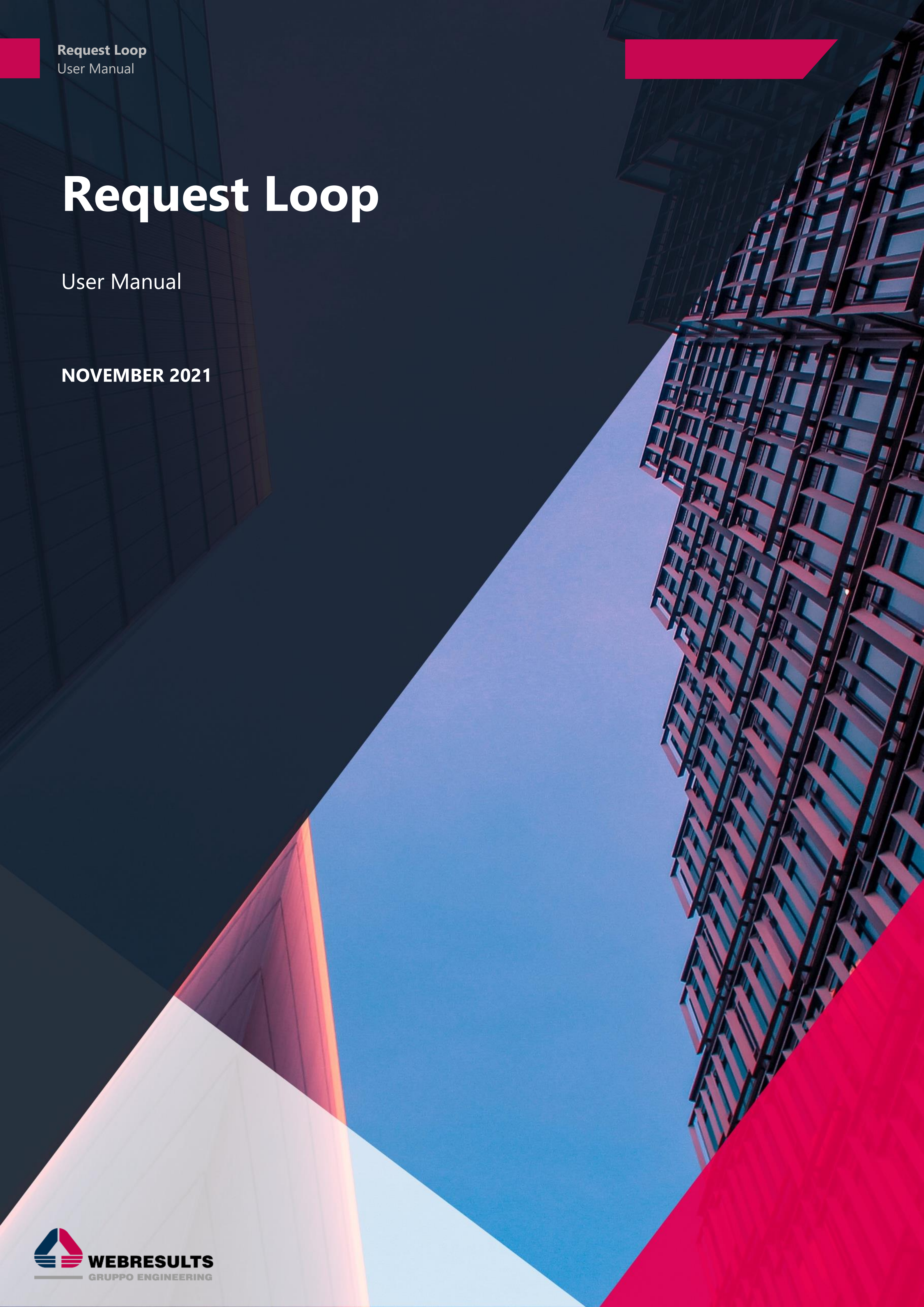


Request Loop

User Manual

NOVEMBER 2021

WEBRESULTS
GRUPPO ENGINEERING

TABLE OF CONTENTS

1 INTRODUCTION

1.1 WHAT IS IT?

Request Loop is a free tool meant for developers who want to debug webservices communication both inbound and outbound. This tool has been imagined as a quick disposal package that anyone can install in a DE org or a sandbox (even production but it is unlikely and not suggested to debug directly in production), use until necessary and then uninstall to clean up everything.

The tool is composed by 2 features:

- **Request Bin**: an inbound webservice that can receive any supported HTTP call and log it for further analysis. This tool can also simulate the response of a valid service (just like the famous Requstb.in online service). Imagine you need to get the SOAP payload of an Apex webservice: no Salesforce tool is available for this porpoise and with Request Loop you can inspect the content message on the fly safely.
- **Request Client**: a tool to send outbound callouts from Salesforce to outside systems. This tool can be used to simulate an external system call from within Salesforce to test a service without the need of a complete Apex implementation.

Although it is true that there is plenty of services online that do almost the same things, having an AppExchange package lets developers debug within the platform using the customer's orgs, reducing chances that CRM data is passed outside the org safe bounds.

1.2 DISCLAIMER

Request Loop is a free AppExchange app delivered to the Salesforce Ohana for free. We do not deliver any assistance nor warranties for future improvements.

1.3 WHAT IS THE SUGGESTED AUDIENCE?

The suggested audience is developers/architects or administrators with integration knowledge who wants a way within the CRM to debug inbound / outbound callouts.

It is highly suggested that the tool is installed on DE orgs, trial orgs or sandboxes and not on production orgs.

2 INSTALLATION & SETUP

2.1 PACKAGE INSTALLATION

The first thing is to install the package from the AppExchange to admins only: this avoids applying useless visibility and access on the package objects/fields to all internal profiles.

2.2 PACKAGE SETUP

To let users access the Request Loop app and the main custom objects and fields, assign the **Request Loop User** permission sets to the selected users: if you plan to work with administrator users only, you don't need this configuration, as the package installation already granted administrators the necessary access to the package's resources.

To enable the inbound webservice that is used to collect inbound webservice calls, assign to an internal user (suggested an Administrator user) the **Request Loop Integration** permission set: this user will have access to all required custom objects/fields and to the Apex class that enables a custom REST service available for callins.

The new service is now available at:

`https://[MY_DOMAIN].my.salesforce.com/services/apexrest/wrts_reqlop/v1.0/bin/[BIN_KEY]`

Where `[BIN_KEY]` is the unique key of a given bin (see next chapter for details): have a look at the **Request Bin URI** field of a **Request Bin** record after record creation to get the actual url.

