Make **IQM Reports** App User Documentation

Introduction

Make (formerly Integromat) is an online tool that allows users to save time by creating automated workflows, or “Scenarios”, that process and transfer data between web applications. Scenarios allow you to chain web apps together so that when a specified trigger event occurs in an app, Make automatically executes a workflow that sends data to or performs actions in one or more apps. The IQM Reports app on Make allows you to integrate IQM into these workflows, providing a convenient way to automatically export the ad-serving reporting data of your running campaigns to your platform of choice.

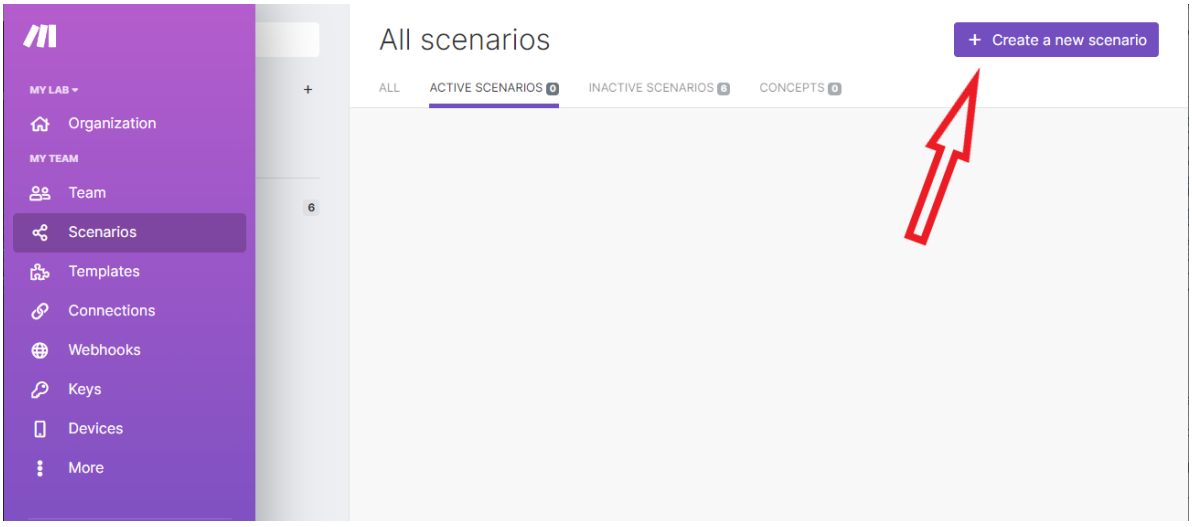
Prerequisites

To use the IQM Reports app on Make, you must have an IQM account and a Make account. If you do not have an IQM account, you can sign up for one here: https://open.iqm.com/#/signup.

Getting Started

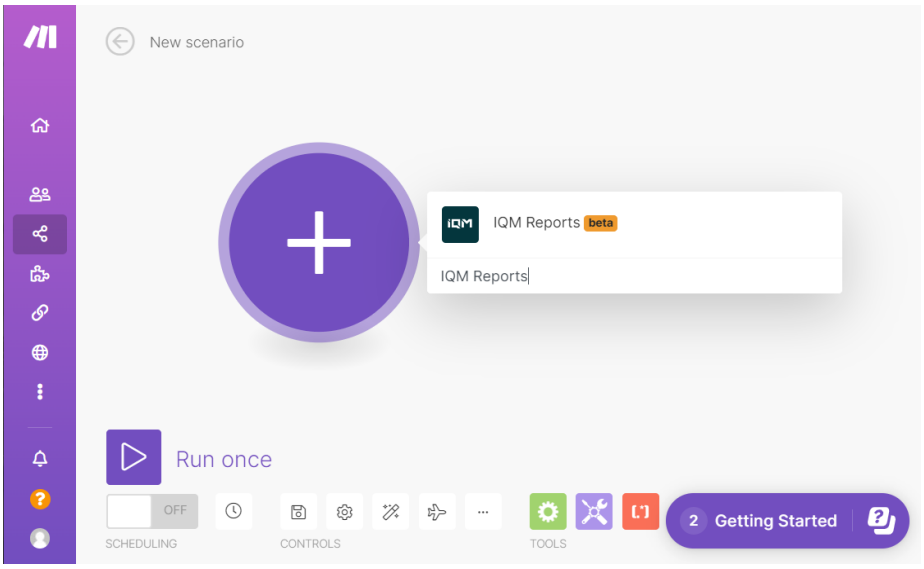
To create a workflow, or “scenario”, which exports your ad-serving data to your app of choice, use the following instructions.

1. To set up a Make scenario, first log into the Make platform: https://www.make.com/en.
2. In the **Scenarios** tab, click on the **Create a new scenario** button. This will take you to the scenario editor.

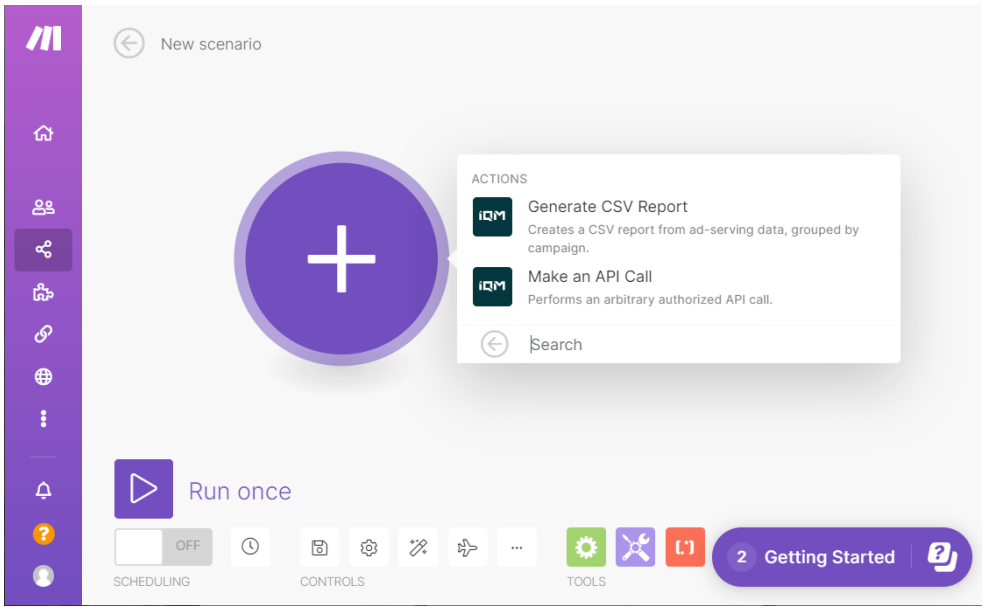


Scenarios consist of a trigger and one or more modules; the trigger determines when the scenario will run, and each module represents a web app that Make will pass data to and/or receive data from. Each module can receive data from any previous module in the scenario as inputs, called “mapping”. For more information about mapping, see https://www.make.com/en/help/mapping/mapping.

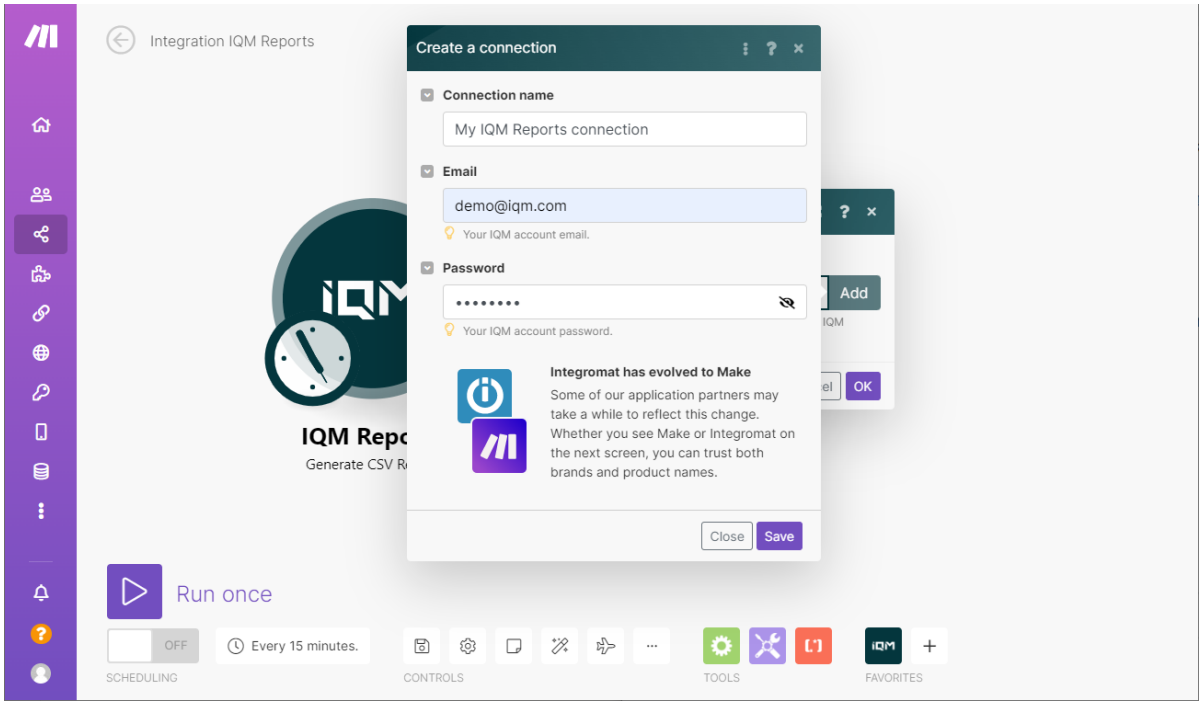
3. Click on the blank module to open the app list. Search for and select **IQM Reports** from the app list in the module.



4. Under **Actions**, select **Generate CSV Report**. This action generates a CSV file containing ad-serving report data for your running campaigns, and allows you to map the file into subsequent modules as input.



