Webhooks - GO1

Using our proprietary webhook system (v3), you can receive real-time notifications of the events happening in the Go1 ecosystem. For example, you can configure the webhook via the Public API that can notify the developer whenever the content in Go 1 is completed or decommissioned.

In this documentation, we have covered the following topics that will help you navigate throughout the webhook system of Go1:

- Why Use Webhooks
- Available Webhook Endpoints
- Event Types
- The Webhook Header
- Creating A New Webhook
- Edit a Webhook
- Enable/Disable Webhook
- Webhook Validation with Signature
- Webhooks Authentication
- Webhooks Retries
- The Webhook Event Object

Let’s get started 🚀

Why Use Webhooks

Go1 has upgraded its webhook version, offering you relatively improved usability with several key advantages for utilizing the webhook service.

- Webhooks for enrollment events will trigger when the event occurs on any connected child portal. This means you only need to create one enrollment webhook for all your customers.

- The Go1 webhook service also supports retries. If a webhook was not successfully received, for instance then Go1 will attempt to send the webhook again.

- The webhook offers to keep the data synched between the developer’s own application and the Go1 platform.

The Webhooks for Developers

Go1 webhooks offers unique advantages to keep the data synchronized between the developer's own application and Go1.

With Go1 webhook system, the developers receive real-time notifications (as soon as any change is executed) without the need to poll out the Go1 API to check for the "changes" executed by an object, hence, keeping the developer's application always synced.

- With real-time notifications generated by the webhook, the developers can keep their application up to date with respect to all the events occurring in Go1.
- The developer's system will make fewer calls to the Go1 API and far less data will need to be relayed between the system.
- The developers will not need to implement the logic to "diff" the state of your system with what is stored in the Go1 platform.

Available Webhook Endpoints

Go1 provides the following types of webhook endpoints:

Webhook Endpoint	Type	Description
`POST /webhooks`	POST	Creates a new webhook that will fire events to the given url.
`GET /webhooks`	GET	Returns a list of your webhooks.
`PATCH /webhooks/{id}`	PATCH	Updates any of the following fields on an existing webhook: name, url, secret_key, event types, and status.

Event Types

Go1 offers two types of webhook events **enrollment complete** and **content decommission.**

Event Type	Event Trigger Description
`enrollment.complete`	Triggered whenever a user completes a learning object. When configured for a master portal, the webhook fires for all child portal completion events.
`content.decommission`	Triggered when a Go1 course is flagged for retirement by its

	content provider.

The Webhook Header

The header is the standard to develop a webhook, the key-value parameters that are sent as part of the webhook request.

The upgraded v3 Webhooks are hosted on the Go1 v3 Public API.

The following header is generated by the webhook service and will be sent with all requests.

GO1 Sample Header

```
curl -X POST "https://gateway.go1.com/webhooks" \
-H "Content-Type: application/json" \
-H "Authorization: {access_token}" \
-H "Api-Version: 2022-07-01"
```

GO1 Header Description

Header	Description
`Content-Type`	The representation header is used to indicate the original media type of the resource.
`Authorization`	Allowing access to a protected resource
`Api-Version`	The api version of the resource. **Important**: When sending a v3 API request, the users are required to explicitly set the API version by sending an `Api-Version` header with your request. For example: "Api-Version: 2022-07-01"

Creating a New Webhook

Go1 can be configured and sent as an HTTP POST request to any URL.

POST /webhooks

Step 1: Click on **create a webhook**.

Step 2: Click on the **Try it** button to configure the webhook.

Step 3: A panel will appear at the right with the sections described below

Section	Description
Parameters	You can send the parameters with your webhook request by adding the parameters if any.
Headers	You can configure the Headers with the values and you can add more headers if required.
Body	You will have a sample request body in which you can also customize the request body as per your needs.
HTTP Request	By default HTTP request is selected. You can have multiple methods to execute the POST webhook using different programming languages.

Step 4: Click on the **Send** button to execute the webhook request with the parameters defined.

Parameters (Key)	Type	Description	Required/Optional
name	string	The name or description of what the webhook is used for.	Optional
url	string	The URL of the webhook endpoint	Required
secret_key	string	An optional secret key, identifying each event as coming from a trusted Go1 source.	Optional
event_types	string	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]	Required

Response Code

If the POST URL hit the API then it will return the response code to the user to acknowledge whether the POST URL is executed successfully or has some error(s).

API Response Code	Description
201	Webhook was created successfully.

