Oracle Visual Builder Blog

Follow:

REST Backends

Upload / Download files from OCI Storage in a Visual Builder app

September 21, 2021 | 7 minute read

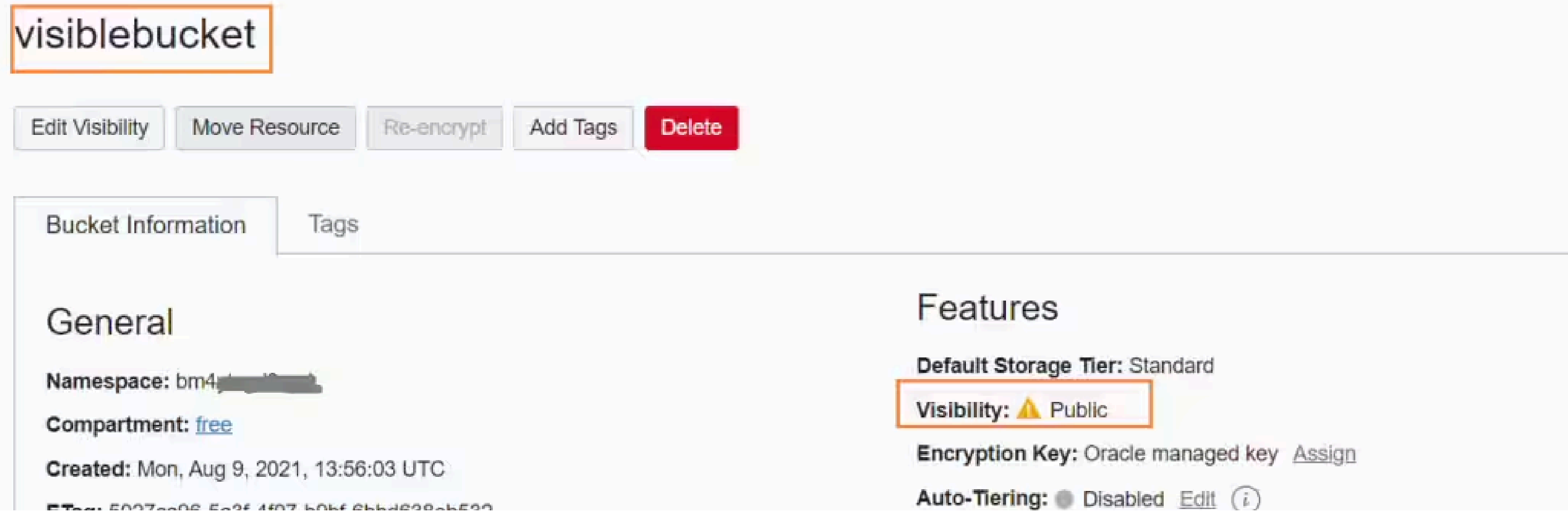
Aparna Gaonkar
Product Manager

OCI Object Storage is a versatile service which is very commonly used to store application data like images, files etc. In this blog, we discuss how to upload and download an object present in the OCI Object Storage from a Visual Builder app.

First we need to have a bucket that represents where we will be storing our objects.

Create a Bucket in OCI Storage

From the OCI Console, navigate to Storage → Buckets. Create a Bucket that you will be using for storage of files. For simplicity this Bucket is marked as Public, so that it doesnt require authentication, but you could achieve the same with Private visibility buckets as well.



visiblebucket

General

Namespace: bm4[redacted]

Compartment: free

Created: Mon, Aug 9, 2021, 13:56:03 UTC

ETag: 5027aa06-5a3f-4f07-b0bf-6bbd638ab532

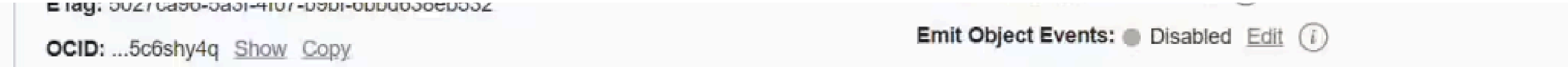
Features

Default Storage Tier: Standard

Visibility: ⚠ Public

Encryption Key: Oracle managed key Assign

Auto-Tiering: ● Disabled Edit ⓘ



OCID: ...5c6shy4q Show Copy

Emit Object Events: Disabled Edit

Upload from VB

Below are the steps to implement the upload use case:

