PROJEQTOR

WEB HOOK

TECHNICAL DOCUMENTATION

VERSION: V 1.0

REFERENCE: PLUGIN – WEBHOOK

SOMMAIRE

INTRODUCTION 3
SETUP 3
WEBHOOK FEATURE 4
AUTHENTICATING THE WEBHOOK NOTIFICATION 4
TRIGGER A WEBHOOK 5
EXAMPLE OF USE 5

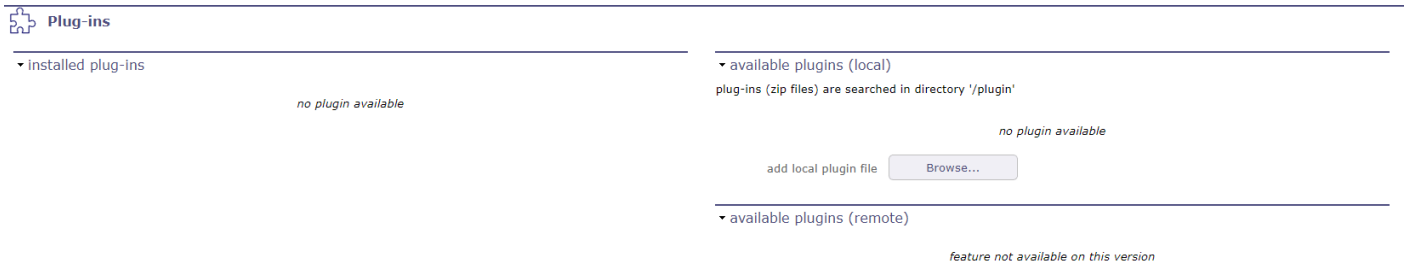
INTRODUCTION

The purpose of this document is to describe how the "Webhook" plugin works.

Webhooks are "user-defined HTTP callbacks". They are usually triggered by some event, such as pushing code to a repository.

SETUP

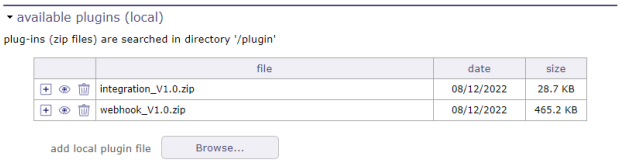
When acquiring a plugin, you get a .zip file. To install it, go to the Plugins Management screen.



In the section of installed plugins, you have the list of plugins already in place with the version of the latter and on which version of ProjeQtOr you installed it.

In the available plugins section, click on the browse button to upload your .zip file.

Once the file is uploaded, it appears in a table.
Note: if your zip file is a package containing several plugins, all plugins will appear in the table.



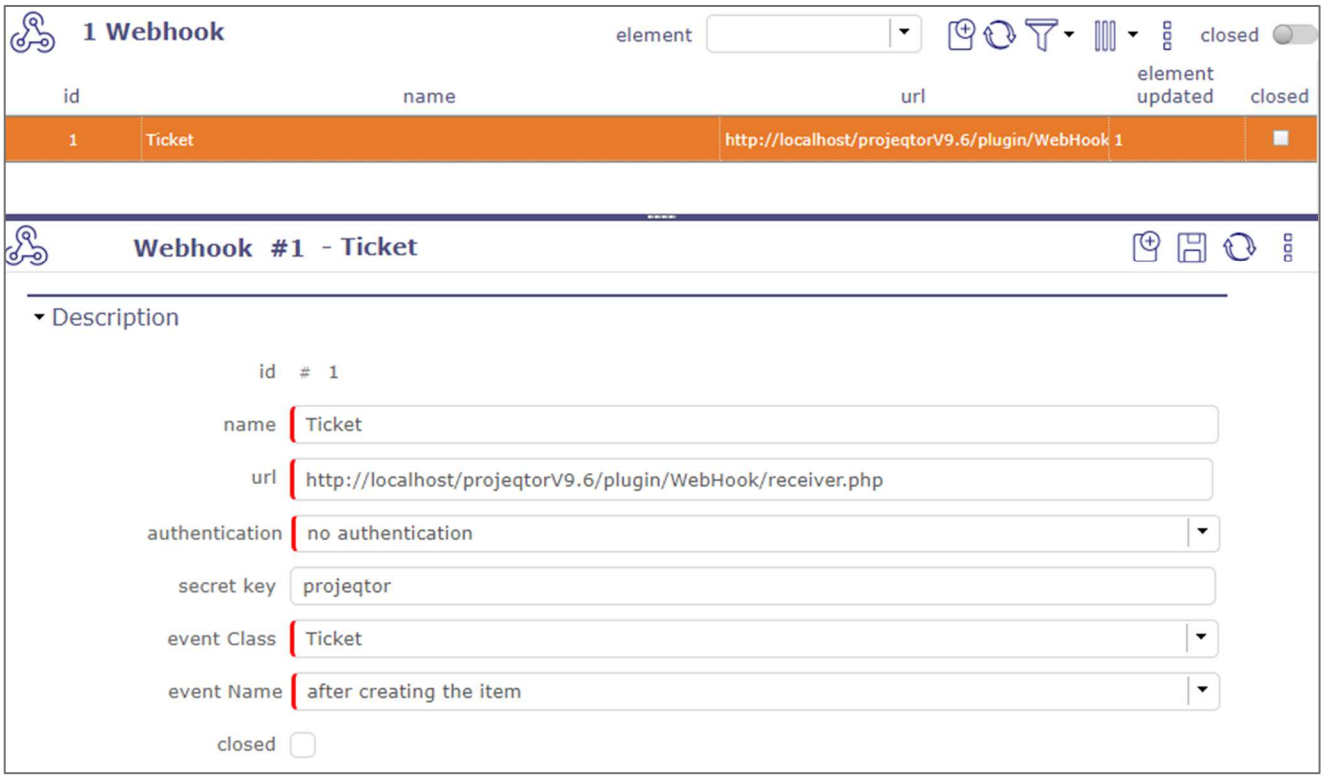
Click on  to install your plugin
Click on  to view the plugin metadata.
Click on  to cancel the installation of this plugin.