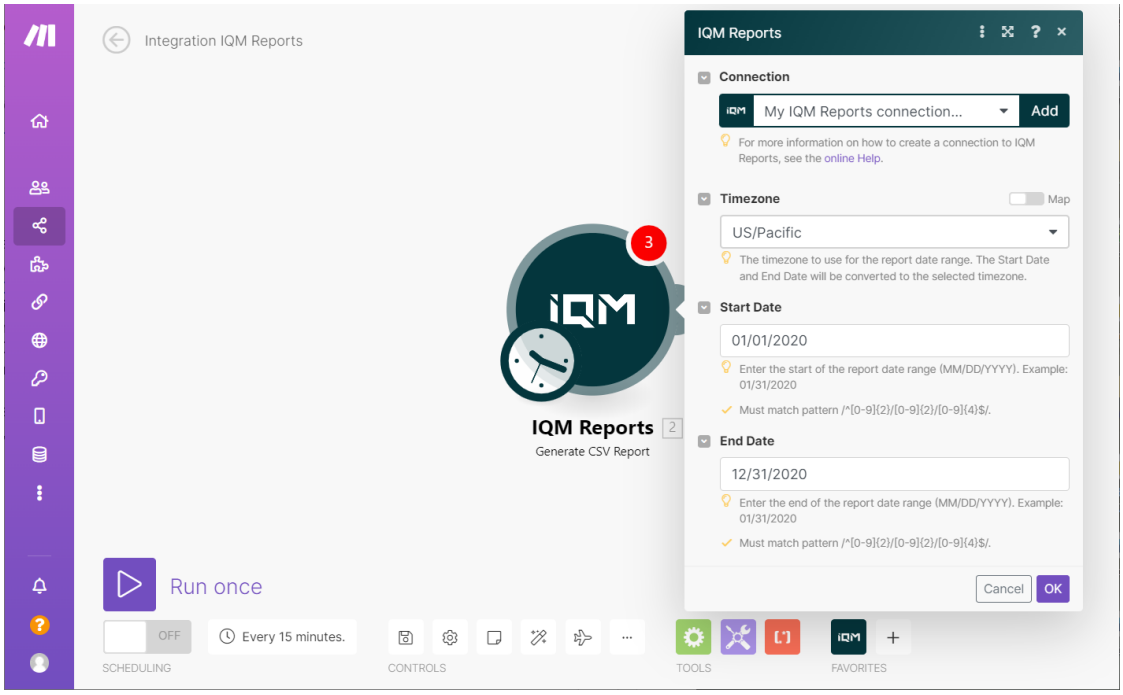
5. Click **Add** to connect the scenario to your IQM account. You will be prompted for your IQM account email and password; enter them and click **Save** to connect to your IQM account. If multiple IQM accounts will be used in different scenarios or modules within the same Make account, you can give the connection a unique name under **Connection name**.



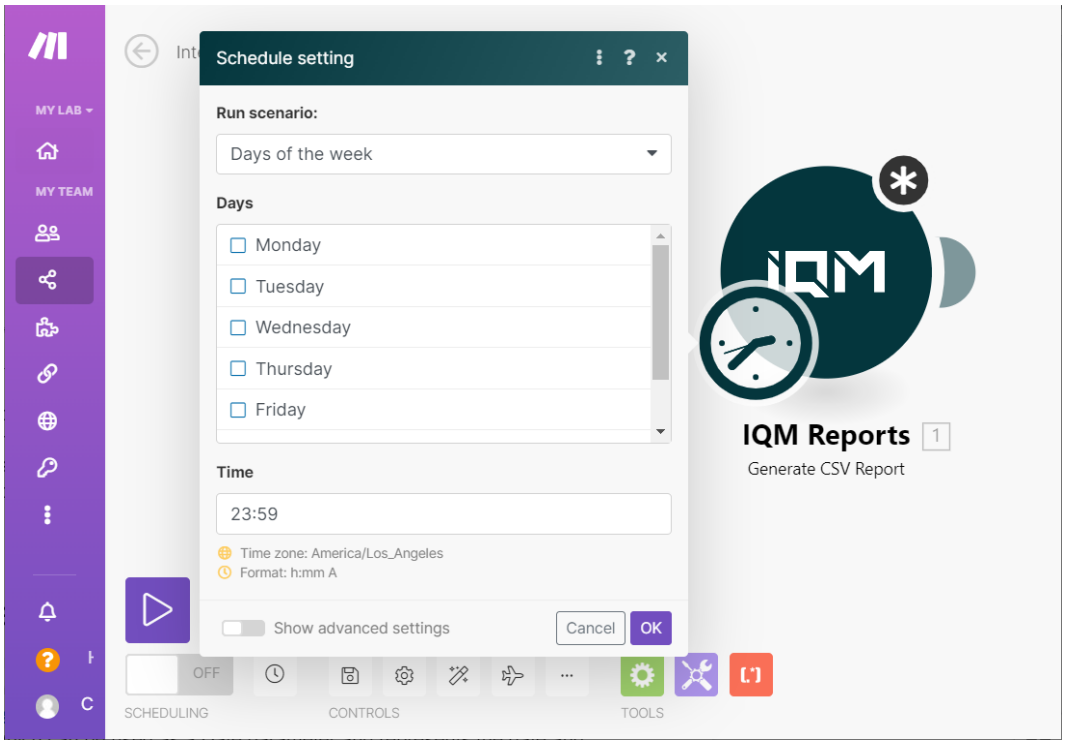
6. The Generate CSV Report action takes three required parameters: **Timezone**, **Start Date**, and **End Date**. The module will return ad-serving data for running campaigns from the time period between midnight on the Start Date to 11:59 PM on the End Date, in the selected Timezone.

   Note that the Start Date and End Date are not values of Make's built-in Date data type; it is Text of the form MM/DD/YYYY, which is then interpreted as a date and time. For instructions on how to use Make's date manipulation functions on these fields, see the **Using Start Date and End Date** section below. For more information on Make data types, see https://www.make.com/en/help/mapping/item-data-types.

   Once the parameter fields are filled out, click **OK** to continue.

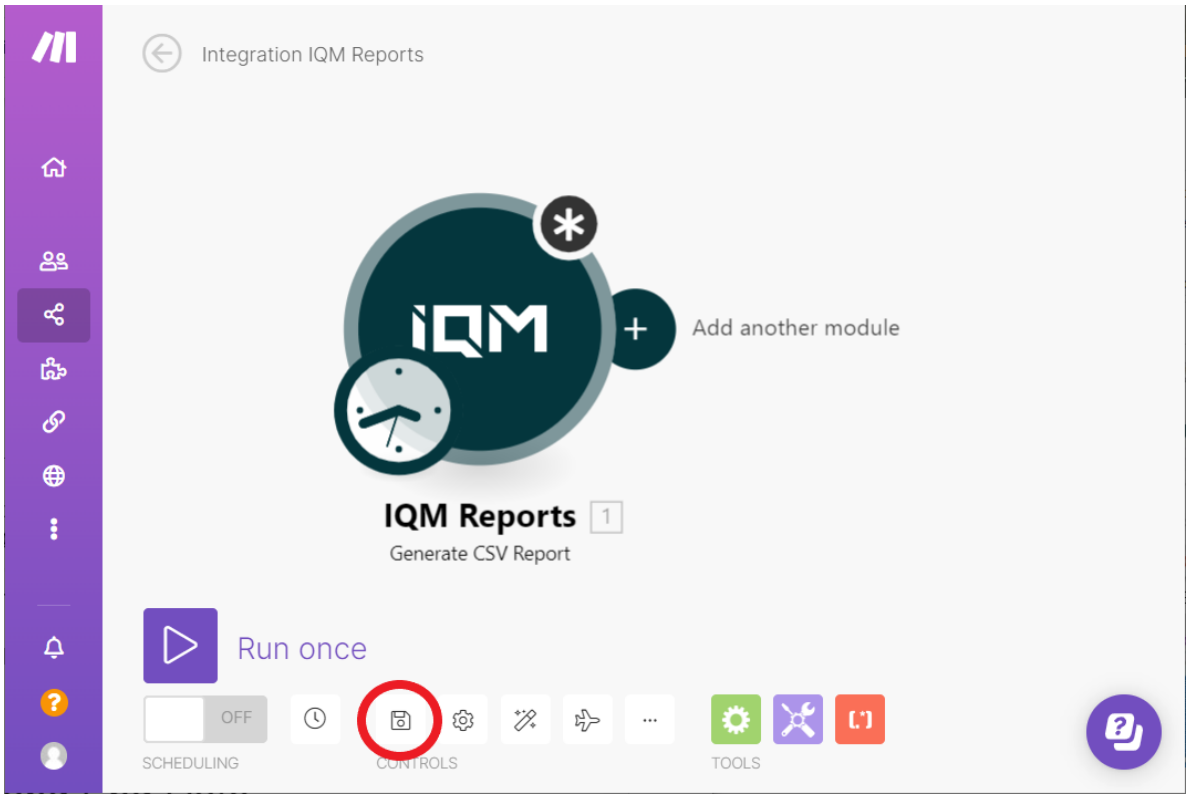


7. Next, click on the clock to customize the trigger to specify when the scenario will run. The built-in trigger for IQM Reports is the default scenario trigger, a scheduler which runs the scenario at specified time intervals. Select the desired interval type from the **Run scenario** dropdown, and specify the time interval, dates, and/or times that the scenario should run.