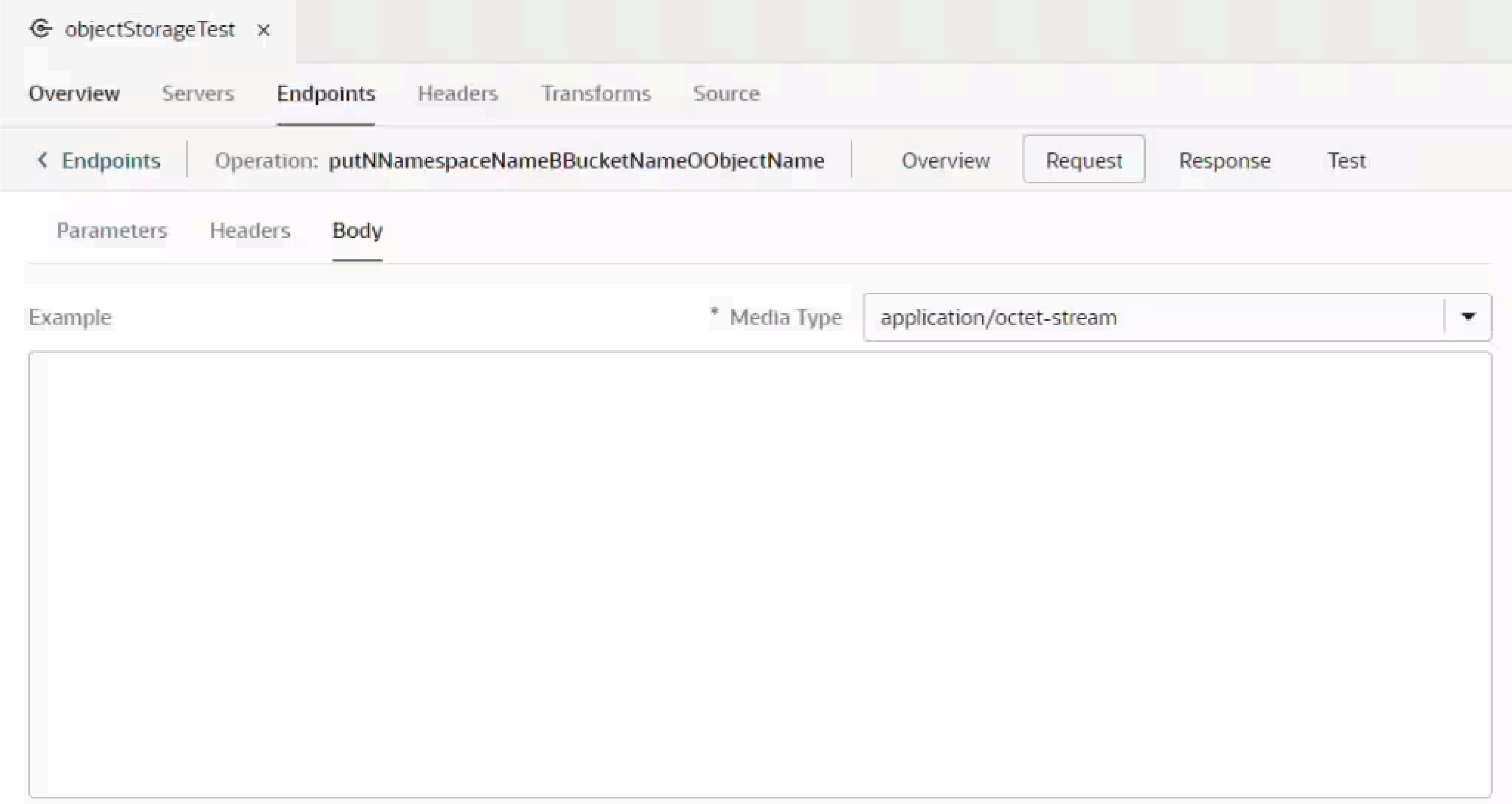
Setup a Service Connection to OCI Storage

We will use the PutObject OCI Storage API that is used for uploading objects to a bucket in the storage. The Content-Type header in this API is defaulted to `application/octet-stream`, but it is a good practice to have it same as the standard MIME type format of the object, so that when you retrieve the object (via the GetObject API), you can get the same MIME format back.

Create a Service Connection to OCI Storage by using the endpoint flow with the following details:

General
Method : PUT
URL : https://[OCI Regional host]/[version]//n/{namespaceName}/b/{bucketName}/o/{objectName} (e.g. https://objectstorage.ap-mumbai-1.oraclecloud.com/n/{namespaceName}/b/{bucketName}/o/{objectName})
Action Hint : Create
Server tab
Authentication = None (since we marked the bucket Public). In case you want to upload to a protected bucket, choose OCI HTTP Signature authentication, and add the necessary details as is documented in this blog
Request tab
Media Type = application/octet-stream. We will override this in the REST Action to be the mime type from the selected file
Response tab
There is no response body for a successful call, so just leave it empty as-is with the default settings.

