RADIX® iot

WHITE PAPER

TWILIO INTEGRATION FOR MANGO

OVERVIEW

This document will take you through the steps set to configure a scripting event handler to send text messages through the Twilio API. We will walk through a number of advanced features available in the Mango Scripting environment.

REQUIREMENTS

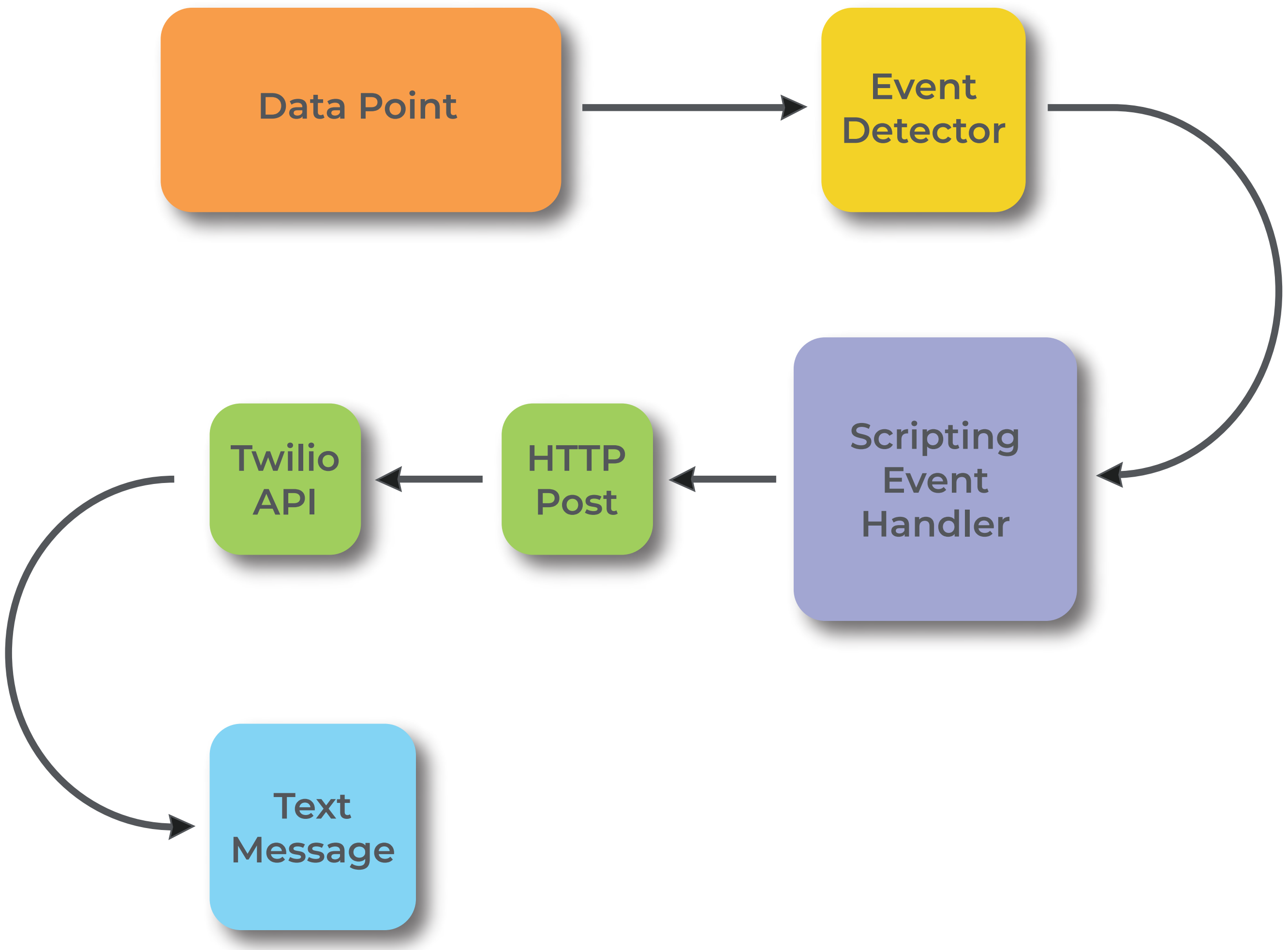
- A Mango v4 instance with access to the internet
- An active Twilio developer account
- A cell phone to receive text messages

Twilio Account

Before you get started you will need to create an account with Twilio. Here is a great guide on how to send your first text message https://www.twilio.com/en-us/use-cases/commerce-communications/account-notifications/build

Once you have successfully sent a text message with your Twilio we can move on to setting up data points.

