INTEL CORPORATION

COSBench Adaptor Development Guide

Version 2.2.0

Wang, Yaguang

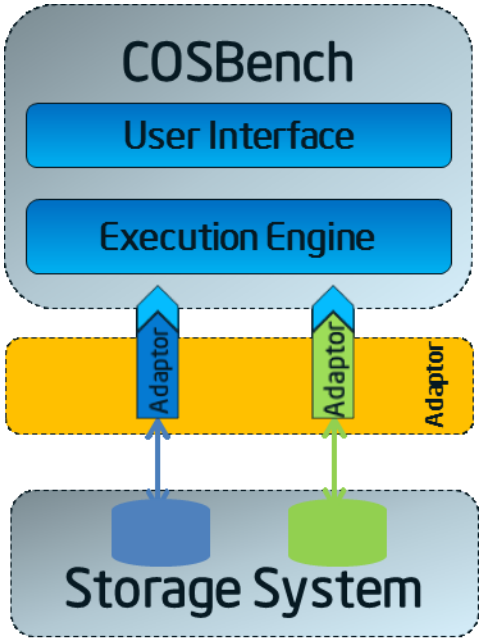
3/28/2013

This document describes how to extend COSBench to support new storage system by developing corresponding Adaptor. It depicts the model COSBench assumed, lists interfaces should be implemented in Adaptor, and demonstrates how to adopt new Adaptor into system.

Revision	Date	Author	Reviewer	Description
2.0	11/09, 2012			Initial version for v2.0
2.1	11/22, 2012			Add diagrams and samples
2.2	2/19, 2013			Add deployment section
2.3	3/18, 2013			Add section 5.1.10 for "abort" method in storage API.
2.4	3/28, 2013			. add call flow for authAPI and storageAPI interfaces. . add explanation for arguments in interface definition, and reword some sentences.

1. Introduction

COSBench is a modular, extensible system, and new storage system can be plugged into by feeding storage Adaptor which follows predefined APIs.



2. Glossary

API: the interface exposed from COSBench to enable extensibility, so far two kind of APIs are supported: Auth API and Storage API.

Bundle (or OSGi Bundle or plugin): it's one jar file compliant to OSGi specification.

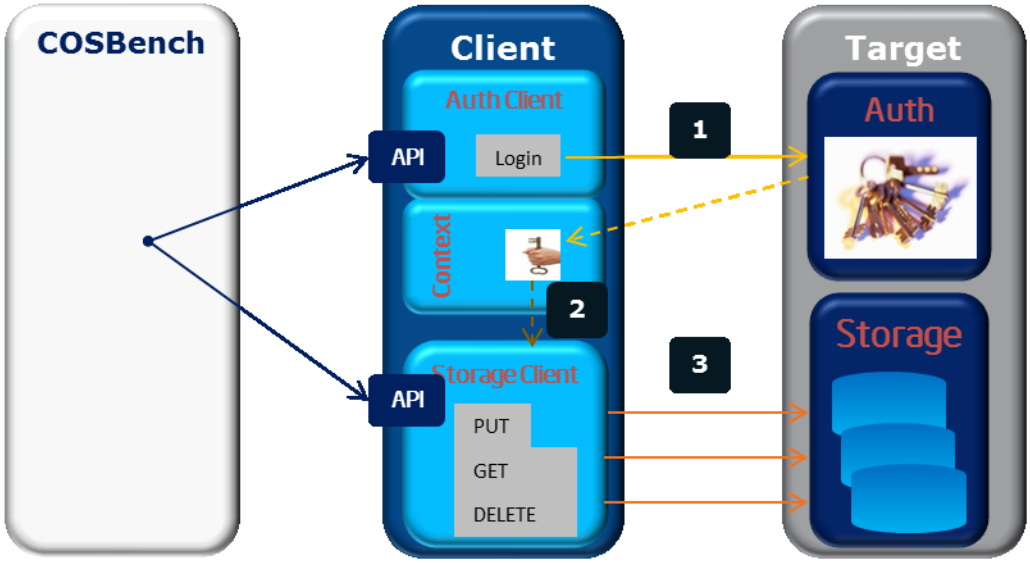
Adaptor: one bundle which implemented Auth API and/or Storage API.

3. General Concepts

3.1 Overview

COSBench defines below model for depict one storage system, there are three major components:

i) API
ii) Client
iii) Context.