Clicking on Create button will create the Service Connection as below

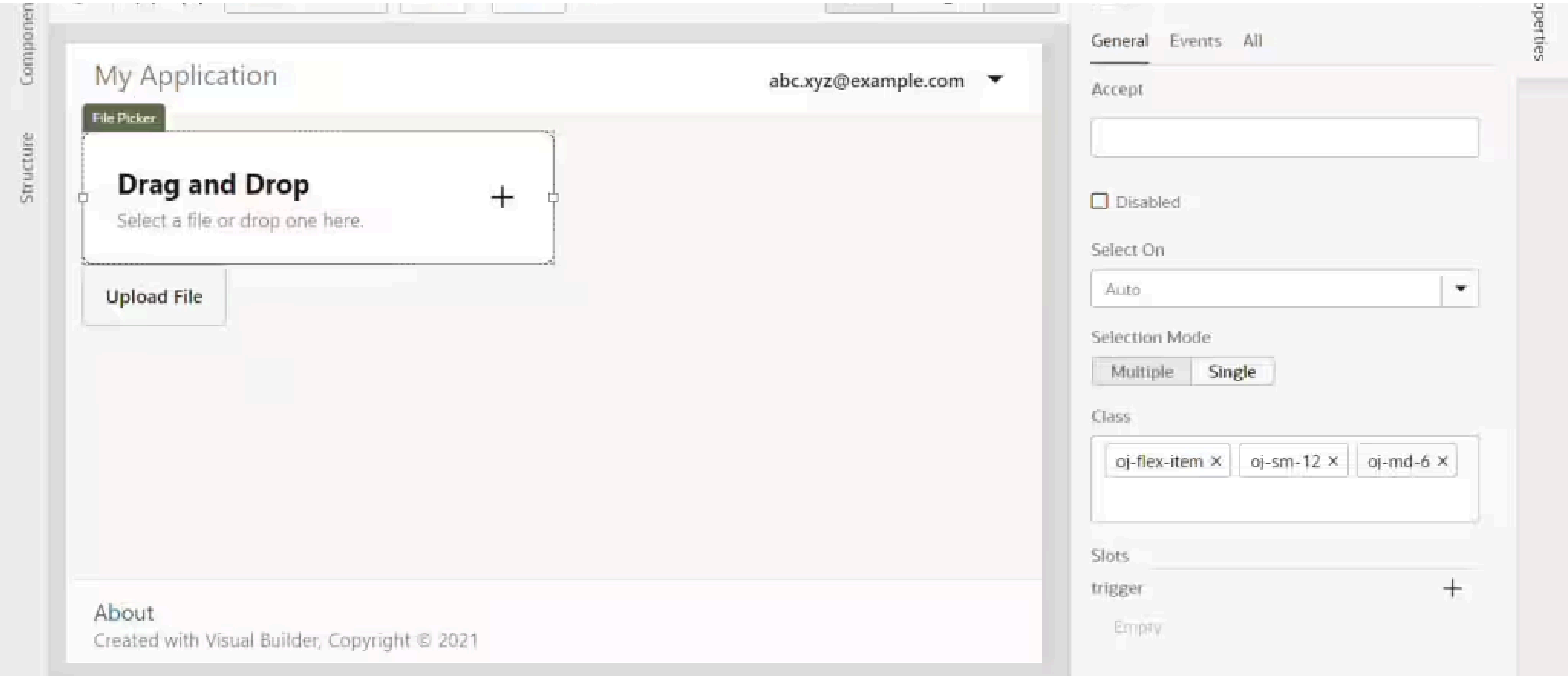


Operation: putNNamespaceNameBBucketNameOObjectName

* Media Type application/octet-stream

Create a webapp with a File picker

Create a webapp (called *uploadstoragefile* in my example), and on the page designer drag a File Picker component. In the properties tab of the File Picker component, use Selection mode = Single. Drag a button on the page and change the text to "Upload File".



Accept

☐ Disabled

Select On: Auto

Selection Mode: Multiple | Single

Class: oj-flex-item × oj-sm-12 × oj-md-6 ×

Slots

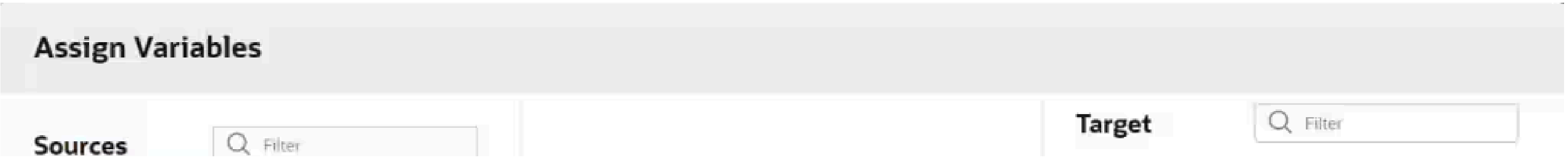
trigger +

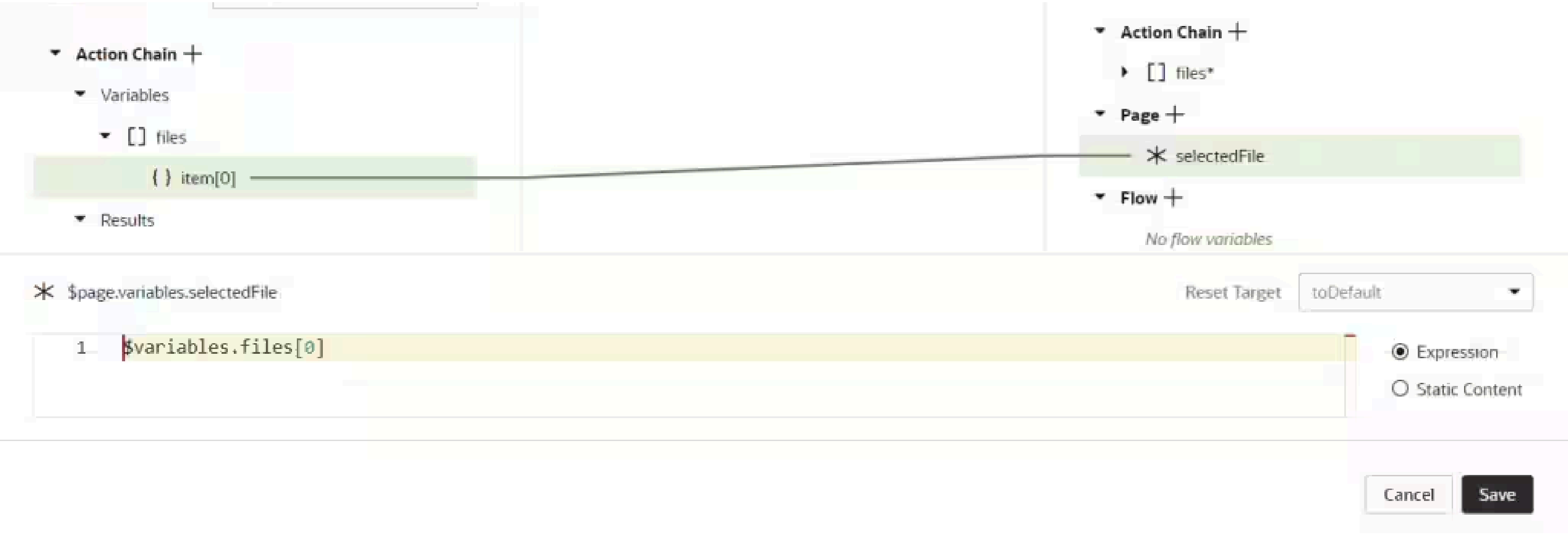
Empty

Also create a page variable called *selectedFile* of type Any. This will be used to store the file we select from the File Picker

Create an action chain to assign a variable to store the selectedFile

In the property inspector of the File Picker, click the events tab, select *Selected File* event and proceed to create an action chain attached to this event. Within the action chain, add *Assign variables* action from the palette and set the *selectedFile* variable to the action chain to input parameter *variables.files[0] as shown*





