CFG Repository Management Component

API specification version 1.2.0

Component name: REPO - CFG Repository Management
Component deployment name: REPO

Demo server endpoint: http://161.74.26.58/service/rest/v1/cfgrepo/
Demo server GUI: http://161.74.26.58/

Authored by: University of Westminster

Collaboratively editable “live” version of this document:
https://docs.google.com/document/d/11CqiG2sK3dqLBcFsk-fOXgQU-fAV5S3uVO0-nAzJd5A/edit?usp=sharing

Changelog

2018-11-28: Version 1.2.0

- (Minor) Add CFGUM authentication realm, the user can use the token generated by CFGUM as the password to call the API.
- (Minor) Restructure the artefact metadata ontology

2018-09-28: Version 1.1.0

- (Minor) Change some response message of the API according to most recent understanding of CFG project, e.g. metadata ontology
- (Patch) Fix some types

2018-06-25: Version 1.0.0

- Initial release

Introduction

Repository component (REPO) in Cloudifactoring is built on an Open-source repository management framework called Nexus Repository OSS,

https://www.sonatype.com/nexus-repository-oss

REPO's RESTful API was developed as a Nexus plugin that extends Nexus' original RESTful services to meet the needs of the Cloudifacturing project.

As a general artifact manager, Nexus Repository OSS supports various types of popular artifacts:

- Maven
- Npm
- PyPI
- YUM
- Docker
- Bower
- Git Large File Storage (LFC)
- Raw Binary File
- And More

In Cloudifacturing, a plug-in implemented by the University of Westminster supports a new type of artifact called ***Reference Only Artefact***, which stores only metadata references to executable artifacts stored in the external integrated execution engine, e.g. CloudFlow, CloudBroker, Flowbster …

The metadata ontology of ***Reference Only Artefact*** is shown as follows:

- **Id:** The unique identification for the artefact stored in the REPO
- **downloadUrl:** An URL link to the downable asset of the artefact, e.g. a zip file to store the configurations of external executable artefacts.
- **path:** The typical repository storage structure to the asset of the artefact, (*groupid/engineid/projectid/version/assetname*), e.g. *uow/cloudflow/p1/1.0.0/conf.zip*, which means the configuration file named conf.zip was uploaded and stored in the path uow/cloudflow/p1/1.0.0/, the artefact is developed by University of Westminster on the CloudFlow platform, and the artefact belongs to project P1 and the version number is 1.0.0
- **repository:** The repository name where the artefact is created
- **format:** The type of artefact
- **checksum**
    - **sha1**
    - **md5**
- **cfg**
    - **groupid:** The unique identification for the ISV
    - **engineid:** The unique identification for the external integrated execution

engine, e.g. cloudflow, cloudbroker, flowbster
    - **projectid:** The unique identification for the project that related to the artefact
    - **version:** The version number of the artefact
    - **serviceid:** The unique identification for the referenced external executable artefact
    - **serviceUrl:** The URL to call the API to run the referenced external executable artefact
    - **serviceConf:** The configurations are stored as string
    - **serviceConfType:** The format of the configurations, e.g. XML, JSON, YAML
- **Metadata:** Any additional metadata
    - **Key1:value1**
    - **Key2:value2**

An overview of operations on the artefact is shown below:

Operations	**Actor who can perform operations**	**Related RESTful API function**
Browse	End user and Developer	· List repositories: *GET /repositories/* · List artefacts: *GET /artefacts/* · Get an artefact details: *GET /artefacts/{id}*
List	End user and Developer	· List repositories: *GET /repositories/* · List artefacts: *GET /artefacts/*
Search	End user and Developer	· Search artefacts: *GET /search/artefacts*
Download	End user and Developer	· Get an artefact details: *GET /artefacts/{id}* · Download an artefact from ***downloadURL***
Upload	Developer	· Upload an artefact: *POST /artefacts/*
Delete	Developer	· Delete an artefact: *DELETE /artefacts/{artefact_id}*
Custom Metadata Management	Developer	· List artefact's custom metadata: *GET /artefacts/{artefact_id}/metadata* · Add/Update artefact's custom metadata: *PUT /artefact/{artefact_id}/metadata* · Delete artefact's custom metadata by metadata key: *DELETE /artefacts/{artefact_id}/metadata/{key}*

Repositories

A repository is a central place in which artifacts are kept and maintained in an organized way. You can list, show details for repositories.

