Current Date and Time

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

Save the current date and/or time in a specific timezone for a response being collected in Alchemer.

Effort: ✔ ✔ ✔

Solution

The current date and time the page is presented will be saved to one or more Hidden Value Actions. The title of the Hidden Value Action determines what will be saved. An additional Hidden Value Action contains a setup string to choose the timezone and additional settings.

Step 1: Add elements to save the date/time values

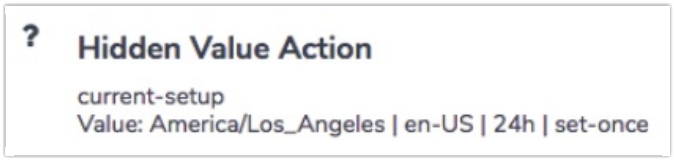
Add one or more **Hidden Value Actions** with the following titles to save the current date or time in various formats. The title of the Hidden Value Action determines what will be saved in that Action:

Hidden Value Action Title	Example value	Description
current-date	12/09/2020	Save the date in the format defined below
current-time	14:47:52	Save the time in the format defined below
current-hour	16	The current hour, 0-23

current-day-of-week	T	The current day of the week: UMTWRFS
current-timestamp	12/09/2020 14:47:52	Save both the date and time
current-filterable-timestamp	20201209144752	Save a format that can be used in Alchmer reports to filter for a range of dates and times using <= and >= comparisons
current-iso-timestamp	2020-12-09T15:47:52.248Z	Save timestamp in the ISO format
current-milliseconds	1607528920248	Save current time in milliseconds since 01-01-1970, useful for Javascript
current-day-of-week	Wed	Save current day of week in short format specific to the Locale in the setup in Step 2
current-hour	14	Save current hour, 1-23

Step 2: Set the format to save the date/time

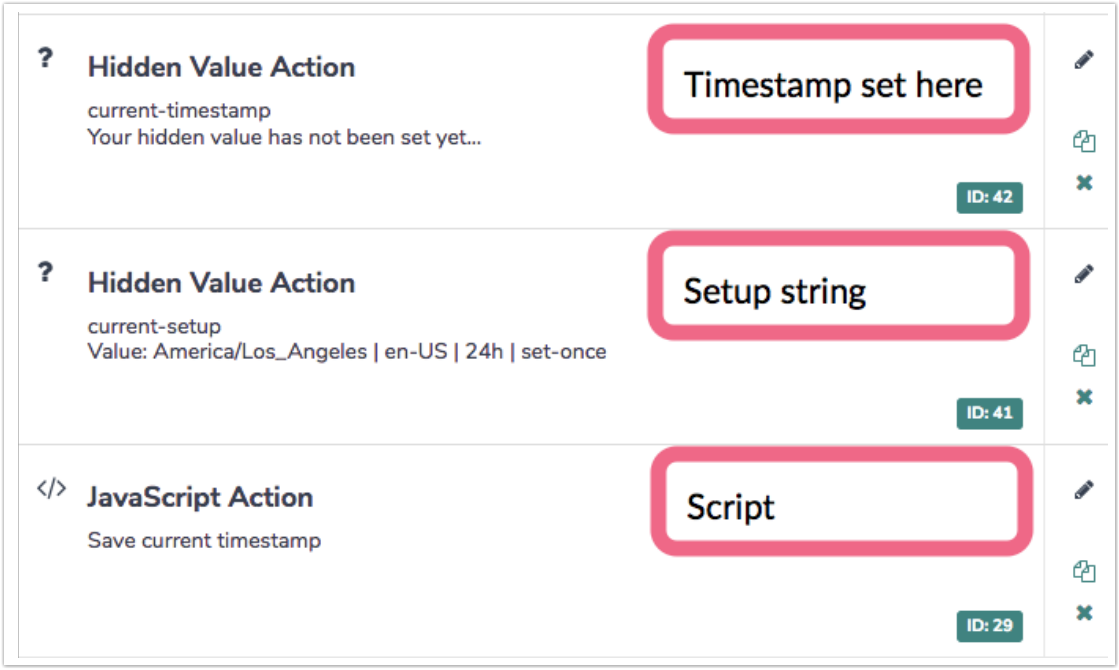
Add a **Hidden Value Action** with the title current-setup to define the timezone and basic format. Enter your setup string in the **"Populate with the following"** field separating entries with vertical bars, for example: America/Los_Angeles | en-US | 24h | set-once



Example setting	All Settings
America/Los_Angeles	Worldwide timezone, see: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
en-US	The locale (date format, like EU dd/mm vs US mm/dd), see: https://en.wikipedia.org/wiki/Language_localisation **Note:** en-CA format is yyyy/mm/dd
24h	12h for am/pm 24h for EU / US military time

set-once	set-once to only set the first time the page is displayed set-always to reset each time the page is displayed.