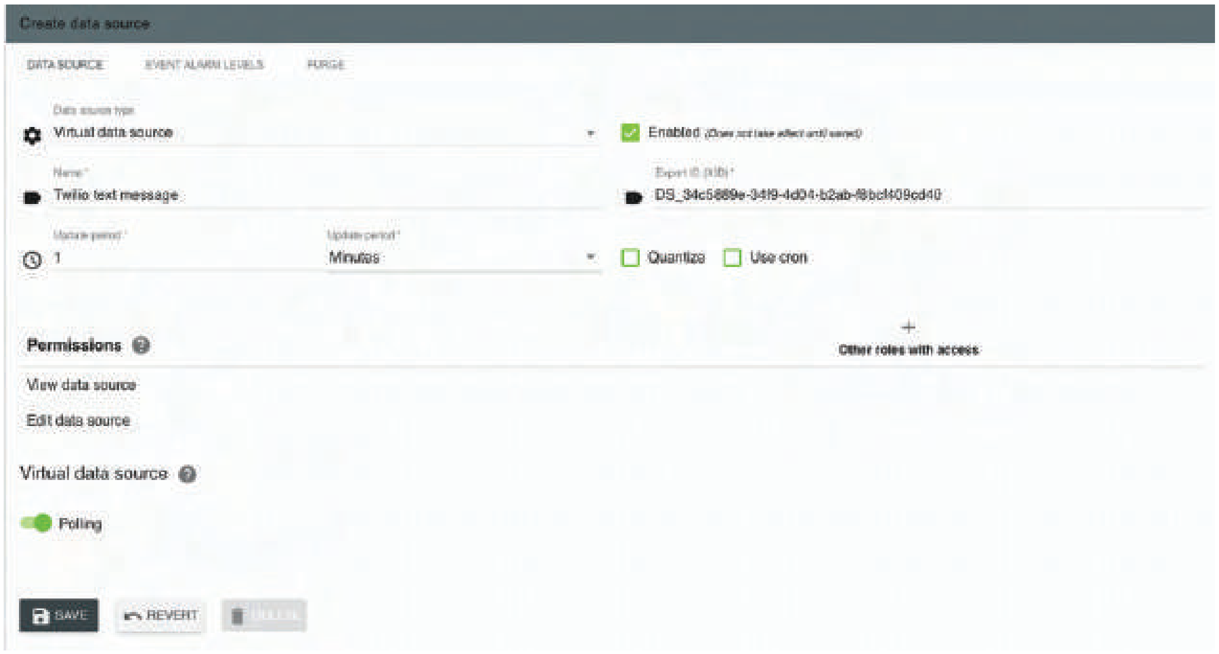
Below is a diagram of what we will be configuring within Mango



DATA POINT EXAMPLE

For this, we will create a virtual data point that will be used to test the text message. You could replace this virtual point with any other data point in a production system..

1. Create a Virtual data source within your Mango instance. We are making a virtual point as a test. But you can connect real data sources for the production system.