When the zip is installed, the application restarts. Go to the Plugin menu to access it.
After installation, the menus are available for the profile “administrator”.

WEBHOOK FEATURE

The webhooks are usually triggered by some event, in ProjeQtOr, these events are creation, update or delete of an item.

When the event occurs, the source site makes an HTTP request to the URL configured for the webhook. Users can configure them to cause events on one site to invoke behavior on another. Because webhooks use HTTP, they can be integrated into web services without adding new infrastructure.



Screen Webhook: example of webhook screen layout.

AUTHENTICATING THE WEBHOOK NOTIFICATION

When the client, i.e., the originating website or application, makes a webhook call to the third-party server, the incoming POST request must be authenticated to avoid a spoofing attack.
Its timestamp is checked to avoid a replay attack.

Different techniques to authenticate the client are used:

- The terminal can choose to maintain a list of IP addresses for known sources from which requests will be accepted.
- HTTP basic authentication can be used to authenticate the client.
- The webhook can include information about what type of event it is, as well as a shared secret or digital signature to verify the webhook.
- An HMAC signature can be included as an HTTP header.

ProjeQtOr webhook plugin gives the choice between:

- not checking the authentication
  authentication is required on third-party server with the IP addresses known by the server

- simple HTTP authentication with username and password.

In both cases, a HMAC signature is defined as from the payload and a secret key.
HTTP_PROJEQTOR_SIGNATURE_SHA256 contains the HMAC hashed payload with secret key.

PHP Code generating the signature is:

```
$signature ="sha256=".hash_hmac("sha256",$payload,$secret);
```

TRIGGER A WEBHOOK

Webhooks are triggered by an event on ProjeQtOr server (your own instance).

On the Webhook screen you define a webhook line for each element and event you want to trigger.
Select the event class (element to manage) and then the events to trigger between proposed ones:

- After creating the item
- After updating the item
- After deleting the item
-

Then define the URL to call for this event, and security options.
The payload sent is the complete description of the item in JSON format.

EXAMPLE OF USE

You can test the functionality of the plugin using the “receiver.php” file located in the “plugins/Webhook/” folder.

This file receives the request after the event is triggered and creates a “receiver.log” log file containing the information sent by the webhook.

In order to use this file, just create a webhook then define the following url:
http://yourProjeqtorServer/plugin/webhook/receiver.php

Here is an example of receiver.log

```
========================================

 payload received at 2022-01-14 17:54:54

----------------------------------------

Event=afterUpdate

Signature=sha256=d43e097431ec3880ee725ec1d6430cf9d3b6f6ee4be3a15f0697567dcd11c490

ContentType=application/json

ContentLength=1524

RequestTime=1642182894 (0 seconds ago)

Class=Ticket

Payload={"id":9,"reference":"001-001-INC-009","name":"TEST
999","idTicketType":16,"idProject":1,"externalReference":"xxxxxxxxxxxxxxxxxxxx","idUrgency":null
,"creationDateTime":"2022-01-13                         20:02:28","lastUpdateDateTime":"2022-01-14
18:52:52","idUser":1,"idContact":null,"Origin":{"id":null,"originType":null,"originId":null,"refT
ype":"Ticket","refId":9,"idle":0},"idTicket":null,"idContext1":null,"idContext2":null,"idContext3
":null,"description":"","idActivity":null,"idStatus":1,"idResolution":null,"isRegression":0,"idAc
countable":1,"idResource":1,"idCriticality":null,"idPriority":null,"idMilestone":null,"initialDue
DateTime":"2022-01-21                                   00:00:00","actualDueDateTime":"2022-01-21
00:00:00","WorkElement":{"id":9,"refType":"Ticket","refId":9,"idActivity":null,"idProject":"1","r
efName":"TEST
999","_tab_3_1":["estimated","real","left","ticketWork"],"plannedWork":0.5,"realWork":0,"leftWork
":0.5,"realCost":null,"leftCost":0,"_spe_run":null,"_spe_dispatch":null,"idUser":1,"ongoing":"0",
"ongoingStartDateTime":"","done":0,"idle":0,"_nbColMax":3},"handled":0,"handledDateTime":null,"pa
used":0,"pausedDateTime":"
","done":0,"doneDateTime":null,"solved":0,"idle":0,"idleDateTime":null,"cancelled":0,"result":"",
"idProduct":null,"idComponent":null,"idOriginalProductVersion":null,"idOriginalComponentVersion":
null,"idTargetProductVersion":null,"idTargetComponentVersion":null,"delayReadOnly":"0"}

----------------------------------------

check secret

receiveSignature=sha256=d43e097431ec3880ee725ec1d6430cf9d3b6f6ee4be3a15f0697567dcd11c490
expectedSignature=sha256=d43e097431ec3880ee725ec1d6430cf9d3b6f6ee4be3a15f0697567dcd11c490

   => MATCH

----------------------------------------
```