3.2 APIs

COSBench defines two sets of API to adapt new storage systems:

i) Auth API, to handle authentication related work.
ii) Storage API, to handle data access related work.

The implementation of both API can be included in one OSGi bundle or separated into two OSGi bundles, one general guideline is if the authentication mechanism is for general purpose like Open Auth, or Kerbose, suggestion is to implement the Auth API in individual bundle.

3.3 Client

Client is a class which encapsulates all interactions with target storage or authentication system.

3.4 Context

Normally, it's necessary to exchange information between authentication client and storage client, typical information like authenticated token returned from authentication system, which is

required by storage client for following storage operations. That kind of shared information will be maintained in one context; so far the context is a set of key-value pair.

4. Auth API

4.1 Credential

Credential is used to authenticate one storage client, in COSBench, it is one parameter list including all authentication related information, e.g. username, password, authentication url. For different storage systems, the parameter list could be different.

4.2 Interfaces

```
public void init(Config config, Logger logger);
public void dispose();
public Context getParms();
public AuthContext login();
```

4.2.1 init()

```
    /**
     * Initializes an <code>Auth-API</code> with parameters contained in the
     * give <code>config</code>, whose content depends on the specific Auth
     * type. Normally, it will also initialize one client for authentication.
     *
     * @param config
     *          - one instance from com.intel.cosbench.config.Config, which
     *          includes parameters for authentication, and it will be passed
     *          from execution engine.
     * @param logger
     *          - one instance from com.intel.cosbench.log.Logger, which
     *          delivers logging capabilities to Auth-API, and it will be passed
     *          from execution engine.
     */
public void init(Config config, Logger logger);
```

4.2.2 dispose()

```
/**
 * Release the resources held by the Auth API.
 */
public void dispose();
```

4.2.3 getParms()

```
/**
 * retrieve parameters and current settings used by the AuthAPI.
```

```
 * @return Context - one com.intel.cosbench.context.Context instance which contains all parameters configured for the authentication mechanism.
 */
public Context getParms();
```

4.2.4 login()

```
/**
 * trigger backend authentication mechanism.
 * @return AuthContext - one com.intel.cosbench.context.AuthContext instance which contains all parameters configured for the authentication mechanism if authentication is succeed, otherwise, exception will raise.
 */
public AuthContext login();
```

5. Storage API

5.1 Interfaces

```
public void init(Config config, Logger logger);
public void dispose();
public Context getParms();
public void setAuthContext(AuthContext info);
public InputStream getObject(String container, String object, Config config);
public void createContainer(String container, Config config);
public void createObject(String container, String object, InputStream data, long length, Config config);
public void deleteContainer(String container, Config config);
public void deleteObject(String container, String object, Config config);
```

5.1.1 init()

```
    /**
     * Initializes a <code>Storage-API</code> with parameters contained in the
     * give <code>config</code>, whose content depends on the specific storage
     * type. Normally, it will also initialize one client for storage access.
     *
     * @param config
     *            - one instance from com.intel.cosbench.config.Config, which
     *            includes parameters for authentication, and it will be passed
     *            from execution engine.
     * @param logger
     *            - one instance from com.intel.cosbench.log.Logger, which
     *            delivers logging capabilities to Storage-API, and it will be passed
     *            from execution engine.
     */
public void init(Config config, Logger logger);
```

5.1.2 dispose()

```
/**
 * Release the resources held by the Storage API.
 */
public void dispose();
```

5.1.3 getParms()

```
/**
 * retrieve parameters and current settings used by the StorageAPI.
 * @return Context - one Context instance which contains all parameters configured for the storage.
 */
public Context getParms();
```

5.1.4 setAuthContext()

```
/**
 * associate authenticated context with Storage API for further storage operations.
 * @param info - one AuthContext instance, normally, it's the return from login() in Auth API.
*/
public void setAuthContext(AuthContext info);
```

5.1.5 getObject()

```
/**
 * download an object from a container.
 *
 * @param container - the name of a container.
 * @param object - the name of an object to be downloaded.
 * @param config - the configuration used for this operation.
 * @return inputStream - the inputStream of the object content. If null that means the object does't
 * exist or something bad happened.
 */
public InputStream getObject(String container, String object, Config config);
```

5.1.6 createContainer()

```
/**
 * create a container.
 *
 * @param container - the name of a container.
 * @param config - the configuration used for this operation.
 */
public void createContainer(String container, Config config);
```

