Feb 2021

THE WHISPIR POSTMAN COLLECTION

A Developer's Guide

whispir

Document version history

DATE	VERSION NO.	AUTHOR	PURPOSE
26 August 2019	V0.1	Jason Hendry	Draft created. Input from Elizabeth G. with formatting and editing in new template
27/08/2020	v0.2	Jason Hendry	Minor update to training video link
17/02/2021	v0.3	Jason Hendry	Minor Update after additions to services, some reformatting.

Document approval

DATE	APPROVER'S NAME

Whispir Limited (ABN 89 097 654 656)

Level 15, 360 Collins Street
Melbourne, Victoria 3000
+61 (03) 8630 9900

Contents

1. Introduction

About the Whispir platform

Whispir is a transactional messaging platform. You don't need to log in or maintain state in order to send a message using the Whispir API. Each request carries all the authentication required, including all the details of the message, recipient and addresses.

Further, with a single request you can send tens of thousands of messages for mass customer engagement scenarios, or individually for One Time Password and other personalised services. It all depends on your use case and your desired implementation model.

Whispir's REST-based transactional API makes it easy to develop host-to-host applications integrated into line-of-business solutions to trigger immediate cut through using SMS, Email, Web, Push and Voice messages. All these channels can be monitored for status and replies giving rise to round trip, conversational messaging applications. .

About this document

This document will guide you on the proper use of the Whispir REST API Postman Collection and how to test and develop your integrated client system. Each API call is documented with an example and all the required HTTP URLs, headers and bodies as needed.

Document conventions

Some values have been `{{parameterised}}` to identify where Postman environment values are inserted.

`{{region}}`	The identifier appropriate to your market AU - au US - us AP - ap AP1 - ap1 NZ - nz IT - it EDU - edu
`{{base-url}}`	https://api.region.whispir.com See https://developers.whispir.com/docs/new-api-url
`{{workspace-id}}`	ID to a nominated workspace; for example, D85E1B2FE68A9EF9
`{{x-api-key}}`	An API access token provided by Whispir Support Contact support@whispir.com
`{{resource-id}}`	ID to a base64 encoded resource used for parameterised bulk messaging
`{{template-id}}`	ID of a parameterised template containing SMS, email and rich content messaging

`{{auth}}`	Base64 digest of Whispir credentials

2. Using the Whispir REST API

There are two different approaches to integrating clients to the Whispir API. The first approach is simpler requiring only the knowledge of the structure of Whispir's REST URLs. The second approach is more complex and requires maintaining state between calls.

URL parameterisation

The first approach involves parameterising URLs with known values from the Whispir account. This is useful for simple client applications that have well established resources such as workspaces, templates and callbacks and known locations.

In this case, the client simply substitutes values into parameters in the API call to complete the details in a single call. For example, to send an SMS using a known workspace, you would replace the workspace parameter in the URL with a known value:

```
POST https://api.{{region}}.whispir.com/workspaces/{{workspace-id}}/messages
```

Since the Workspace is a long lived asset that isn't likely to change, there's little impact in substituting the workspace-id directly into the client code.

However, this approach doesn't take into account that permissions on the Workspace might one day change, causing the client code to fail. This creates significant issues in more complex solutions.

URL discovery

The second approach uses a technique called Hypertext As The Engine Of Application State.[1] This technique provides clients with the URLs they need to send messages and create resources.

For example, a client application could discover the available workspaces based on the credentials provided:

```
GET https://api.{{region}}.whispir.com/workspaces
```

The result of this call lists all the workspaces visible to the API user. Included in each oem of the response is a URL to retrieve details of the item in the workspace collection.


```
{
    "workspaces": [
        {
            "id": "174157CA505F4142",
            "projectName": "A Space Odessey",
            "projectNumber": "2001",
            "status": "A",
            "billingcostcentre": "Hollywood",
            "link": [
                {
                    "uri": "https://api.au.whispir.com/workspaces/174157CA505F4142",
                    "rel": "self",
                    "method": "GET",
                    "host": "api.au.whispir.com",
                    "port": -1
                }
```


[1] https://en.wikipedia.org/wiki/HATEOAS

```
            ]
        },
        ...
```

Retrieval of an item in the workspace collection returns the workspace entity, and an array of management property objects. The `rel` parameter gives some indication of the purpose of the method and can be used to identify the correct method to use.

