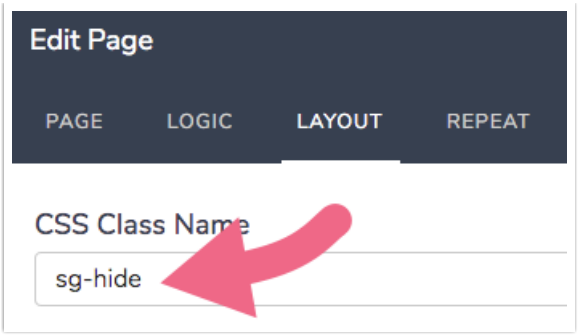
Effort: ✔ ✔ ✔ (a Javascript developer is necessary modify the Javascript for the specific data structure returned by the API called by the Webhook Action, however, this is typically a simple process)

Solution

Alchemer users develop an example using a free API for getting sunrise/sunset information (https://sunrise-sunset.org/api) and prepopulate a few Hidden Value Actions.

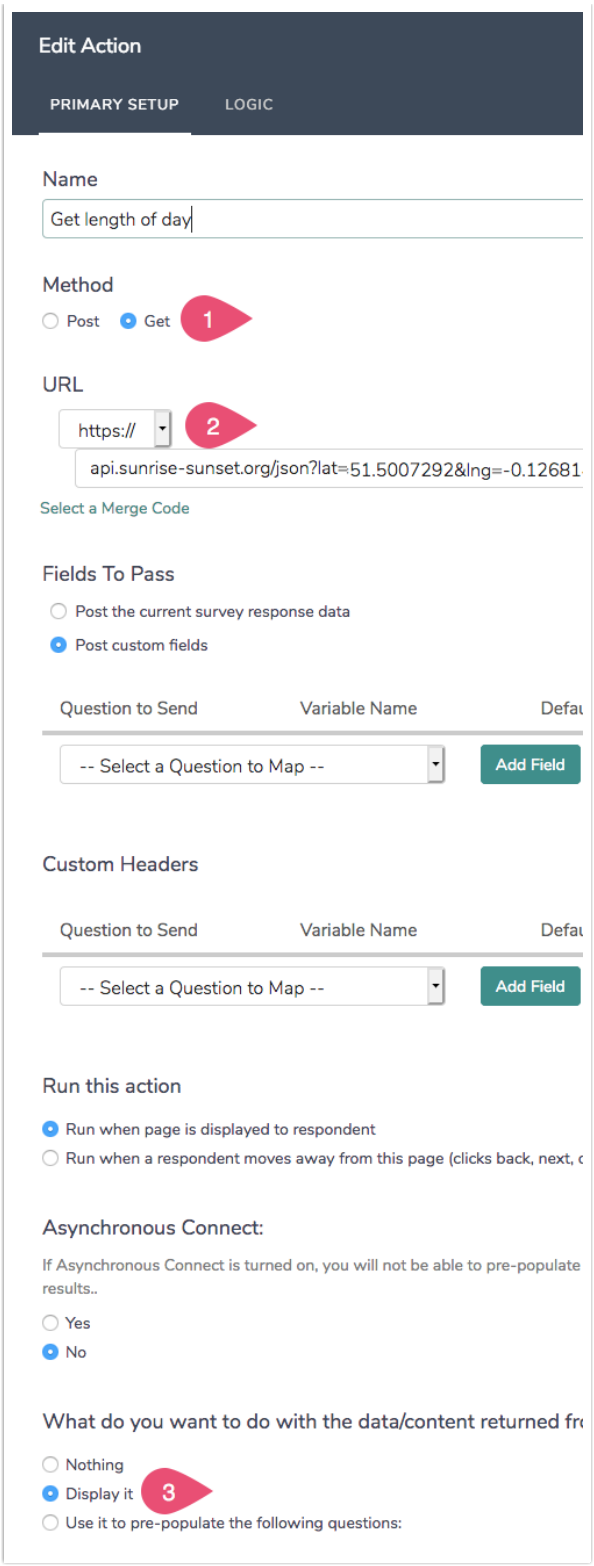
Step 1: Add Webhook Page

1. Add a new page
2. Set the page **Layout > Class Name** to **sg-hide to hide the page**



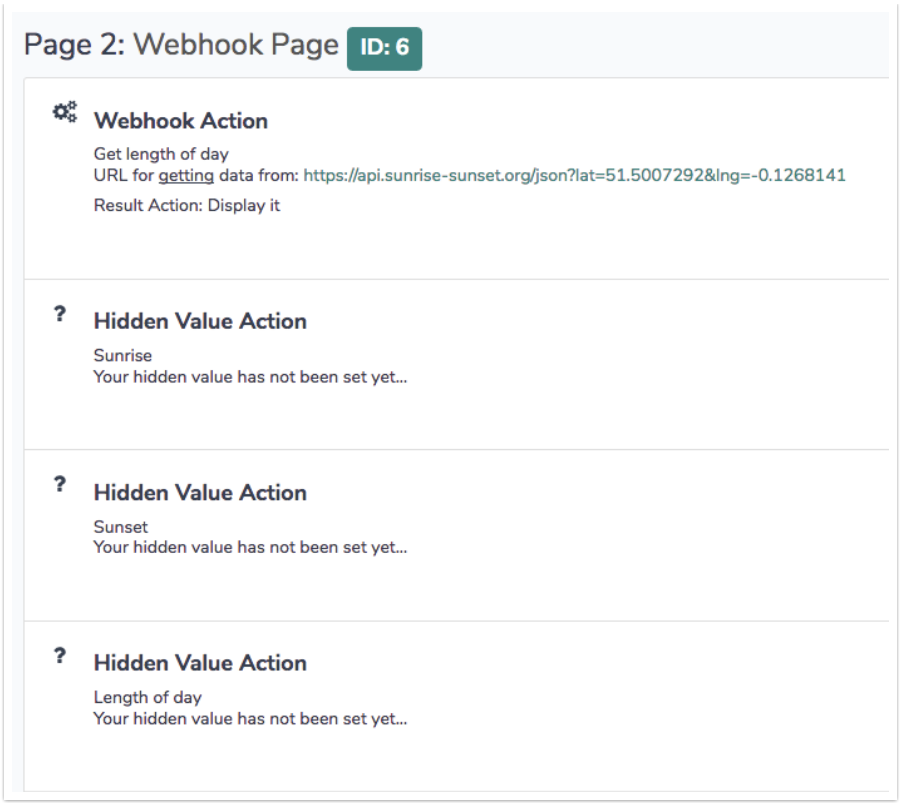
2. Add a **Webhook Action** to the page where:

   (1) The **Method** Get requests data from the API
   (2) The URL calls the API, this this example we'll get the data for London:
   https://api.sunrise-sunset.org/json?lat=51.5007292&lng=-0.1268141
   (3) The returned data is displayed so the Javascript can access it



Step 2: Add Hidden Values to Prepopulate

Add Hidden Values to the Webhook page to save the information from the API call. In this example, save sunrise, sunset, and length of the day. The Webhook page now looks like this:



Step 3: Add Javascript

Add the Javascript Action at the end of this article to the Webhook page to interpret the JSON data returned by the webhook call. In our example the webhook returns JSON like this:

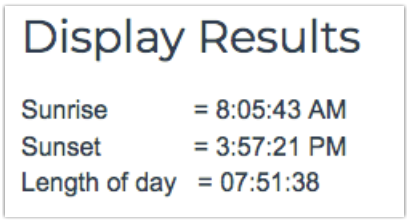
```
{
  "status": "OK",
  "results": {
    "sunrise": "8:05:30 AM",
    "sunset": "3:56:35 PM",
    "solar_noon": "12:01:02 PM",
    "day_length": "07:51:05",
    "civil_twilight_begin": "7:25:25 AM",
    "civil_twilight_end": "4:36:39 PM",
    "nautical_twilight_begin": "6:42:18 AM",
    "nautical_twilight_end": "5:19:47 PM",
    "astronomical_twilight_begin": "6:01:32 AM",
    "astronomical_twilight_end": "6:00:33 PM"
  }
}
```

Our Javascript will confirm the Status is OK then save the sunrise, sunset, and day_length properties to the Hidden Value Actions we added to the Webhook Page.

A Javascript developer can adjust the code in getWebhookData() to match the structure of your JSON.

Step 4: Make use of the data

For this example, we'll display the three Hidden Value Actions on the next page.