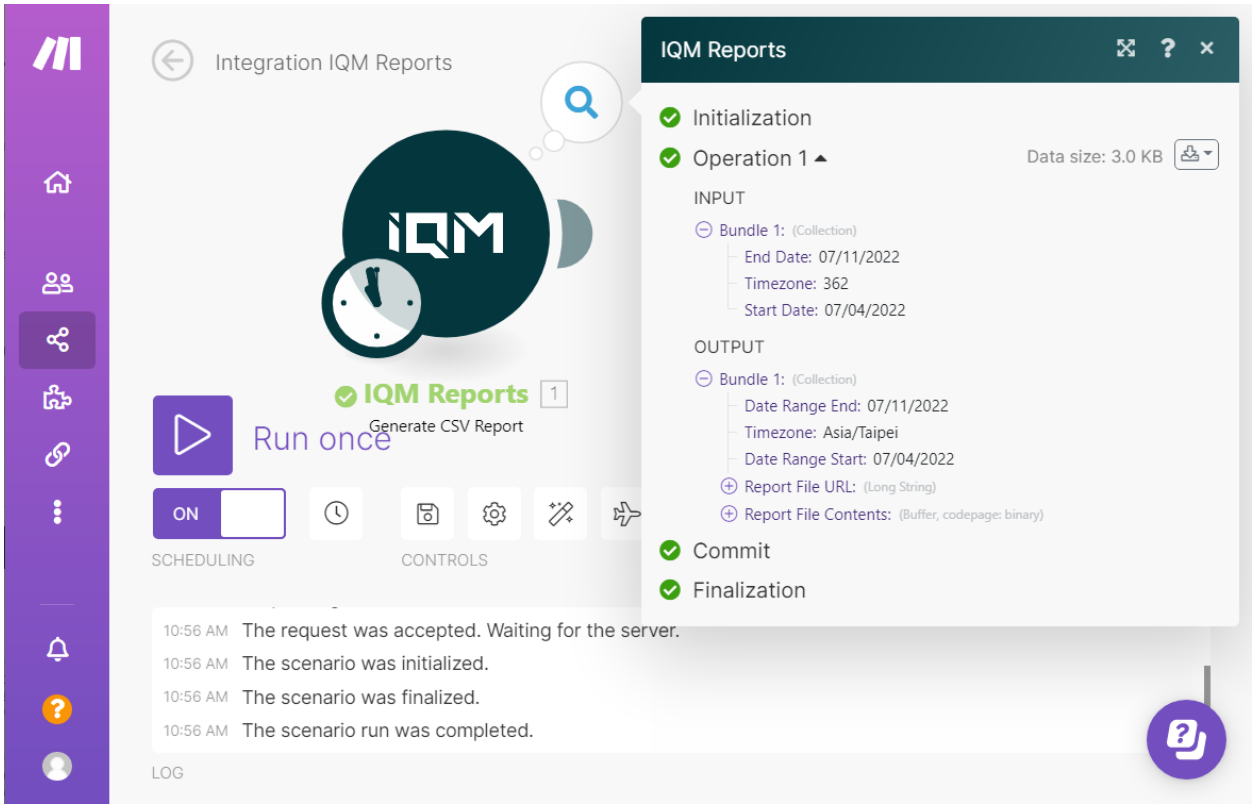


Or, to use a different trigger, right-click on the IQM Reports module and select **Add a module**, then search for and select the app with the desired trigger. See https://www.make.com/en/help/modules/types-of-modules for more information about the different types of modules, including triggers.

8. To test the IQM Reports module and view mappable outputs, save the scenario by clicking the save button circled below, then click **Run Once**.



9. To view the results of the scenario, click on the bubble that appears above the module. This will show the input and output bundles for the module if the run was successful, or the relevant error messages if it is not.

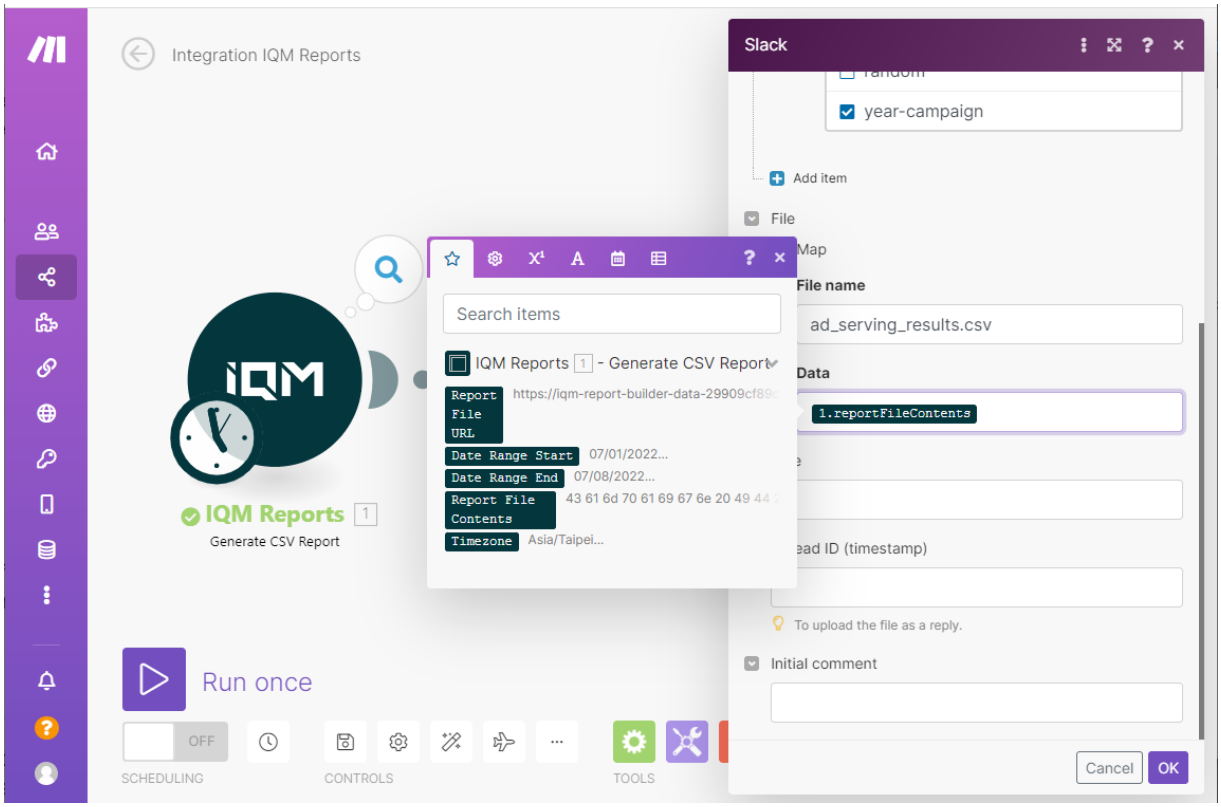


On a successful run, the output bundle contains the values **Date Range Start**, **Date Range End**, **Timezone**, **Report File URL**, and **Report File Contents**. The Date Range Start, Date Range End, and Timezone output values are the same as those entered into the Start Date, End Date, and Timezone input fields, respectively. The **Report File URL** is a temporary URL that points to the CSV file generated by the module; the **Report File Contents** contains the contents of the CSV file generated by the module. These output values can be mapped as inputs to any subsequent module added to the scenario after the IQM Reports module; for example, an Email module could be added to send the Report File Contents as a .csv attachment in an email whenever the scenario runs, or a Dropbox module could be added to save the file. In general, the Report File Contents value can be mapped into any field that expects a file.

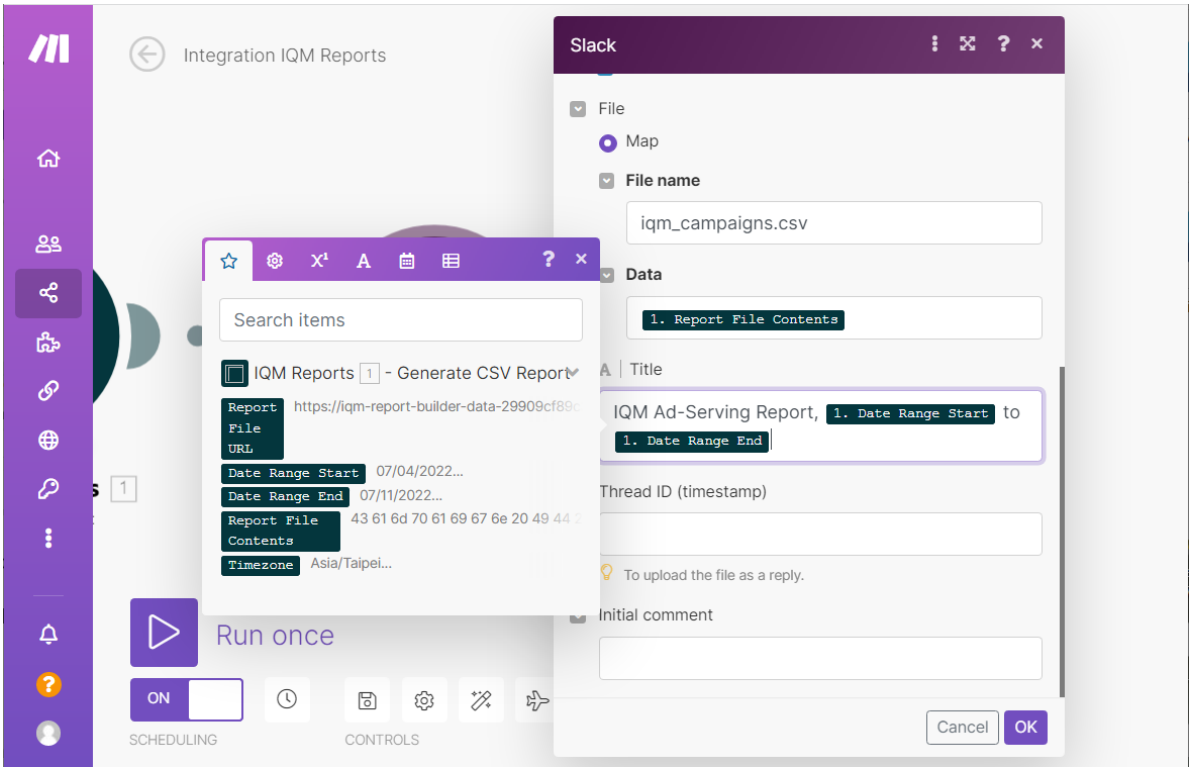
10. To pass the CSV report generated by IQM Reports to other web apps, select **Add another module** and select the desired app from the app list. In this example, we'll periodically send a Slack message with the CSV report attached, so we'll select the Slack app.