$page.variables.selectedFile

Reset Target: toDefault

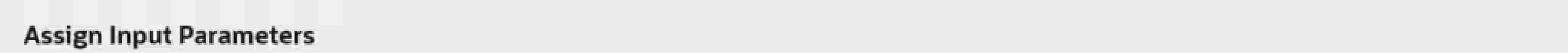
```
$variables.files[0]
```

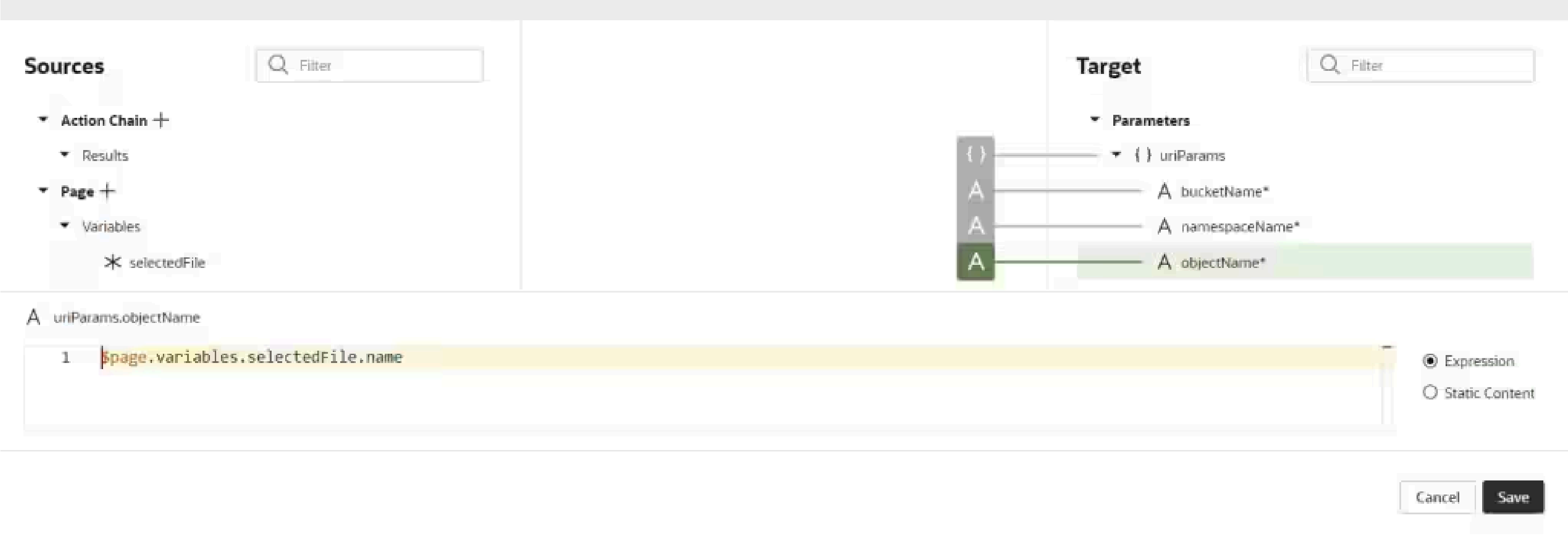
● Expression
○ Static Content

Create an action chain to upload the file from the file picker to service

In the property inspector of the "Upload File" button, click the events tab, select *ojAction* event and proceed to create an action chain attached to this event. Within the action chain, add *Call REST* action from the palette and set the following details on it:

Endpoint = select the endpoint of the PutObject Service Connection
Parameters
bucketName = the name of your bucket. In my example, it is *visiblebucket*
namespaceName = the namespace to which your bucket belongs to. E.g. *bm4yxxxx*
objectName = *$page.variables.selectedFile.name* - This is the name of the file you selected.
Body = *$page.variables.selectedFile* - This represents the file blob
Content-Type = *[[$page.variables.selectedFile.type]]*





Sources

Action Chain
Results
Page
Variables
selectedFile

Target

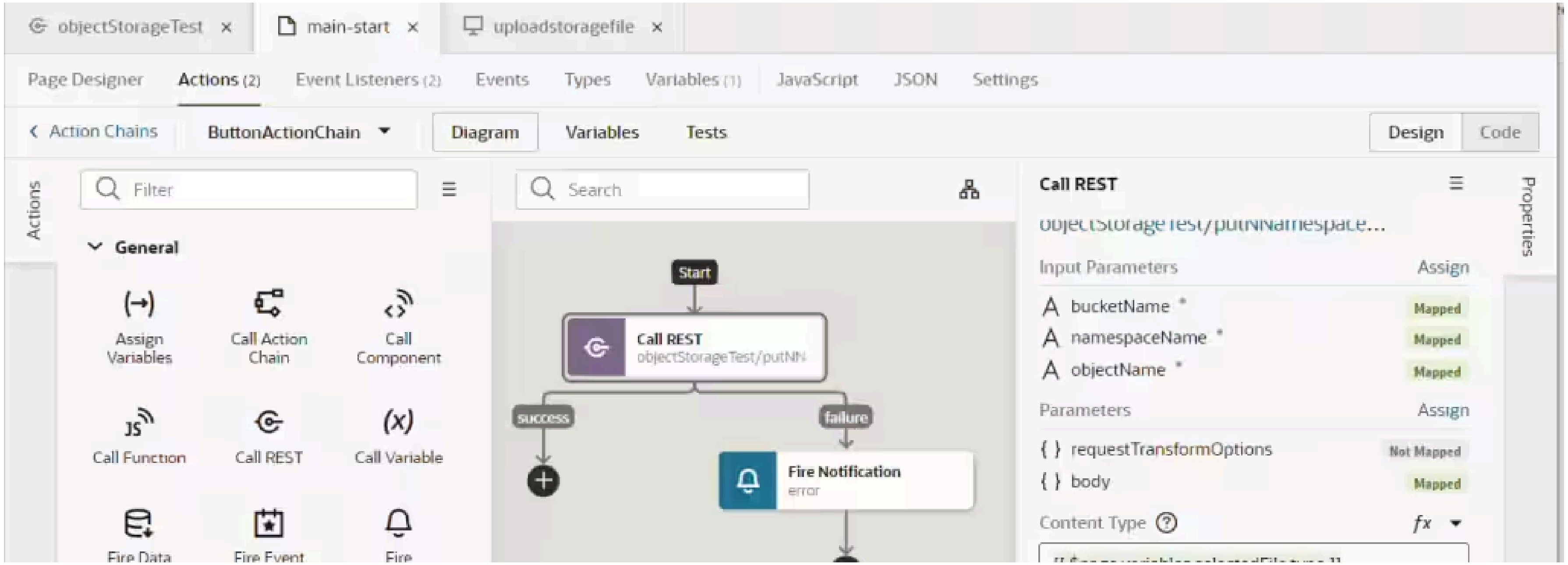
Parameters
{ } uriParams
bucketName*
namespaceName*
objectName*