In this way, the Whispir REST API is self documenting with respect to the methods available on a specific entity.


```
{
    "id": "174157CA505F4142",
    "projectName": "A Space Odessey",
    "projectNumber": "2001",
    "status": "A",
    "billingcostcentre": "Hollywood",
    "link": [
        {
            "uri": "https://api.au.whispir.com/workspaces/174157CA505F4142",
            "rel": "self",
            "method": "GET",
            "host": "api.au.whispir.com",
            "port": -1
        },
        {
            "uri": "https://api.au.whispir.com/workspaces/174157CA505F4142",
            "rel": "editWorkspaces",
            "method": "PUT",
            "type": "application/vnd.whispir.workspace-v1+xml,application/vnd.whispir.workspace-v1+json",
            "host": "api.au.whispir.com",
            "port": -1
        },
        {
            "uri": "https://api.au.whispir.com/workspaces/174157CA505F4142/messages",
            "rel": "retrieveMessage",
            "method": "GET",
            "host": "api.au.whispir.com",
            "port": -1
        },
        {
            "uri": "https://api.au.whispir.com/workspaces/174157CA505F4142/messages",
            "rel": "createMessage",
            "method": "POST",
            "type": "application/vnd.whispir.message-v1+xml,application/vnd.whispir.message-v1+json",
            "host": "api.au.whispir.com",
            "port": -1
        },
        ...
}
```


A client application will typically locate the method desired (e.g.: `editWorkspace`) and extract the associated URL and type information. This method can be used on all entities and methods in the Whispir API.

3. The Whispir Postman Collection

Selecting the right approach for your scenario

Whispir is a communication platform with an API that provides a versatile way to interact with Whispir's functions and resources.

Postman https://www.getpostman.com/ is a popular tool for testing web based APIs. It organises API requests into collections which can be scripted with parameters and include tests for service validation.

The Whispir Postman Collection provides most of the Whispir REST API methods to demonstrate and validate their use in typical customer scenarios.

Request Strategy

The Whispir Postman collection demonstrates the more complex **HATEOAS client** strategy to utilise the information returned from Whispir to be flexible and versatile in as many contexts as possible.

The Whispir Postman collection combines the HATEOAS approach with Postman Environment variables to remember values returned by Whispir for use as parameters in subsequent requests. As you work with the Postman collection, you'll notice the list of environment variables will grow as new values are added.

4. Best Practice Messaging

Developers should keep the following recommendations in mind while developing integrated client solutions.

Whispir Default Content Type

If no `Content-Type` header is provided, Whispir defaults to responses in XML.

e.g:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns3:error xmlns:ns2="http://schemas.api.whispir.com/dap"
xmlns:ns3="http://schemas.api.whispir.com">

   <errorSummary>The resource that you have requested does not exist. Please check
the identifier and try again </errorSummary>

   <errorText>Not Found</errorText>

</ns3:error>
```

Correct headers

Always include the following headers to ensure your request is authorised, and that you are sending and receiving content in the correct format.

Accept and Content-Type headers are used to specify the API version and the dialect of any body content. There is normally only one content type for each request, with a choice of JSON or XML body content