List repositories: GET /repositories

Lists existing repositories.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token

Response

Parameters:

Name	In	Description
List of repositories • Name • Format • Type • url	body	List all the REPO repositories

Status codes:

Success:

- 200 - OK

Error:

- 401 - Unauthorized

Example:


```
[
    {
        "name": "cfg-snapshots",
        "format": "cfg",
        "type": "hosted",
        "url": "https://localhost:8081/repository/cfg-snapshots"
    },
    {
        "name": "cfg-releases",
```

```
        "format": "raw",
        "type": "hosted",
        "url": "https://localhost:8081/repository/cfg-releases"
    },
    {
        "name": "test-isv-releases",
        "format": "test-format",
        "type": "test-type",
        "url": "https://localhost:8081/repository/test-isv-releases",
    }
]
```

Get repository information: GET /repositories/{repository_name}

Get repository information by repository name.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token

Response

Parameters:

Name	In	Description
name	body	The repository name
format	body	The repository format, e.g. raw, maven2, npm ...
type	body	The repository type, e.g. hosted, proxy ...
url	body	The url link to the repository

Status codes:

Success:

- 200 - OK

Error:

- 401 - Unauthorized
- 404 - Not found

Example:

```
{
    "name": "cfg-snapshots",
```

```
    "format": "raw",
    "type": "hosted",
    "url": "https://localhost:8081/repository/cfg-snapshots"
}
```

Artefacts

A artefact is managed by the ISV developer. Developer can list, upload, show details for, download and delete artefacts.

Upload an artifact: POST /artefacts

Upload an artefact to a previously created repository.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
content-type	header	multipart/form-data
repository (required)	query	Name of the repository to which you would like to upload the artefact
cfg.asset1 (required)	Body (multipart/form-data)	Artefact to be uploaded, file part
cfg.asset1.filename (required)	Body (multipart/form-data)	Artefact file name, String
cfg.groupId (required)	Body (multipart/form-data)	Group ID, e.g. ISV name, String
cfg.engineId (required)	Body (multipart/form-data)	Engine ID, e.g. cloudflow, flowster, cloudbroker, String
cfg.projectId (required)	Body (multipart/form-data)	Project ID, the project the upload artefact belong to, String
cfg.version (required)	Body (multipart/form-data)	The version of the project, String
cfg.serviceId (required)	Body (multipart/form-data)	A reference id to the executable artefact stored in external integrated execution engine, e.g. Cloudflow, Cloudbroker,

		Flowster
cfg.serviceUrl (required)	Body (multipart/form-data)	The URL to run the reference executable artefact
cfg.serviceConf	Body (multipart/form-data)	Any configurations that need to run reference executable artefact
cfg.serviceConfType	Body (multipart/form-data)	The configuration type, e.g. String, XML, YAML, ZIP
cfg.metadata (required)	Body (multipart/form-data)	Platform relevant metadata, e.g. {“key1”:”value1”,”key2”:”value2”},JSON string

Response

Status codes:
Success:

- 204 - No Content

Error:

- 400 - Bad Request: Some content in the request body was invalid
- 401 - Unauthorized
- 403 - Forbidden
- 404 - Not Found: Repository
- 422 - Unprocessable Entity: Repository name is required

List artefacts in a repository: GET /artefacts

Lists existing artifacts in a repository.

Request

Parameters:

Name	**In**	**Description**
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
continuationToken	query	A token returned by a prior request. If present, the next page of results are returned
repository (required)	query	List the artifacts in a repository by a repository name.

		For example: `/artifacts?repository=cfg`

Response

Parameters:

Name	In	Description
items	body	A list of artefacts with basic information; empty array if there are none List of artefacts ● id ● downloadUrl ● path ● repository ● format ● checksum ○ sha1 ○ md5 ● cfg ○ groupid ○ engineid ○ projectid ○ version ○ serviceid ○ serviceUrl ○ serviceConf ○ serviceConfType ● Metadata ○ Key1:value1 ○ Key2:value2
continuationToken	body	A token returned by a prior request. If present, the next page of results are returned

Status codes:

Success:

- 200 - OK: Request was successful

Error:

- 401 - Unauthorized
- 404 - Not found
- 422 - Repository name is required

Example:


```
{
  "items" : [ {
    "downloadUrl" :
"http://161.74.26.58/repository/cfg/uow/cloudbroker/p1/1.0.0/conf.json",
    "path" : "uow/cloudbroker/p1/1.0.0/conf.json",
```

```
    "id" : "Y2ZnOjkzYjliOWViOWE3ZWNiMDZiMzc5NjIzZGZlZjM0OTZj",
    "repository" : "cfg",
    "format" : "cfg",
    "checksum" : {
      "sha1" : "6eb04e36de53b7cd3a4422089216a9ad4a8335d0",
      "md5" : "658cba07ea2107aafad31adc0cc8a455"
    },
    "metadata" : {
      "additionalEngineMetadata" : "value"
    },
    "cfg" : {
      "groupId" : "uow",
      "serviceUrl" : "serviceurl",
      "serviceId" : "serviceid",
      "projectId" : "p1",
      "version" : "1.0.0",
      "serviceConf" : "serviceconf",
      "serviceConfType" : "serviceconftype",
      "engineId" : "cloudbroker"
    }
  }, {
    "downloadUrl" :
"http://161.74.26.58/repository/cfg/uow/cloudflow/p2/1/conf.xml",
    "path" : "uow/cloudflow/p2/1/conf.xml",
    "id" : "Y2ZnOjkzYjliOWViOWE3ZWNiMDY0NTkxYzE3OTk4OTg1ODgz",
    "repository" : "cfg",
    "format" : "cfg",
    "checksum" : {
      "sha1" : "fc131546ea6ef553878e2c0cbcda6144760826f1",
      "md5" : "dbadc276a8cd9349d6b3f3e923bca1de"
    },
    "metadata" : {
      "key1" : "value1",
      "key2" : "value2"
    },
    "cfg" : {
      "groupId" : "uow",
      "serviceUrl" : "surl1",
      "serviceId" : "sid1",
      "projectId" : "p2",
      "version" : "1",
      "serviceConf" : "sconf",
      "serviceConfType" : "xml",
      "engineId" : "cloudflow"
    }
  }],
  "continuationToken" : null
}
```

Get an artefact: GET /artefacts/{artefact_id}

Get a single artefact by ID.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
artefact_id (required)	path	ID of the artifact to get

Response

Parameters:

Name	In	Description
Artefact ● id ● downloadUrl ● path ● repository ● format ● checksum ○ sha1 ○ md5 ● cfg ○ groupid ○ engineid ○ projectid ○ version ○ serviceid ○ serviceUrl ○ serviceConf ○ serviceConfType ● Metadata ○ Key1:value1 ○ Key2:value2	body	Artifact details

Status codes:

Success:

- 200 - OK

Error:

- 400 - Bad Request: Some content in the request was invalid

- 401 - Unauthorized
- 403 - Forbidden
- 404 - Artefact Not found

Example:


```
{
  "downloadUrl" :
"http://161.74.26.58/repository/cfg/uow/cloudbroker/p1/1/conf.xml",
  "path" : "uow/cloudbroker/p1/1/conf.xml",
  "id" : "Y2ZnOjcxYWZlYTU0MGUyM2RkZTUxNWUwMGUzNzYyOGM2MTM0",
  "repository" : "cfg",
  "format" : "cfg",
  "checksum" : {
    "sha1" : "fc131546ea6ef553878e2c0cbcda6144760826f1",
    "md5" : "dbadc276a8cd9349d6b3f3e923bca1de"
  },
  "metadata" : {
    "key1" : "value1",
    "key2" : "value2"
  },
  "cfg" : {
    "groupId" : "uow",
    "serviceUrl" : "surl1",
    "serviceId" : "sid1",
    "projectId" : "p1",
    "version" : "1",
    "serviceConf" : "sconf",
    "serviceConfType" : "xml",
    "engineId" : "cloudbroker"
  }
}
```


Delete an artefact: DELETE /artefacts/{artefact_id}/

Delete an existing artefact by id.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
artefact_id (required)	path	ID of the artefact to delete

Response

Status codes:
Success:

- 204 - No Content

Error:

- 401 - Unauthorized
- 403 - Forbidden
- 404 - Artifact not found

Example:

```
{}
```

Metadata

In addition to the predefined metadata auto-generated by the repository format. Users can add their own custom metadata to the artifacts for easy searching.

List artefact's metadata: GET /artefacts/{artefact_id}/metadata

List the custom metadata by artefact id.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
artefact_id (required)	path	ID of the artefact to get

Response

Parameters:

Name	In	Description
Artifact metadata • groupid • engineid • projectid • version • serviceId • serviceUrl • serviceConf	body	Artefact details with generic metadata and platform-related metadata

• serviceConfType • Other platform-related metadata		

Status codes:
Success:

- 200 - OK

Error:

- 401 - Unauthorized
- 403 - Forbidden
- 404 - Artifact not found

Example:

```
{
 "metadata" : {
    "key1" : "value1",
    "key2" : "value2",
    "groupId" : "uow",
    "serviceUrl" : "surl1",
    "serviceId" : "sid1",
    "projectId" : "p2",
    "version" : "1",
    "serviceConf" : "sconf",
    "serviceConfType" : "xml",
    "engineId" : "cloudflow"
  }
}
```

Add/Update artifact's custom metadata: PUT /artefacts/{artefact_id}/metadata

Add/Update artifact's metadata

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
artefact_id (required)	path	ID of the artefact to get

metadata	body	Metadata map, e.g.
		{
		"key1":”value1”,
		"key2":”value2”
		}

Response

Parameters:

Name	In	Description
Artifact metadata • groupid • engineid • projectid • version • serviceId • serviceUrl • serviceConf • serviceConfType • Other platform-related metadata	body	Artefact metadata