5.1.7 createObject()

```
/**
 * upload an object into a container.
```

```
 *
 * @param container - the name of a container.
 * @param object - the name of an object to be uploaded.
 * @param data - the inputStream of the object content.
 * @param length - the length of object content.
 * @param config - the configuration used for this operation.
 */
public void createObject(String container, String object, InputStream data, long
length, Config config);
```

5.1.8 deleteContainer()

```
/**
 * delete a container.
 *
 * @param container - the name of a container to be deleted.
 * @param config - the configuration used for this operation.
 */
public void deleteContainer(String container, Config config);
```

5.1.9 deleteObject()

```
/**
 * delete an object.
 *
 * @param container - the name of a container.
 * @param object - the name of an object to be deleted.
 * @param config - the configuration used for this operation.
 */
public void deleteObject(String container, String object, Config config);
```

5.1.10 abort()

```
/**
  * Aborts the execution of an on-going storage operation (HTTP request) if
  * there is one.  The method expects to provide  one approach to abort outstanding
  * operation gracefully when the worker hits some termination criteria.
  */
public void abort();
```

6. Adaptor Development

6.1 Overview

One adaptor is actually one OSGi (http://www.osgi.org) bundle, a few IDEs support the development of OSGi bundles, the IDE used for COSBench is Eclipse SDK 3.7 Indigo (http://www.eclipse.org) .

To develop one new adaptor involves following steps:

- import a few dependencies,
- setup source code structure and,
- modify configurations.

6.2 Plugin dependencies

6.2.1 Third-party libraries (xxx is version name)

```
- other libraries needed based on abcStor specific requirements.
- com.springsource.org.apache.commons.codec-1.30.jar
- org.apache.httpcomponents.httpclient_4.1.3.jar
- org.apache.httpcomponents.httpcore_4.1.4.jar
- cosbench-api_xxx.jar
- cosbench-config_xxx.jar
- cosbench-http_xxx.jar
- cosbench-log_xxx.jar
```

6.2.2 COSBench bundles

As there are two APIs need implemented, two major bundling modes can work, other cases like leveraging existed auth or storage bundle don't include.

- Two in one
  a. abcStor+abcAuth bundle
- One plus one
  a. abcStor bundle (Storage API)
  b. abcAuth bundle (AuthAPI)

6.3 Source Code Structure

Normally, there are two folders: "*api*" and "client", and "*api*" wraps all required API declarations, and calls to functionalities included in "client", and "client" expects to include storage/auth specific implementation, and so far classes in this folder don't need any COSBench bundles, this is based on considerations for below two cases:

i) If there is tested code in adaptor developer site, developer can put the code into "client" folder, and add necessary wrapper in "api" to call into them.

ii) If it's a fresh new implementation, developer can focus on storage specific implementations in client and build tests based on all stuff in this folder without the involvement of COSBench code or osgi plugin related configurations, which would be easy development efforts.

Below is a reference source code structure with abcStor as example, full abcStor project is stored in the "*plugin*" folder in COSBench package.

```
src
    com.abc
            api
                    abcStorage.java
                    abcStorFactory.java
            client
                abcStorClient.java
                abcStorConstants.java
                abcStorClientException.java
```

6.4 Configurations

6.4.1 META-INF\MANIFEST

Below is one sample for MANIFEST file, adaptor developer needs modify to reflect their development requirements. Normally, adaptor developer needs review the "Import-Package" list.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Abc Storage Client Bundle
Bundle-SymbolicName: abc-stor
Bundle-Version: 2.2.0.GA
Bundle-Vendor: Abc Co.
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Import-Package: com.intel.cosbench.api.auth,
 com.intel.cosbench.api.context,
 com.intel.cosbench.api.storage,
 com.intel.cosbench.client.http,
 com.intel.cosbench.config,
 com.intel.cosbench.log,
 org.apache.commons.codec;version="[1.3.0,2.0.0)",
 org.apache.commons.codec.net;version="[1.3.0,2.0.0)",
 org.apache.http;version="[4.1.4,5.0.0)",
 org.apache.http.client;version="[4.1.3,5.0.0)",
 org.apache.http.client.methods;version="[4.1.3,5.0.0)",
 org.apache.http.conn;version="[4.1.3,5.0.0)",
 org.apache.http.entity;version="[4.1.4,5.0.0)",
 org.apache.http.message;version="[4.1.4,5.0.0)",
 org.apache.http.params;version="[4.1.4,5.0.0)",
 org.apache.http.util;version="[4.1.4,5.0.0)"
```

6.4.2 META-INF\spring\plugin-context.xml

Below is for "one plus one" bundling mode, so there will be two plugin projects:

abcAuth

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:osgi="http://www.springframework.org/schema/osgi"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/osgi
        http://www.springframework.org/schema/osgi/spring-osgi.xsd">

    <bean name="authFactory" class="com.abc.api.abcAuth.AbcAuthFactory" />

    <osgi:service ref="authFactory" context-class-loader="service-provider"
            interface="com.intel.cosbench.api.auth.AuthAPIFactory">
    </osgi:service>

</beans>
```