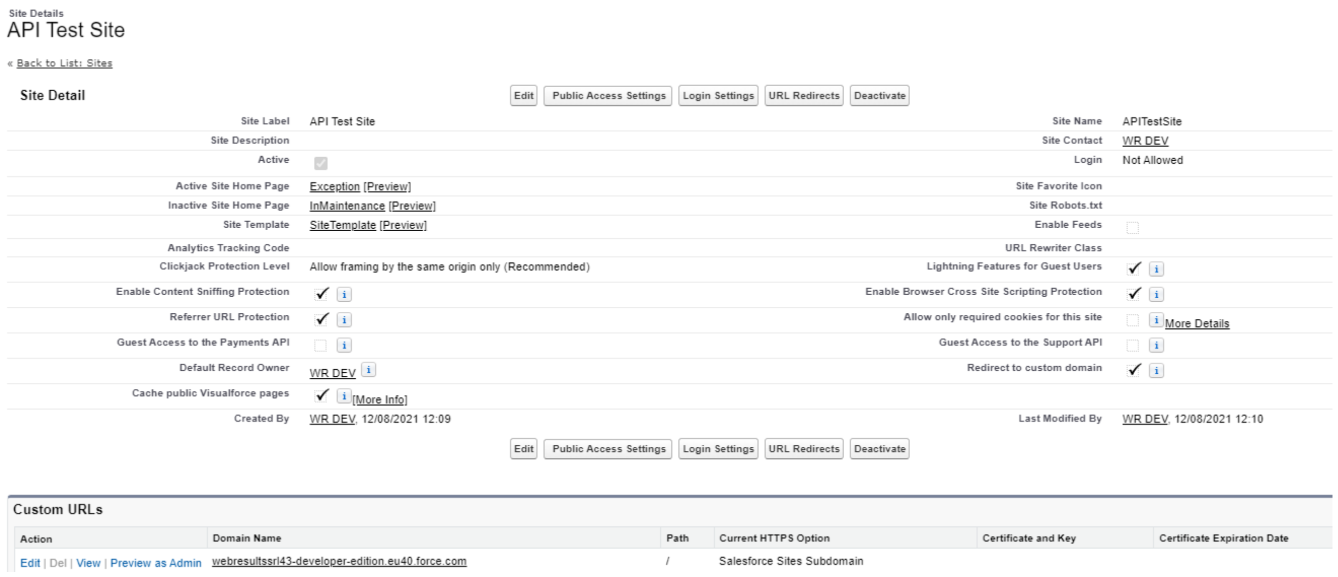
Once the service is enabled, you need a valid *session id* to access the service which, if you are testing an Apex callout where you cannot change the code to insert the new authentication header, can be a little complicate. We have 2 solutions:

- create a named credential with OAuth authentication to the same Salesforce user (it requires the creation of a Connected App that points to the same Salesforce instance) or a simple unauthenticated named credential (but you need to set up manually the "`Authorization: Bearer XXXX`" header on each request)
- publish the Request Bin webservice as a public service in a public Site or Community (see below) and other configurations noted in the following section

2.2.1 SETUP REQUEST BIN AS GUEST USER

Although not suggested for security reasons, you can expose the Request Bin webservice as a public service: as long as you are dealing with sandbox data and not actual customer data, this can be no problem at all.

Create a new Salesforce Site (it works with a community as well) as shown below:


Site Details
API Test Site

« Back to List: Sites

Site Detail

Site Label	API Test Site	Site Name	APITestSite
Site Description		Site Contact	WR DEV
Active	✓	Login	Not Allowed
Active Site Home Page	Exception [Preview]	Site Favorite Icon	
Inactive Site Home Page	InMaintenance [Preview]	Site Robots.txt	
Site Template	SiteTemplate [Preview]	Enable Feeds	☐
Analytics Tracking Code		URL Rewriter Class	
Clickjack Protection Level	Allow framing by the same origin only (Recommended)	Lightning Features for Guest Users	✓
Enable Content Sniffing Protection	✓	Enable Browser Cross Site Scripting Protection	✓
Referrer URL Protection	✓	Allow only required cookies for this site	☐ More Details
Guest Access to the Payments API	☐	Guest Access to the Support API	☐
Default Record Owner	WR DEV	Redirect to custom domain	✓
Cache public Visualforce pages	✓ [More Info]		
Created By	WR DEV, 12/08/2021 12:09	Last Modified By	WR DEV, 12/08/2021 12:10

Custom URLs

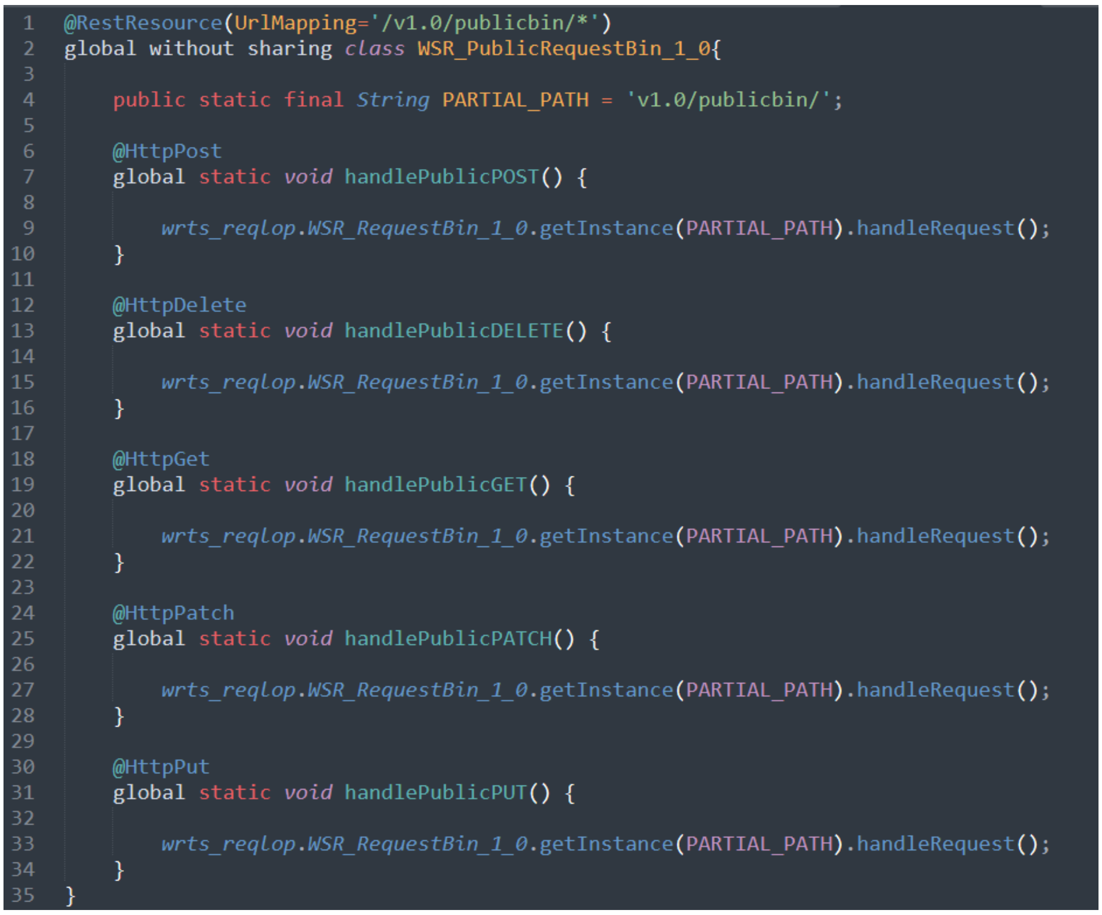
Action	Domain Name	Path	Current HTTPS Option	Certificate and Key	Certificate Expiration Date
Edit | Del | View | Preview as Admin	webresultssrl43-developer-edition.eu40.force.com	/	Salesforce Sites Subdomain		

