ramtech Orama group

REACT

RAMTECHGLOBAL.COM

REACT Webhooks Configuration guide

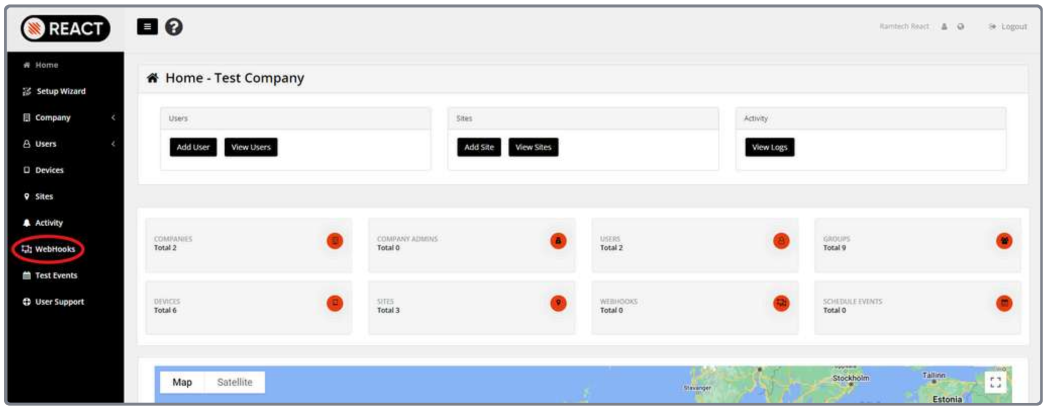
Webhooks can automatically trigger an external service as a result of an event in React. This can be useful e.g. for API integrations, where a user might wish to connect React with an external monitoring system.

Users supply an HTTP(S) address, and optionally configure payload fields, authentication and formatting. The webhook is then linked to one or more workflows in React and is triggered each time the relevant workflow stage is encountered. This results in an HTTP(S) payload arriving at the 3rd party system.

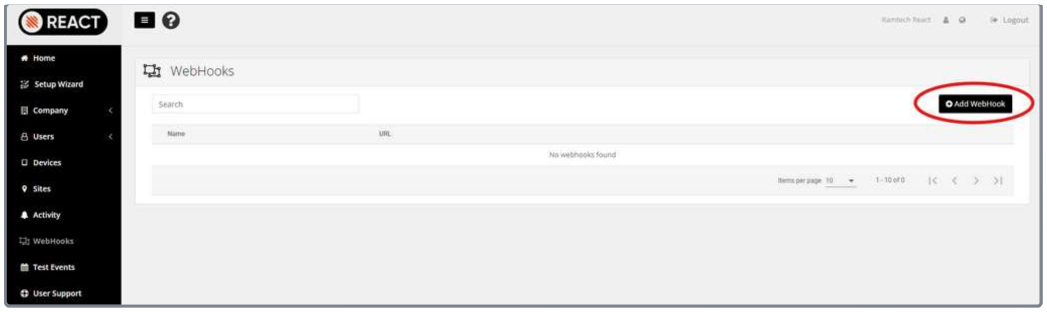
Configuration

Create Webhook

In the React web portal, click Webhooks in the left-had navigation menu.



Then click Add Webhook.



The fields for configuring a webhook are as follows:

- **Company**: If you are logged in as a Super Admin, you will be able to select the company against which the webhook will be registered. Otherwise, this field will be pre-populated with the company to which your account belongs.
- **Name**: This is just a label for the webhook. It must be unique per master company.
- **URL**: This is the HTTP(S) address to which data will be sent when the associated event is triggered. HTTP or HTTPS addresses can be used, but note that HTTPS addresses should have a valid SSL certificate installed.
- **Request Method**: This is the HTTP verb that will be used for the request, and can be any of GET, POST, PUT, PATCH and DELETE. Note that using GET will result in parameters being appended to the URL.
- **Format**: The default payload format is JSON. Custom formats are available, the names of which can be typed in directly having clicked on the custom radio button. If the custom format requested is not found a message to this effect will be given on screen. Some custom formats are very customer-specific, this is the reason they are not explicitly advertised.
- **Include Default Payload**: When switched on, the payload will include the below fields:
  - **Include Default Payload**: When switched on, the payload will include the below fields:
    - **eventTypeName**: The type of event, e.g. fire or SOS.
    - **eventTypeCode:** Four-digit, unchanged code aligning to the type of event
    - **eventTypeUUID:** Universally unique event type identifier
    - **locationName**: The location the event was raised from.
    - **siteName**: The name of the site from where the event was raised.
    - **latitude**: Geographical event location.
    - **longitude**: Geographical event location.
    - **createdAt**: Event timestamp in local timezone.
    - **createdAtUtc**: Event timestamp in UTC/GMT.
    - ***unitNumber**: Device identifier. This field is only populated when the event is raised by hardware such as a fire call point. It is not populated for events raised via the mobile app.
    - **eventUuid**: Universally unique event identifier.
    - **eventStatus**: Current event state, e.g. created, ended.
    - **siteId**: Alpha-numeric site ID, typically base station serial number. Note that a site can have multiple IDs, and as such this field is given in array syntax.
    - **siteUuid**: Site ID in React (as opposed to base station site IDs, above).
    - **deviceType**: The type of device that raised the event. This field is only populated when the event is raised by hardware (not the mobile app).