abcStor

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:osgi="http://www.springframework.org/schema/osgi"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/osgi
        http://www.springframework.org/schema/osgi/spring-osgi.xsd">

    <bean name="storageFactory" class="com.abc.api.abcStor.AbcStorageFactory" />

    <osgi:service ref="storageFactory" context-class-loader="service-provider"
            interface="com.intel.cosbench.api.storage.StorageAPIFactory">
    </osgi:service>

</beans>
```

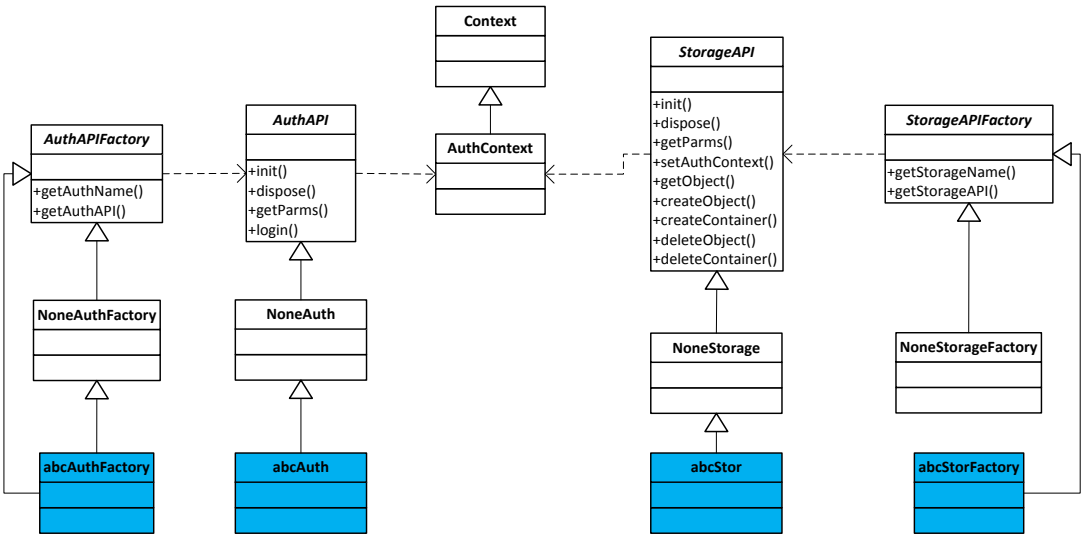
Below is for “two in one” bundling mode, and there is one plugin project, and two osgi services are defined in one plugin-context.xml:

abcStor + abcAuth

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:osgi="http://www.springframework.org/schema/osgi"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/osgi
        http://www.springframework.org/schema/osgi/spring-osgi.xsd">

    <bean name="authFactory" class="com.abc.api.abcAuth.AbcAuthFactory" />

    <osgi:service ref="authFactory" context-class-loader="service-provider"
            interface="com.intel.cosbench.api.auth.AuthAPIFactory">
    </osgi:service>

    <bean name="storageFactory" class="com.abc.api.abcStor.AbcStorageFactory" />

    <osgi:service ref="storageFactory" context-class-loader="service-provider"
            interface="com.intel.cosbench.api.storage.StorageAPIFactory">
    </osgi:service>

</beans>
```

