

Receive API

Author Kodmyran AB

Introduction

The receive API accepts incoming webhooks: HTTP calls that another system makes to Kodmyran Commerce when something happens on its side, such as a warehouse system reporting that an order has shipped. It is the opposite of the outgoing webhooks, where Kodmyran Commerce calls you.

A receiver is an extension point rather than a finished integration. The platform takes care of the URL, the rate limit and the logging; what happens to the message is decided by the receiver's implementation, a PHP class. The one shipped with the platform puts every message on a queue for a Wolfpack worker to process. Anything specific to a sender, such as checking its signature, goes in an implementation of your own.

Setting up a receiver

Receivers are created in the admin under *Configuration* → *API* → *Receiver webhook*. Each receiver has:

Field	Meaning
Name	A name for your own reference
Implementation	The fully qualified name of the class that handles the message
Argument	One free text value passed to the implementation, e.g. the name of a queue
Active	An inactive receiver refuses all calls
Magic	A 32 character identifier, generated when the receiver is saved

The magic cannot be edited. To change it, create a new receiver and deactivate the old one once the sender has switched over.

Endpoint

The sender makes a POST to the receiver's URL, ending in its magic:

```
POST https://testaccount.kodmyran.io/admin/api/receive/3f9a1c0e7b2d4e6f8a1b3c5d7e9f0a2b
Content-Type: application/json
```

```
{"event": "shipped", "order": "100245"}
```

The call needs no API key or other credentials, so a sender can be pointed at it without any setup on its side.

Anyone who knows the URL can post to it: the magic is the only thing that identifies the caller. Treat the URL as a secret, and if the sender signs its messages, check the signature in the implementation.

The body is passed to the implementation exactly as received. The Content-Type must still be one the REST API accepts; for a webhook that means application/json, or multipart/form-data. A charset parameter is fine.

Responses

Situation	Answer
Anything but POST	405
Another Content-Type than above	400, "Bad content-type"
Unknown or malformed magic	400, "Invalid receive webhook"
The receiver is not active	400, "Requested receive webhook is disabled"
The rate limit is reached	429

Otherwise the answer is whatever the implementation prints. Errors are sent as JSON in the same form as the other APIs:

```
{"callStatus":"FAIL","message":"Invalid receive webhook"}
```

Every call is logged in the API log, with the body and headers received and the output returned.

Rate limit

Receivers are limited to 120 calls a minute by default. The limit is shared by all receivers in the account, so one busy sender uses up the allowance of the

others. As in the other APIs, the X-Commerce-APILimit and X-Commerce-RemainingAPILimit headers tell the sender where it stands.

The AMQP receiver

The receiver shipped with the platform is `Kodmyran\Commerce\Admin\API\Receivers\AmqpReceiver`. Its argument is the name of an AMQP queue. It wraps the body with some metadata and publishes it on that queue:

```
{
  "meta": {
    "receivedAt": 1760180400,
    "receivedBy": "AMQP Webhook Receiver",
    "sourceIP": "203.0.113.10",
    "sourcePort": "51234",
    "serverName": "testaccount.kodmyran.io",
    "requestUri": "/admin/api/receive/3f9a1c0e7b2d4e6f8a1b3c5d7e9f0a2b"
  },
  "data": {
    "payload": "{\"event\": \"shipped\", \"order\": \"100245\"}"
  }
}
```

The payload is the body as a string, not decoded. The sender then gets:

```
{"callStatus":"OK","message":"Message accepted and put into queue"}
```

The work itself is done by a Wolfpack worker listening on the queue. Answering at once and processing later keeps the call short, which most senders require, and lets the worker retry without involving the sender.

The AMQP receiver does not check anything about the message. It suits senders you trust, or messages that are only a hint to fetch the real data from the sender.

Writing your own receiver

An implementation is a class implementing `Kodmyran\Commerce\Core\ApiReceiveWebhook`:

```
interface ApiReceiveWebhook
{
    public function onMessage(AdminLib $adm, string $req, array $hdrs, string $arg): void;
}
```

Parameter	Content
\$adm	The admin library, for database access, configuration, AMQP and logging
\$req	The body as received
\$hdrs	The request headers, as an array from header name to value
\$arg	The receiver's argument

Everything the method prints is sent to the sender and logged. Set the Content-Type with header() if the sender cares about it. Throwing an exception answers with an error.

The usual way to build one is to copy AmqpReceiver into your custom code and add what the sender needs, typically a check of its signature before the message is queued. Each sender signs in its own way, with its own header, secret and algorithm, which is why the platform has no generic check. A sketch:

```
class WmsReceiver implements ApiReceiveWebhook
{
    public function onMessage(AdminLib $adm, string $req, array $hdrs, string $arg): void
    {
        // Header names arrive as the sender wrote them, and in lower case over HTTP/2
        $hdrs = array_change_key_case($hdrs, CASE_LOWER);
        $secret = $adm->getConfig("custom.wms.webhookSecret", "", 0);
        $expected = hash_hmac('sha256', $req, $secret);

        if (!strlen($secret) || !hash_equals($expected, strval($hdrs['x-signature'] ?? "")))
            throw new S4SFieldValueException('Bad signature');
    }

    // From here on, as AmqpReceiver: wrap the message and publish it on the queue in $arg
}
```

Keep the secret in a setting rather than in the receiver's argument, which is shown in the admin.

Enter the class's fully qualified name as the receiver's implementation. A class that does not implement the interface is refused with "Invalid webhook implementation".

Payment providers are the exception: their callbacks are handled by the payment module itself, not by a receiver.