11. Just as before, you'll be prompted to select or create a connection to authorize Make to connect to your Slack account. Follow the steps as before to connect your Slack account to Make. Then, from the action list, select **Upload a File**.

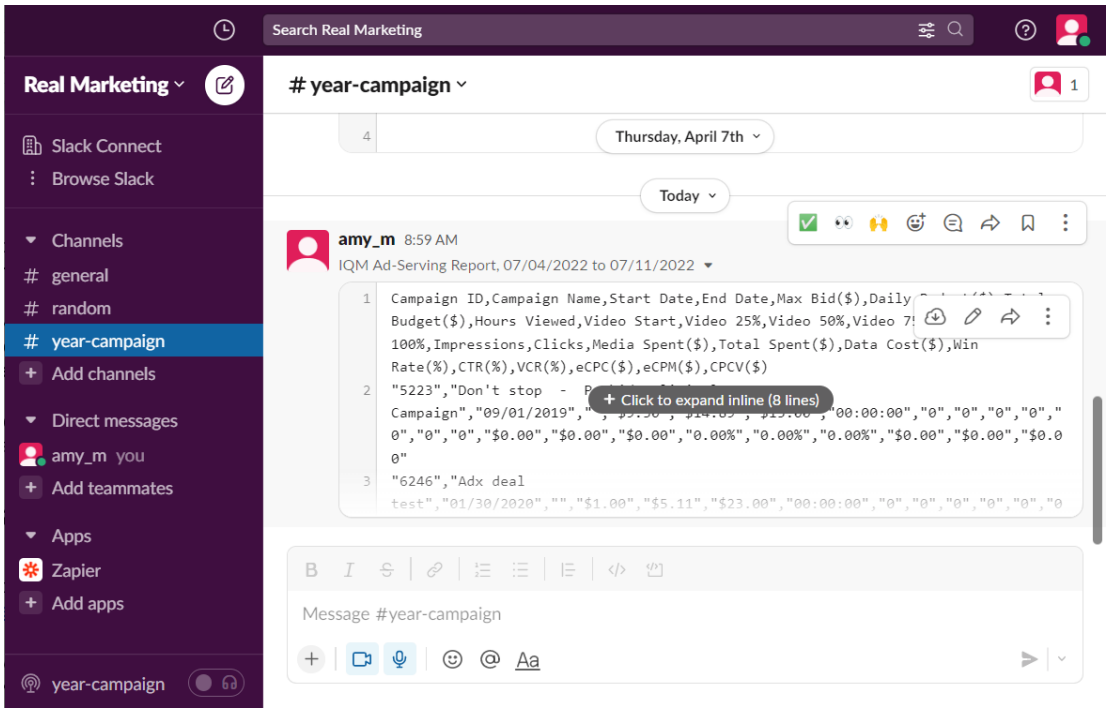
12. In the module settings, Fill in the **Channels** that you want the file uploaded to. Then, under **File**, type in the filename to use for the file attachment (including the .csv extension), and map the **Report File Contents** from the output bundle of IQM Reports into the **Data** field. Slack's Upload a File action creates and sends a Slack message; mapping the Report File Contents to Data will cause the CSV file generated by IQM Reports to be added to the Slack message as an attachment.



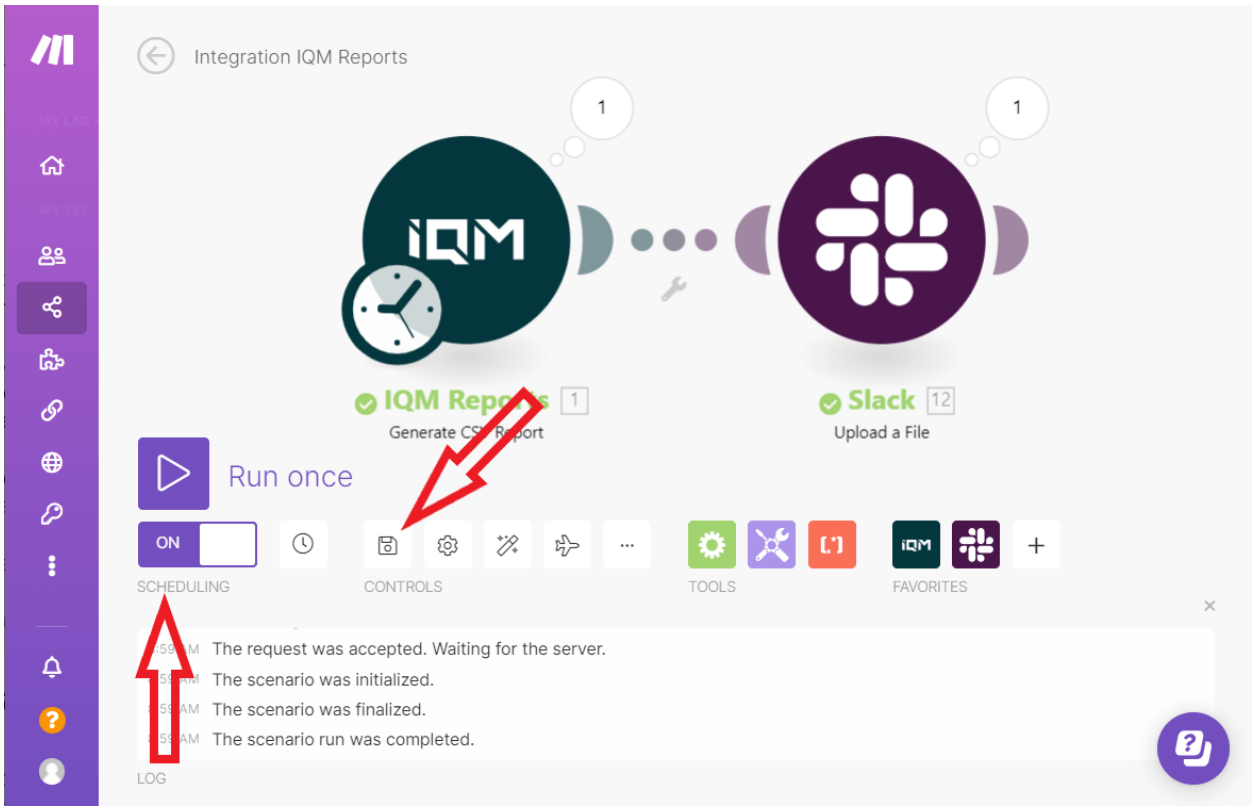
13. Fill in the other input fields as appropriate. Note that, with the mapping panel that appears when clicking on a mappable input, you can map any values from the IQM Reports output bundle into input fields of compatible types. For example, you can map the **Date Range Start** and **Date Range End** fields into the **Title** field as shown below. If a mapped item is of a different type than the field it is mapped into, Make will convert the mapped item to the appropriate data type if possible; see https://www.make.com/en/help/mapping/type-coercion for more information.



14. Click the **OK** button in the module settings panel. Then, click **Run Once** in the scenario editor to test the scenario. This will cause a single run of the scenario; if successful, it will cause a message to be posted to the Slack channel chosen in Step 12 with the CSV file generated by IQM Reports attached.



15. Click the Save icon in the **Controls** section to save the scenario, and set the **Scheduling** switch to **On**. This will cause the scenario to run periodically according to the trigger settings from Step 7.

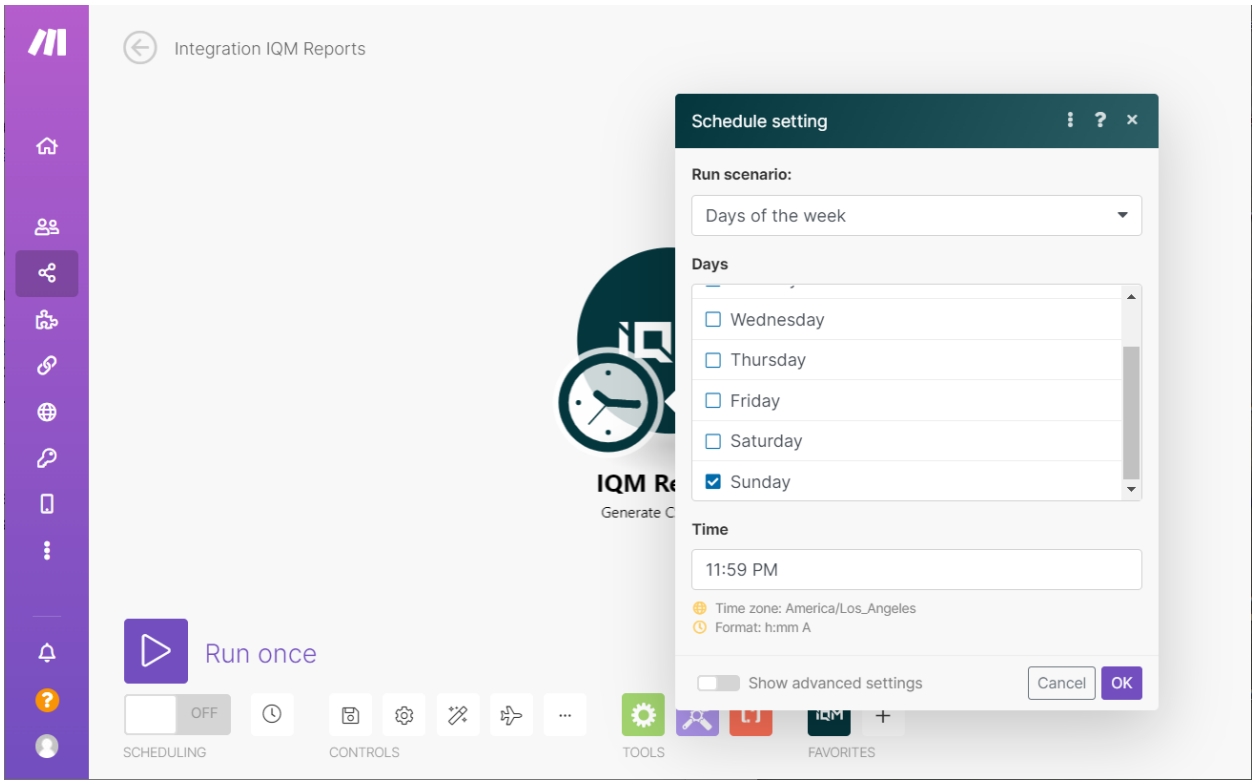


The Make scenario will now run according to the settings. It can be edited or disabled at any time by selecting it in the **Scenarios** page.