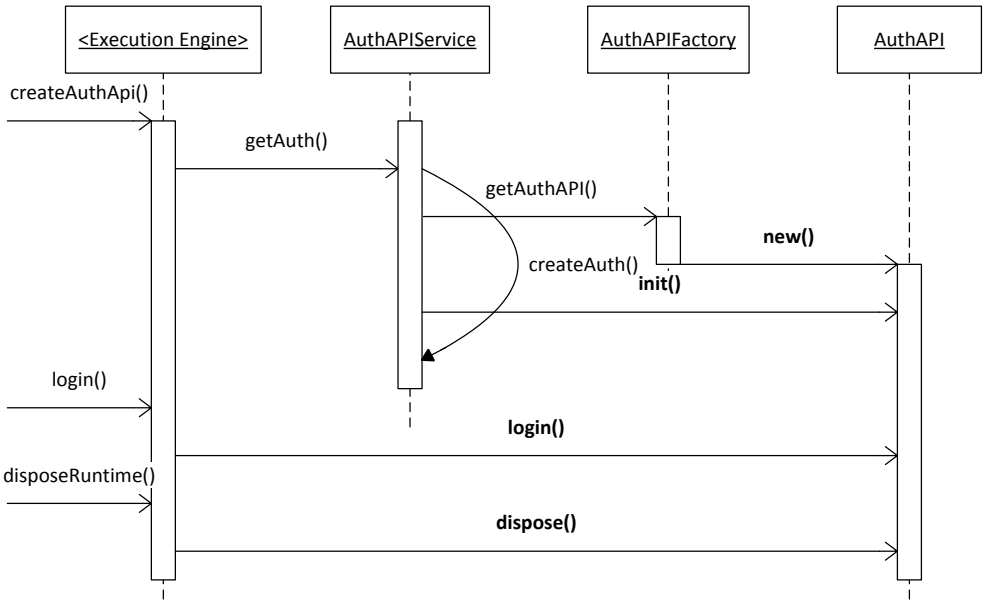
7. Class Diagram & Call Flow

Below is the class diagram which shows relationship between classes which is related to adaptor development.



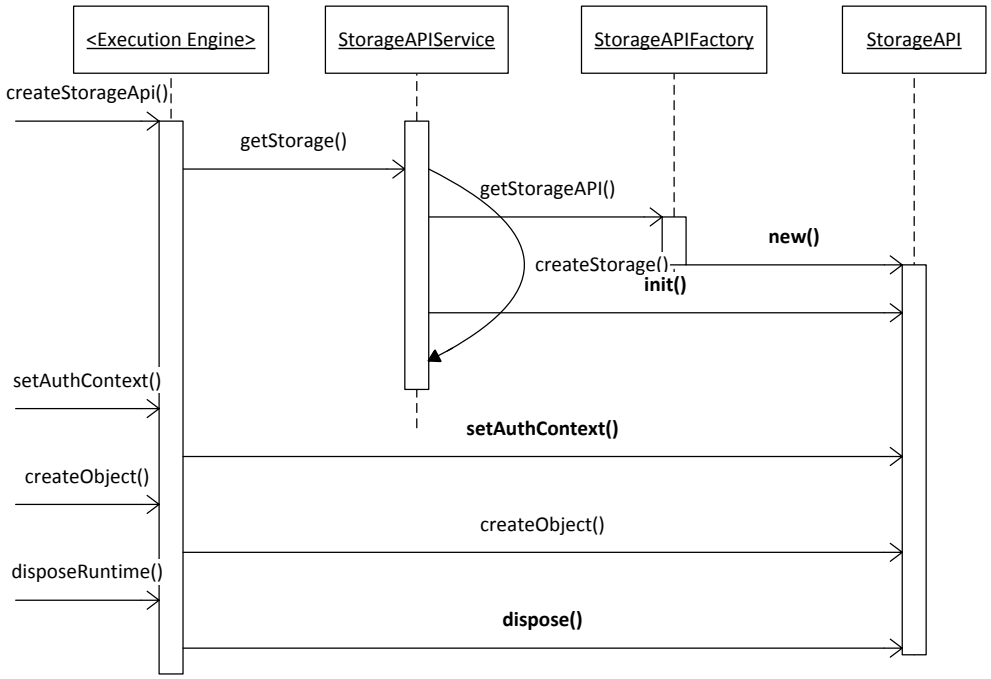
7.1 AuthAPI lifecycle

Below is the sequence for how authAPI (abcAuth) is initialized, performed and disposed, the procedure is driven by COSBench execution engine.