This Site is only used to expose the webservice publicly, so you don't need to configure anything else (anyone will ever access the site via browser): if you already have a Site this step is optional. **Make sure the site is active.**

To enable the webservice you need to assign the **Request Loop Integration** permission set to the Guest User of the above site. Click on **Public Access Settings** > **View Users** and select the ***Site Guest User, [Site Name]***; then click on **Edit Assignments**on the **Permiossion Set Assignments** section, select the **Request Loop Integration** permission set and save.

Since Site’s Guest users cannot access custom objects if not enabled, we need to create a new Apex class to remove the sharing restrictions (the webservice query the bins and create the request records).

Here is an example:



```
@RestResource(UrlMapping='/v1.0/publicbin/*')
global without sharing class WSR_PublicRequestBin_1_0{

    public static final String PARTIAL_PATH = 'v1.0/publicbin/';

    @HttpPost
    global static void handlePublicPOST() {

        wrts_reqlop.WSR_RequestBin_1_0.getInstance(PARTIAL_PATH).handleRequest();
    }

    @HttpDelete
    global static void handlePublicDELETE() {

        wrts_reqlop.WSR_RequestBin_1_0.getInstance(PARTIAL_PATH).handleRequest();
    }

    @HttpGet
    global static void handlePublicGET() {

        wrts_reqlop.WSR_RequestBin_1_0.getInstance(PARTIAL_PATH).handleRequest();
    }

    @HttpPatch
    global static void handlePublicPATCH() {

        wrts_reqlop.WSR_RequestBin_1_0.getInstance(PARTIAL_PATH).handleRequest();
    }

    @HttpPut
    global static void handlePublicPUT() {

        wrts_reqlop.WSR_RequestBin_1_0.getInstance(PARTIAL_PATH).handleRequest();
    }
}
```

WSR_PublicRequest
Bin_1_0.class

The class must then be enabled on the **Site** > **Public Access Settings** > **Enabled Apex Class Access** section.

The **PARTIAL_PATH** constant is used to setup a new path for the webservice: make sure that this constant equals the value on the **`@RestResource(UrlMapping='xx')`** annotation excluding the “*” character.

In this condition the final endpoint will be:

*https://[SITE_DOMAIN].[ORG_INSTANCE].force.com/[SITE_PATH_IF_ANY]/services/apexrest/**v1.0/publicbin**/[BIN_KEY]*

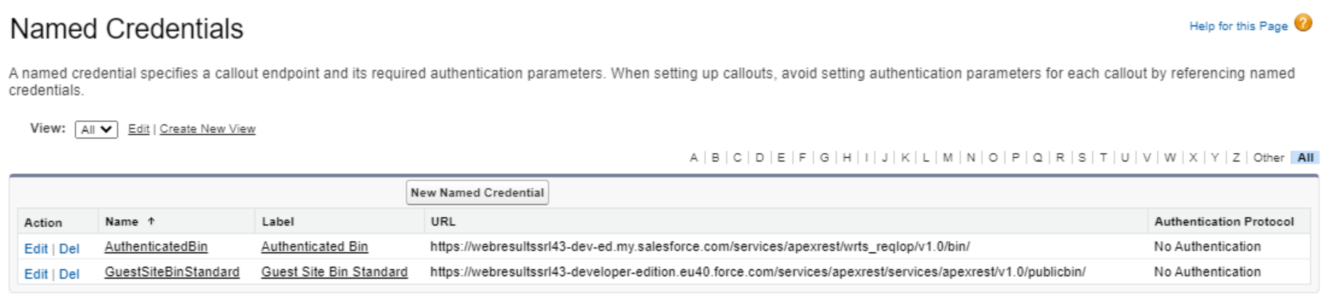
Using this method, the new endpoint overrides the package webservice Request Bin default endpoint.

2.2.2 ENABLE NAMED CREDENTIALS FOR REQUEST BIN ENDPOINTS

To speed up testing, you would need to create named credentials to store the main path of the Request Bin records.

The **Request Client** tool lets you input your own callout endpoint (that you have to explicitly enable via the **Setup** > **Remote Site Settings** configuration) or helps you with an autocomplete feature showing the available named credentials.

Here is an example of named credential configuration:


Named Credentials

Help for this Page

A named credential specifies a callout endpoint and its required authentication parameters. When setting up callouts, avoid setting authentication parameters for each callout by referencing named credentials.