uriParams.objectName

```
$page.variables.selectedFile.name
```

● Expression
○ Static Content

Cancel
Save



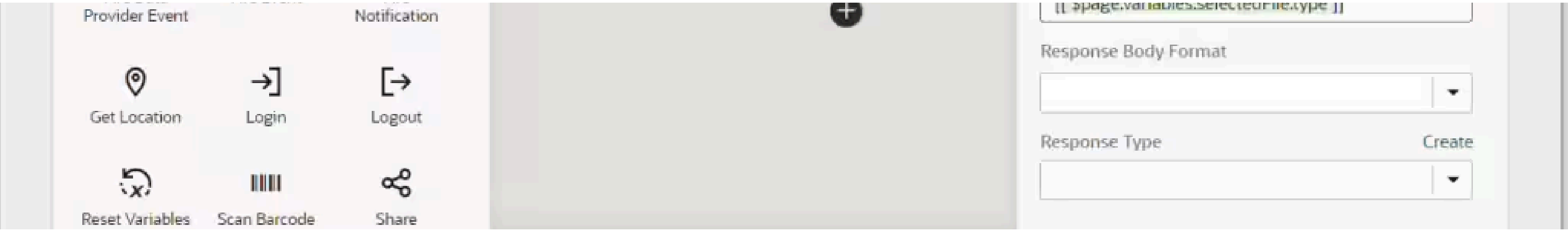
Call REST
objectStorageTest/putNN

Call REST
objectStorageTest/putNNamespace...

Input Parameters Assign
bucketName * Mapped
namespaceName * Mapped
objectName * Mapped

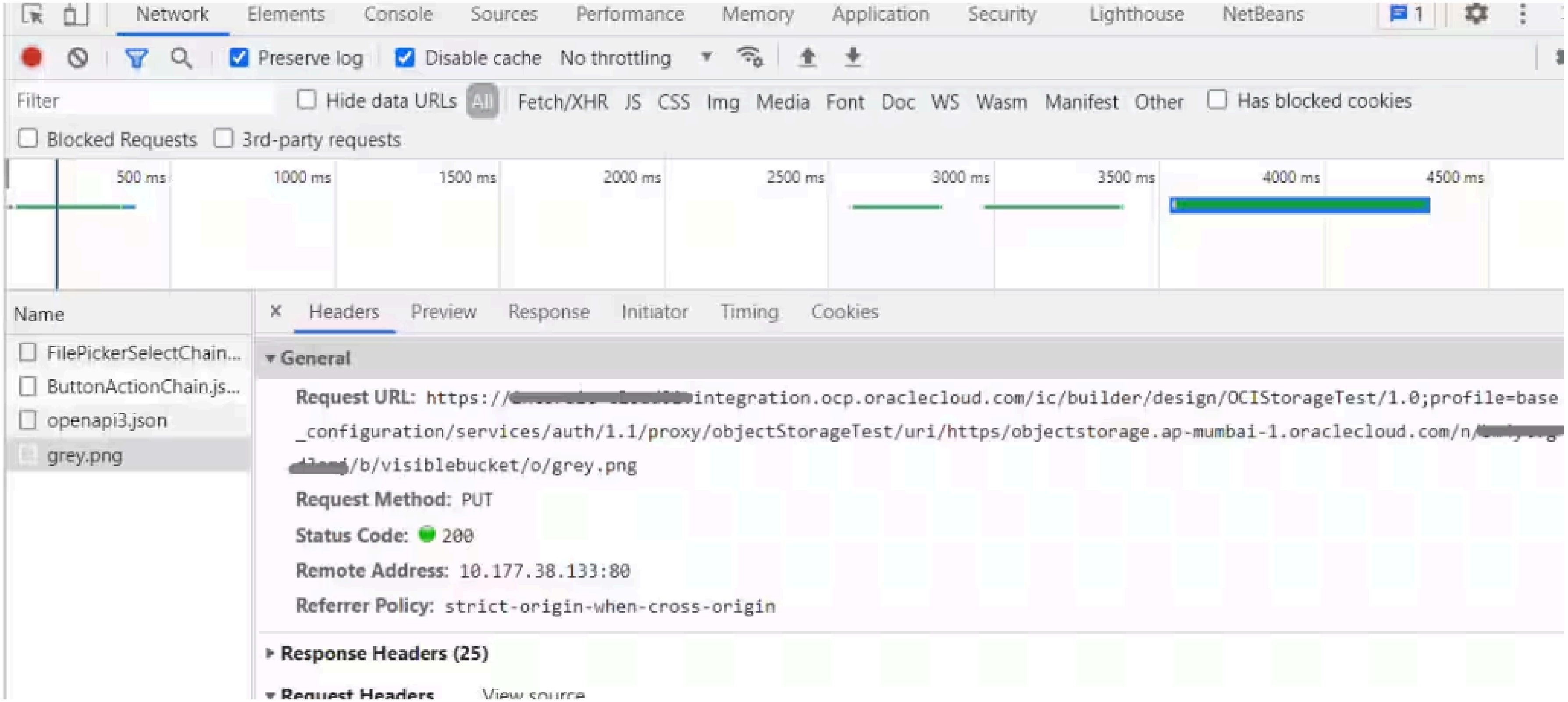
Parameters Assign
{ } requestTransformOptions Not Mapped
{ } body Mapped