  ***As of release 1.18.0, the unitNumber field no longer contains a letter suffix to indicate hardware category. This field is now numeric only, and the deviceType field has been added to explicitly state the associated hardware type. So what might previously have been "0001F" is now simply "0001".*

When default payload is not selected, payload will be empty but for any custom fields configured (below).

- **Force Site Location**: With this option set, any devices that are not GPS capable, and so don’t supply event coordinates, will have their latitude and longitude substituted with that of the site they’re on.
- **Certificate**: You can optionally select an SSL certificate to facilitate authentication with the 3rd party. Configuring certificates is detailed below.
- **Custom Parameters**: You can optionally set any number of custom parameters to be send alongside or instead of the default payload above - see below.

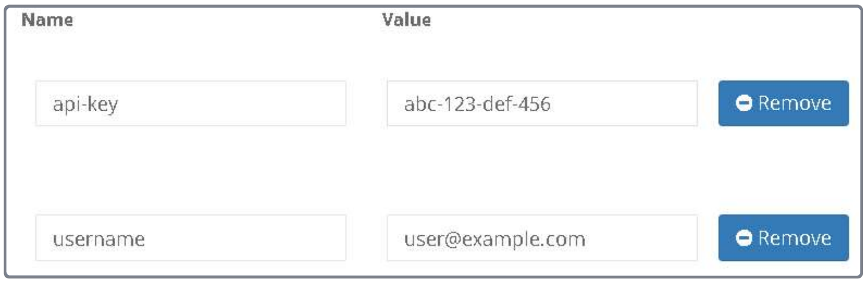
Custom Parameters

Body

Custom parameters can serve two purposes: either to send hard-coded key-value pairs, or to rename any of the fields in the default payload.

Hard-Coded Parameters

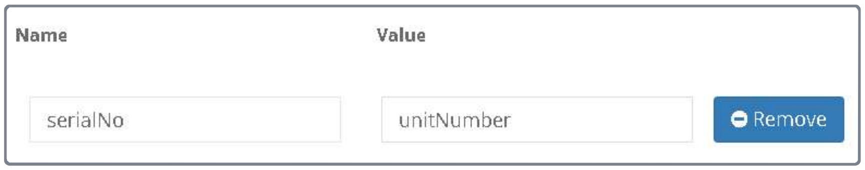
An example use-case for hard-coded custom parameters might be as below, where we’re sending an API key and username. These values will be sent unchanged every time the webhook is triggered.



Name	Value
api-key	abc-123-def-456
username	user@example.com

Renaming Parameters

The second use of custom parameters is to rename (or re-key, as the software term might be) elements of the default payload. Say for example the receiving system expected a serial number field, and the value of interest was held in default field *unitNumber*. We could set a custom parameter as below to rename this element of the default payload:



Name	Value
serialNo	unitNumber

If unitNumber is 123456, then the payload will include *“serialNo”: 123456*, and there will be no field named unitNumber.

What we’ve done is simply use the name (or key) of a default parameter as the value of a custom parameter. This can be done for any default parameter.

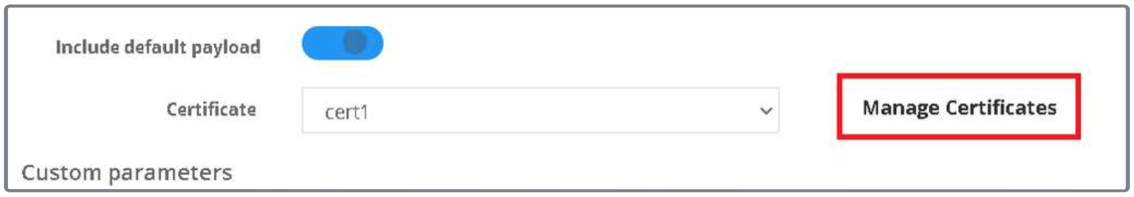
Note that you are not obliged to include the default payload when doing this - data will be plucked from the default set regardless and placed in your custom parameter.