View: All | Edit | Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

New Named Credential

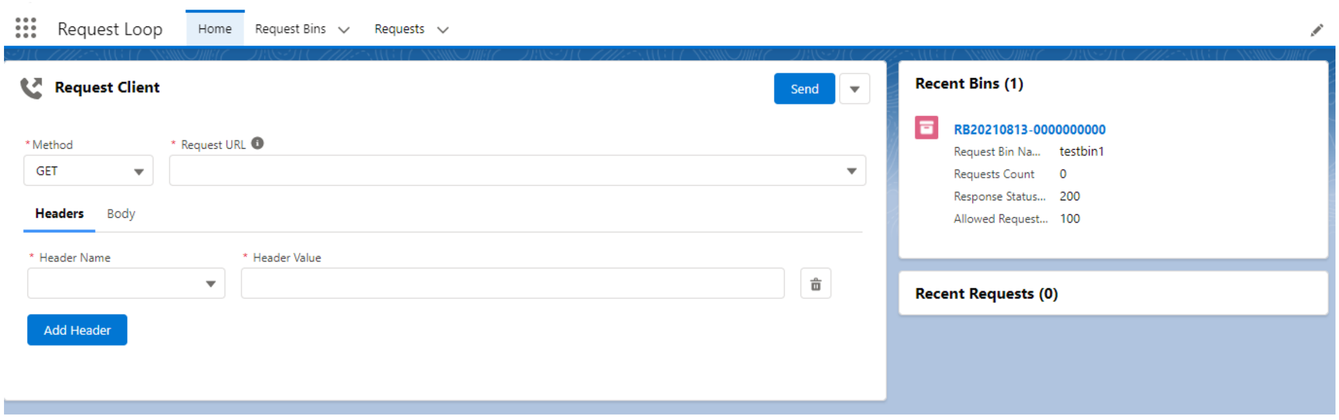
Action	Name ↑	Label	URL	Authentication Protocol
Edit | Del	AuthenticatedBin	Authenticated Bin	https://webresultssrl43-dev-ed.my.salesforce.com/services/apexrest/wrts_reqlop/v1.0/bin/	No Authentication
Edit | Del	GuestSiteBinStandard	Guest Site Bin Standard	https://webresultssrl43-developer-edition.eu40.force.com/services/apexrest/services/apexrest/v1.0/publicbin/	No Authentication

Both named credentials are unauthenticated, the first points to the standard package service (and needs the *Authorization* header), the second points to the guest site public service.

Both named credentials must be completed with the Request Bin key and additional path/parameters required by your call testing.

3 FEATURES DESCRIPTION

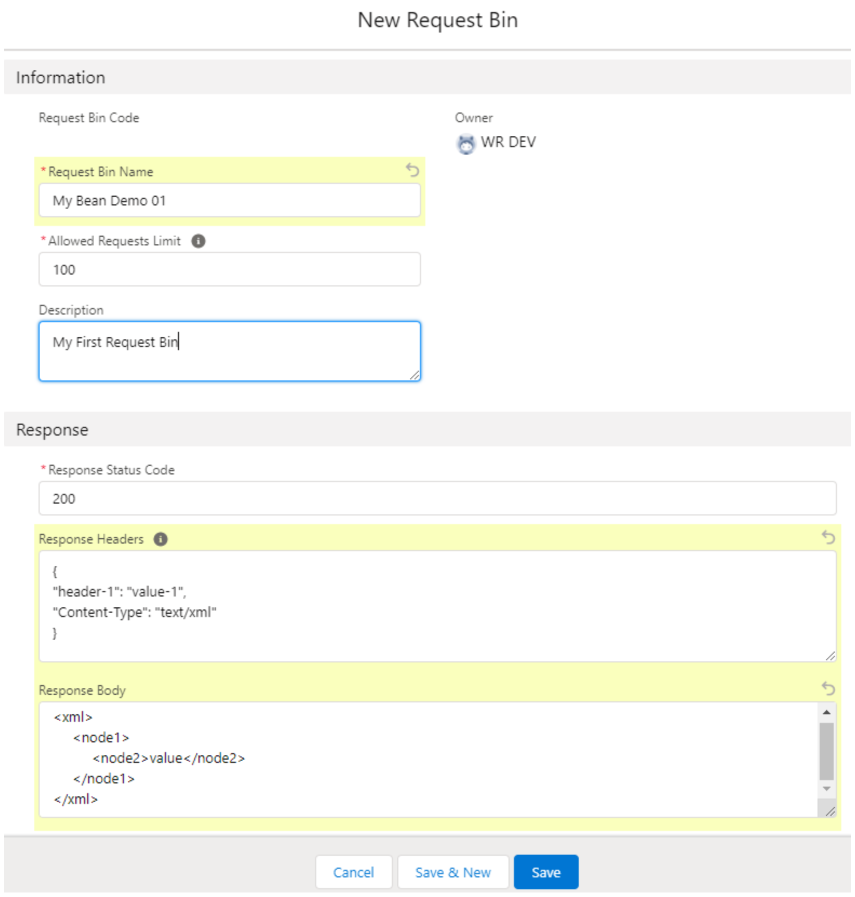
To access **Request Loop** simply look for the *Request Loop* application:



The **Request Client** feature is nested into the *Home* tab, while the **Request Bin** configuration is available through regular record creation via the *Request Bins* tab.

3.1 REQUEST BIN

To configure a new Request Bin (i.e. a container for inbound requests) simply create a new Request Bin record:


New Request Bin

Information

Request Bin Code

Owner: WR DEV

*Request Bin Name: My Bean Demo 01

*Allowed Requests Limit: 100

Description: My First Request Bin

Response

*Response Status Code: 200

Response Headers:

```
{
"header-1": "value-1",
"Content-Type": "text/xml"
}
```

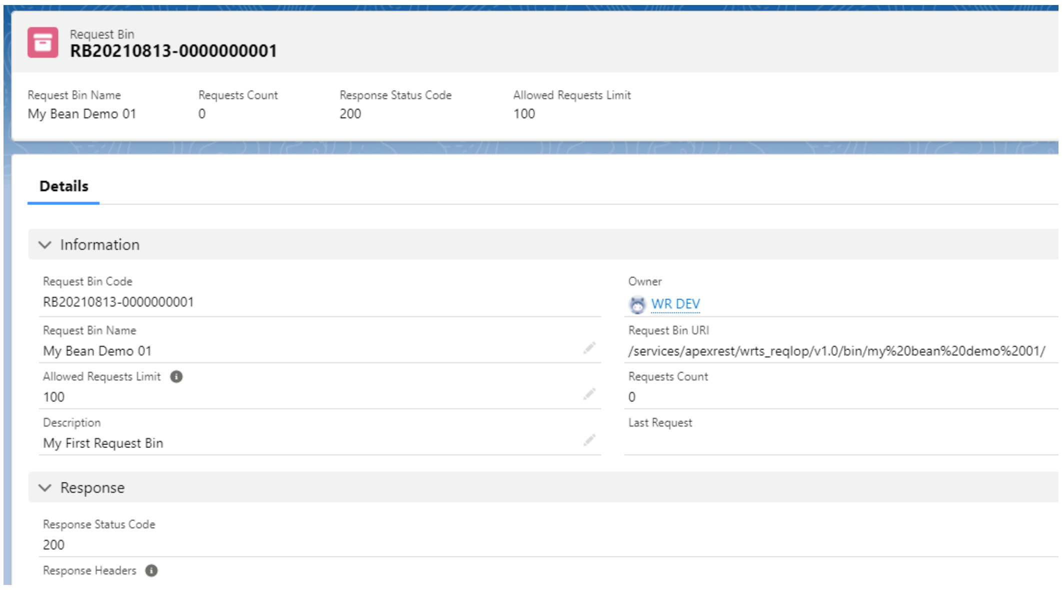
Response Body:

```
<xml>
    <node1>
        <node2>value</node2>
    </node1>
</xml>
```

