Abstract

Swift Storlets extend Swift with the capability to run computation near the data in a secure and isolated manner. With Swift Storlets a user can write code, package and deploy it as a Swift object, and then explicitly invoke it on data objects as if the code was part of the Swift pipeline. We use the term Storlet to refer to the binary code deployed as a Swift object. Invoking a Storlet on a data object is done in an isolated manner so that the data accessible by the computation is only the object's data and its user metadata. Moreover, the computation has no access to disks, network or to the Swift request environment.

API Summary

- Storlets
  - Storlets
    GET - List All Storlets [/storlet]
  - Storlet
    PUT - Upload a New Storlet [/storlet/{storlet_object_name}]
- Dependency
  - Operations on dependencies
    PUT - Upload a New Dependency [/dependency/{dependency_object_name}]
- Storlets Invocation
  - Invocation
    GET - Invoke a storlet upon object download [/{container_name}/{object_name}]
    PUT - Invoke a storlet upon object upload [/{container_name}/{object_name}]

API Specification

Storlets

Storlets **[/storlet]**

Storlets are uploaded to a special container called 'storlet'. Therefore, '/storlet' shows up in the url of various operations, such as to upload a storlet or to list all of the available storlets.

List All Storlets

GET /storlet

Response 200 (application/json)

Go to example

Storlet [/storlet/{storlet_object_name}]

Upload a New Storlet

PUT /storlet/{storlet_object_name}

You may write a storlet in Java according to the instructions in the Developer's manual and upload it to the Object Store. The storlet is uploaded to the container named 'storlet', so '/storlet' appears in the url. The storlet may depend on other existing libraries, which must first be uploaded to the dependency container. When uploading a storlet, the X-Object-Meta-Storlet-Dependency header requires a value that is a comma separated list of dependencies. The main_class_name parameter for the X-Object-Meta-Storlet-Main header specifies the class in which the invoke method of the storlet is defined. The X-Object-Meta-Storlet-Language header specified the language in which the storlet is run. At present, only 'Java' is supported. The X-Object-Meta-Storlet-Interface-Version header should be provided and set to the value '1.0'. Although not currently used, the X-Object-Meta-Storlet-Object-Metadata header must be provided and set to 'no'. See the Storlets Developer's manual for details of the signature of the invoke method. The content-type of the request should be set to 'application/octet-stream'.

Request (application/octet-stream)

Response 201

Go to example

Dependency

Operations on dependencies [/dependency/{dependency_object_name}]

Upload a New Dependency

PUT /dependency/{dependency_object_name}

You may create and upload your own libraries to assist in running a storlet. These dependencies are uploaded to a special container named 'dependency'. For an executable dependency (such as a compiled c program), you must specify the X-Object-Meta-Storlet-Dependency-Permissions header. Otherwise, that header may be omitted. The name of the library containing the dependency (i.e. dependency_name) goes into the URI. The content-type of the request should be set to 'application/octet-stream'.

Request (application/octet-stream)

Response 201

Go to example

Storlets Invocation

Once the storlet and its dependencies are deployed the storlet is ready for invocation. Storlets can be invoked in 2 ways: (1) Invocation upon GET. In this case the user gets a transformation of the object residing in the store (as opposed to the actual object). One use case for GET is anonymization, where the user might not have access to certain data unless it is being anonymized by some storlet. (2) Invocation upon PUT. In this case the data kept in the object store is a transformation of the object uploaded by the user (as opposed to the original data or metadata). A typical use case is metadata enrichment, where a Storlet extracts format specific metadata from the uploaded data and adds it as Swift metadata.

Invocation involves adding an extra header ('X-Run-Storlet') to the Swift original PUT/GET requests.

Invocation [/{container_name}/{object_name}]

Invoke a storlet upon object download

GET /{container_name}/{object_name}

An additional header ('X-Run-Storlet') must be provided to inform the system to run a storlet.

Request

Response 200

Go to example

Invoke a storlet upon object upload

PUT /{container_name}/{object_name}

An additional header ('X-Run-Storlet') must be provided to inform the system to run a storlet.

Request

Response 201

Go to example

Examples

Storlets

Storlets

[/storlet]

List All Storlets **GET /storlet**

Response 200 (application/json)

Headers

Content-Type: **application/json**

Body

```
[
    storlet1,
    storlet2,
    storlet3
]
```

Storlet

[/storlet/{storlet_object_name}]

Upload a New Storlet **PUT /storlet/{storlet_object_name}**

Request (application/octet-stream)

Headers

```
Content-Type: application/octet-stream
'X-Object-Meta-Storlet-Language': 'Java'
'X-Object-Meta-Storlet-Interface-Version': '1.0'
'X-Object-Meta-Storlet-Dependency': dependencies
'X-Object-Meta-Storlet-Object-Metadata': 'no'
'X-Object-Meta-Storlet-Main': main_class_name
```

Response 201

Dependency

Operations on dependencies

[/dependency/{dependency_object_name}]

Upload a New Dependency **PUT /dependency/{dependency_object_name}**

Request (application/octet-stream)

Headers

```
Content-Type: application/octet-stream
'X-Object-Meta-Storlet-Dependency-Version': '1'
'X-Object-Meta-Storlet-Dependency-Permissions': '0755'
```

Response 201

Storlets Invocation

Invocation

[/{container_name}/{object_name}]

Invoke a storlet upon object download **GET /{container_name}/{object_name}**

Request

Headers

```
'X-Run-Storlet': storlet_name
```

Response 200

Invoke a storlet upon object upload **PUT /{container_name}/{object_name}**

Request

Headers

```
'X-Run-Storlet': storlet_name
```

Response 201