Example page:



Step 3: Add a new Javascript Action

Paste the below script into a new Javascript Action:

```
/* Alchemer, v02

   Save the current date and time as a specific timezone

   Documentation and updates: https://help.alchemer.com/help/current-date-and-time
*/

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

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

  // Hidden Value Action titles to set (if they exist on the page)
  const DATE_TITLE              = 'current-date'
  const TIME_TITLE              = 'current-time'
  const TIMESTAMP_TITLE         = 'current-timestamp'
  const FILTERABLE_TIMESTAMP_TITLE = 'current-filterable-timestamp'
  const ISO_TIMESTAMP_TITLE     = 'current-iso-timestamp'
  const MILLISECONDS_TITLE      = 'current-milliseconds'
  const DOW_TITLE               = 'current-day-of-week'
  const HOUR_TITLE              = 'current-hour'

  const SETUP_TITLE             = 'current-setup'
```

```
  /***
  * 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 Error:\n\n" + 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
  }

  /***
  * Parse the Hidden Value titled SETUP_TITLE
  *
  * (return) setup object, example:
  *       {
  *         timezone: "America/Los_Angeles",
  *         locale:   "en-US",
  *         hours:    "24h",
  *         set:      "set-once"
  *       }
  */
  const parseSetup = () => {
    const qid = getQidByTitle(SETUP_TITLE)
    assert(qid, `Missing element with title '${SETUP_TITLE}'`)
    const setupString = getElemByQid(qid).value

    const aParsed = setupString.match(/(.+)\|(.+)\|(.+)\|(.+)/)

    console.log("aParsed = \n", aParsed)
    assert(aParsed && aParsed.length === 5, "The current-setup string is not in expected format:\n\n'America/Los_
Angeles | en-US | 24h | set-once'")

    const setup = {
      timezone: aParsed[1].trim(),
      locale:   aParsed[2].trim(),
      hours:    aParsed[3].trim(),
      set:      aParsed[4].trim()
    }

    assert(setup.hours === "12h" || setup.hours === "24h", `current-setup hours must be '12h' or '24h', not: ${setu
p.hours}`)
    assert(setup.set === "set-once" || setup.set === "set-always", `current-setup must be 'set-once' or 'set-always'
, not: ${setup.set}`)
```

```
  return setup
 }

 /***
  * get the the date settings used by Intl.DateTimeFormat()
  *
  * setupObj (obj) see parseSetup() for a definition
  * return (obj) date settings used by Intl.DateTimeFormat
  */
 const getDateSettings = (setupObj) => {
  return {
   timeZone: setupObj.timezone,
   month:    "2-digit",
   day:      "2-digit",
   year:     "numeric" // 4-digit year
  }
 }

 /***
  * get the the time settings used by Intl.DateTimeFormat()
  *
  * setupObj (obj) see parseSetup() for a definition
  * return (obj) date settings used by Intl.DateTimeFormat
  */
 const getTimeSettings = (setupObj) => {
  return {
   timeZone: setupObj.timezone,
   hour:     "2-digit",
   minute:   "2-digit",
   second:   "2-digit",
   hour12:   setupObj.hours === '12h'
  }
 }

 /* ----------------------------------------
    Helper: Get a SurveyGizmo element on the page.
          Ex: In survey 1234567 on page ID 12 the call getSgId(123, "element")
             returns HTML element for "sgE-1234567-12-123-element"
    ---------------------------------------- */
 function getSgElemById(qid, oid = "element") {
  let surveyInfo = SGAPI.surveyData[Object.keys(SGAPI.surveyData)[0]]
  let id = "sgE-" + surveyInfo.id + "-" + surveyInfo.currentpage + "-" + qid + "-" + oid
  let elem = document.getElementById(id)
  if (!elem)
   alert("Javascript error: can't find element with id = " + id)
  return elem
 }

 /***
  * getDate()
  *
  * setupObj (obj) see parseSetup() for a definition
  * dt (Date)
  * (return) Date string of current time using settings above
  */
 const getDate = (setupObj, dt) => new Intl.DateTimeFormat(setupObj.locale, getDateSettings(setupObj)).format