In this example we are setting:

- The Request Bin unique name (only letters, numbers, spaces and special chars "_", "-" and "." Allowed and the value must be greater or equal to 3 in size
- The maximum number of allowed requests per bin (this avoids creating thousands of request records without control)
- A description
- The response code (should be a 3 digit string)
- Response headers (optional) in JSON format
- Response Body (optional) in any format (plain text, JSON, base 64, JSON, …)

Once saved, a formula shows the actual endpoint:



Request Bin
RB20210813-0000000001

Request Bin Name	Requests Count	Response Status Code	Allowed Requests Limit
My Bean Demo 01	0	200	100

Details

Information

Request Bin Code: RB20210813-0000000001
Owner: WR DEV
Request Bin Name: My Bean Demo 01
Request Bin URI: /services/apexrest/wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/
Allowed Requests Limit: 100
Requests Count: 0
Description: My First Request Bin
Last Request:

Response

Response Status Code: 200
Response Headers:

Remember that, if you are not using the package's webservice, the Request Bin URI changes from:

```
/services/apexrest/wrts_reqlop/v1.0/bin/[encoded_bin_key]
```

to:

```
/services/apexrest/[PARTIAL_PATH]/[encoded_bin_key]
```

As shown on chapter "2.2.1 SETUP REQUEST BIN AS GUEST USER".

To test out the Request Bin feature, we'll be using the **Request Client** in the next chapter.

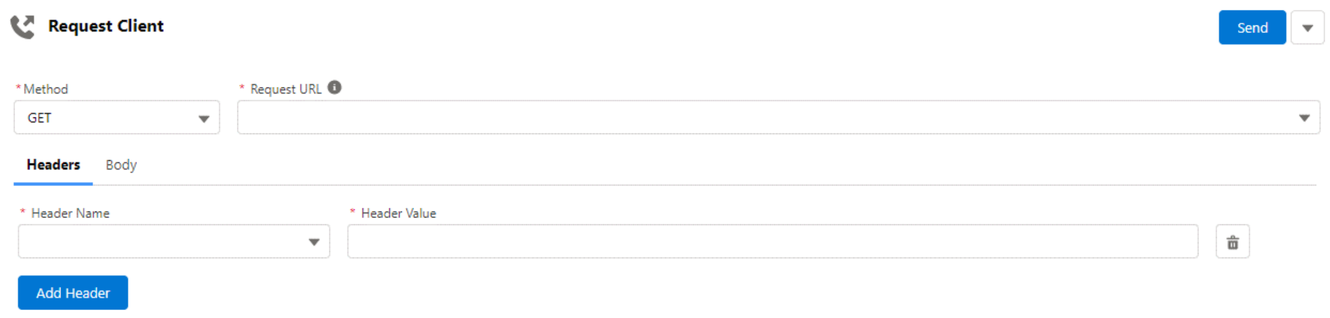
You can now use this endpoint to:

- Grab the content body from any Salesforce callout (REST or SOAP, no matter the protocol): you can switch the endpoint of the custom Apex webservice to one of the defined named credential
- Make an external system point to this endpoint to get what's happening in the request (standard API or custom services, both REST and SOAP)

3.2 REQUEST CLIENT

Now that we have a configured Request Bin that can take any incoming request, we'll have a look at the **Request Client** too that can generate a callout by hand.

Click on the *Home* tab of the **Request Loop** app:



Request Client

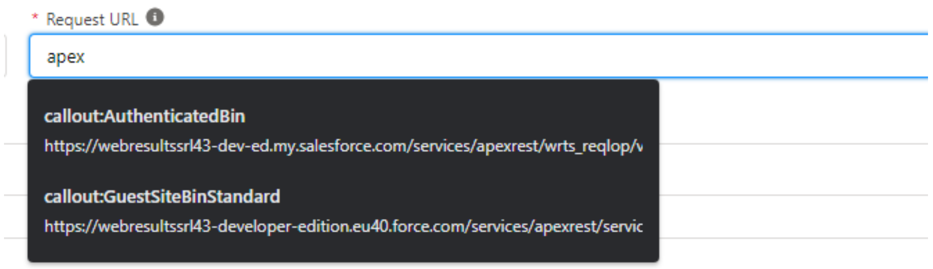
* Method: GET
* Request URL:

Headers | Body

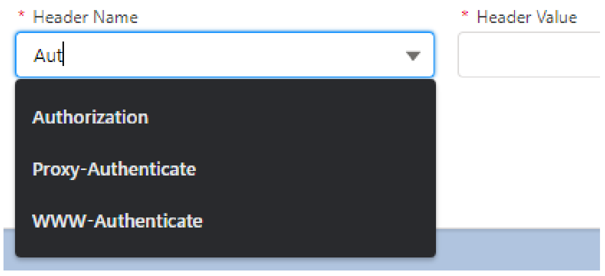
* Header Name:
* Header Value:

It can be configured with:

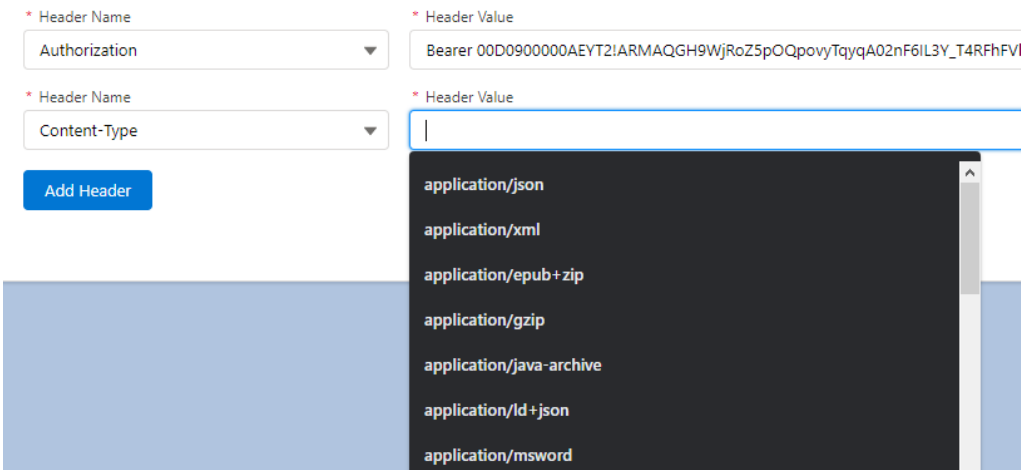
- Supported HTTP method (Salesforce supports a set of HTTP methods, GET / PATCH / PUT / POST / DELETE)
- The Request URL, which helps you with an autocomplete behavior:


* Request URL
apex
callout:AuthenticatedBin
https://webresultssrl43-dev-ed.my.salesforce.com/services/apexrest/wrts_reqlop/v
callout:GuestSiteBinStandard
https://webresultssrl43-developer-edition.eu40.force.com/services/apexrest/servic

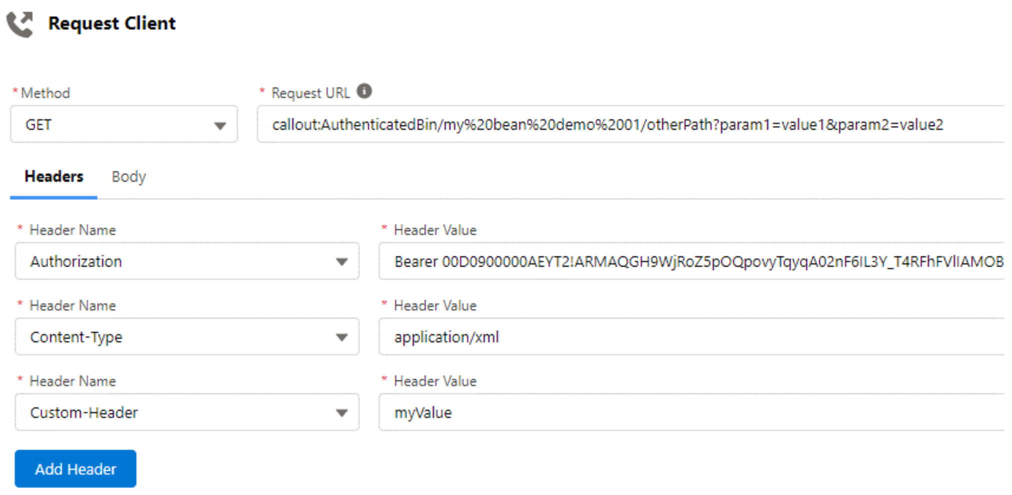
- Request headers, with an autocomplete features for the main standard headers:



The *Content-Type* header has an autocomplete behavior for the value as well, showing the main standard content types:



Here is an example of a Request Bin configuration, ready to be fired (we have used the authenticated service):



Request Client

* Method: GET

* Request URL: callout:AuthenticatedBin/my%20bean%20demo%2001/otherPath?param1=value1&param2=value2

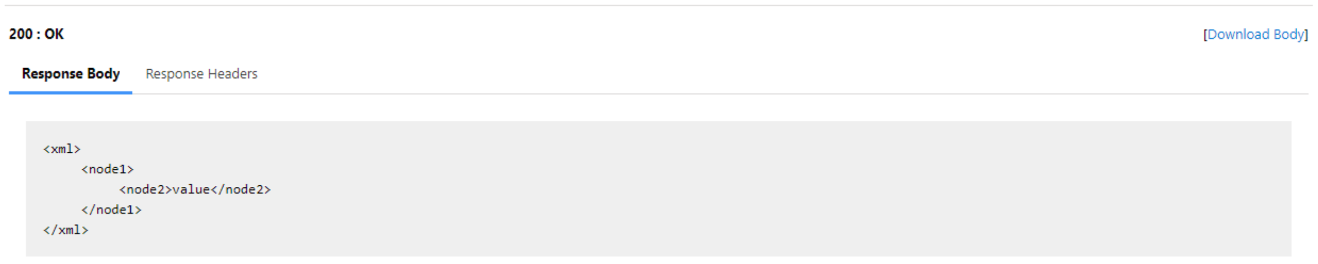
Headers | Body

* Header Name	* Header Value
Authorization	Bearer 00D0900000AEYT2!ARMAQGH9WjRoZ5pOQpovyTqyqA02nF6IL3Y_T4RFhFVlIAMOB
Content-Type	application/xml
Custom-Header	myValue

Take a look at the Request URL which have been changed to include the Request Bin key and other additional path and parameters.

To add a body to the request simply click on the **Body** tab and add any body you want to send (this example shows a GET request which should not have any body at all).

Now, click the **Send** button to send the callout:

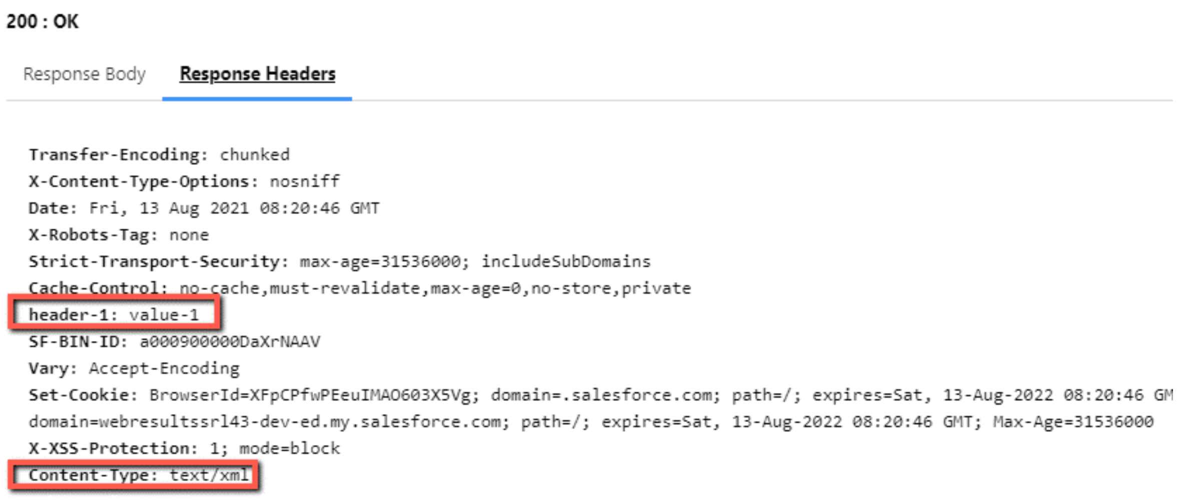


200 : OK

```
<xml>
     <node1>
          <node2>value</node2>
     </node1>
</xml>
```

The Request Bin has responded (as configured) a “200 OK” with a custom XML body shown on the **Response Boby** panel (that you can download using the **Download Body** link on the right side of the response section).