Javascript code:

```
/* Alchemer v01

   Prepopulate HVas with Webhook JSON response.

   Documentation and updates: https://confessions.knowledgeowl.com/help/webhook-action-parse-json-returned-from-webhook

   This example code expects the JSON to be in the form:
     {
       "status": "OK",
       "results": {
         "sunrise": "8:05:30 AM",
         "sunset": "3:56:35 PM",
         "solar_noon": "12:01:02 PM",
         "day_length": "07:51:05",
         "civil_twilight_begin": "7:25:25 AM",
         "civil_twilight_end": "4:36:39 PM",
         "nautical_twilight_begin": "6:42:18 AM",
         "nautical_twilight_end": "5:19:47 PM",
         "astronomical_twilight_begin": "6:01:32 AM",
         "astronomical_twilight_end": "6:00:33 PM"
       }
     }
*/

document.addEventListener("DOMContentLoaded", function() {

  const PREPPOP_MAPPINGS = [
    { jsonKey: 'sunrise',    qid: 15 },
    { jsonKey: 'sunset',     qid: 16 },
    { jsonKey: 'day_length', qid: 17 },
  ]

  // ******************************
  // *** no changes needed below ***
  // ******************************

  const LOG = false

  /***
   * Test boolean value, alert() and throw Error if it's false
   *
   * bool (t/f) value to test
   * msg (string) message to alert and throw in new Error
   */
  const assert = (bool, msg) => {
    msg = "Javascript Assert Error: " + msg
    if (!bool) {
```

```
    alert(msg)
    console.error(msg)
    const err = new Error(msg)
    console.error(err)
    throw err
    }
}

/***
 * Get an element based on its Question ID
 *
 * qid {int/string} question ID
 * section = "element" {string} the final section of the element id
 * return {element} looks for id's in the form: "sgE-1234567-12-123-element"
 */
const getElemByQid = (qid, section = "element") => {
  const id = "sgE-" + SGAPI.survey.surveyObject.id + "-" + SGAPI.survey.pageId + "-" + qid + "-" + section
  const elem = document.getElementById(id)
  assert(elem, "Javascript: can't find element with id = " + id + ", section = " + section)
  return elem
}

/***
 * Get the data object from the JSON returned by the webhook
 *
 * return (object) The 'results' object built from the JSON
 */
const getWebhookData = () =>  {

  const raw = document.querySelector('.sg-http-content').innerText
  if (LOG) console.log("raw = ", raw)
  if (!raw || raw[0] !== '{') {
    console.warn("The Webhook call didn't return JSON")
    return {}
  }

  const jsonObj = JSON.parse(raw)
  if (LOG) console.log("webhook jsonObj = ", jsonObj)
  if (jsonObj['status'] !=="OK") {
    console.error("Webhook status = ", jsonObj['status'])
    return {}
  }

  return jsonObj['results']
}

/***
 * Hide the JSON returned from the Webhook call in case we're displaying data on
   this page.
 */
const hideWebhookJSON = () => {
   document.querySelector('.sg-http-content').classList.add('sg-hide')
 }

/***
 * Prepopulate HVA's with JSON data
 *
 * prepopMappings (array of objects) mapping of jsonKey to prepop into QID
 * data (object) Data object parsed from JSON returned by the Webhook
 */
const prepop = (prepopMappings, data) => {
  prepopMappings.forEach(prepopMapping => {
    const val = data[prepopMapping.jsonKey]
    getElemByQid(prepopMapping.qid).value = (val !== undefined ? val : "")
```

```
    })
    console.log("done with prepop")
  }

  /***
   * Check if this page is being shown because user clicked the Back button on the following
   * page.  If so, keep moving back.
   *
   * return (t/f) true if moving back, when true caller should do nothing more
   */
  const movingBack = () => {
    var isMovingBack = SGAPI.surveyData[Object.keys(SGAPI.surveyData)[0]].page_direction === -2
    if (!isMovingBack)
      return false

    document.querySelector("#sg_BackButton").click()
    return true
  }

  /***
   * main()
   */

  // Propagate moving Back
  if (movingBack())
    return

  hideWebhookJSON()

  // save data from WebHook response to elements on page
  const data = getWebhookData()
  prepop(PREPPOP_MAPPINGS, data)

  // Click Next or Submit to move to next page, leading semi-colon is required
  ;(document.querySelector("#sg_NextButton") || document.querySelector("#sg_SubmitButton")).click()
})
```

Related Articles