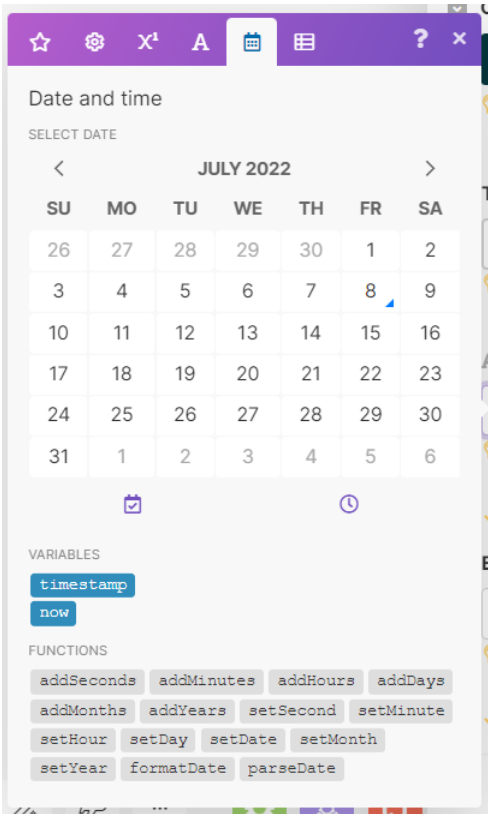
Using Start Date and End Date

Make provides several useful built-in functions in the scenario editor for manipulating Date values; for example, you can add or subtract days or hours from a given Date value. See https://www.make.com/en/help/functions/date---time-functions for more information. It also provides the **now** variable, which can be used as a Date parameter and represents the date and time at which the scenario runs. As mentioned above, the Start Date and End Date parameters are of type Text, not Date. This means that a few extra steps are needed to use Make's built-in date-manipulation functions on these fields.

One common use case is a Scenario which is triggered at regular intervals, such as once per week. In these cases, you probably want IQM Reports to return reporting data from that same time interval. For example, the scenario below is scheduled to run once per week, every Sunday at 11:59 PM (as per the logged-in user's timezone). Note that the Schedule Settings trigger (and other Date fields) use the logged-in user's profile timezone, unlike the IQM Reports **Start Date** and **End Date** fields, which are interpreted according to the **Timezone** dropdown field.

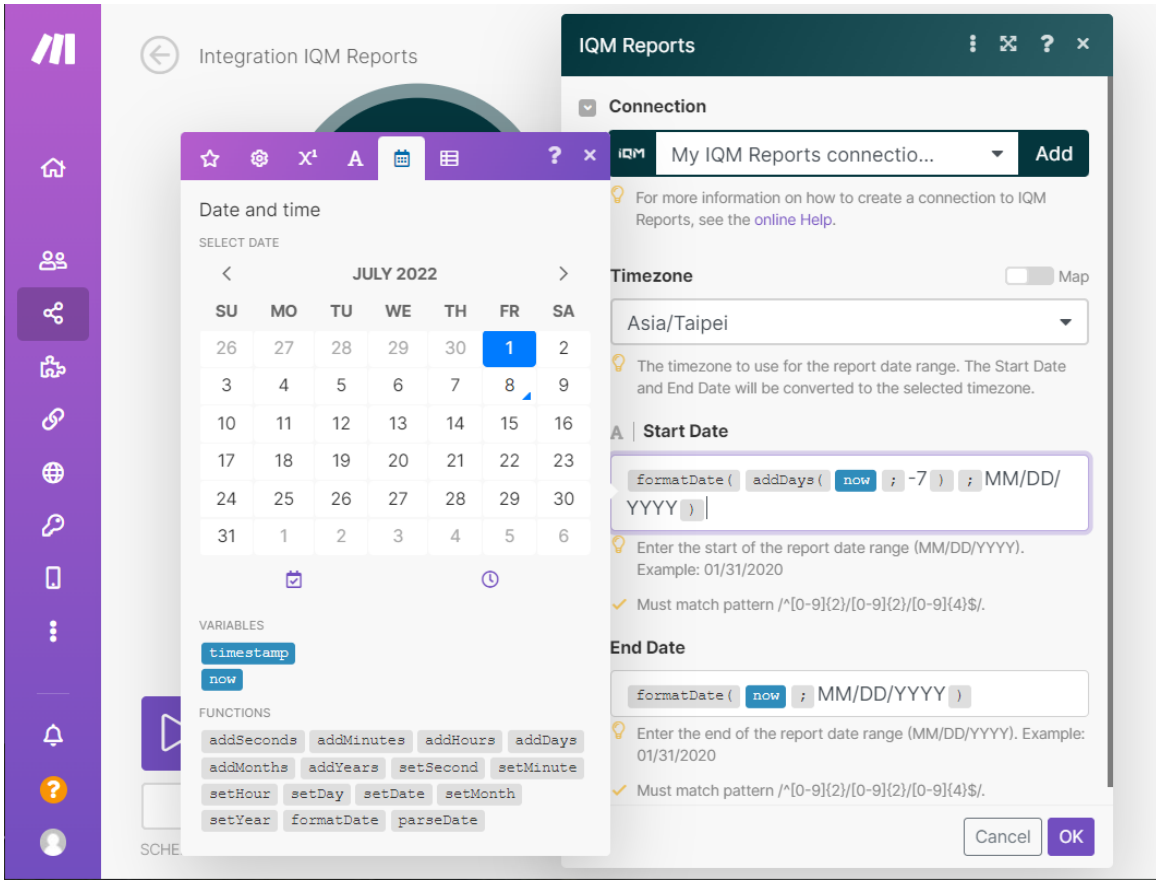


To generate the ad-serving report for the past week in IQM Reports, we can use the **addDays** function, the **formatDate** function, and the **now** variable. These functions, along with similar date-manipulating functions, are found in the **Date and time** tab in the mapping panel which appears after clicking on a mappable input, as shown below.



To set the **Start Date** to one week before the scenario runs, first select the **formatDate** function from the mapping panel. The formatDate function converts the Date passed in its first parameter to a Text value in the format determined by the “token” passed in its second parameter. For the second parameter, type in **MM/DD/YYYY**. For more information about the formatDate function, see https://www.make.com/en/help/functions/tokens-for-date-time-formatting. For the first parameter to formatDate, insert the **addDays** function. Set the first parameter of addDays to the **now** variable, and the second parameter to **-7**. This will subtract seven days from the date and time at which the scenario runs, each time it runs.

For the **End Date** field, use the **formatDate** function to convert the **now** variable to Text using the same token as before, MM/DD/YYYY.

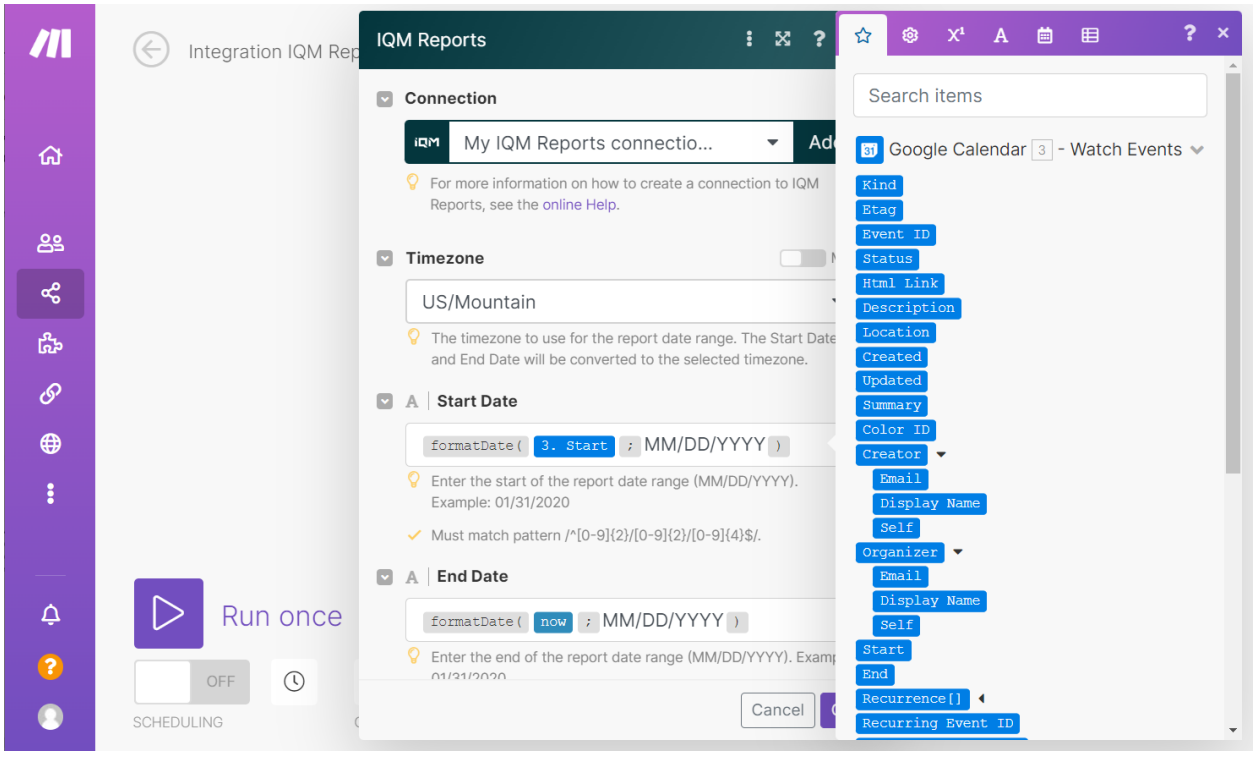


The above settings for **IQM Reports** will cause the module to generate a report for the past week each time it is run, with the **End Date** set to the date (according to the user's profile timezone) when the scenario runs and the **Start Date** seven days before the End Date. These dates are then interpreted according to the selected value in the **Timezone** field. You can use similar formulas to calculate the Start Date for any desired time interval.