The **Response Headers** shows the headers exchanged by the external system (Request Bin again):



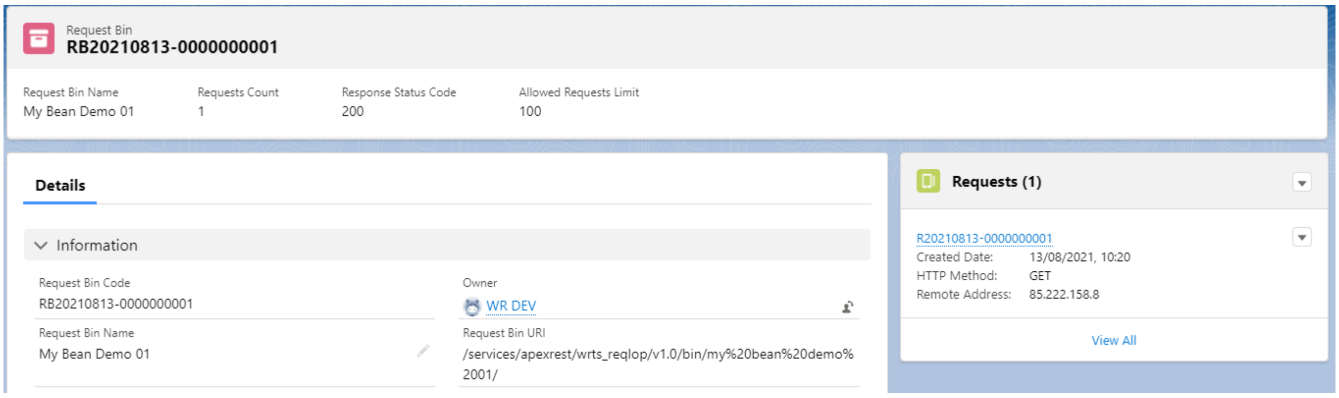
200 : OK

```
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
Date: Fri, 13 Aug 2021 08:20:46 GMT
X-Robots-Tag: none
Strict-Transport-Security: max-age=31536000; includeSubDomains
Cache-Control: no-cache,must-revalidate,max-age=0,no-store,private
header-1: value-1
SF-BIN-ID: a000900000DaXrNAAV
Vary: Accept-Encoding
Set-Cookie: BrowserId=XFpCPfwPEeuIMAO603X5Vg; domain=.salesforce.com; path=/; expires=Sat, 13-Aug-2022 08:20:46 GM
domain=webresultssrl43-dev-ed.my.salesforce.com; path=/; expires=Sat, 13-Aug-2022 08:20:46 GMT; Max-Age=31536000
X-XSS-Protection: 1; mode=block
Content-Type: text/xml
```

As you can see the headers contain the headers set in the Request Bin configuration.

3.2.1 **REQUEST BIN**'S REQUESTS

Jump back to the Request Bin record we have used to grab the Request Client callout.



Request Bin
RB20210813-0000000001

Request Bin Name	Requests Count	Response Status Code	Allowed Requests Limit
My Bean Demo 01	1	200	100

Details

Information

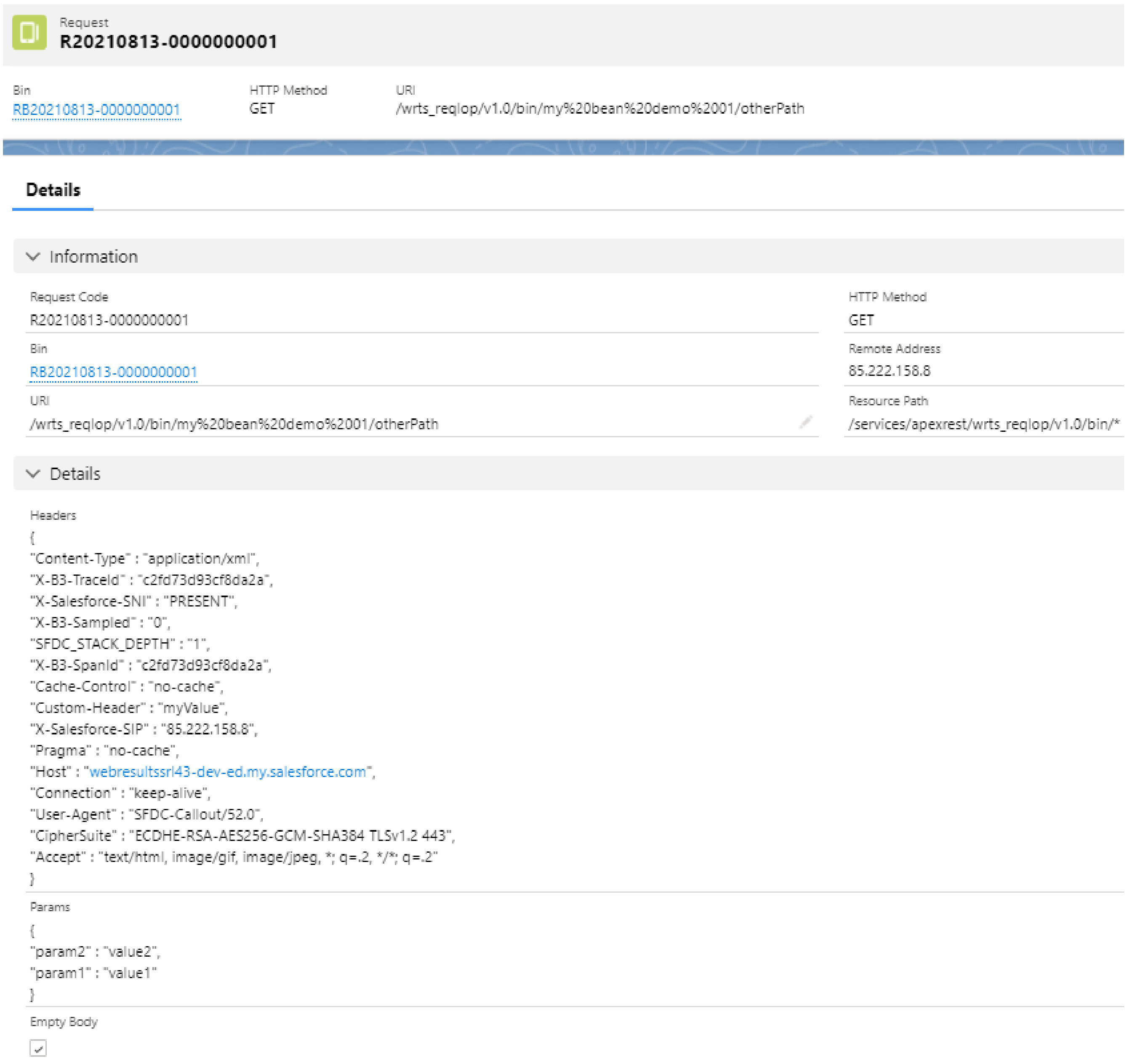
Request Bin Code: RB20210813-0000000001
Owner: WR DEV
Request Bin Name: My Bean Demo 01
Request Bin URI: /services/apexrest/wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/