400	Invalid POST parameters provided.
403	Invalid permission to create the webhook.

Response Object

The 201 code will return the response object:

```
{
 "id": "xUjKifleke4u",
 "portal_id": "11905591",
 "name": "My webhook for course completion",
 "url": "https://webhook-consumer.com",
 "secret_key": "MyS3cretK#y",
 "event_types": ["enrollment.complete"],
 "status": "active",
 "created_by": "19491473",
 "created_date": "2022-03-29T01:29:36.000Z"
 "last_updated_by": "19491473",
 "last_updated_date": "2022-03-29T01:29:36.000Z"
}
```

Object Properties	Description
id	ID for the webhook configuration.
portal_id	Go1 portal against which the webhook is configured
name	Name of the webhook
url	The webhook URL
secret_key	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
status	The status of the webhook. It can be active or inactive.
created _by	The user ID of the user who created the webhook.
created_date	Time at which the webhook was created (ISO 8601 in UTC).
last_updated_by	Webhook updated by whom
last_updated_date	The date when the webhook was last updated.

The other codes will not return anything in the form of an object.

GET /webhooks

The GET method is used to retrieve all the webhooks available in the Go1 portal. It returns a list of webhooks in the form of an object response.

Response Code

If the GET URL hit the API then it will return the response code to the user to acknowledge whether the GET URL is executed successfully or has some error(s).

API Response Code	Description
200	The GET URL executed successfully with returning the webhooks object
403	Invalid permission to fetch the webhooks.

Response Object

The 201 code will return the list of webhooks in the form of an object:


```
{
 "total": 1,
 "hits": [
 {
 "id": "xUjKifleke4u",
 "portal_id": "11905591",
 "name": "My webhook for course completion",
 "url": "https://webhook-consumer.com",
 "secret_key": "MyS3cretK#y",
 "event_types": [
 "enrollment.complete"
 ],
 "status": "active",
 "created_by": "19491473",
 "created_date": "2022-03-29T01:29:36.000Z"
 "last_updated_by": "19491473",
 "last_updated_date": "2022-05-19T01:20:16.000Z"
 }
 ]
}
```

Object Properties	Description
total	The total number of webhooks returned.
hits	The array of webhook objects.
id	ID for the webhook configuration.
portal_id	Go1 portal against which the webhook is configured
name	Name of the webhook
url	The webhook URL
secret_key	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
status	The status of the webhook. It can be active or inactive.
created _by	The user ID of the user who created the webhook.
created_date	Time at which the webhook was created (ISO 8601 in UTC).
last_updated_by	Webhook updated by whom
last_updated_date	The date when the webhook was last updated.

Edit a Webhook

You can update a webhook in the Go1 portal by providing a PATCH request in the HTTPS request URL.

By sending the params in the body, you can update the webhooks fields such as name, url, secret_key, event_types, fire_child_portal_events, and status.

PATCH /webhooks/{id}

This is the PATCH webhook URL you need whenever you are going to update the webhooks. The {id} in the URL is defined by the content you want to update.

PATCH Request Sample

```
{
 "name": "My new webhook name",
```

```
 "url": "https://newUrlForWebhook",
 "secret_key": "my-secret-key-changed",
 "event_types": ["enrollment.complete"],
 "fire_child_portal_events": true,
 "status": active
}
```

Parameters (Key)	Type	Description
name	string	A name or description of what the webhook is used for.
url	string	The URL of the webhook endpoint.
secret_key	string	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	string	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
fire_child_portal_events	string	The parameter can be configured to include events triggered by all connected portals that can be defined as a boolean value. For example: “true” or “false”
status	string	The status of the webhook. It can be active or inactive.

Response Code

If the PATCH URL hits the API then it will return the response code to the user to acknowledge whether the PATCH URL is executed successfully or has some error.

API Response Code	Description
200	The Webhook was updated successfully.
401	The user is unauthenticated.
403	The user is not authorized.
404	The Webhook is not found.
500	The internal server error.

Response Object

The 200 code will return the response object of the webhook you have updated.

```
{
 "id": "xUjKifleke4u",
 "portal_id": "11905591",
 "name": "My new webhook name",
 "url": "https://newUrlForWebhook",
 "secret_key": "my-secret-key-changed",
 "event_types": ["enrollment.complete"],
 "status": "active",
 "created_by": "19491473",
 "created_date": "2022-03-29T01:29:36.000Z"
 "last_updated_by": "19491473",
 "last_updated_date": "2022-05-19T01:20:16.000Z"
}
```


Object Properties	Description
id	ID for the webhook configuration.
portal_id	Go1 portal against which the webhook is configured
name	Name of the webhook
url	The webhook URL
secret_key	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
status	The status of the webhook. It can be active or inactive.
created _by	The user ID of the user who created the webhook.
created_date	Time at which the webhook was created (ISO 8601 in UTC).
last_updated_by	Webhook updated by whom
last_updated_date	The date when the webhook was last updated.