Content Type fx



Running/Debugging the application

Run the web app using the preview button. You should see the action run successfully and the file being created in your bucket. You can get a detailed view of what happens when you see the Developer Tools Network console and the JavaScript Console



□ FilePickerSelectChain...
□ ButtonActionChain.js...
□ openapi3.json
□ grey.png

▾ General

Request URL: `https://[redacted]integration.ocp.oraclecloud.com/ic/builder/design/OCIStorageTest/1.0;profile=base_configuration/services/auth/1.1/proxy/objectStorageTest/uri/https/objectstorage.ap-mumbai-1.oraclecloud.com/n/[redacted]/b/visiblebucket/o/grey.png`

Request Method: PUT

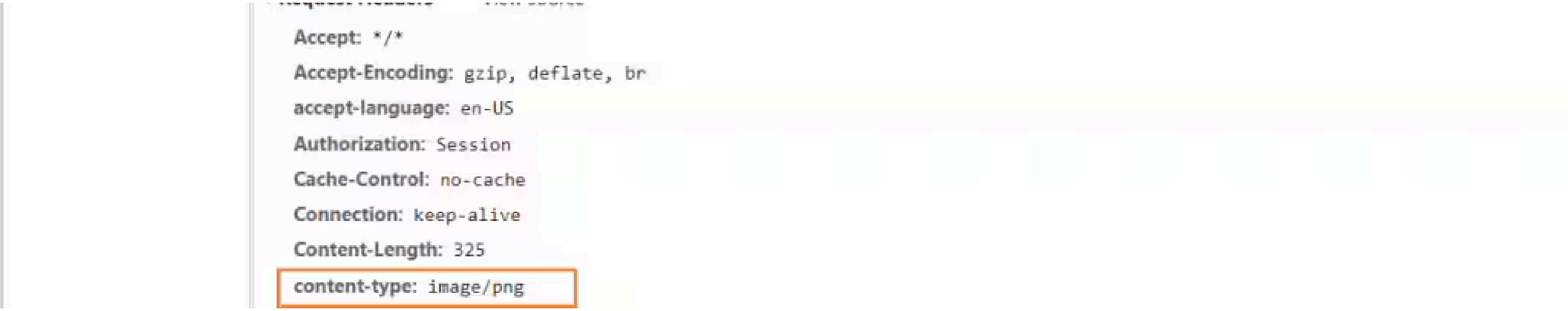
Status Code: ● 200

Remote Address: 10.177.38.133:80

Referrer Policy: strict-origin-when-cross-origin

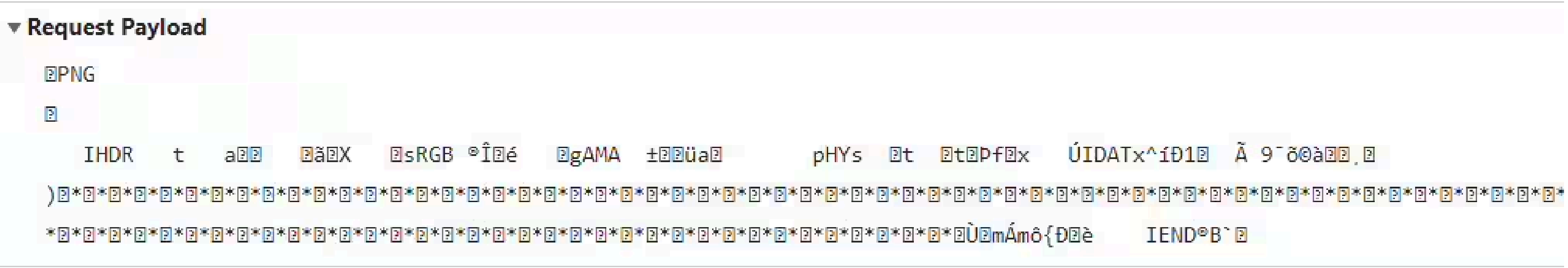
▸ Response Headers (25)

▾ Request Headers View source



```
Accept: */*
Accept-Encoding: gzip, deflate, br
accept-language: en-US
Authorization: Session
Cache-Control: no-cache
Connection: keep-alive
Content-Length: 325
content-type: image/png
```

If you scroll down in the Network console, you can also see the binary payload of the file



▾ Request Payload

```
�PNG
�
   IHDR   t   a��   �ã�X   �sRGB ®Î�é   �gAMA  ±��üa�        pHYs  �t  �t�Pf�x   ÚIDATx^íÐ1�  Ã 9¯õ®à��¸�
)�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*
*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�*�ÙmÁmô{Ð�è    IEND®B`�
```