Headers

Custom header parameters can be set in much the same way as body parameters, and these are of course added to the HTTP header. Be aware though that parameter renaming/re-keying as described above does not apply to headers, they’re just for hard-coded values.

SSL Certificates

If you need to use an SSL certificate for authentication you can configure them via the Manage Certificates link on the webhook config page:



From here you can add, edit and delete certificates.

When adding a certificate, you should provide files for the certificate itself and associate key file, and optionally provide passphrases for each (assuming they’re set on the respective files).

Please note that files and passphrases you upload here are not further encrypted, and are visible to other administrators within your company.

Once created, certificates can be selected in the webhooks form as above.

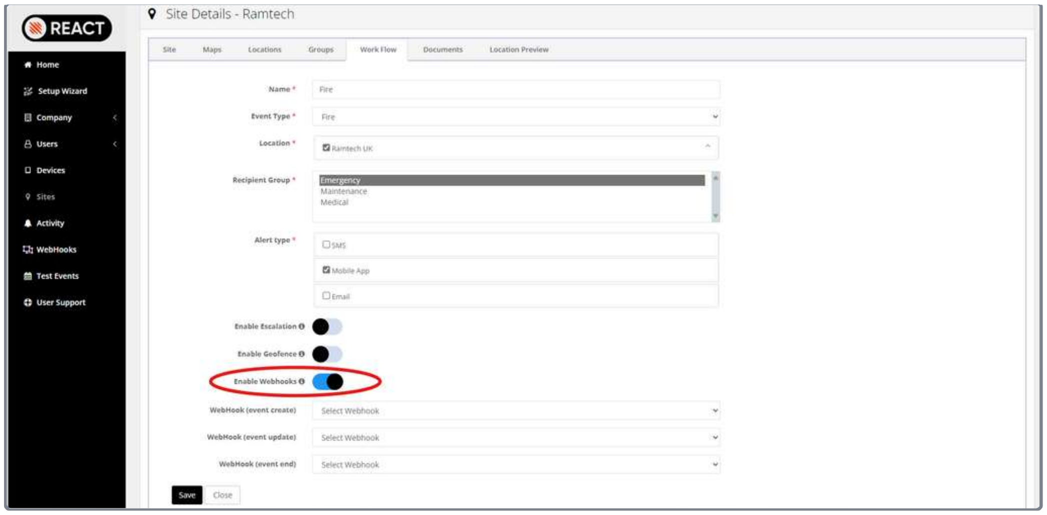
Responses

A typical response to a successful webhook request would be a HTTP 200 status. Note though that we do not enforce or act on any specific response within React - it’s a simple send and forget mechanism. Response codes are recorded in our server logs for debugging purposes should they be needed.

Connecting to a Workflow

Once a webhook has been configured it can be selected for triggering whenever a given workflow has itself been triggered.

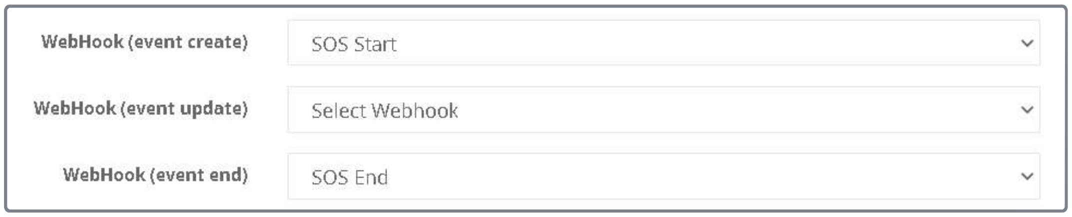
To enable webhooks for a workflow, you must first move the slider as below within the “Work Flow” tab of the site configuration.



Name *	Fire
Event Type *	Fire
Location *	☑ Ramtech UK
Recipient Group *	Emergency (selected), Maintenance, Medical
Alert type *	☐ SMS
	☑ Mobile App
	☐ Email
Enable Escalation	Off
Enable Geofence	Off
Enable Webhooks	On
WebHook (event create)	Select Webhook
WebHook (event update)	Select Webhook
WebHook (event end)	Select Webhook

Then, in the workflow config screen there are three stages against which you can register a webhook: created, updated and ended. You can register a webhook against any or all of these stages. The same webhook can be used for multiple stages, or you can select different ones according to your needs.

It might for example be useful to have different webhooks for different stages where an external system uses different event types rather than discreet stages, e.g. there might be distinct SOS start and SOS end event types, rather than a single SOS event type with multiple stages. In this scenario, you might configure webhooks as below, where one sends a custom parameter for SOS start and the other sends one for SOS end.



