Web Service OSR REST:

Web Service User Guide

Index

- This document explains the operations related to loading or replacing files for the MasterBase® attachments sending service.

Definition

- Endpoint: the complete address that includes a base URL plus the required parameters (mandatory or optional) for execution.

- Base URL: the root address for using the Web Services.

This Web Service has only one method:

POST Method: Used for loading and replacing archives.

https://api-scl01.masterosr.com/in/v2/
https://api-scl02.masterosr.com/in/v2/

Each execution of a Web Service delivers:

Result Code: Response

- The RESPONSE is a code that represents the result of an execution:
  - 201 : satisfactory result
  - 400 : problem with execution
  - 500 : problem with service (internal)

Payload:

- The information attached to the so-called Web Service and necessary for the action to be executed.

Web Service OSR REST R4:

To perform a load, you must use the POST method.

The endpoints are:

https://api-scl01.masterosr.com/in/v2/{IdCliente}/{IdContainer}
https://api-scl02.masterosr.com/in/v2/{IdCliente}/{IdContainer}

Donde:

ID Client	Numeric identifier of client`s account	Required parameter
ID Container	Identifier of container on which the load will be completed	Required parameter

- Use valid credentials for this Web Service.
- For this method it is necessary to include a XML structure in the PAYLOAD and it must be of the type "multipart/form-data".
- This load only accepts publication of one file at a time. If you want to publish a larger number of files, repeat the process as many times as necessary.

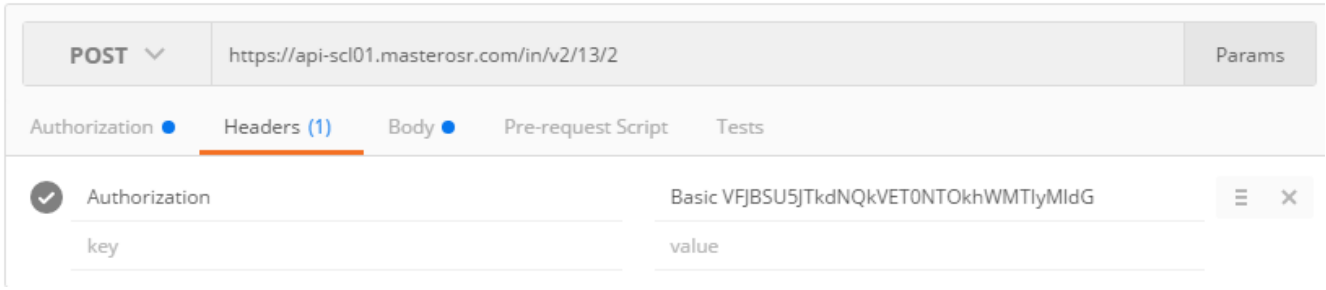
For Web services it is recommended to use tools that allow loading of files, for example, Postman.

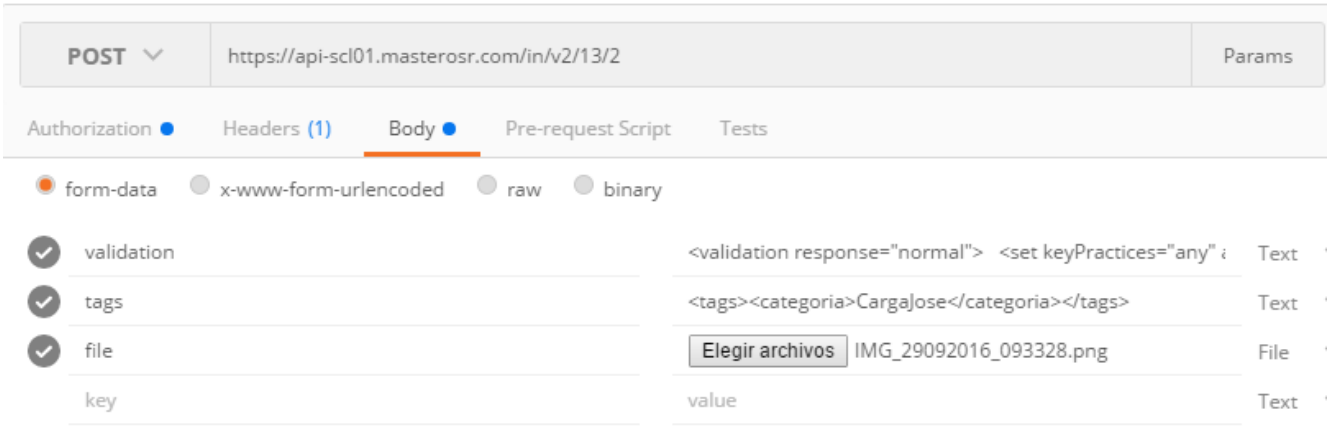
The XML must be created according to the following specification:

Heading				
Heading	Type	Required	Description	Example
Content-Type	string	Yes	Http message format	multipart/form-data
Content-Length	int	Yes	Http message size	> 1
Autorization	Basic	Yes	Credentials to access service	
X-OSR-ClientPath	string	Optional	Unique file path	/misarchivos/test.txt In case no value is assigned for this parameter, the system generates a unique identifier.
X-OSR-ExpirationDate	Date time	Optional	File expiration date	2015-12-30T00:00:00

Body			
Nombre	Type	Description	Example
validation	String(XML)	XML validation, contains file access restriction parameters.	<validation response="normal"> <set keyPractices="any" allowHttp="true"> <httpAuth validation="anonymous"/> </set> </validation>
tags	String(XML)	XML Tags, contains tags to guide file search.	<tags> <categoria>viajes</categoria> <campain>promoción USA</campaña> </tags>
File	file	File to upload.	

Example:





Result:

The extract of the list result (if successful) will reflect the following structure:

<OSRResponse>
<statusCode>201</statusCode>
<statusDescription>Created</statusDescription>
<location Hash="e976af72d9a6644d7422c16d64ce597a"
FileSize="33">https://api.masterosr.com/out/v1/1/2/demoApi/archivos/saludos_3.text</locatio
n>
</OSRResponse>

To replace files, you must use the POST method.

The endpoints are:

https://api-scl01.masterosr.com/in/v2/{IdCliente}/{IdContainer}
https://api-scl02.masterosr.com/in/v2/{IdCliente}/{IdContainer}

Where:

IDClient	Numeric identifier of client´s accounts	Required parameter
IDContainer	Identifier of container on which the load will be completed	Required parameter

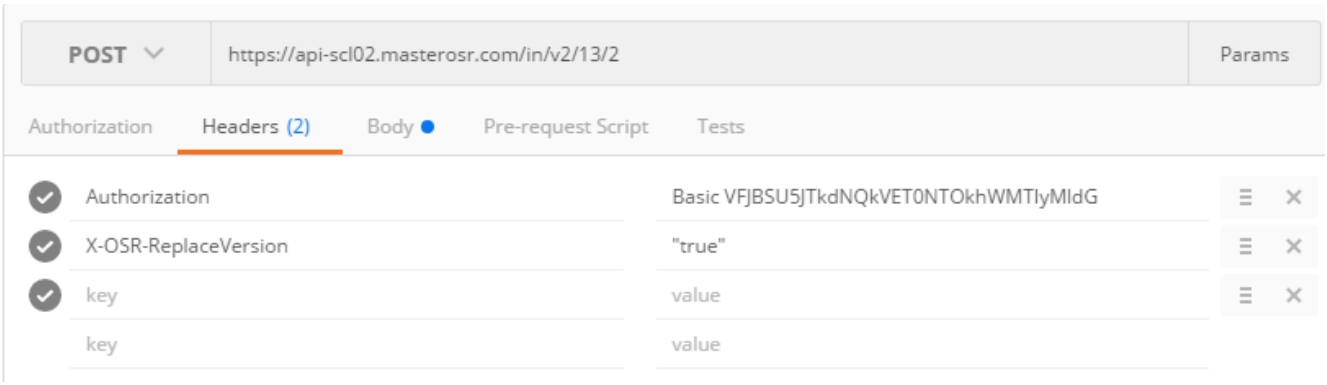
- Use valid credentials for this Web Service.
- For this method it is necessary to include a XML structure in the PAYLOAD and it must be of the type "multipart/form-data".
- This load only accepts publication of one file at a time. If you want to publish a larger number of files, repeat the process as many times as necessary.

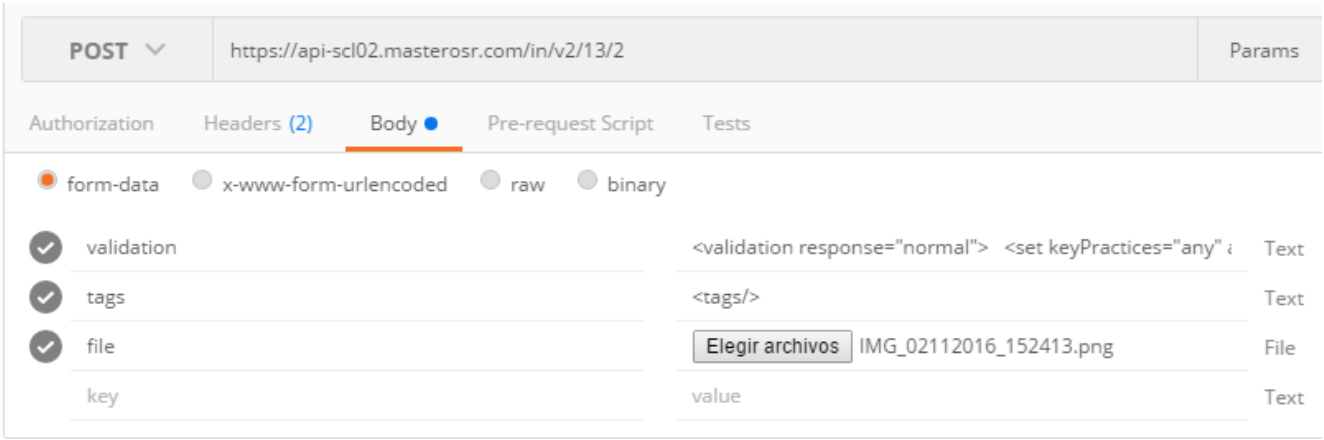
For Web services it is recommended to use tools that allow the loading of files, for example, Postman.