Now, let us complete the download use case for the file as well

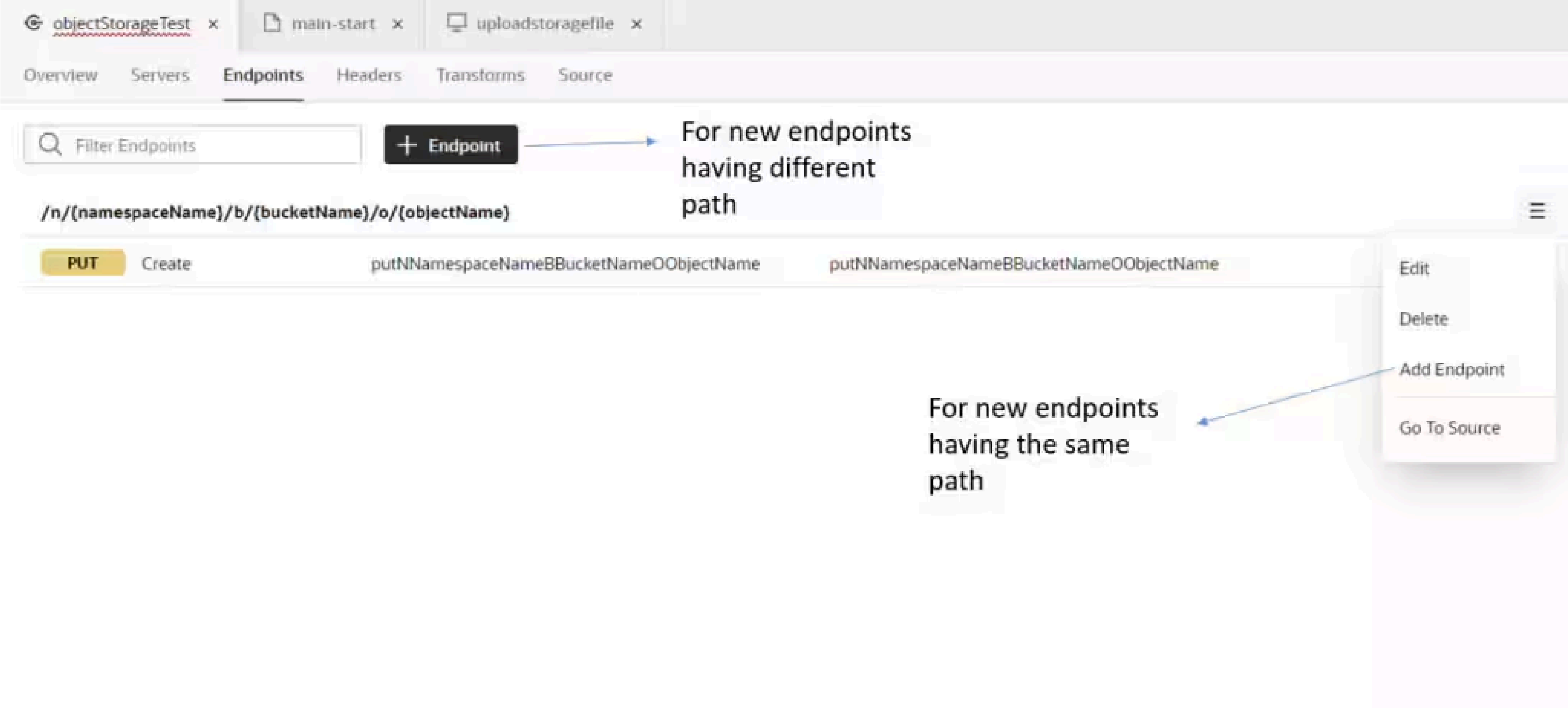
Download from OCI Storage

Add the GetObject endpoint to the Service Connection

I am going to utilize the same Service Connection and add the GetObject endpoint to it. This reduces the number of service connections in your app, and is more effecient than creating a separate Service Connecction for each endpoint.

Since the PutObject and GetObject have the same path, I am going to select the path and click *Add Endpoint*. If they had different paths, I

Since the PutObject and GetObject have the same path, I am going to select the path and click *Add Endpoint*. If they had different paths, I would have chosen the *Add Endpoint* on the top of the Service Connection.


objectStorageTest ×
main-start ×
uploadstoragefile ×
Overview
Servers
Endpoints
Headers
Transforms
Source
Filter Endpoints
+ Endpoint
For new endpoints having different path
/n/{namespaceName}/b/{bucketName}/o/{objectName}
PUT
Create
putNNamespaceNameBBucketNameOObjectName
putNNamespaceNameBBucketNameOObjectName
Edit
Delete
Add Endpoint
Go To Source
For new endpoints having the same path

Create the Endpoint with the following details:

Method = GET
Action Hint = Get One
URL = same as PutObject
Response tab
Add an example response body as given in the GetObject REST API documentation
`{ "ETag": "3A635DC17E6B0AADE053C20610AC8015", "MD5": "36Cut+u0HeVugCulKkuVzAg==", "Size": "13", "Trace-ID":`

```
{ "ETag" : "3A63FDC17F6B9AADE053C20610AC801F" , "MD5" : "36Gwtu9HovxgCwIkK+VgAg==" , "Size" : 12 , "Trace ID" :
"d13fe5a5-0358-9d32-a25c-d6864b5b9a64" }
```

Copy code snippet

Click Save to create the endpoint.

Create a webapp with a download button

Create another webapp (called *downloadstoragefile* in my case), and on the page designer drag a button and change the text to *Download File*

Create a JavaScript function to download the blob returned from OCI Storage

From the page designer, navigate to JavaScript and proceed to create a page module function as below:

Copy code snippet

```
PageModule.prototype.downloadFile = function (data, mimeType, filename) {

const blob = new Blob([data], {

type: mimeType

});

// IE/Edge
if (window.navigator && window.navigator.msSaveOrOpenBlob) {

window.navigator.msSaveOrOpenBlob(blob);

return;

}

var link = document.createElement('a');

link.href = URL.createObjectURL(blob);

link.download = filename;

link.click();

// Firefox: delay revoking the ObjectURL

setTimeout(function() {

URL.revokeObjectURL(blob);

}, 100);

};
```

This function will be responsible for taking the binary data from the Rest call and converting into a JavaScript Blob which can be downloaded with a proper name