Create data source

Data source type: Virtual data source

✓ Enabled (Does not take effect until saved)

Name*: Twilio text message

Export ID (XID)*: DS_34c5889e-34f9-4d04-b2ab-f8bcf409cd40

Update period*: 1

Update period*: Minutes

☐ Quantize ☐ Use cron

Permissions

Other roles with access

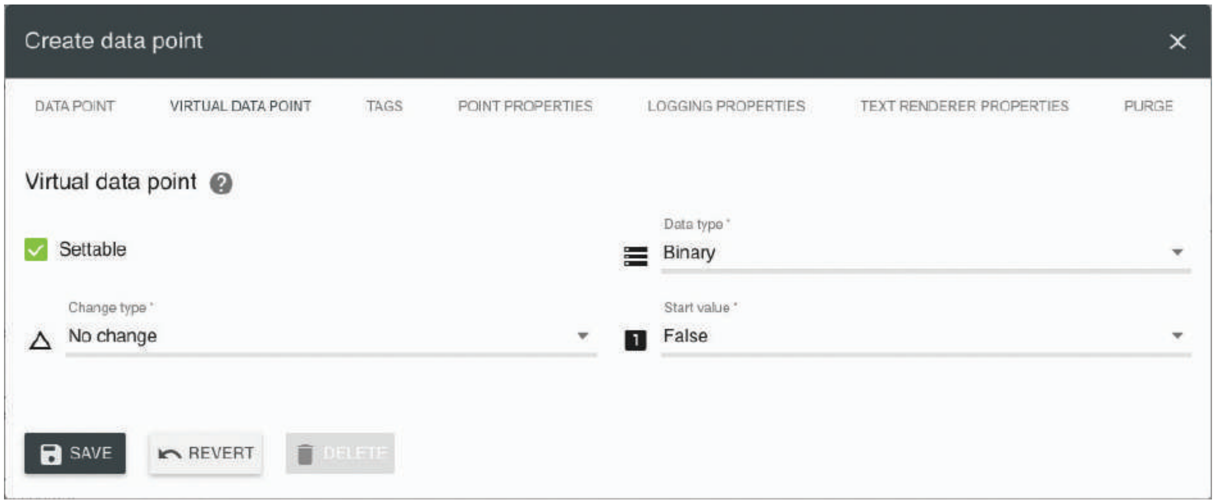
View data source

Edit data source

Virtual data source

Polling

2. Create a binary virtual data point that is settable.



Create data point

Virtual data point

✓ Settable

Data type*: Binary

Change type*: No change

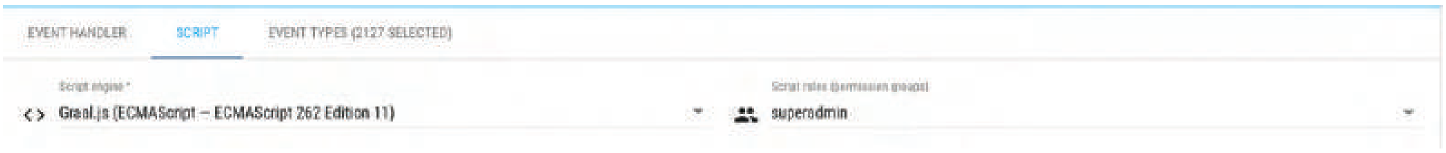
Start value*: False

3. Before we create an event detector we will first create a scripting event handler that will send the text message. Before you do this you will need to make sure you have your Twilio account SID, Auth token, and Twilio number on hand. If you have not set this up yet, please follow the Twillio guide at https://www.twilio.com/docs/usage/tutorials/how-to-use-your-free-trial-account#sending-sms-and-mms-messages

If you navigate to this path: "/ui/help/event-handlers/help/script-event-handler" on your Mango instance you will find the help doc on the scripting event handler. You will see that the scripting event handler provides us with 3 callbacks that will be executed by the event handler on those specific events.

```
/**
 * Called when the Mango event is raised.
 */
function eventRaised(event) {
   console.log(event)
}

/**
 * Called when the Mango event is
 * acknowledged (the event may still be active).
 */
function eventAcknowledged(event) {
   console.log(event);
}

/**
 * Called when the Mango event returns to normal
 * or is deactivated (e.g. on shutdown).
 */
function eventInactive(event) {
   console.log(event);
}
```

Make sure you are using the Graal.js engine and ensure that your script has super admin privileges. The Graal.js is a Javascript engine that allows you to run Javascript code within the Java Virtual Machine. Providing you with a quick and flexible scripting environment.