Heading				
Heading	Type	Required	Description	Example
Content-Type	string	Yes	Http message format	multipart/form-data
Content-Length	int	Yes	Http message size	> 1
Autorization	Basic	Yes	Credentiales to access service	
X-OSR-ClientPath	string	Yes	Unique file path to be replaced	/misarchivos/test.txt
X-OSR-ReplaceVersion	string	Yes	Indicates the replacement for the file	true

Body			
Name	Type	Description	Example
File	file	File to upload	

Example:





Result

The result (if successful) will reflect the following structure:

```
<OSRResponse>
 <statusCode>200</statusCode>
 <statusDescription>Created</statusDescription>
 <location Hash="e976af72d9a6644d7422c16d64ce597a"
FileSize="33">https://api.masterosr.com/out/v1/1/2/demoApi/archivos/saludos_3.text</locatio
n>
```

</OSRResponse>

Here is an example to integrate these methods

Loading files:

```
public static string Upload(string urlBase, string customer, string
container, string username, string password, string validation, string tags,
string filePath, string clientPath, DateTime? expDate, double? timeout)
{
    #region Validaciones de parámetros.
   // Parameter validations are intentionally omitted, in order to simplify.
    #endregion

   // the HttpClientHandler object is created, to establish credentials and
other parameters.
    using (var handler = new HttpClientHandler())
    {
       // required  to send credentials
        handler.PreAuthenticate = true;
       // the provided credentials are added.
        handler.Credentials = new NetworkCredential(username, password);
       // if necessary a proxy can be defined
        //handler.Proxy = new WebProxy("10.1.1.1", 8080);

        var baseApiUri = new Uri(urlBase);
        var customerApiUri = new Uri(baseApiUri, string.Format("{0}/{1}",
customer, container));

       // creates the HttpClient object that will be responsible for sending
the query.
        using (var client = new HttpClient(handler))
        {
           // if the timeout parameter has value.
            if (timeout.HasValue)
            {
                client.Timeout = TimeSpan.FromSeconds(timeout.Value);
            }
            else
            {
               //Default value.
                client.Timeout = TimeSpan.FromSeconds(30);
            }

           // create the request object that will contain the message (header
and body
            using (var request = new HttpRequestMessage(HttpMethod.Post,
customerApiUri))
            {
               // If the expiration date parameter has value, the X-OSR-
ExpirationDate header is added.
                if (expDate.HasValue)
                {
                    request.Headers.Add("X-OSR-ExpirationDate",
expDate.Value.ToString("s"));
```

```
                }

                // if the client path parameter has value, the X-OSR-
ClientPath header is added.
                if (!string.IsNullOrWhiteSpace(clientPath))
                {
                    request.Headers.Add("X-OSR-ClientPath", clientPath);
                }

                using (var content = new MultipartFormDataContent())
                {
                   // XML validation is added inside the object.
                    var validationContent = new StringContent(validation);
                    validationContent.Headers.ContentType = null;
                    content.Add(validationContent, "\"validation\"");

                   // XML validation is added inside the object.
                    var tagsContent = new StringContent(tags);
                    tagsContent.Headers.ContentType = null;
                    content.Add(tagsContent, "\"tags\"");

                    // the StreamContent object is created to add the file to
the message.
                    using (var fileContent = new StreamContent(new
FileStream(filePath, FileMode.Open, FileAccess.Read)))
                    {
                        // defines the layout of the file.
                        fileContent.Headers.ContentDisposition = new
ContentDispositionHeaderValue("form-data")
                        {
                            Name = "\"file\"",
                            FileName = "\"" + Path.GetFileName(filePath) +
"\""
                        };

                        // indicates the contentType of the file.
                        fileContent.Headers.ContentType = new
MediaTypeHeaderValue(MimeTypes.GetMimeType(filePath));
                        // the file is added to the message content.
                        content.Add(fileContent, "file");

                        // the content is assigned to the search.
                        request.Content = content;

                        // the search is made and the answer is obtained.
                        using (var response =
client.SendAsync(request).Result)
                        {
                            // ensures the generation of an exception if the
response is not successful.
                            response.EnsureSuccessStatusCode();

                            // read the content of the response
                           var responseString =
response.Content.ReadAsStringAsync().Result;
                            // the answer is returned.
                            return responseString;
```

```
                                    // return only the URL of the file.
                                    //return response.Headers.Location.AbsoluteUri;

                                }// response
                            }

                        }// content
                    }// request
                }// client
            }// handler
        }
```