Create an action chain to download from OCI Storage

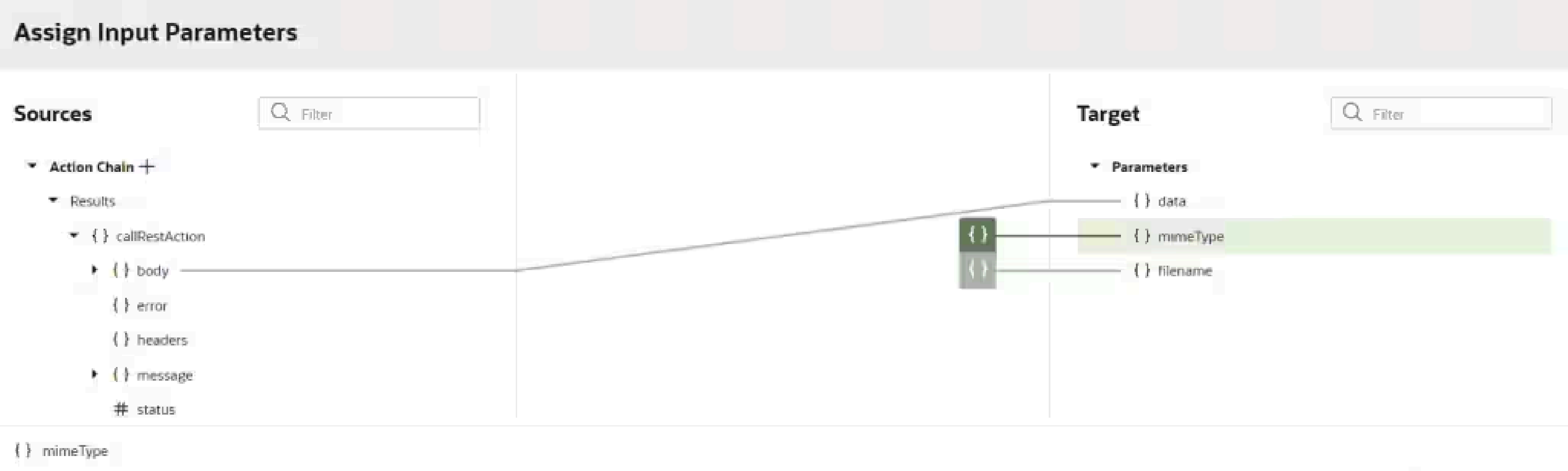
In the property inspector of the 'Download File' button, click the events tab, select *ojAction* event and proceed to create an action chain attached to this event. Within the action chain, add *Call REST* action from the palette and set the following details:

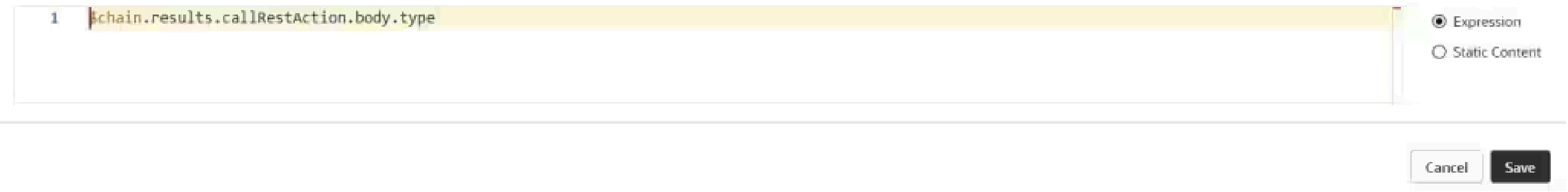
attached to this event. Within the action chain, add Call REST action from the palette and set the following details:

Endpoint = select the endpoint of the GetObject Service Connection
Parameters
bucketName = the name of your bucket. E.g. visiblebucket
namespaceName = the namespace to which your bucket belongs to. E.g. bm4yxxxx
objectName = $page.variables.selectedFile.name - name of the object in object storage (e.g. grey.png)

After the REST action, add the *Call Function* action to the chain with the following details

Function = downloadFile
Parameters
data - Map this to the body of the result from the rest action i.e. *$chain.results.callRestAction.body*
mimeType - mimetype of the returned blob i.e. *$chain.results.callRestAction.body.type*
filename - The name by which you want the file to be downloaded i.e. *download.png*





```
$chain.results.callRestAction.body.type
```

● Expression
○ Static Content

Run the app using the preview button and click on the download button. You can see more details from the developer tools JavaScript console and the Network console.

This completes the procedure on how to upload/download objects from OCI Object Storage service. The same process can be used for other data sources whose REST APIs have `application/octet-stream` MIME type. Also check out similar blogs on REST services in VB:

REST APIs with different formats

- Consuming REST APIs with multipart/form-data
- Consuming REST APIs with application/x-www-form-urlencoded payload

Other blogs in the VB - OCI Integration series

- Using OCI Signature authentication for services
- Connecting to Oracle Functions from Visual Builder
- Connecting to API GW from Visual Builder
- Connecting to API Gateway with IDCS authentication

Aparna Gaonkar
Product Manager

Previous Post

Running Visual Builder Apps Outside of Visual Builder

Shay Shmeltzer | 3 min read

Next Post

Aggregate Data from Business Objects with Functions

Shay Shmeltzer | 3 min read

Resources for

About
Careers
Developers
Investors
Partners
Startups

Why Oracle

Analyst Reports
Best CRM
Cloud Economics
Corporate Responsibility
Diversity and Inclusion
Security Practices

Learn

What is Customer Service?
What is ERP?
What is Marketing Automation?
What is Procurement?
What is Talent Management?
What is VM?

What's New

Try Oracle Cloud Free Tier
Oracle Sustainability
Oracle COVID-19 Response
Oracle and SailGP
Oracle and Premier League
Oracle and Red Bull Racing Honda

Contact Us

US Sales 1.800.633.0738
How can we help?
Subscribe to Oracle Content
Try Oracle Cloud Free Tier
Events
News