Enable/Disable Webhook

PATCH /webhooks/{id}

The users can activate or deactivate the webhooks by PATCH URL to define the “active”/”inactive” in the status parameter.

PATCH Request Sample

```
{
 "name": "My new webhook name",
 "url": "https://newUrlForWebhook",
 "secret_key": "my-secret-key-changed",
 "event_types": ["enrollment.complete"],
 "fire_child_portal_events": true,
 "status": active
}
```

Parameters (Key)	Type	Description
name	string	A name or description of what the webhook is used for
url	string	The URL of the webhook endpoint.
secret_key	string	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	string	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
fire_child_portal_events	string	The parameter can be configured to include events triggered by all connected portals and that can be defined as a boolean value. For example: “true” or “false”
status	string	The status of the webhook. It can be active or inactive.

Response Code

If the PATCH URL hit the API then it will return the response code to the user to acknowledge whether the PATCH URL is executed successfully or has some error.

API Response Code	Description
200	The Webhook was updated successfully.

401	The user is unauthenticated.
403	The user is not authorized
404	The Webhook is not found.
500	The internal server error

Response Object

The 200 code will return the response object of the webhook you have updated.

```
{
 "id": "xUjKifleke4u",
 "portal_id": "11905591",
 "name": "My new webhook name",
 "url": "https://newUrlForWebhook",
 "secret_key": "my-secret-key-changed",
 "event_types": ["enrollment.complete"],
 "status": "active",
 "created_by": "19491473",
 "created_date": "2022-03-29T01:29:36.000Z"
 "last_updated_by": "19491473",
 "last_updated_date": "2022-05-19T01:20:16.000Z"
}
```

Object Properties	Description
id	ID for the webhook configuration.
portal_id	The Go1 portal against which the webhook is configured
name	Name of the webhook
url	The webhook URL
secret_key	An optional secret key, identifying each event as coming from a trusted Go1 source.
event_types	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
status	The status of the webhook. It can be active or inactive.
created_by	The user ID of the user who created the webhook.

created_date	Time at which the webhook was created (ISO 8601 in UTC).
last_updated_by	Webhook updated by whom
last_updated_date	The date when the webhook was last updated.

Webhook Validation with Signature

Verification of Go1 Request

In order to secure the webhooks, the system verifies the request which is coming from the Go1 portal to hit the webhook endpoints.

Receiving the API hit calls from Go1 can optionally sign the webhook events which are sent to your endpoints by including a signature in each request's header. This allows you to verify that event requests were sent by Go1 and not by an unauthorized third party.

In order to activate the signature, create or update your webhook with a "secret_key" as described in our API Documentation.

Note: The secret key length cannot exceed 64 characters.

After setting up a "secret_key" all requests made to your webhook endpoint will include a "Go1-Signature" value in the header. An example payload could look like this:

```
POST /webhooktest HTTP/1.1
Host: api.go1.com
Content-Length: 252
Content-Type: application/json
Go1-Signature:
t=1588141753,v1=fa818dc59df5b5c7c42517dedf24d85ca2184201d03ed24bb6e2c7bf766
b7e89
User-Agent: GO1 API v1.0

{
"type":"user.create",
"fired_at":"2020-04-29T06:29:13+0000",
"data": {
"Id":"8191190",
"email":"new.user@go1.com",
"full_name":"New User",
"first_name":"New",
"last_name":"User",
```

```
"profile_picture":null,
"status":null,
"created_time":"1588141753",
"account":null
}
}
```

Verification of Webhook Signature

As you have observed in the code block above, the Go1-Signature is included in the header.

```
Go1-Signature:
t=1588141753,
v1=fa818dc59df5b5c7c42517dedf24d85ca2184201d03ed24bb6e2c7bf766b7e89
```

The signature is generated by two partitions:

Signature Property	Description
“t”	The timestamp when the signature is created. The timestamp can be used to prevent replay attacks, by rejecting requests containing a timestamp that is too far from the past.
“v”	This contains the signature strings. The signature itself is generated using a hash-based message authentication code (HMAC) with the SHA-256 algorithm.

Step 1: Extract the timestamp and signature from the request header. Split the header value of "Go1-Signature", using the "," character as the separator, to retrieve an array of components. Then split each component, using the "=" character as the separator, to get a key and value pair.