Script engine *: Graal.js (ECMAScript – ECMAScript 262 Edition 11)

Script roles (permission groups): superadmin

To begin we will be providing a help function that will utilize the Twilio API and create an HTTP post. It is beyond the scope of this document to explain the full Mango scripting environment. The helper function provided creates an HTTP post according to the Twilio API and injects your message and Twilio authentication tokens into the post.

Help function:
Function parameters: Body: The message that will be sent via text message

- From: Your Twilio number
- To: The receiving number
- Token: Your Twilio authentication token
- SID: Your Twilio account number

```
function sendSMS(body, from, to, token, SID) {
// Create the authorization header using the provided token and SID.
let auth = ${SID}:${token};
// Create an HTTP client and set up the request to send an SMS message using the Twilio API.
const client = HttpClients.createDefault();
const requestBuilder = RequestBuilder.post()
.setUri(https://api.twilio.com/2010-04-01/Accounts/${SID}/Messages.json)
.addHeader('Authorization', 'Basic ' + Base64.getEncoder().encodeToString(StandardCharsets.UTF_8.encode(auth).array()))
.addHeader('Content-Type', 'application/x-www-form-urlencoded');

// Add the message content, sender phone number, and recipient phone number as parameters in the request.
const params = [
new BasicNameValuePair('Body', body),
new BasicNameValuePair('From', from),
new BasicNameValuePair('To', to)
];

// Set the request entity to be the encoded form parameters.
try {
const entity = new UrlEncodedFormEntity(params);
requestBuilder.setEntity(entity);
} catch (e) {
console.error(e);
}

// Build the HTTP POST request and execute it using the HTTP client.
const httpPost = requestBuilder.build();
const response = client.execute(httpPost);

// Return the HTTP response object.
return response;
}
```

Next, we will add some code to the eventRaised function which is called when any event detector that has been linked to the event handler goes active.

When Mango calls this function it passes the event information into the eventRaised() function. We can use this to build our text messages. You can see below that we extract the alarm level, point tag, event message, and event timestamp and use a javascript string literal to create the text message. We then pass this as well as the other three variables to the sendSMS function.

```
function eventRaised(event) {
        let context = event.getContext();
        let body = `${event.getAlarmLevel()} event at ${context.point.getTags().Site}   -  ${event.
getMessageString()} has transitioned to active at ${event.getFullPrettyActiveTimestamp()}`
        sendSMS(body,'+1111222333','+1111222333',"YOUR TOKEN","YOURSID")
}
```