```
Accept: application/vnd.whispir.message-v1+json
```

Method	Header	Comment

ALL	`x-api-key`	Missing x-api-key will result in 403 Forbidden and message body
		`{`
		`"message": "Forbidden"`
		`}`

GET	`Accept`	Specify the format for receiving Whispir data.
PUT, POST	`Content-Type` `Accept`	PUT and POST both send content to Whispir to create and update resources and entities. Methods that return the created resource in the body will use the `Accept` content type specified if appropriate.
DELETE		Technically DELETE requires neither Accept or Content-Type, however you may elect to specify `Accept` type in the event of an error message returned.

Operate through a workspace

Whispir employs an internal organisational unit called a Workspace. A workspace helps to segregate access to resources and functionality using Permissions and Roles assigned to Users.

Whispir's default workspace, My Company, should not be used for any but the most simple messaging scenarios since most customers find it difficult to migrate once the integration pattern has been established.

Operating through workspaces helps to separate security concerns, message and resource management, usage and reporting. All the examples in this document utilise a workspace.

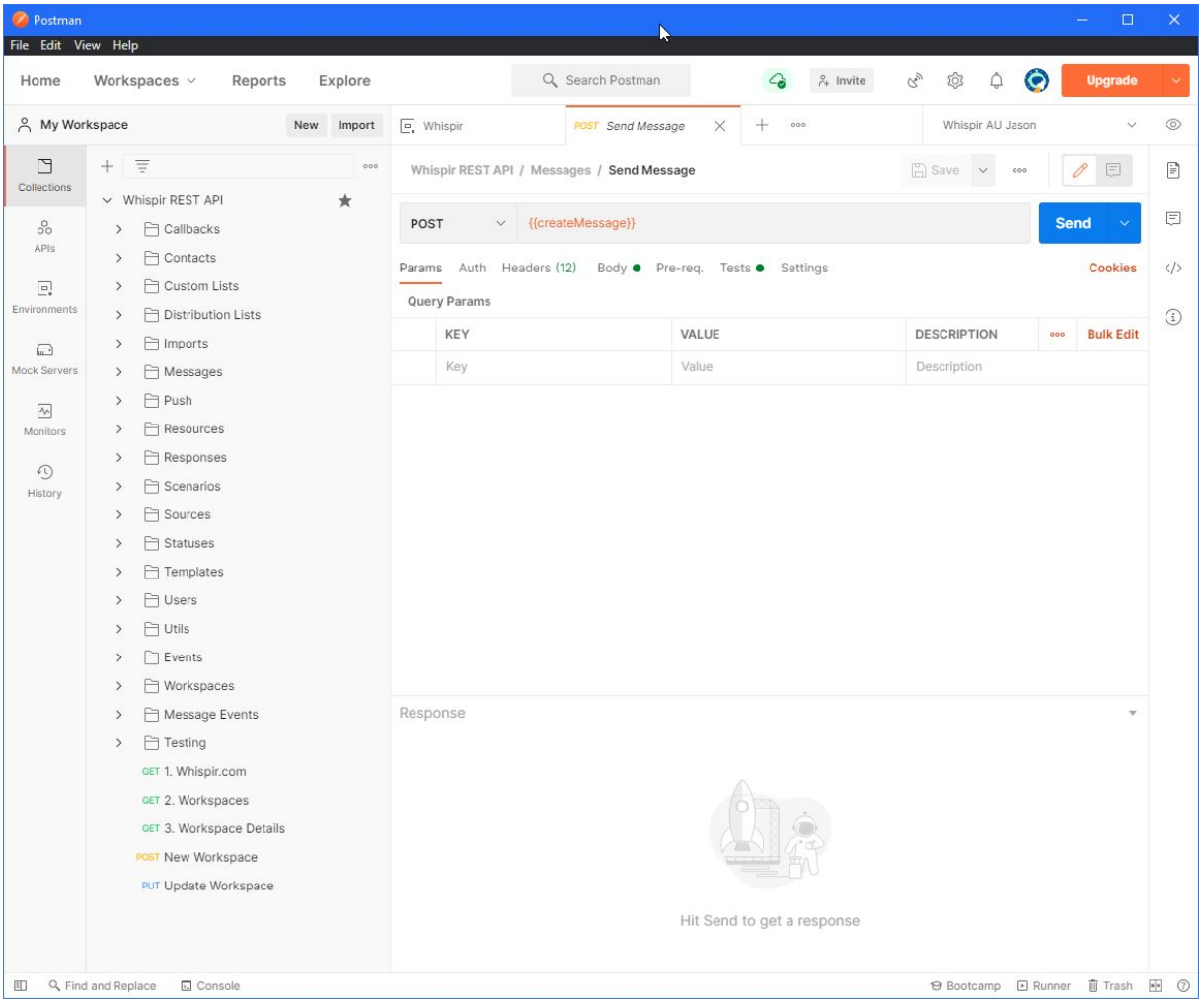
It's important to follow the set up procedure to initialise the `workspace-id` environment variable.

Modern development practices recommend development, testing and production applications are assigned their own workspaces.

