Implementing ExperienceSampler Google Tools

Data Collector

Google option. For the Google option, researchers will need to go to Google Drive and create three spreadsheets in Google Sheets. One sheet will be used to store the raw data (`database`), one sheet will be used to store spliced data (`splicedDataset`), which will then be used to check compliance, and one sheet will be used to store compliance rates (`compliance`). For each spreadsheet, note the ID of the sheet. The sheet id is used to tell the web app where to write the data and is found in the URL for the Google sheet. It is a long string of letters, numbers, and symbols that appears before "/edit#gid=…" Researchers can also create a Google Document that will act as a log of the web app's activities.

To create your own data collector web app, go to Google Script (http://www.google.com/script/start/) and select a Blank Project in the top left of the menu of the pop-up dialog. You can copy and paste the script from this page (https://script.google.com/d/1P5dCtwPQxsXYFcN68sE8egkfyhKs0WDcXqFDXm3jactwxvbUXP8sLJlR/edit?usp=sharing). You will then replace the variables `databaseID`, `splicedDatabaseID`, `complianceID`, and `DocumentID`, in the public code with the ids you stored for the raw data spreadsheet, the spliced data spreadsheet, the compliance data spreadsheet, and the web app log, respectively. Save the script.

You are now ready the test and potentially deploy the data storage web app. Click on the debug icon, which looks like a bug and is located beside the dropdown menu titled **Select function**. The script will then ask for Authorization. Select **Continue** and then **Accept** to authorize the web app. To deploy as a web app, a project version needs to be created. Go to **Publish** then **Deploy as web app**. In the dropdown menu for **Who has access to the app**, select

Anyone, even anonymous. Then click **Deploy**. Copy the URL that appears below **Current web app URL**. You need this URL to input into your ExperienceSampler code so that the app knows where to send the data.

saveData functions. Next, researchers will need to tell the ExperienceSampler where to send the data by locating the `saveData` and `saveDataLastPage` functions in the JavaScript file and inputting the URL for the Google web app. In these functions, the only other property that you should double-check is the `type` property, which should be set to "`get`". The `type` property indicates how the data will be sent to the web app.

The only difference between the `saveData` and `saveDataLastPage` functions is that participants will be notified whether the data was sent to the server successfully when the `saveDataLastPage` function is executed but not when the `saveData` function is executed. These functions ensure that the data is being sent to the server as often as possible so that participant data files are as up-to-date as possible. This allows for the most accurate records of response compliance (i.e., using the `saveData` function) that can be checked daily using our automated compliance checking program but not annoy participants by notifying them every time their data has been sent successfully. We chose to only notify participants when they would naturally expect to receive such confirmation – when they have completed the survey. If you choose not to notify participants, then you only need to implement the `saveData` function; however, we would advise you to use the `saveDataLastPage` function during testing to ensure that your data is being sent and saved correctly.

ExperienceSampler HTML file. Once you have set up your server or your Google database, you will need to add the URL of your server or Google Data Collector Script into the index.html file in your ExperienceSampler file. This will let ExperienceSampler know that your

Data Collector has permission to access the data, and the Data Collector is not doing anything malicious. If you are using the Google option, you will need two copy two URLs into the HTML file. If you are using the Server option, you will only need to copy one URL into the HTML file. This is the only change you have to make to the HTML file.

First, open your index.html file in your www folder. You will see a line that contains Content-Security-Policy. In this line, you will add the additional property of `connect-src`, followed by the URL of your data collector. For the Google option, you will need to the same URL that you used in the `saveData` and `saveDataLastPage` functions. After your Google data collector URL, you will need to type `https://script.googleusercontent.com/macros/echo` with a space separating the two URLs and a double quotation mark after echo. Your entire Content-Security-Policy should be `<meta http-equiv="Content-Security-Policy" content="default-src 'self' 'unsafe-inline' 'unsafe-eval' data: gap: https://ssl.gstatic.com; style-src 'self' 'unsafe-inline'; media-src *; connect-src https://script.google.com/macros/s/AKfycbzzbp0437BkTqx95W9THF9JhWcydzn-K-FJTbwIHF23-S0JbDXG/exec https://script.googleusercontent.com/macros/echo">`. If you are using a server, you will simply need to input the URL of your server.