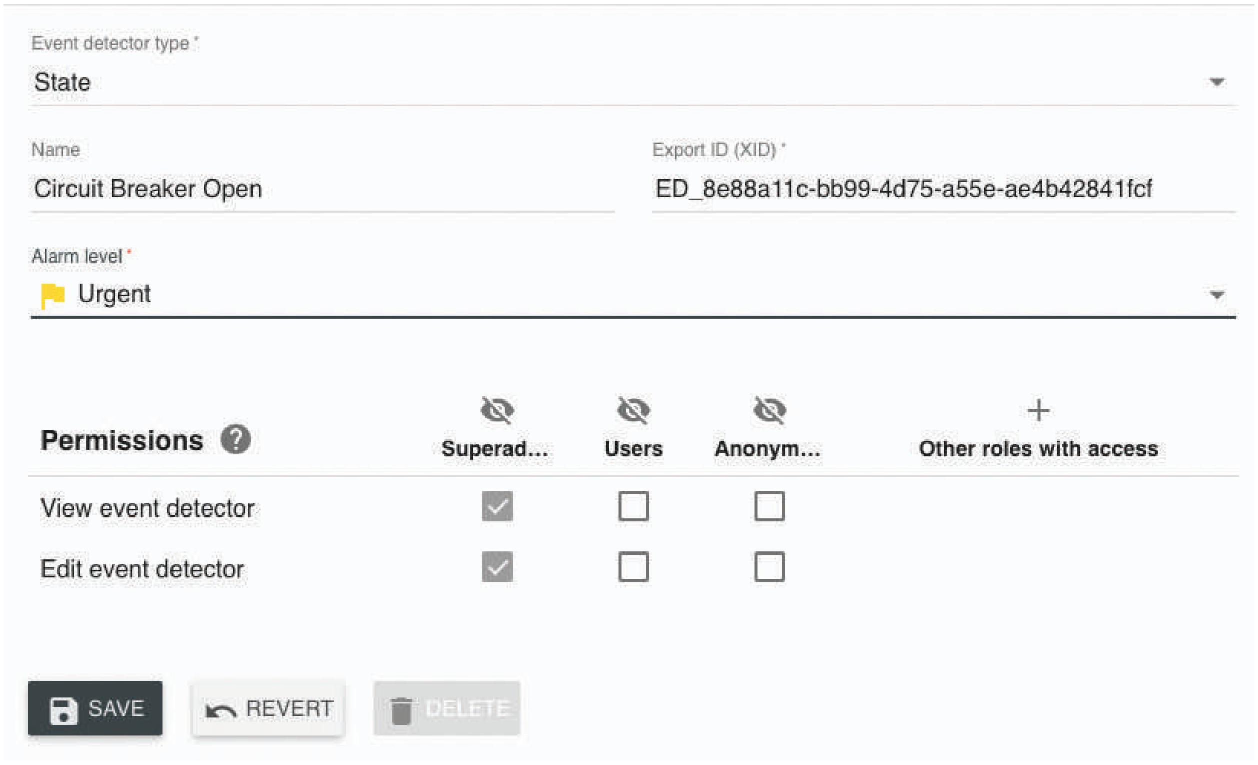
Lastly, we need to import some Java types for the sendSMS function. These will be put right at the top of the script.

```
const HttpClients = Java.type('org.apache.http.impl.client.HttpClients');
const RequestBuilder = Java.type('org.apache.http.client.methods.RequestBuilder');
const UrlEncodedFormEntity = Java.type('org.apache.http.client.entity.UrlEncodedFormEntity');
const BasicNameValuePair = Java.type('org.apache.http.message.BasicNameValuePair');
const Base64 = Java.type('java.util.Base64');
const StandardCharsets = Java.type('java.nio.charset.StandardCharsets');
```

This is how your code block should look at the end.

```
const HttpClients = Java.type('org.apache.http.impl.client.HttpClients');
const RequestBuilder = Java.type('org.apache.http.client.methods.RequestBuilder');
const UrlEncodedFormEntity = Java.type('org.apache.http.client.entity.UrlEncodedFormEntity');
const BasicNameValuePair = Java.type('org.apache.http.message.BasicNameValuePair');
const Base64 = Java.type('java.util.Base64');
const StandardCharsets = Java.type('java.nio.charset.StandardCharsets');

/**
 * Called when the Mango event is raised.
 */
function eventRaised(event) {
	let context = event.getContext();
	let body = `${event.getAlarmLevel()} event at ${context.point.getTags().Site}   -  ${event.getMessageString()} has 
transitioned to active at ${event.getFullPrettyActiveTimestamp()}`
	sendSMS(body,'+1111222333','+1111222333',"YOUR TOKEN","YOURSID")
}

/**
 * Called when the Mango event is
 * acknowledged (the event may still be active).
 */
function eventAcknowledged(event) {


}

/**
 * Called when the Mango event returns to normal
 * or is deactivated (e.g. on shutdown).
 */
function eventInactive(event) {

}

function sendSMS(body, from, to, token,SID) {


    let auth = `${SID}:${token}`
    console.log(auth)
    const client = HttpClients.createDefault();
    const requestBuilder = RequestBuilder.post()
        .setUri(`https://api.twilio.com/2010-04-01/Accounts/${SID}/Messages.json`)
        .addHeader('Authorization', 'Basic ' + Base64.getEncoder().encodeToString(StandardCharsets.UTF_8.encode(auth).
array()))
        .addHeader('Content-Type', 'application/x-www-form-urlencoded');

    const params = [
        new BasicNameValuePair('Body', body),
        new BasicNameValuePair('From', from),
        new BasicNameValuePair('To', to)
    ];

    try {
        const entity = new UrlEncodedFormEntity(params);
        requestBuilder.setEntity(entity);
    } catch (e) {
        console.error(e);
    }

    const httpPost = requestBuilder.build();

    const response = client.execute(httpPost);
    return response;
}
```