5. Set up

After installing Postman and importing the Whispir Postman Collection and Whispir Postman Environment definitions, you'll have access to the Whispir REST requests and a set of Environment variables to use in the Whispir collection.

You can request the latest Whispir Postman Collection from support@whispir.com .



"We recommend making duplicates of the collection and the environment so you can always restore to a known good configuration."

Whispir Environment

The Whispir Postman Collection relies heavily on Postman Environment Variables to satisfy parameterised values. Some variables are used in every request, while others are more specialised.

The Whispir environment includes the following values:

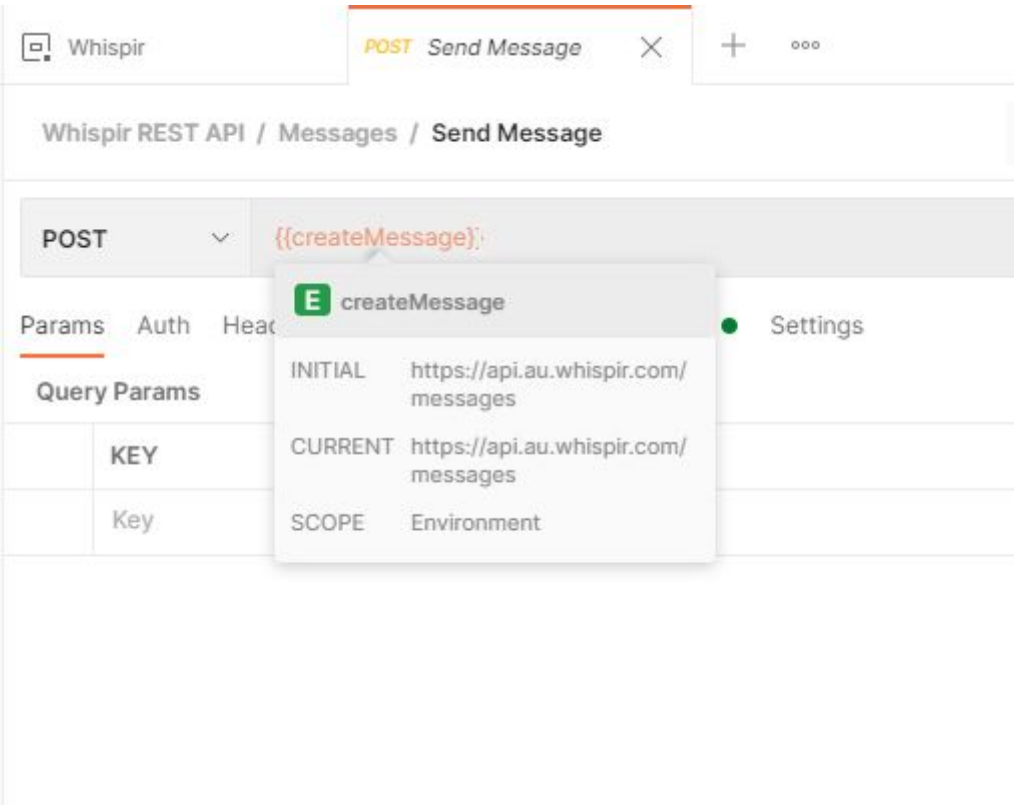
base-url	The regional variant of the whispir api
	`api.au.whispir.com`
	`api.us.whispir.com`
	`api.ap1.whispir.com`
	`api.it.whispir.com`
	`api.edu.whispir.com`

x-api-key	Supplied by Whispir

whispir-username	Supplied by Whispir
whispir-password	Supplied by Whispir
test-sms	Any valid mobile, cell, long code or short code
test-email	An email address to send test content
workspace-name	The name of a Whispir Workspace to use
callback-url	[optional] A Callback endpoint.

Set all the values in the table above to ensure the proper operation of the Whispir Postman Collection. In some cases, you may need to execute requests in a specific order to satisfy environment variables. This is indicated by a numbered sequence of requests, 1, 2, 3, … etc