In general, whenever you need to use Make's datetime manipulation functions (addDays, setHour, etc) on the dates passed to the **Start Date** and **End Date** fields, simply wrap the datetime manipulation function in a formatDate function to convert the Date result to a Text value of the form MM/DD/YYYY.

Likewise, in order to map a Date value from a previous module into **Start Date** or **End Date**, select the desired Date value from the mapping panel and wrap it in a formatDate function. For example, you might map the date of an event returned from a calendar module into the Start Date or End Date fields, as in the example below.

Below, the **Start** value in the output bundle from Google Calendar is of type Date, so the formatDate function is used to convert it to Text.



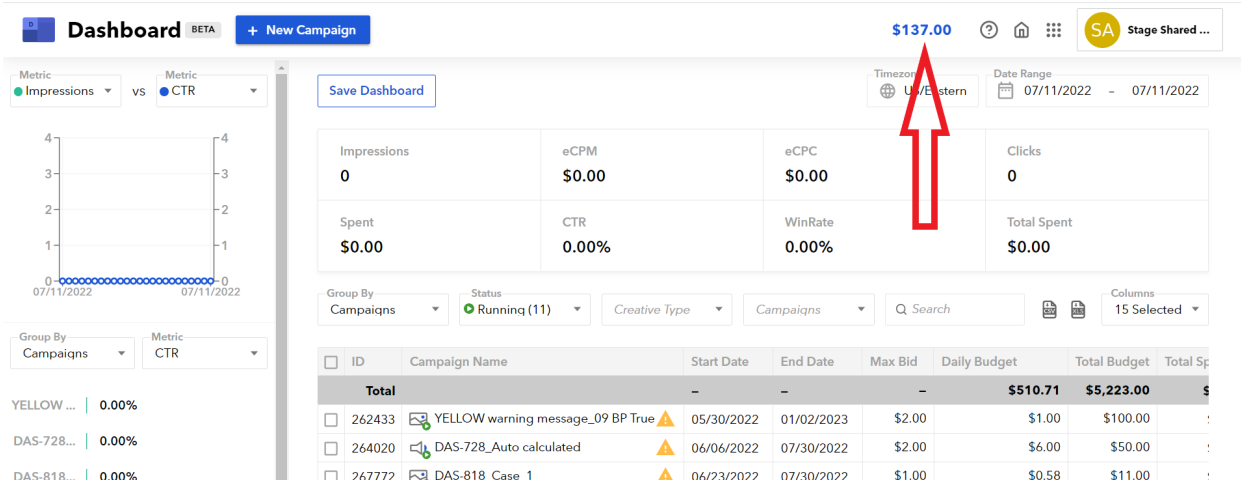
The Universal Module: Make an API Call

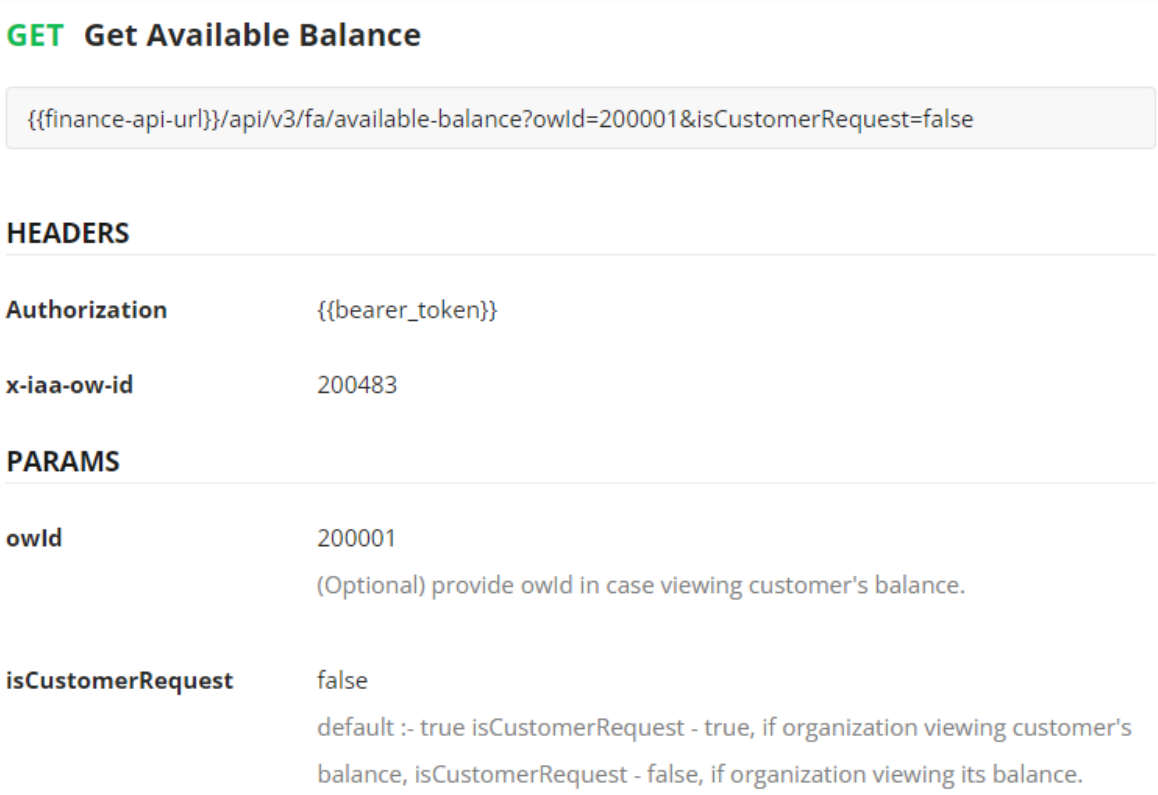
The IQM Reports app offers two actions: **Generate CSV Report** and **Make an API Call**. Behind the scenes, the former action sends an *API call*–a structured request to retrieve or change data–over the Internet to IQM servers to retrieve your ad-serving data. It then passes that data to the Make scenario. Specifically, it requests ad-serving data from the https://app.iqm.vote/api/v2/rb/serving/result API endpoint, then transforms the data into a form that can be mapped to subsequent modules.

IQM offers many more API endpoints that can be used to retrieve data about your campaigns, creatives, audiences, and more. See https://api.iqm.com/ for a list of API endpoints. The **Make an API Call** action allows you to use most IQM API endpoints in a Make scenario. **Use this action with caution!**

The following example shows the use of the **Get Available Balance** endpoint in the Make an API Call action. (See https://api.iqm.com/#d0be7f8f-6a46-46cf-bdd6-b74402857d39 for documentation on this endpoint.)

This is the same endpoint that is used to retrieve the available balance shown in the IQM Dashboard app:

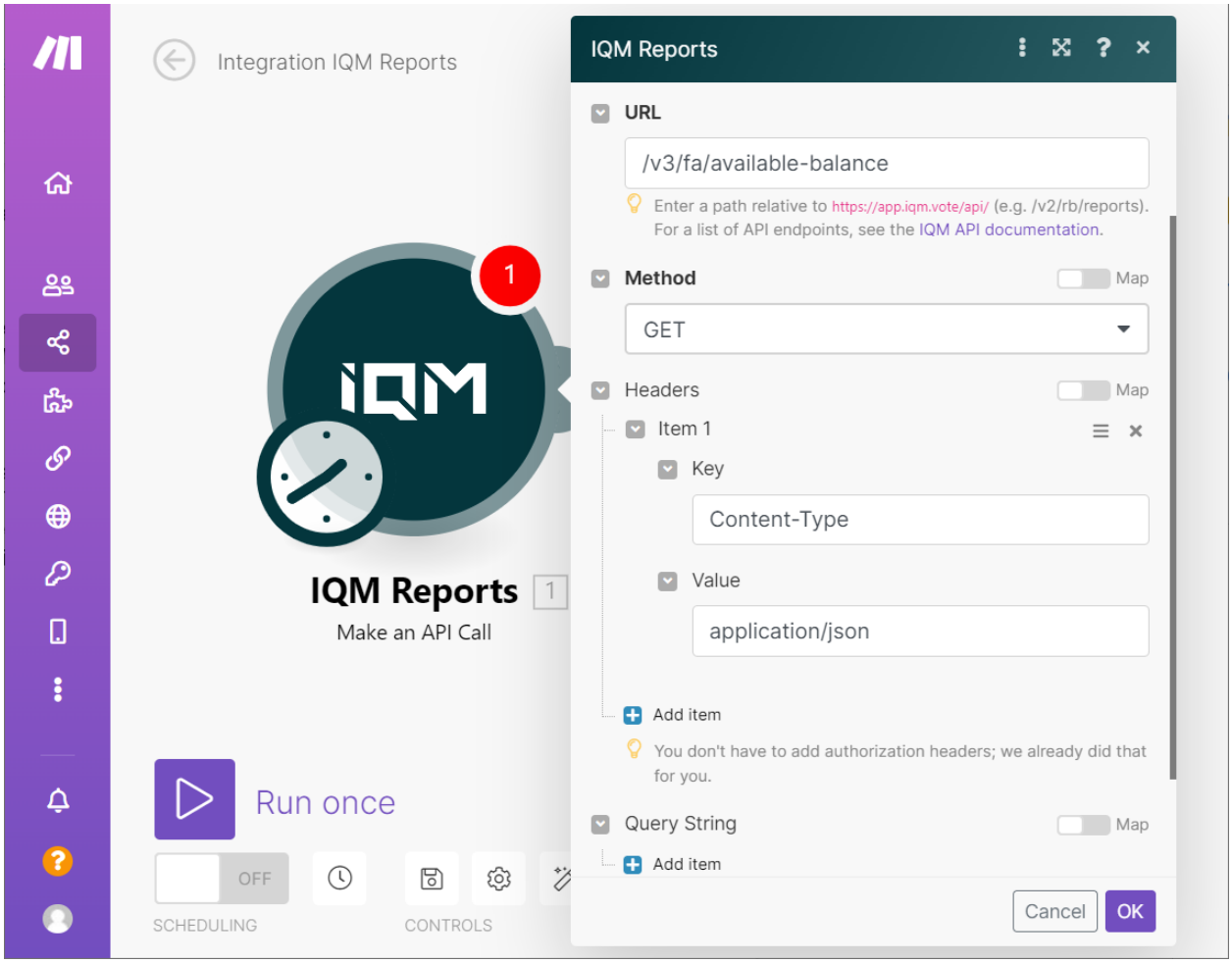




For the **URL** parameter in the module settings, enter the API endpoint relative to /api (including a leading forward slash, /). This is the part of the endpoint between **/api** and the **?** in the URL in the documentation screenshot:
{{finance-api-url}}/api/v3/fa/available-balance?owId=200001&isCustomerRequest=false becomes **/v3/fa/available-balance**.