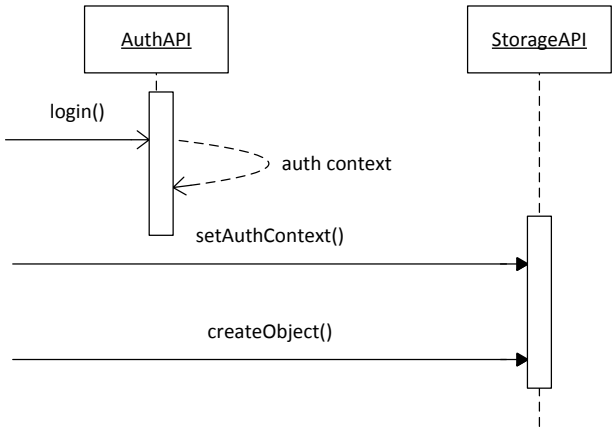
7.2 StorageAPI lifecycle

Below is the sequence for how storageAPI (abcStorage) is initialized, performed and disposed, the procedure is driven by COSBench execution engine.



7.3 AuthAPI & StorageAPI Interaction

Inside COSBench, the execution engine will call login() on AuthAPI (abcAuth) to authenticate the user, after successfully get valid auth context, it will call setAuthContext() to pass returned auth context to StorageAPI (abcStor). At this time, COSBench can issue object storage related operations through information from auth context. The flow is depicted as following:



8. Parameters

New auth or storage adaptor could require different parameters, and those parameters will be defined in "config" attribute in workload configuration file as key-value pairs with ";" as separator as following (With abcAuth as example):

```
<auth type="abcauth"
config="username=test;password=testing;auth_url=http://192.168.10.1:8080/auth/v1.0
;timeout=10000" />
```

Normally the parameters will be read in "init()" to help initialize client.

9. Sample Project

There are sample projects shipped with release package under "ext" folder, one auth sample and one storage sample are included.

```
libs
    com.springsource.org.apache.commons.codec-1.3.0.jar
    org.apache.httpcomponents.httpclient_4.1.3.jar
    org.apache.httpcomponents.httpcore_4.1.4.jar
modules
```

```
        cosbench-api_xxx.jar
        cosbench-config_xxx.jar
        cosbench-http_xxx.jar
        cosbench-log_xxx.jar
    adaptor
        abc-auth
            src
                com.abc
                    api
                        abcAuth.java
                        abcAuthFactory.java
                    client
                        abcAuthClient.java
                        abcAuthConstants.java
                        abcAuthClientException.java
            META-INF
        abc-stor
            src
                com.abc
                    api
                        abcStorage.java
                        abcStorFactory.java
                    client
                        abcStorClient.java
                        abcStorConstants.java
                        abcStorClientException.java
            META-INF
```

10. Deployment

To deploy adaptor bundles into COSBench, a few additional steps should take.

10.1 Register adaptor bundle

The osgi configuration file is located at conf/.driver/config.ini; new adaptor bundle needs register itself in it. Normally, COSBench related modules are at near the end of the file as following:

```
modules/cosbench-castor,\
modules/cosbench-log4j,\
modules/cosbench-log@5\:start,\
modules/cosbench-config@6\:start,\
modules/cosbench-http@6\:start,\
```

```
modules/cosbench-core@6\:start,\
modules/cosbench-api@6\:start,\
modules/cosbench-mock@6\:start,\
modules/cosbench-ampli@6\:start,\
modules/cosbench-swift@6\:start,\
modules/cosbench-keystone@6\:start,\
modules/cosbench-swauth@6\:start,\
modules/cosbench-abcstor@6:\start,\
modules/cosbench-driver@6\:start,\
modules/cosbench-tomcat@6\:start,\
modules/cosbench-core-web@6\:start,\
modules/cosbench-driver-web@6\:start
```

10.2 Update bundle list

"**start-driver.sh**" is the launching script for driver, it will check if bundle is loaded successfully. In order to ensure adaptor bundle is loaded successfully, modification in the script should be done by updating its bundle list as following:

```
OSGI_BUNDLES="cosbench-log_${VERSION} cosbench-tomcat_${VERSION} cosbench-config_${VERSION} cosbench-core_${VERSION} cosbench-core-web_${VERSION} cosbench-api_${VERSION} cosbench-http_${VERSION} cosbench-mock_${VERSION} cosbench-ampli_${VERSION} cosbench-swift_${VERSION} cosbench-abcstor_${VERSION} cosbench-keystone_${VERSION} cosbench-driver_${VERSION} cosbench-driver-web_${VERSION}"
```