Webhooks

Current version when this document was written	v5.3.38
Last verified as accurate	February 25, 2021

What are Webhooks?

Webhooks are automated messages, data, and commands sent from an app to a web repository when an event is triggered.

Webhooks can be sent from *CoreOne for Labs* to a **“Receiving URL“** when any specified trigger events are initiated.

You can think of them as push notifications.

Webhooks v APIs

Webhooks are similar to APIs—but simpler, instantaneous, and automated.

During the early days of email, you had to check to see if you received anything. Now, however, you get a push notification when an email is delivered to your inbox. The push notification contains some or all of the email data so you can determine how urgent it may be. This is why using webhooks is more efficient than an API for events in *CoreOne for Labs*.

For example, when a new accession is created, a webhook can submit desired accession data to another system. Rather than hitting the server 720 times an hour for a 5 second refresh rate to ask if there is new data, data can be instantaneously submitted/transmitted when it occurs (3-5 times per hour on avg).

Implementing Webhooks for CoreOne for Labs

For example, when a new accession record is created, a webhook is triggered. The specified data from the newly created accession record would be sent to the receiving URL. Once the data submission is validated, it is delivered as designed.

1. Set up a secret and a **“Receiving URL.”** The secret validates incoming webhook data.

2. Tell *CoreOne for Labs*/the webhook when and where to send the specified data. For example, every time a new accession is created, send accession #, x, y, z to **“Receiving URL.”**

3. Send a test.

4. Do something with the data submission: slack notification, data to another app, etc.

Configuration

You need to specify WebhookURL and WebhookSecret

All Webhooks are sent to the configured WebhookURL - if there are some you aren't interested in, you should just ignore them.

The WebhookURL will receive an HTTP/HTTPS POST from CoreOne for Labs

- The message body will contain a JSON packet. It will contain a hash which varies depending on the specific Webhook as described in the section below.
- A HTTP header called `X-Coreone-Webhook-Type` will be provided, indicating the type of Webhook that has been triggered; these are detailed below.
- A HTTP header called `X-Coreone-Webhook-Hash` will also be provided. This allows you to verify that the message hasn't been tampered with. It is a hex digest of the JSON data sent in the body of the webhook. It is calculated using HMAC with SHA-256 hash function with the key being the value stored in WebhookSecret.

Available Webhooks

Accession

accession/status_change This webhook is fired when an accession's status has been changed.

accession/client_change This webhook is fired when an accession's client has been changed.

accession/receive This webhook is fired when an accession has been marked as 'Received'.

Accession Lab Section

accession_lab_section/create This webhook is fired when an accession has been assigned a lab section.

accession_lab_section/destroy This webhook is fired when an accession's lab section has been removed.

accession_lab_section/release This webhook is fired when an accession's lab section has been released for review.

accession_lab_section/unrelease This webhook is fired when an accession's lab section has been unreleased.

Attachment

attachment/fail This webhook is fired when an attachment's processing has failed.

attachment/process This webhook is fired when an attachment has been processed successfully.

Client

client/create This webhook is fired when a new client has been created.

client/destroy This webhook is fired when a client record has been deleted.

client/update This webhook is fired when a client record has been updated. This includes when the client's credit status has been updated.

Clinical Histories

clinical_histories/create This webhook is fired when the clinical history has been entered for an animal.

clinical_histories/destroy This webhook is fired when the clinical history of an animal has been deleted.

clinical_histories/update This webhook is fired when the clinical history of an animal has been edited.

Comment

comment/create This webhook is fired when a new comment has been added to an accession.

comment/destroy This webhook is fired when a comment has been deleted from an accession.

comment/update This webhook is fired when a comment has been edited.

Credit Note

credit_note/create This webhook is fired when a credit note has been created.

HL7 Message

hl7message/fail This webhook is fired when an HL7 message's processing has failed.

hl7message/process This webhook is fired when an HL7 message has been processed successfully.

Instrument

instrument/fail This webhook is fired when an instrument message's processing has failed.

instrument/generate_instrument_file This webhook is fired when a user in the UI generates an instrument file. This file contains *only* the instrument name and filename of the generated order file.

instrument/process This webhook is fired when an instrument message has been processed successfully.

instrument/request_instrument_file This webhook is fired when a user in the UI generates an instrument file. This file contains the exact parameters the user requested to be further used with the API endpoint `/api/v1/instruments/generate_file` to generate an order file.

Invoice

invoice/client_change This webhook is fired when the client on an invoice has been changed.

invoice/close This webhook is fired when an invoice has been closed.

invoice/create This webhook is fired when a new invoice has been created.

Leaf

leaf/fail This webhook is fired when an incoming scan has been processed but has not been automatically matched to an accession.

leaf/match This webhook is fired when an incoming scan has been processed and matched with an accession because it had a *CoreOne for Labs* barcode which was successfully identified.

NAHLN

nahln_transmission/fail This webhook is fired when a NAHLN transmission's processing has failed.

nahln_transmission/process This webhook is fired when a NAHLN transmission has been processed successfully.

Payment

payment/create This webhook is fired when a payment has been made.