Ensuring Compliance

Once your participants start their experience sampling data collection period, you will want to ensure that they are completing their questionnaires. For example, if they are completing one nightly survey, you may want to check whether they are completing their surveys, so that you can remind them in the morning to complete it if they have forgotten. In contrast, if you are signaling participants multiple times a day, you may want participants to complete a certain number of surveys each day even though you signal them more often (e.g., you choose to signal participants six times each day, but only require them to complete 5 each day). Moreover, you may choose to compensate participants based on the number of questionnaires they complete. Thus, you will need some method of tracking your participants' compliance. In this section, we will demonstrate how to implement two tools that you can use to manage participants' compliance. The first tool checks will count the number of surveys participants have completed at the end of each day. The second tool will send the participant a reminder email if they forgot to complete their questionnaire or if they have completed fewer questionnaires than the required minimum. These tools, however, are currently only available for the Google data storage option because they both use Google Script and can only interact with Google Sheets.

Checking compliance. We have created a script using Google script to check participant's compliance at the end of each day. The script can be downloaded from this page (https://script.google.com/d/1I-Uo_phZEM94YYvjWMwZPi5hGG3V3YwBSm4yOabD4Q-t0bvntCiHNw-i/edit?usp=sharing). Below we will outline how to implement the compliance checker. This script checks each participant's data sheet and determines the number of entirely completed questionnaires at the end of each day. The script will then record this to the Compliance Google spreadsheet.

The script. Now, we will briefly describe what each of the functions within the script does so that you understand how the script works. The compliance checker script performs two primary functions. The first function splices the variable name into its three components: the unique key, the variable name, and the time stamp. The second function uses the unique key to determine how many questionnaires the participant has completed on a given day and record this value in the compliance spreadsheet along with a date stamp of when the responses were checked. This second function only includes completed questionnaires in this count. That is, if a participant does not complete a questionnaire, it is not included in the day's count. The script does this by looking a unique key that also has the tag "completed" appended to it. The `renderLastPage` function in ExperienceSampler only appends this tag to the unique key when the participant has completed the entire questionnaire. These functions are only executed during the participant's experience sampling data collection period. The compliance checker script does this by finding the dates in the compliance spreadsheet and then checking whether the day the script is being run falls between the experience sampling data collection period date ranges using the `startOfDataCollectionPeriod` and the `endOfDataCollectionPeriod` functions.

If your participants attend an in-lab information session as part of your experimental protocol, you may encourage your participants to try using ExperienceSampler during this session. This practice questionnaire should not count towards the compliance total; thus, you can implement an additional function that will exclude this practice intake data. This function, `intakeSessionTestData`, only checks the intake session data, and will only run in the time between the intake session and the first day of the experience sampling data collection period.

To have the script run automatically each day, we will use a feature of Google Scripts called triggers. A trigger tells the script to run if a specific condition has been met. The triggers can be either event-based (e.g., every time the spreadsheet is updated), or time-based (e.g., at a certain time each day). Because the splice data function can take a while to run depending on how much data has been collected on a given day, we would advise you to run this function one hour before the compliance checking (i.e., `checkCompletedSurveys`) function.

Implementing the script. First, you will need to create your compliance spreadsheet. This spreadsheet should have the following headers in this order: **Participant ID, Participant's Name, Participant Email, App Installation Day, First Day of Data Collection, Day After Last Day of Data Collection, Compliance 1, Compliance Check Date 1, Compliance 2, Compliance Check Date 2, … Compliance X, Compliance Check X**. If you wish to use our compliance email reminder tool as well, which we will describe in greater detail below, you should also add the following to the header: **Compliance1 Email Sent, Compliance2 Email Sent, … ComplianceX-1 Email Sent**. You can see an example of a compliance sheet on this page (https://docs.google.com/spreadsheets/d/16VZAKlW0thWRyWwwxvUrq2xfjSJHgWKTZN7oDq5gDvw/edit?usp=sharing). In both cases, **X** is the number of days of data collection.

Next, you will have to set up the script. First, download our Compliance Checker script. You will then copy this script into a new Google Script project. To do this, go to Google Script and create a blank project. Then copy our code into the script. In this code, you will have to make two changes. First, you will need to tell the script the Google Sheet ID of your compliance spreadsheet. Remember that the sheet id is found in the Google Sheet URL, and it is a long string of letters, numbers, and symbols that appears before "/edit#gid=…". You will also need to tell the script the Google Sheet ID of your spliced database.