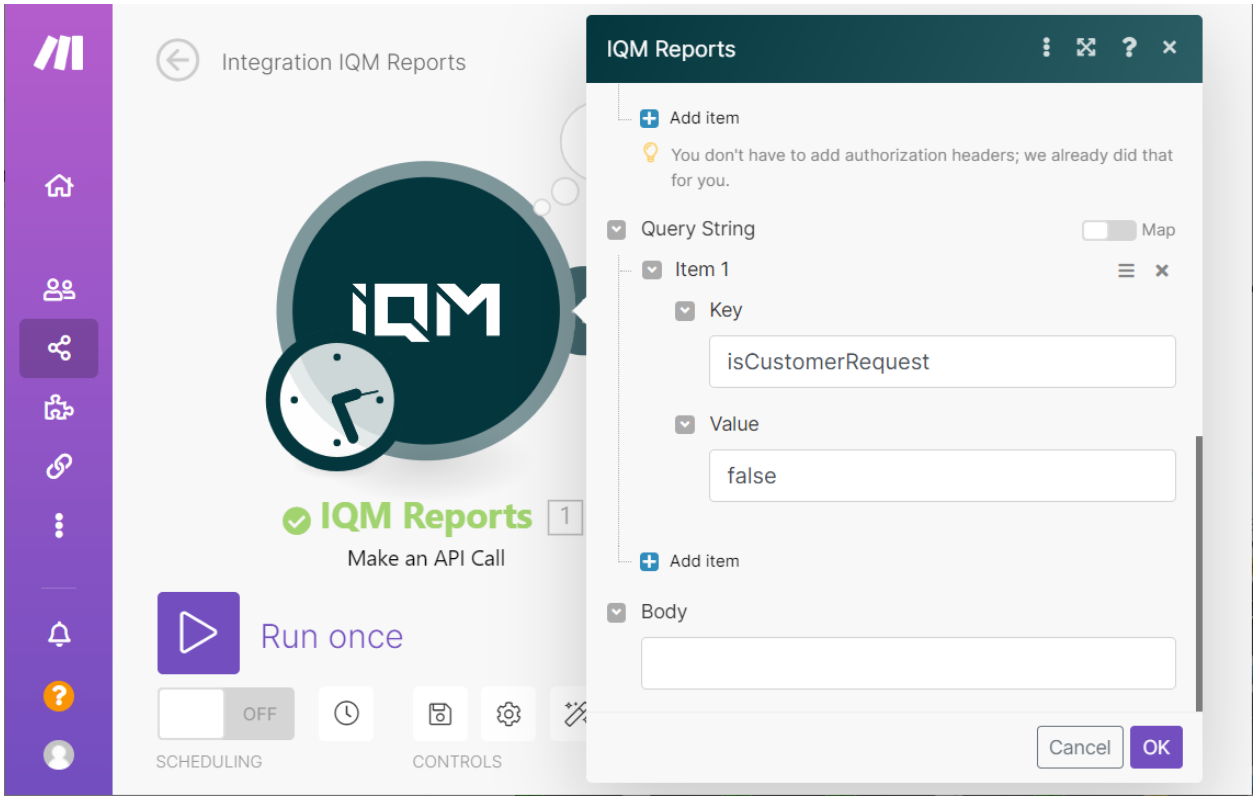
The **Method** field determines the HTTP method that the API call will use. As per the documentation above, keep the **Method** field as **GET**. Generally, the GET HTTP method means that the API call will read data from your IQM account without modifying data; it is safe to experiment with.

The **Headers** section determines the HTTP headers that will be sent with the API call. Headers contain information about the HTTP request, such as authorization details that allow the server to identify the user making the request and determine if the client can access protected resources. The API documentation above lists two headers: **Authorization** and **x-iaa-ow-id**. These headers are added automatically by Make, so there is no need to add them in the module settings; leave the Headers section as-is.



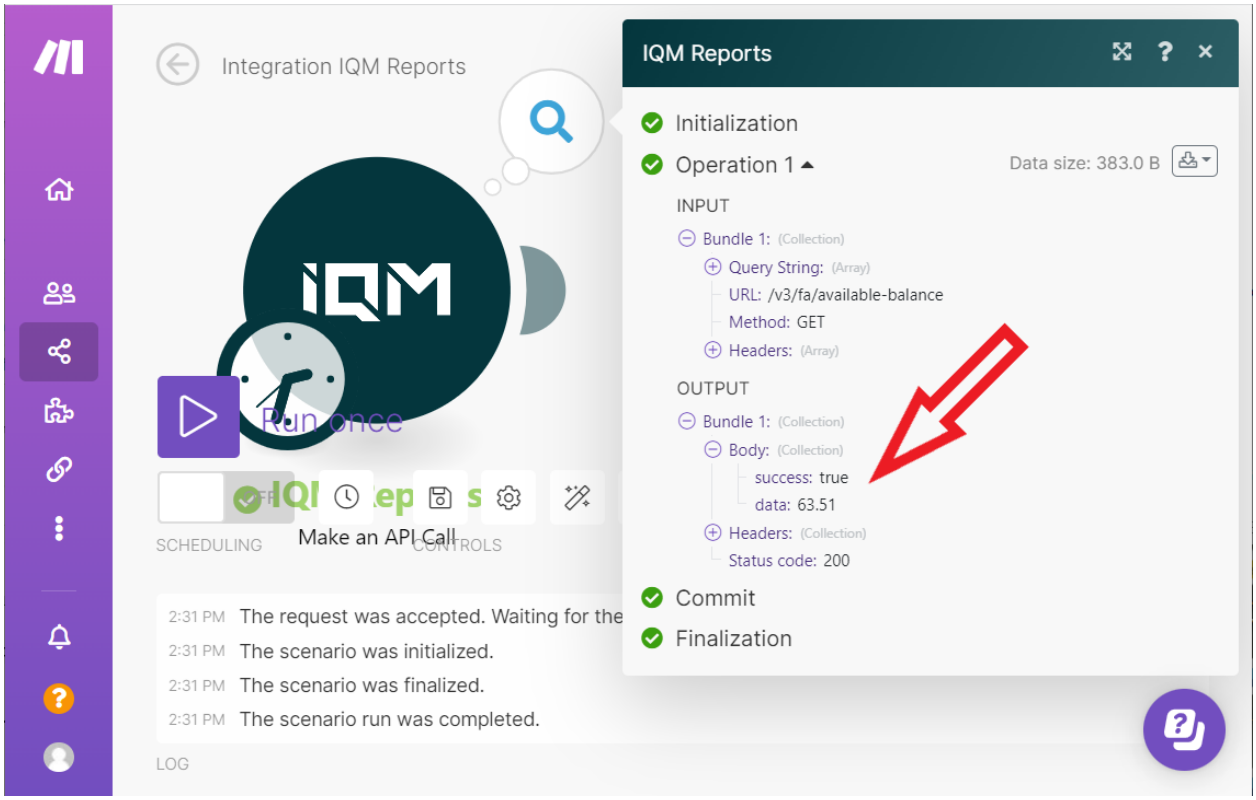
The **Query String** section allows you to add query string parameters to the API call. These parameters are added to the URL as key-value pairs. Query string parameters add extra information to a GET API call; for example, they usually tell the server how to filter and sort the requested data before it is returned in the API response. In the URL **{{finance-api-url}}/api/v3/fa/available-balance?owId=200001&isCustomerRequest=false**, the **?** marks the start of the query string. The two parameters in this query string are **owId** and **isCustomerRequest**; they are passed the values ‘200001’ and ‘false’, respectively. The server uses these parameters to determine which user’s available balance to return.

In this example, we’ll retrieve the available balance for the entire organization. This requires the query string parameter **isCustomerRequest=false**. To add this parameter, add an item under **Query String** with the **Key** and **Value** shown in the below screenshot. (To request the available balance for a specific customer, set the Value for the isCustomerRequest Key to true, and add an additional key-value pair to specify the customer’s owId in the Query String section.)