4. We are now ready to create an event detector on our point and link it to the event handler. Head back to the virtual data point and create an event detector. An important thing to note is that in the above example, we pull a tag from the data point. If your point does not have that tag the script will error out.



Event detector type *
State

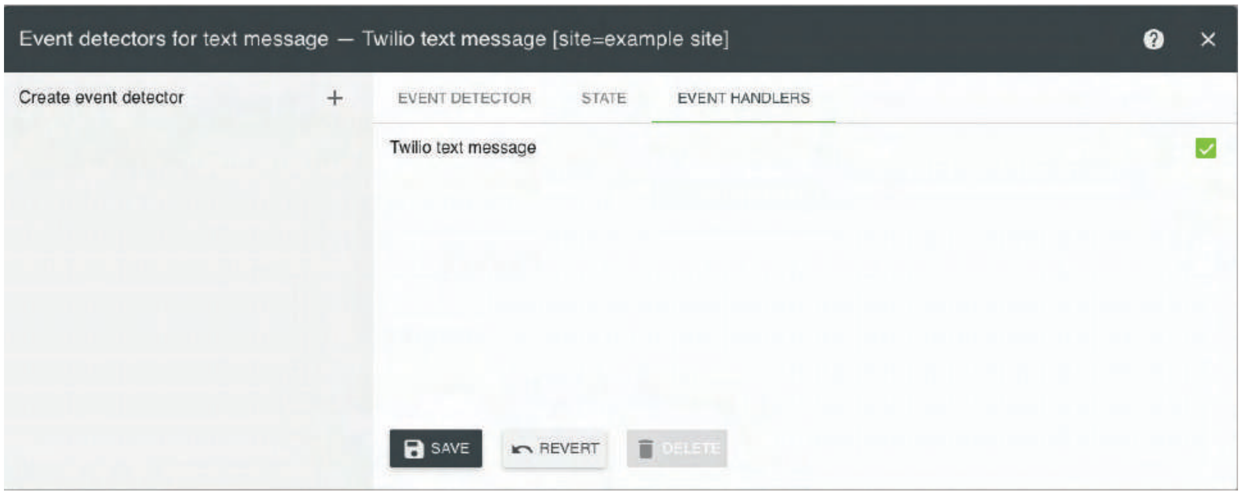
Name
Circuit Breaker Open

Export ID (XID) *
ED_8e88a11c-bb99-4d75-a55e-ae4b42841fcf

Alarm level *
Urgent

Permissions	Superad...	Users	Anonym...	Other roles with access
View event detector	✓	☐	☐	
Edit event detector	✓	☐	☐	

Now link the event detector to your event handler.



Event detectors for text message — Twilio text message [site=example site]

Twilio text message ✓

TEST THE CONFIGURATION

To test your configuration navigate to the point details page in Mango of the data point that you created above and toggle the value in a manner that will trigger the event detector you just configured. If all goes well you should receive a text message. If you do not get a message, see below for some troubleshooting tips.

Twilio trial account

- The free account uses a toll-free number which limits the amount of characters in a text message. If you are using a free account try reducing the text message length.

Authentication

- Log your HTTP response so you can debug. Your tokens may be incorrect. This will log the Twilio API response to your Mango ma.log file. An error code will be provided in the response. The definition of each error code can be found here : https://www.twilio.com/docs/api/errors

```
const response = client.execute(httpPost);
console.log(response)
return response;
```

Published by
Radix IoT 2023

Radix IoT, LLC
14555 N. Dallas Parkway #125
Dallas, TX 75254
United States

For more information, please contact
our Customer Support Center.
Email: info@radixiot.com

radixiot.com