The value for the key t corresponds to the timestamp, and v1 corresponds to the signature.

Step 2: Calculate your own signature. Concatenate the following components to a string:

- The timestamp

- The "." character

- The JSON payload as a string,

For example, for the payload you should have a string looking like this:

```
'1588141753.{
"type":"user.create",
"fired_at":"2020-04-29T06:29:13+0000",
"data": {
"id":"8191190",
"email":"new.user@go1.com",
"full_name":"New User",
"first_name":"New",
"last_name":"User",
"profile_picture":null,
"status":null,
"created_time":"1588141753",
"account":null
}
}'
```

Now use the "**secret_key**" which you have set up for this webhook earlier to compute an HMAC with the SHA256 hash function:

This is the PHP and Node.js example hash function:

```
#PHP Example:
$calculatedSignature = hash_hmac('sha256', 'TIMESTAMP.JSON_PAYLOAD',
'MY_SECRET_KEY');

#Node.JS Example
const crypto = require("crypto");
let hmac = crypto.createHmac("sha256", 'MY_SECRET_KEY');
let calculatedSignature = hmac.update(Buffer.from('TIMESTAMP.JSON_PAYLOAD',
'utf-8')).digest("hex");
```

Step 3: Compare the signature you have received in the request with the expected one which you calculated yourself. If they match, the request and payload come from a sender who knows your **“secret_key”**.

Security with the Webhooks

In addition to validations the webhooks are allocated the IP Ranges. The Go1 Platform is using the IP range 20.188.251.48/28 (Australia East) and 13.67.161.224/28 (Central US) for traffic from its Web/API fleet. This whole range is solely allocated to Go1. You can expect connections from webhooks to come from those IPs and allow them.

Webhooks Authentication

The access_token used in the authorization parameter of the webhook header helps in navigating the Go1 portal on which the user will receive the webhook events.

Go1 uses the OAuth 2.0 standard as a framework for third parties to gain authorization and make API requests on behalf of a Go1 user.

In Go1, following OAuth 2.0 specifications, OAuth clients are used to request access authorization from users, to act on their behalf, and will produce *tokens* that can be submitted to APIs to verify requests.

For more information about the Authentication and Authorization, you can visit the link here.

Webhooks Retries

When a webhook is fired, a retry pattern will trigger if either of the following occurs:

Go1 receives no response from your configured endpoint within 10 seconds.
Go1 receives an HTTP response code that is outside the 200 range (eg, 500 - Internal Server Error)

The retry pattern is as follows:

Attempts	Time Slots
Attempt 2	After 30 seconds
Attempt 3	After 15 minutes (later than attempt 2)
Attempt 4	After 4 hours (later than attempt 3)
Attempt 5	After 24 hours (later than attempt 4)

Important note: At this time, if the 5th attempt does not receive a success response, then the webhook will fail quietly without an alert or notification.

The Webhook Event Object

Once a webhook is created, the webhook event object will be fired to the given endpoint URL that is specified when creating your webhook.

Sample Webhook Event Object:

```
{
  "id": "hg4JWUDbT55B",
  "event_type": "enrollment.complete",
  "webhook_version": "3.0.0",
  "sent": "2022-03-29T01:29:36.676Z",
  "attempt_number": 1,
  "url": "https://webhook.site/dd15f234-5160-43e5-b405-a6765cdc8292?",
  "webhook_id": "auto generate uuid will be here",
  "data": {
    "id": "101916887",
    "lo_id": "741712",
    "portal_id": "11861197",
    "event_time": "2022-03-29T01:29:36.000Z"
  }
}
```

Object Properties	Description
id	Unique identifier for the object. Represents the webhook send attempt and will be consistent across all retries.
event_type	The event for which the webhook was triggered. i.e. ["enrollment.complete"] or ["content.decommission"]
webhook_version	Hardcoded 3.0.0 until subsequent versions are also supported.
sent	Time at which the webhook attempt was sent (ISO 8601 in UTC).
attempt_number	The attempt number as per the retry pattern.
url	The endpoint URL of the configured webhook.
webhook_id	Auto-generated ID for the webhook object itself.
data	Data object as per event type configured (below)

`

Sample Data Object

```
{
    "id": "101916887",
    "lo_id": "741712",
    "portal_id": "11861197",
    "event_time": "2022-03-29T01:29:36.000Z"
```

```
}
```

Object Properties	Description
id	The ID of the corresponding enrolment.
lo_id	The learning object for which the completion occurred.
portal_id	The Go1 portal against which the completion occurred.
event_time	Time at which the enrolment was completed (ISO 8601 / UTC).