Once you have made these changes to the script, you should save your script. We named our script Compliance Script. Then you will need to authorize the script to run. That is, you need to provide your script with permission to look at the data in your spreadsheet. To do this, you need to click on the debug icon, which looks like a bug and is located beside the dropdown menu in the toolbar. Next, we want our script to run automatically each day. To do this, we will have to set triggers. A trigger tells the script to run if a specific condition has been met. For example, you can tell the script to run every time the spreadsheet is updated or you can tell the script to run between two specific times every day. We will use a time-based trigger to our script. We will want to specify two time-based trigger. The first one will splice the long variable string into a unique key, a variable name, and a timestamp. The second one will count the number of completed questionnaires. To specify a time-based trigger, you first click on the clock icon that appears beside the play icon. Then click **Add a new trigger,** then select **spliceData** in the first dropdown menu and **Day timer** in the third menu, and **3am to 4am** in the last menu. Then you create your second trigger by selecting Select **checkCompletedSurveys** in the first dropdown menu, **Day timer** in the third dropdown menu, and **4am to 5am** in the last dropdown menu. We selected these times for our triggers because most participants will probably be sleeping and not be providing new data; however, you can select the most appropriate trigger times based on the population of participants they are studying. If you have chosen to use the `intakeSessionTestData` function, you will need a third trigger. You can run this function between **2am to 3am**.

Compliance email reminders. Once you have determined the number of questionnaires each participant has completed, you may want to remind participants who have completed fewer than acceptable. You can use our compliance emailer Google Script. You can copy and paste the script from this page (https://script.google.com/d/1L2rEefLaftrDnZRVMosO7cuW5dAiAbmLzuDOcjHD9iOpgAGVCugLI_A3/edit?usp=sharing). Below we will outline how to implement the compliance emailer.

This script checks will send an email to each participant if they do not meet a specific condition. This condition should be the number of questionnaires that participants should complete each day. The script will then record whether an email was sent (i.e., Email Sent) or whether compliance is met (i.e., Survey Completed) to the Compliance Google spreadsheet.

The compliance emailer consists of four sections. In the first section, you tell the script what each column in the Compliance Spreadsheet represents. For example, you will tell the script that the second column is the participant's name, `var name = column[1]`. It is important to note that the column numbers in this section use zero-based character numbering. Thus, you have to subtract 1 from the actual column number (e.g., column 2 in the spreadsheet is referred to as column 1 in this section of the emailer script).

The second part of the script determines whether the day that the script is being run falls between the experience sampling data collection period for a specific participant. It does this by counting the number of days since the first day of data collection. This value will be used to determine whether an email should be sent for a specific participant.

The third section specifies the conditions that need to be met in order to send a reminder email. The condition of this if statement states that if the day the script is being executed falls within the data collection date range, and the participant has completed fewer than a certain number of questionnaires, and an email has not been sent, and a survey completed has not been posted to the spreadsheet, the compliance emailer should send an email to the participant. Because each day of the experience sampling period has its own compliance column and own email column, you will need to specify this condition multiple times, from the second day of the data collection period to the last day of the data collection period. You would not want to remind them the day the experience sampling period has ended because participants do not need to complete questionnaires on that day. If you are using ExperienceSampler for a daily diary study, you may want the script to remind participants to complete the questionnaire up until the day after the end of the data collection period.

The final section specifies the conditions when the script should note that all the questionnaires have been completed. It is very similar to the third section, except instead of specifying that the participant has completed an insufficient number of questionnaires, you would specify that participants have completed the acceptable number of questionnaires or more. The action to be executed if this condition is met is to indicate that all questionnaires have been sent in the Email Sent column for that specific day. We programmed the script to do this so that it does not continuously email participants when they have completed the questionnaire. It also prevents the script from writing the status of the email in the wrong column.

Implementing the script. Setting up the compliance emailer is very similar to setting up the compliance checker script. The only difference is that you have to paste in a different script and specify more values because there will be more variability in the emailer. For example, different researchers will have different acceptable numbers of questionnaires, and different researchers will collect data for different amounts of time.

First, download our Compliance Emailer script. You will then copy this script into a new Google Script project. To do this, go to Google Script and create a blank project. Then copy our code into the script.

In this code, you will have to make several changes. First, you will need to tell the script the Google Sheet ID of your compliance spreadsheet. Remember that the sheet id is found in the Google Sheet URL, and it is a long string of letters, numbers, and symbols that appears before

"/edit#gid=…". You will also need to tell the script the Google Sheet ID of your spliced database.

Now you will tell the script what each column in the Compliance Spreadsheet represents. This section appears after `var column = data[i]` line that appears inside the `if (i>0)` statement. Remember that in this section, we use zero-based character numbering, so column 1 in the spreadsheet is referred to as column[0] in the script. You will need to declare a new variable for each column in your Compliance spreadsheet (e.g., var name = column[1]).

At the end of the second section, you will need to customize one line. This line determines which column the script should note the status of the compliance email, var `complianceEmailColumn`. To calculate the compliance email column number, you determine the zero-based character number of the first email compliance column (e.g., column titled Compliance 1 Email Sent), and then subtract the `daysFromTheFirstDataDay` (i.e., days that have passed since the first day of the participant's experience sampling data collection period). For example, your first compliance email column may have a zero-based character number of 28, so your compliance email column statement would be `var complianceEmailColumn = 28 - daysFromFirstDataDay`.