Status codes:

Success:

- 200 - OK

Error:

- 400 - Bad Request: Some content in the request was invalid
- 401 - Unauthorized
- 403 - Forbidden
- 404 - Artifact not found

Example:

```
{
  "metadata" : {
    "key1" : "value1",
    "key2" : "value2",
    "key3" : "value3",
    "groupId" : "uow",
    "serviceUrl" : "surl1",
    "serviceId" : "sid1",
    "projectId" : "p2",
    "version" : "1",
    "serviceConf" : "sconf",
    "serviceConfType" : "xml",
    "engineId" : "cloudflow"
```

```
  }
}
```

Delete artifacts metadata: DELETE /artefacts/{artefact_id}/metadata/{metadata_key}

Delete a custom metadata by key from an artifact.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
artefact_id (required)	path	ID of the artifact to get
metadata_key	path	Metadata key

Response

Parameters:

Name	In	Description
Artifact metadata • groupid • engineid • projectid • version • serviceId • serviceUrl • serviceConf • serviceConfType • Other platform-related metadata	body	Artifact metadata

Status codes:

Success:

- 200 - OK: Request was successful

Error:

- 401 - Unauthorized
- 403 - Forbidden
- 404 - Artifact / metadata key not found
- 422 - Artifact id & metadata key are required

Example:

```
{
  "metadata" : {
    "key1" : "value1",
    "key2" : "value2",
    "groupId" : "uow",
    "serviceUrl" : "surl1",
    "serviceId" : "sid1",
    "projectId" : "p2",
    "version" : "1",
    "serviceConf" : "sconf",
    "serviceConfType" : "xml",
    "engineId" : "cloudflow"
  }
}
```

Search

Search artefacts by query: GET /search/artefacts

Lists existing artifacts in a repository.

Request

Parameters:

Name	In	Description
Authorization	header	Basic authorization use user name:user password, where the password can be a CFGUM generated access token
continuationToken	query	A token returned by a prior request. If present, the next page of results are returned
q (required)	query	`Query by keywords. e.g 'engineId=cloudbroker AND groupId=uow'`

Response

Parameters:

Name	In	Description
items	body	A list of artefacts meeting searching criteria with basic information; empty array if there are none

		List of artefacts ● id ● downloadUrl ● path ● repository ● format ● checksum ○ sha1 ○ md5 ● cfg ○ groupid ○ engineid ○ projectid ○ version ○ serviceid ○ serviceUrl ○ serviceConf ○ serviceConfType ● Metadata ○ Key1:value1 ○ Key2:value2
continuationToken	body	A token returned by a prior request. If present, the next page of results are returned

Status codes:

Success:

- 200 - OK

Error:

- 401 - Unauthorized
- 403 - Forbidden
- 404 - Not found

Example:


```
{
   "items": [
        {
         "downloadUrl":
         "http://161.74.26.58/repository/cfg/uow/cloudbroker/p1/1.0.0/conf.
         json",

         "path": "uow/cloudbroker/p1/1.0.0/conf.json",

         "id": "Y2ZnOjkzYjliOWViOWE3ZWNiMDZiMzc5NjIzZGZlZjM0OTZj",

         "repository": "cfg",

         "format": "cfg",

         "checksum": {
```

```
      "sha1": "6eb04e36de53b7cd3a4422089216a9ad4a8335d0",
      "md5": "658cba07ea2107aafad31adc0cc8a455"
    },
    "metadata": {
      "additionalEngineMetadata": "value"
    },
    "cfg": {
      "groupId": "uow",
      "serviceUrl": "serviceurl",
      "serviceId": "serviceid",
      "projectId": "p1",
      "version": "1.0.0",
      "serviceConf": "serviceconf",
      "serviceConfType": "serviceconftype",
      "engineId": "cloudbroker"
    }
  },
  {
    "downloadUrl":
    "http://161.74.26.58/repository/cfg/uow/cloudflow/p2/1/conf.xml",
    "path": "uow/cloudflow/p2/1/conf.xml",
    "id": "Y2ZnOjkzYjliOWViOWE3ZWNiMDY0NTkxYzE3OTk4OTg1ODgz",
    "repository": "cfg",
    "format": "cfg",
    "checksum": {
      "sha1": "fc131546ea6ef553878e2c0cbcda6144760826f1",
      "md5": "dbadc276a8cd9349d6b3f3e923bca1de"
    },
    "metadata": {
```

```
            "key2": "value2",
            "key3": "value3"
         },
         "cfg": {
            "groupId": "uow",
            "serviceUrl": "surl1",
            "serviceId": "sid1",
            "projectId": "p2",
            "version": "1",
            "serviceConf": "sconf",
            "serviceConfType": "xml",
            "engineId": "cloudflow"
         }
      }
   ],
   "continuationToken": null
}
```