Requests (1)

R20210813-0000000001
Created Date: 13/08/2021, 10:20
HTTP Method: GET
Remote Address: 85.222.158.8

View All

The **Request** object stores all requests sent to the bin:



Request
R20210813-0000000001

Bin	HTTP Method	URI
RB20210813-0000000001	GET	/wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/otherPath

Details

Information

Request Code: R20210813-0000000001
HTTP Method: GET
Bin: RB20210813-0000000001
Remote Address: 85.222.158.8
URI: /wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/otherPath
Resource Path: /services/apexrest/wrts_reqlop/v1.0/bin/*

Details

Headers

```
{
"Content-Type" : "application/xml",
"X-B3-TraceId" : "c2fd73d93cf8da2a",
"X-Salesforce-SNI" : "PRESENT",
"X-B3-Sampled" : "0",
"SFDC_STACK_DEPTH" : "1",
"X-B3-SpanId" : "c2fd73d93cf8da2a",
"Cache-Control" : "no-cache",
"Custom-Header" : "myValue",
"X-Salesforce-SIP" : "85.222.158.8",
"Pragma" : "no-cache",
"Host" : "webresultssrl43-dev-ed.my.salesforce.com",
"Connection" : "keep-alive",
"User-Agent" : "SFDC-Callout/52.0",
"CipherSuite" : "ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 443",
"Accept" : "text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2"
}
```

Params

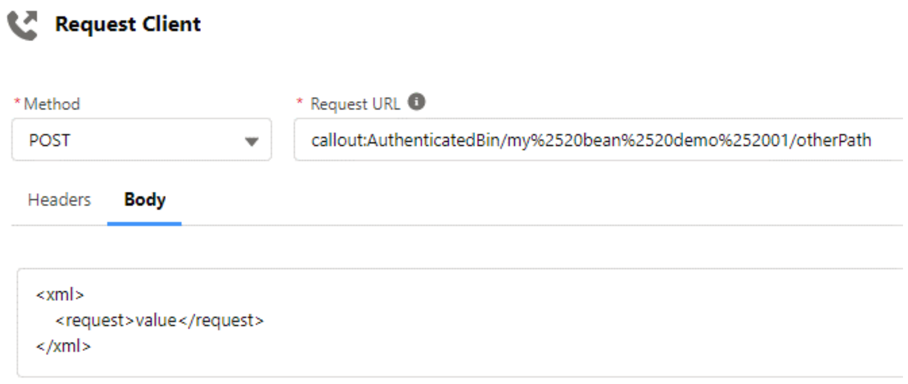
```
{
"param2" : "value2",
"param1" : "value1"
}
```

Empty Body: ✓

As expected we have:

- The callout path (with the additional "/otherPath")
- The headers sent (including the custom "Custom-Header")
- The parameters sent in the URI
- The evidence that no body was sent

If a body was sent via the Request Client configuration:



Request Client

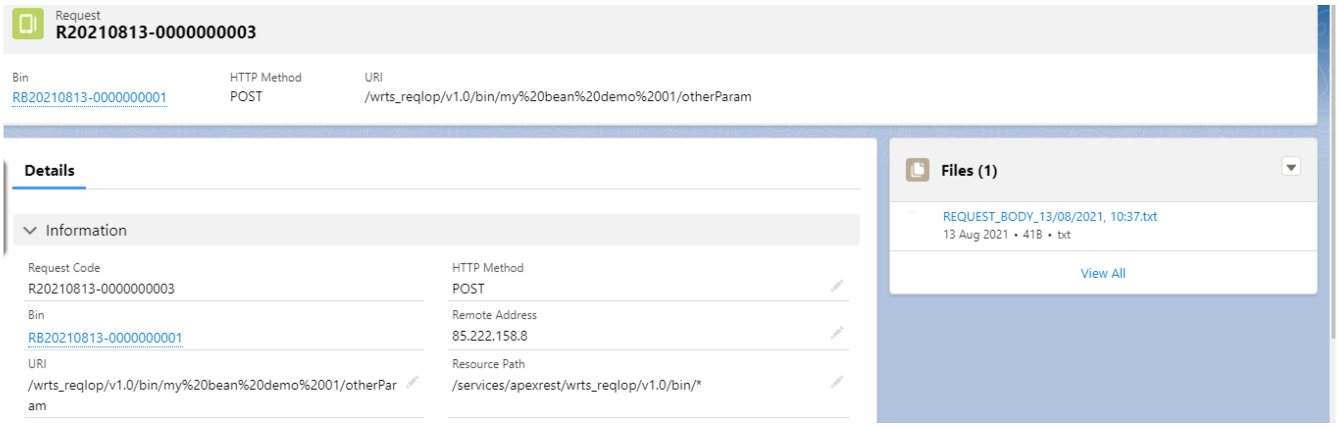
* Method: POST

* Request URL: callout:AuthenticatedBin/my%2520bean%2520demo%252001/otherPath

Headers | **Body**

```
<xml>
   <request>value</request>
</xml>
```

The Request Bin's Request record will show a File containing the request data:



Request
R20210813-0000000003

Bin	HTTP Method	URI
RB20210813-0000000001	POST	/wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/otherParam

Details

Information

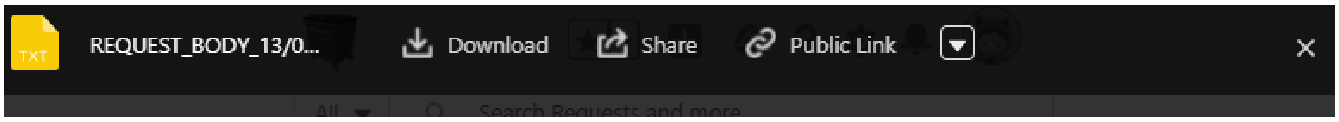
Request Code: R20210813-0000000003	HTTP Method: POST
Bin: RB20210813-0000000001	Remote Address: 85.222.158.8
URI: /wrts_reqlop/v1.0/bin/my%20bean%20demo%2001/otherParam	Resource Path: /services/apexrest/wrts_reqlop/v1.0/bin/*

Files (1)

REQUEST_BODY_13/08/2021, 10:37.txt
13 Aug 2021 • 41B • txt

View All

Which contains the body of the request:



```
<xml>
    <request>value</request>
</xml>
```