In the third section, you will specify the conditions that must be met to send a compliance reminder email. For our example, we will use an experience sampling design that notifies participants six times a day, but we only expect participants to complete 5 questionnaires each day for 7 days. We have six unique sets of conditions that can be met in order for an email to be sent. Because these conditions are for each day in the experience sampling data collection period, they do not all have to be met in order for the email to be sent. Instead, if any one of these sets of conditions is- met, an email should be sent. Thus, we will join each set of conditions with an “or” logical operator (i.e., `||`).

Each set of conditions will have five components. The first component is how many days have passed since the first day of the experience sampling period. This value will always be a negative number because we are subtracting a larger number from a smaller number. Thus, the day after the first day of the experience sampling period will have a value of -1. The second component is how many questionnaires have been completed. This value should be set to be less than the minimum number of questionnaires that should be completed. In our example, we call this variable `c1Met`, and we will set it to be less than 5, `c1Met < 5`. The third component states that c1Met must not be blank, `c1Met !== “”`. This will prevent the script from emailing participants if for some reason the compliance checker did not run. The fourth component states that the compliance email column for the first day’s responses, which we will call `c1Email`, cannot be equal to the email sent value, `c1Email !== sent`. The final component states that the value in the compliance email column for day 1 should also not equal Survey Completed, `c1Email !== “Survey Completed”`. This ensures that the emailer not email the participant if they have completed all their survey’s for the previous day. Put all together, your set of conditions should look like this `(daysFromFirstDataDay === -1 && c1Met == 0 && c1Met !== "" && c1Email !== sent && c1Email !== "Survey Completed")`. Then you repeat this for two days after the first day of the experience sampling period, `|| (daysFromFirstDataDay === -2 && c2Met == 0 && c2Met !== "" && c2Email !== sent && c2Email !== "Survey Completed")`, and so forth until you reach the value of 6, which is equal to the last day of the experience sampling period. `if` statement.

Inside the `if` statement of the third section, you will specify the properties of your email message. First, we will nest another if statement inside the third section that states that an email will only be sent if the participant’s email, which we will call `emailAddress`, is not blank, `if (emailAddress !== “”)`.You will then need to write your form email message in this section, which we will assign to the variable message. To write your message, you will need to place it inside quotation marks. You can also insert your participant’s name into the form email by placing the variable associated with the participant’s name in your message (e.g., `var message = “Hello ” + name + “, ”`). To create new lines in the body of your email message, you will use `\n`. You will also notice that there is a + sign between each set of quotation marks. This let’s the script know that these sets of quotations are supposed to appear in the message together. The you will specify the subject your of your form email, e.g., `var subject = “Yesterday’s Smartphone App Surveys”`. Then you will tell the app to send your email to your participant, `MailApp, sendEmail(emailAddress, subject, message, {name: “Study Name”})`. In this case, you would specify either your lab or study name where is says `Study Name`. Finally, you will tell the script to indicate that an email has been sent, `complianceSheet.getRange(startRow + i, complianceEmailColumn).setValue(sent).`

The final section is very similar to the third section, but much simpler. This section specifies when to note in the Compliance Spreadsheet that all the questionnaires have been completed, so it is an else if statement. The conditions for this statement are identical to the ones set in the third section with one exception. The number of questionnaires that must be completed should be set to be equal to or larger than the minimum number that you require. In our example, we will set this value to be equal to or greater than 5, `c1Met >=5`. Again, we will specify this set of condition for every in the experience sampling period. If one set of these conditions is met, then we want the script to write that Survey Completed in the appropriate email column, `complianceSheet.getRange(startRow + i, complianceEmailColumn).setValue("Survey Completed").`

Once you have made these changes to the script, you should save your script. We named our script Compliance Emailer. Then you will need to authorize the script to run by clicking on the debug icon beside the dropdown menu in the toolbar. Next, we will want to specify two time-based trigger so that the script will execute once every day. Click on the clock icon that appears beside the play icon. Then select **sendComplianceEmails** in the first dropdown menu, **Day timer** in the third dropdown menu, and then select the most appropriate time interval for the population being studied in the last dropdown menu. For this script, we would advise you to select a trigger that is early in the day so that participants can actually increase the number of questionnaires they complete.

Debugging the script. If you script is not working correctly, you can debug it by using the Logger.log function. The Logger.log function is similar to the console function in JavaScript. For example, you can check whether `daysFromTheFirstDataDay` is calculated correctly by including `Logger.log(daysFromFirstDataDay)` in your script.