MasterBase

Files replacement:

```
public static string Replace(string urlBase, string customer, string
container, string username, string password, string filePath, string
clientPath, int? timeout)
{
    // object is created HttpClientHandler, to establish credentials and other
parameters.
    using (var handler = new HttpClientHandler())
    {
        // is forced to send the credentials.
        handler.PreAuthenticate = true;
        // the credentials are added.
        handler.Credentials = new NetworkCredential(username, password);
        // if necessary a proxy can be defined
        //handler.Proxy = new WebProxy("10.1.1.1", 8080);

        var baseApiUri = new Uri(urlBase);
        var customerApiUri = new Uri(baseApiUri, string.Format("{0}/{1}",
customer, container));

        // object HttpClient is created, which will be responsible for sending
the query.        using (var client = new HttpClient(handler))
        {
            // if the timeout parameter has value.
            if (timeout.HasValue)
            {
                client.Timeout = TimeSpan.FromSeconds(timeout.Value);
            }
            else
            {
                // Default value.
                client.Timeout = TimeSpan.FromSeconds(30);
            }

            // the request object is created and will contain the message
(header and body)            using (var request = new
HttpRequestMessage(HttpMethod.Post, customerApiUri))
            {

                // the header X-OSR-ClientPath is added with the parameter
client path.
                request.Headers.Add("X-OSR-ClientPath", clientPath);

                // the header X-OSR-ReplaceVersion is added to indicate to the
service that it is a replacement.
                request.Headers.Add("X-OSR-ReplaceVersion", "true");
using (var content = new MultipartFormDataContent())
                {

                    // the StreamContent object is created to add the file to
the message.                    using (var fileContent = new
StreamContent(new FileStream(filePath, FileMode.Open, FileAccess.Read)))
                    {
                        // the file layout is defined.
                        fileContent.Headers.ContentDisposition = new
ContentDispositionHeaderValue("form-data")
```

```
                        {
                            Name = "\"file\"",
                            FileName = "\"" + Path.GetFileName(filePath) +
"\""
                        };

                        // the contentType of the file is indicated.
                        fileContent.Headers.ContentType = new
MediaTypeHeaderValue(MimeTypes.GetMimeType(filePath));
                        // the file is added to the message content.
                        content.Add(fileContent, "file");

                        // the content is assigned to the query.
                        request.Content = content;

                        // the query is made and the answer is obtained.
                        using (var response =
client.SendAsync(request).Result)
                        {
                            // the generation of an exception is ensured if
the response is not
satisfactory.                               response.EnsureSuccessStatusCode();

                            // the response content is read
                            var responseString =
response.Content.ReadAsStringAsync().Result;
                            // the answer is returned .
                            return responseString;

                            // It could return only the URL of the file.
                            //return response.Headers.Location.AbsoluteUri;

                        }// response
                    }

                }// content
            }// request
        }// client
    }// handler
}
```

Call to methods:

```
class Program
{
    static void Main(string[] args)
                //Important: This example ignores multiple validations and
exception controls, in order to simplify and make the code easier to
understand.
            // Sample Input Parameters            var urlBase = "https://api-
scl01.masterosr.com/in/v2/";
            var customer = "1";
            var container = "2";
            var username = "";
            var password = "xxxxx";
            var validation = "<validation response=\"normal\"><set
keyPractices=\"any\" allowHttp=\"true\"><httpAuth
validation=\"anonymous\"/></set></validation>";
            var tags = "<tags/>";
            var filePath = @"c:\temp\saludo.txt";


            // For more details go to the OSR class, to find the definition of
Upload.            var resultUpload = OSR.Upload(urlBase, customer,
container, username, password, validation, tags, filePath);
            Console.WriteLine(resultUpload);


            // For more details go to the OSR class, to find the definition of
Replace.
            var resultReplace = OSR.Replace(urlBase, customer, container,
username, password, filePath, "misdocumentos/demo/saludo.txt");
            Console.WriteLine(resultReplace);

            Console.ReadKey();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
            Console.ReadKey();
        }
    }
}
```