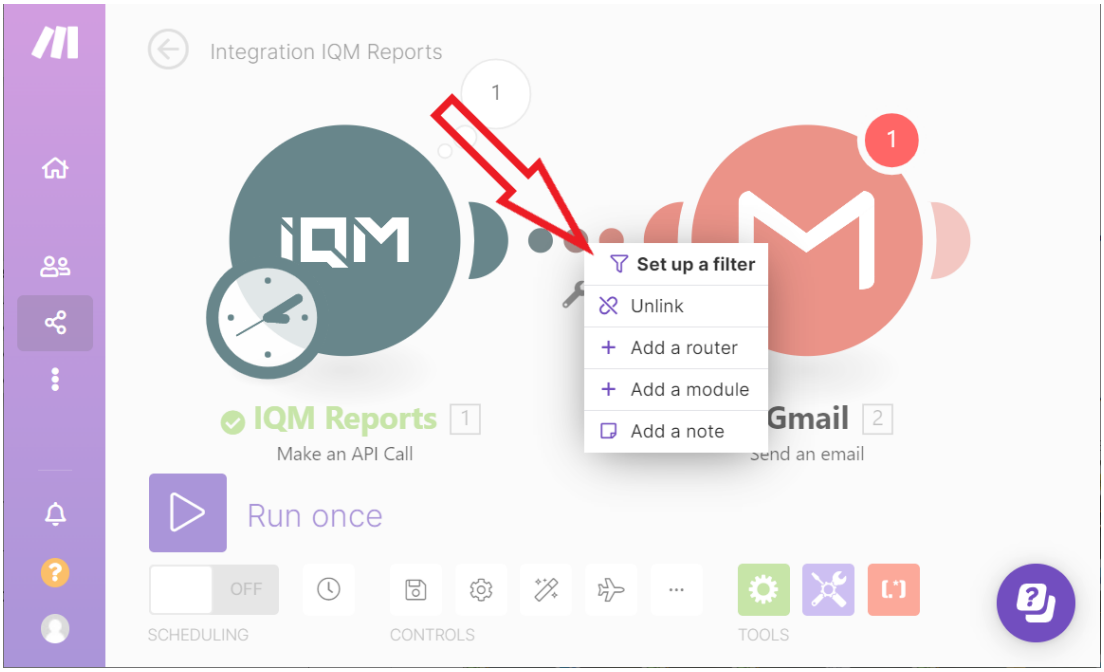


Finally, leave the **Body** field blank. This field is used for POST and PATCH HTTP requests.

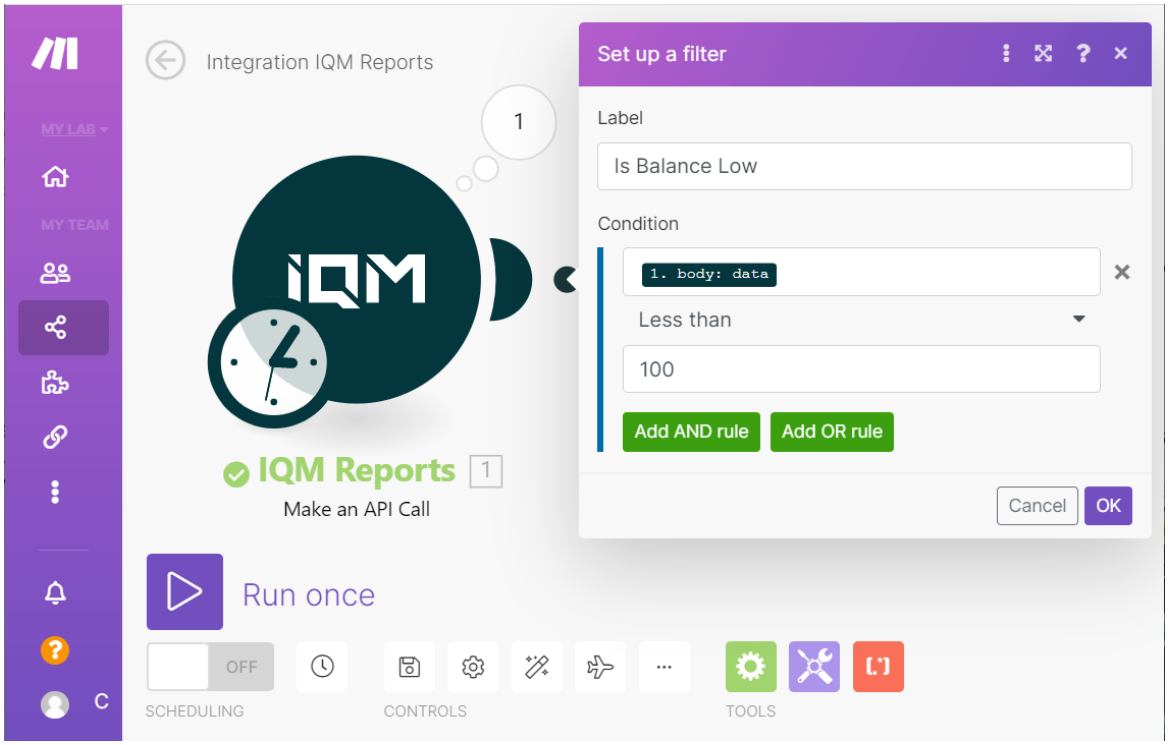
After clicking **OK** in the module settings, click **Run once**. Then, view the results by clicking on the bubble that appears above the module. The **OUTPUT** bundle's **Body** contains the information returned from the API call: A **success** boolean which indicates whether the call completed successfully, and a **data** field that contains the requested information. In this case, the data field contains a number, the requested available balance. This can be mapped to the inputs of subsequent modules just like any other output value.



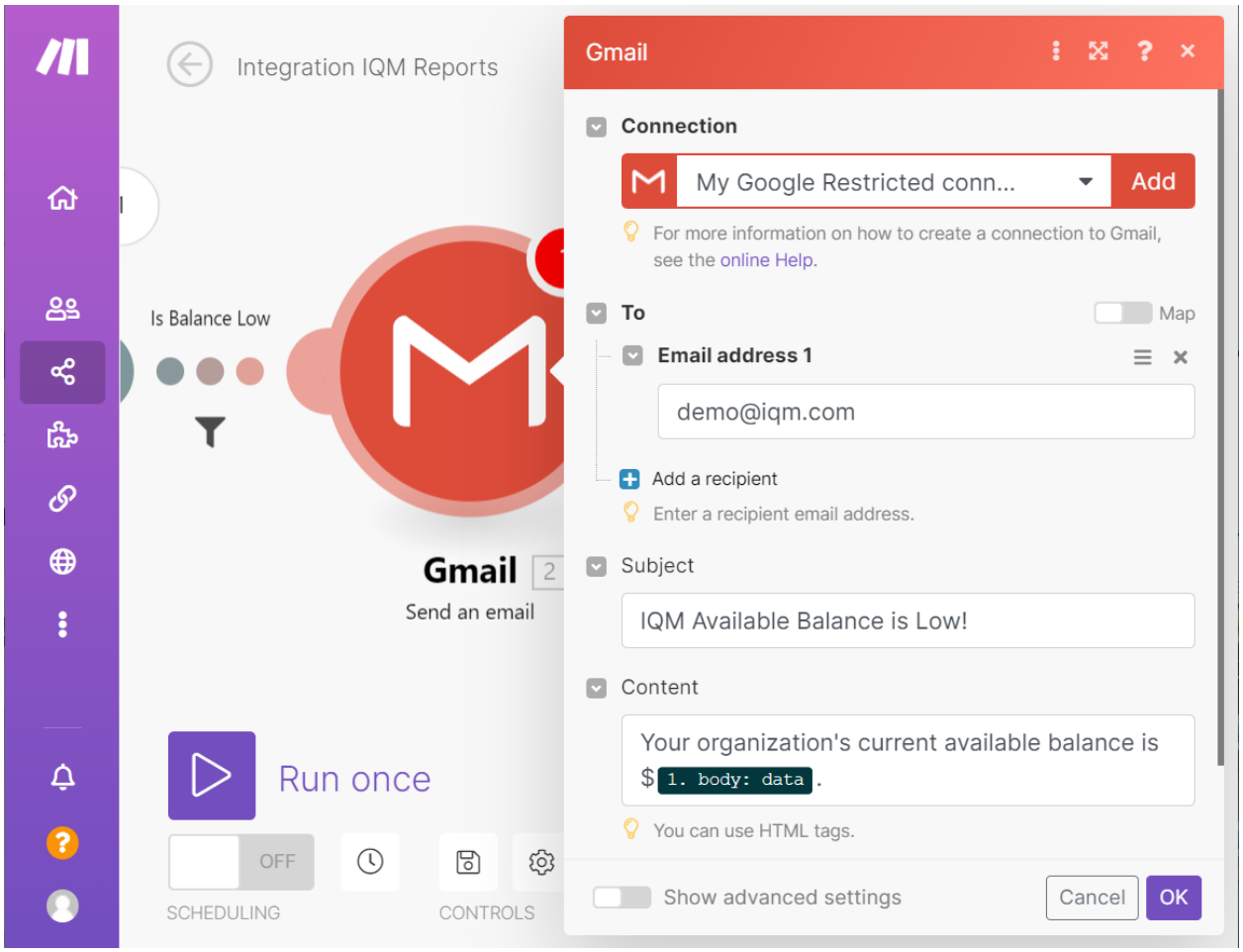
One thing that we can do with this module is check if the available balance returned from the API is low, and conditionally send a reminder via the Gmail app if it is. To do this, add a new module and select the Gmail app from the app list. Then, right-click on the string of dots connecting the IQM Reports module to the Gmail module and select **Set up a filter**. A filter causes the scenario to check if a value from an output bundle matches a given condition. If the condition is met, then the scenario continues executing with the module after the filter; if the condition is not met, then the scenario terminates.



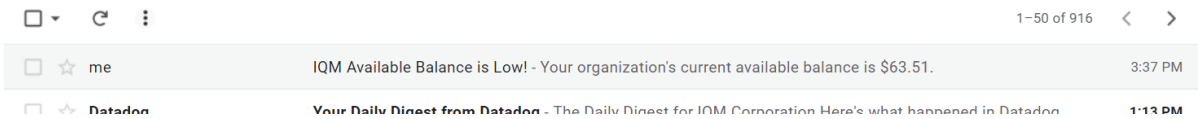
In the **Set up a filter** popup, fill in the condition for the filter by mapping values from the API response output bundle and selecting a condition type from the dropdown, then click **OK**. The following filter will cause the Gmail module to execute only if the **data** field (the available balance) in the API output is less than 100.



Finally, set the trigger to the desired time interval to determine how often the scenario should check the available balance, and fill out the Gmail module settings to specify the recipient and contents of the email reminder. Just as before, click **Run once** to test the scenario, and set the **Scheduling** switch to **On** to enable the scenario.



The scenario will now send an email reminder each time it runs when it detects that the organization's available balance is low:



Make provides a host of other features that allow for a wide variety of automated scenarios; see the Make documentation at https://www.make.com/en/help/home for more information.