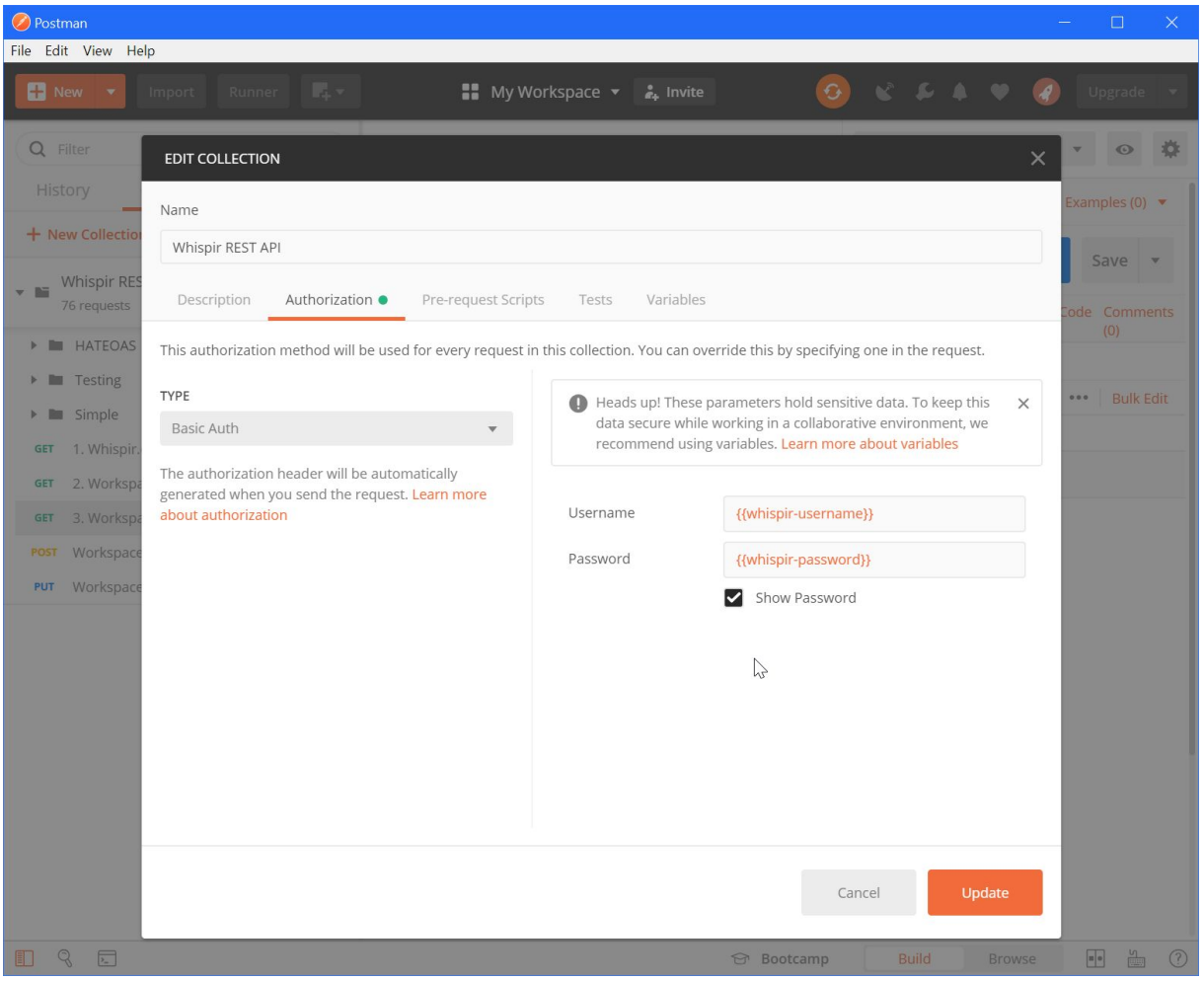
Over time, the Whispir Postman Collection will add environment variables from information provided by Whispir. In most aspects of Postman, hovering the mouse over the variable will display the resolved value.



Authentication

The Whispir Postman Collection uses "inherit authentication from parent" on each request to reduce the configuration burden on you, the user. The Collection comes preconfigured with the

{{whispir-username}}, {{whispir-password}} and {{x-api-key}} environment variables set in the Basic Auth parameters and request headers respectively.



*Note - The use of the `apikey` in the querystring has been removed from the collection.

Connection Validation

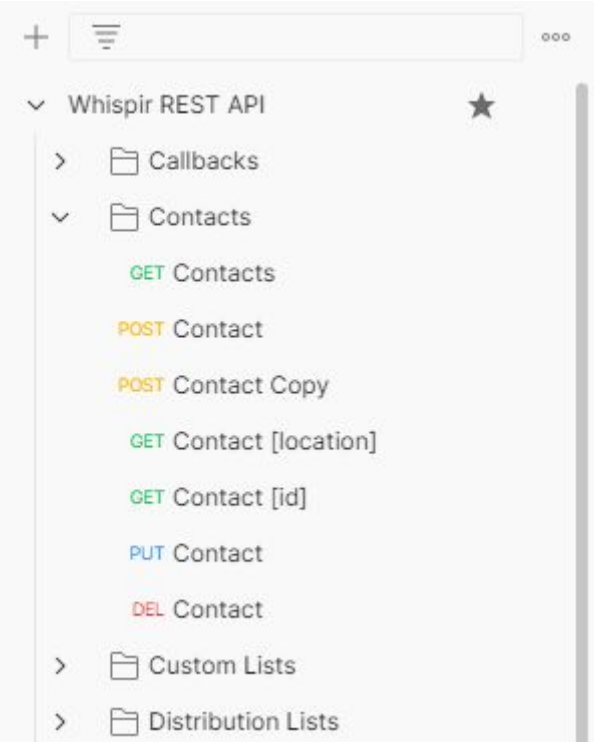
To validate the Postman configuration and your connection to Whispir, perform the following requests in order. If you notice any errors, do not proceed until the error is corrected.

1. Whispir.com - queries the base url and ensures connectivity
2. Workspaces - queries Whispir for all the workspaces
3. Workspace Details - queries Whispir for the Workspace identified as `workspace-name`

Once these steps have been performed, the Whispir Postman Collection is primed with the information it needs to execute most requests.

Whispir Requests

As you go through the collection you'll notice many requests are repeated, and some have [location] or [id] after their name.