(dt)

 /***
  * getTime()
  *
  * setupObj (obj) see parseSetup() for a definition
  * dt (Date)
  * (return) Time string of current time using settings above
  */
```

```
const getTime = (setupObj, dt) => {
  let s = new Intl.DateTimeFormat(setupObj.locale, getTimeSettings(setupObj)).format(dt)
  s = s.replace('24:', '00:') // fixup some LOCALE will use 24 for 12AM times
  return s
}

/***
 * getHour()
 *
 * setupObj (obj) see parseSetup() for a definition
 * dt (Date)
 * return (string) Hour, no leading zero: 1-23
 */
const getHour = (setupObj, dt) => {
  const options = {
    timeZone: setupObj.timezone,
    hour:     "numeric",
    hour12:   false
  }
  let s = new Intl.DateTimeFormat(setupObj.locale, options).format(dt)
  if (s === '24')
    s = '23'
  return s
}

/***
 * getDow()
 *
 * setupObj (obj) see parseSetup() for a definition
 * dt (Date)
 * return (string) short version of weekday name specific to setupObj.locale
 */
const getDow = (setupObj, dt) => {
 const options = {
   timeZone: setupObj.timezone,
   weekday: 'short'
 }
 return new Intl.DateTimeFormat(setupObj.locale, options).format(dt)
}

/***
 * Get a timestamp that Alchemer reports can filter on.  Due to a bug you can
 * only filter using >= or <= on a pure number so 1pm Dec 1 2022 needs to be
 * recorded as: 20221201130000
 *
 * setupObj (obj) see parseSetup() for a definition
 * dt (Date)
 * return (string) yyyymmmddhhmmss: "20221201130000"
 */
const getFilterableTimestamp = (setupObj, dt) => {
  const dateSettings = {
    timeZone: setupObj.timezone,
    month:    "2-digit",
    day:      "2-digit",
    year:     "numeric" // 4-digit year
  }
  const timeSettings = {
    timeZone: setupObj.timezone,
    hour:     "2-digit",
    minute:   "2-digit",
    second:   "2-digit",
    hour12:   false, // false for 24-hr style
  }
  const dtString = new Intl.DateTimeFormat("en-CA", dateSettings).format(dt)
  const tmString = new Intl.DateTimeFormat("en-CA", timeSettings).format(dt)
```

```
  return (dtString + tmString).replace(/\D/g,'') // strip non-numeric
 }

 /***
  * Get the Question ID based on the question / hidden value title
  *
  * title {string} question title
  * return {int/null} the question ID or null if title not found
  */
 const getQidByTitle = (title) => {
  const questionsObj = SGAPI.survey.surveyObject.questions
  const qid = Object.keys(questionsObj).find(key =>
   questionsObj[key].title.toLowerCase() === title.toLowerCase())
  return qid || null
 }

 /***
  * Set the value of qid if it exists and isn't already filled in
  *   when SET_ONLY_ONCE flag is set
  *
  * setupObj (obj) see parseSetup() for the object definition
  * title (string) title for the hidden value action that may be on the page
  * value (string) value to set
  */
 const set = (title, setupObj, value) => {
  const qid = getQidByTitle(title)
  if (qid) {
   // Only set once?  Then don't set again.
   if (setupObj.set === 'set-once' && (getSgElemById(qid).value && getSgElemById(qid).value !== 'not-set'))
    return
   getSgElemById(qid).value = value
  }
 }

 /***
  * main()
  */
 const browserDate = new Date()
 const setupObj = parseSetup()

 set(DATE_TITLE,              setupObj, getDate(setupObj, browserDate))
 set(TIME_TITLE,              setupObj, getTime(setupObj, browserDate))
 set(TIMESTAMP_TITLE,         setupObj, getDate(setupObj, browserDate) + ' ' + getTime(setupObj, browserDate
))
 set(FILTERABLE_TIMESTAMP_TITLE,setupObj, getFilterableTimestamp(setupObj, browserDate))
 set(ISO_TIMESTAMP_TITLE,     setupObj, browserDate.toISOString())
 set(MILLISECONDS_TITLE,      setupObj, browserDate.getTime()) // getTime() is based on UTC and timezone ind
ependent
 set(DOW_TITLE,               setupObj, getDow(setupObj, browserDate))
 set(HOUR_TITLE,              setupObj, getHour(setupObj, browserDate))
})
```

Related Articles