WebHook (event create)	SOS Start
WebHook (event update)	Select Webhook
WebHook (event end)	SOS End

Note that the event update stage is a generic hook point for any activity that signals neither the start nor end of an event. A typical example of an event update would be someone accepting ownership of an event via the React mobile app.

Example Payload

Below is an example JSON payload showing all default fields.

```
"eventTypeName": "Fire",
"eventTypeCode": 1000,
"eventTypeUuid": "a10482d2-50a8-4e6f-a481-b7c86a39a35f",
"locationName": "Construction Site / Main Building / Ground Floor / Call Point",
"siteName": "Construction Site",
"latitude": "52.9442069",
"longitude": "-1.1652912",
"createdAt": "04/07/2024 15:01:39",
"createdAtUtc": "04/07/2024 14:01:39",
"unitNumber": "0001",
"eventUuid": "8e558770-befe-4966-a778-2d3017526761",
"eventStatus": "created",
"siteId": [
  "W0202549D8"
],
"siteUuid": "a9f8b8c5-519c-47b3-8da9-b6bf3b7e968b",
"deviceType": "WES+_868 MHZ_Red_Call Point_Siren"
```

IP Address

Unchanged as of 11/08/2022, all React Webhooks should originate from 35.176.3.55. Updated 19/08/2024.

Change Log

1.28.0

- “Alarm Triggered By” renamed to “Inspection Delay”

1.25.0

- Added eventTypeCode and eventTypeUuid fields.

Event Type Name	eventTypeCode	WES Event Type	eventTypeUUID
Inspection Delay	1018	11	c4fc9326-ff24-4b29-972b-94f99901956c
Approval	1022	NULL	ac5ee9c0-97a5-4651-a516-029ad57fd717
Detector Tamper	1014	2	63e017b2-01c3-4c66-ae84-4eb842b02fc3
Evacuation	1009	NULL	5fe3a0ad-2437-45b3-8278-e348f78625b9
External Tamper	1004	3	6abe2afb-6337-490f-beee-713af115af7e
Fire	1000	0	a10482d2-50a8-4e6f-a481-b7c86a39a35f
Flat Battery	1016	7	ddaf71b9-5012-49dc-b4e9-2edd113c6aa7
GSM Device	1011	NULL	7385e76b-1d44-468e-af71-8da4c8a231cb
Hazard	1021	NULL	6d048581-58d4-4c5e-8650-6fa6da9gbde1
Heart Beat	1025	16	0751efef-5333-403f-94ec-4035116c99f5
Internal Tamper	1013	1	8bdeed4c-6ad2-41ab-a96a-166be9d9ac7a

Latched Unit	1001	10	71eaf444-14e7-4c4f-8cdb-a3adf2d9636d
Low Battery	1012	4	180fadb2-9274-418b-8fc9-900a9984c15e
Low Signal	1015	5	81927a19-e010-4575-b7d8-c31138f7651d
Medical	1003	9	9f551fa5-9fc0-4d85-b13f-89118cbd7b8c
Message	1026	NULL	f6cdf629-2c25-4353-ac50-2f22f083aa1d
Pager	1010	NULL	bcde5e05-be4c-4409-880c-1022e45780cf
SMS	1023	NULL	90aac187-ef61-4c05-8108-0f2d5e1cbb50
SOS	1008	NULL	1dbdf58f-ab6c-414b-9a9f-9e882d4f976e
Stop Work!	1024	NULL	7d548908-cac8-42b7-a445-6d1a9c93e02c
System Test	1006	6	e63ac2ba-02ef-4e5b-a700-ff5842ff5187
Test Message	1007	NULL	22a00e68-b2fe-498c-9a0b-795725566944
Transit Entry	1017	8	9c006bfd-10ae-4d03-8e2b-a84c1777d1c8
Unexpected Unit	1020	13	ecfdba48-0d61-44a3-b85b-5e80b3420102
Unit Info	1027	99	11222c91-993d-41e3-ac73-924589dcbb51
Unit Missing	1019	12	3ff21991-bfc7-4859-856b-23bfe82af8bc
Water	1005	17	45c896e4-3580-4e43-ad56-9f0e27523132

1.21.0

- Addition of Force Site Location option.

1.19.0

- Support for custom headers.
- Webhooks enabled/disabled slider in workflow config.

1.18.0

- Payload field unitNumber is now numeric only, there is no longer a letter suffix to indicate device type/category (new field devieType has been added for this).
- The following payload fields were added:
  - deviceType
  - siteUuid

Hardware specific payload fields such as unitNumber and deviceType will only be populated for alarms raised via hardware (alarms raised via mobile app will give location only).