The requests with the same name, such as POST Contact and PUT Contact are the same URL, they only differ by the HTTP method used in the request to Whispir. The naming convention helps to identify which requests use the same endpoint.

The requests with `[location]` at the end use the Location HTTP header as a means of identifying the resource, typically after a POST request. This is the same URL reported by Whispir as the 'self' link. In some situations where you have the URL but not the ID of an entity, you can modify the `[object-location]` environment value and use a `[location]` method to retrieve the details.

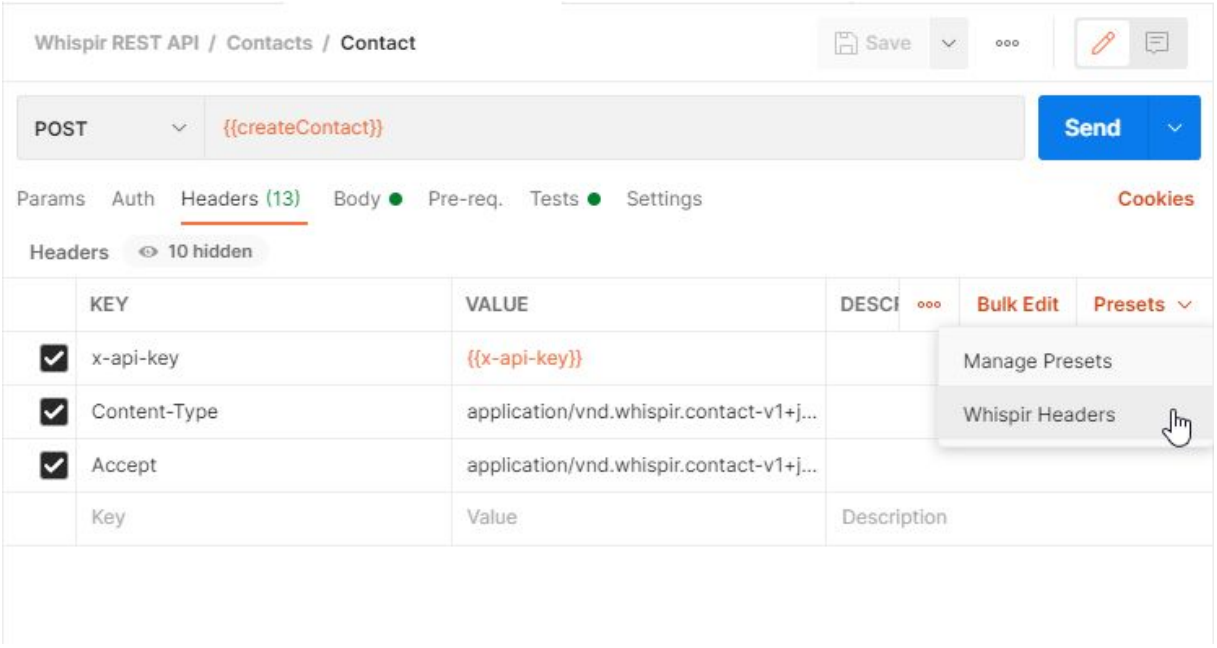
The same also applies to [id] providing a way to get the details of Whispir assets by modifying the `[object-id]` environment variable.

Add a request

There are often instances where you want to add your own request to test a specific Whispir scenario, or copy a returned method or object location from Whispir to evaluate a result.

You only need to perform a couple steps to add a request to the collection:

1. In your request, locate the Headers and choose **Whispir Headers** from the presets
2. Click 'Save' and put the request in a folder or create your own



Following this procedure will ensure your request inherits the authentication configured for the collection and has the correct header parameters.

Additionally, you should set the `Accept` and `Content-Type` headers to ensure the body format is specified.