OpenECOMP SDC

Developer Guide

Product:	OpenECOMP SDC
Title:	Developer Guide
Product version:	v1.0
Publication date:	01/30/2017
Reference number:	OECSDCDEV170130
Revision number	1

Contents

1 Overview of SDC **4**

Technologies used in SDC 4

High-level architecture 4

Data model overview 5

2 SDC projects **6**

asdc-tests project 6

asdctool project 7

catalog-be project 7

catalog-dao project 9

catalog-fe project 11

catalog-model project 12

common-app-api project 13

1 Overview of SDC

The SDC (service, design, and creation) application provides service designers with a GUI interface for creating, testing, and distributing services.

SDC serves as a warehouse of building blocks provided by different vendors. From these building blocks, designers can create any required service. Once those services are created, they can be used by different applications to instantiate these services.

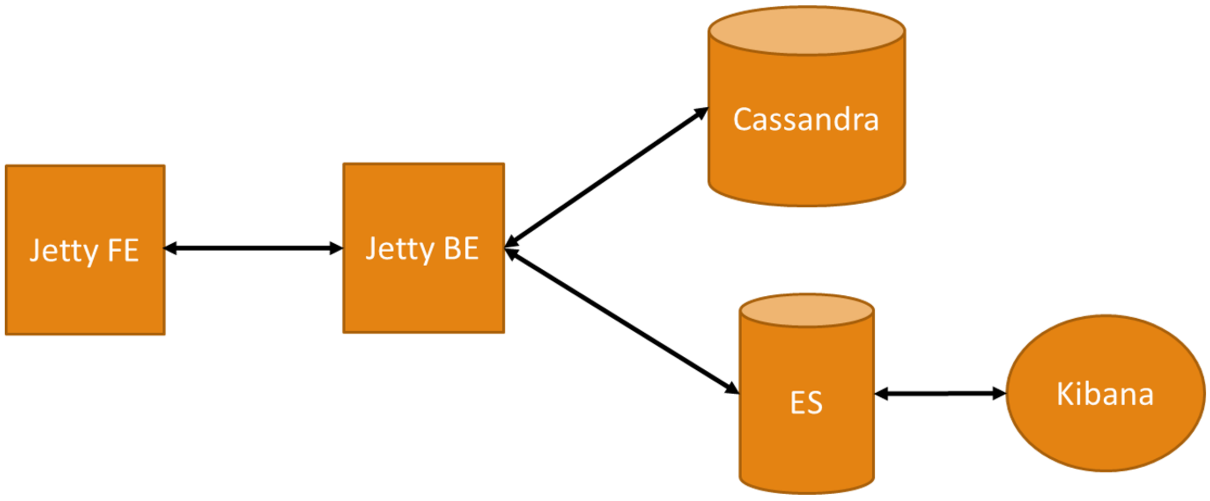
Technologies used in SDC

SDC utilizes the following technologies:

- Back-end technology
  - Java 8
  - JUnit
  - Cassandra
  - Elastic Search
  - Titan Graph DB
  - Kibana
  - Jetty
  - Spring
  - Maven
- Front-end technology
  - TypeScript
  - AngularJS
  - Cytoscape.js
  - NPM
  - Grunt
  - Bower

High-level architecture

The following diagram shows the high-level relationship between the system components.



The SDC architecture uses the Jetty server as an application server.

- The **Jetty front end**:
  - supplies the static content of web pages, and all resources that required by the GUI.
  - serves as a proxy for the REST API requests coming from the GUI.

  Every request originating from the GUI is passed to the Jetty front-end server before it is executed.

- The **Jetty back end** contains all the logic for the SDC.

SDC uses two storage components: Elastic Search (ES) and Cassandra.

- **Elastic Search** is used to index the auditing data received from different operations in the SDC.

  This information can then be analyzed with Kibana. The **Kibana server** enables statistical analysis of the operations done, according to the business logic.

- **Cassandra** is used to store auditing data, artifacts and data model objects.

Data model overview

SDC uses a graph-based data model. The implementation is based on Titan's Graph Database.

Each object in the system is a graph vertex. From the vertex, we use edges to connect to edition objects where the edges represent the relationship between two objects. The full business logic object is represented as a subgraph.

Titan uses Cassandra as storage.

2 SDC projects

The following topics describe the projects included in the SDC application:

- "asdc-tests project" below
- "asdctool project" on the next page
- "catalog-be project" on the next page
- "catalog-dao project" on page 9
- "catalog-fe project" on page 11
- "catalog-model project" on page 12
- "common-app-api project" on page 13

asdc-tests project

This project holds our testing infrastructures. It is a collection of all the integration tests available in a project. Project packages separate the tests according to their subject.

The project uses TestNG as a testing framework. The collection of tests in this project are dedicated for testing the back-end application. You must have a working SDC environment to use these tests.

The project compiles to a JAR file that can be executed to run the tests, according to a predefined configuration.

The following table shows the project structure, and describes the included packages.

src			
	main		
		java (contains all integration tests for SDC)	
			• *org.openecomp.sdc.ci.tests* - packages are separated according to the subject of the tests. • *org.openecomp.sdc.externalApis* • *org.openecomp.sdc.post*
		resources	
			• *conf* – contains all the test configurations. • *scripts* - the start test script. • *testSuits* – the TestNG test suite xml
	test		
		resources	

asdctool project

This project:

- Comprises a collection of stand-alone utilities used in the application. Utilities included in the project are export/import utils, and data migration utilities.
- Utilizes the application's business logic as the basis for all operations it provides.
- Compiles into a JAR file that can be executed with parameters to access the different functionalities that it provides.

The following table shows the project structure, and describes the included packages.

src

main

java

- *org.openecomp.sdc.asdctool.impl* – contains the logic for the different operations that the ASDC tool provides.
- *org.openecomp.sdc.asdctool.main* – the main classes which enable users to call different functionality in the ASDC tool.
- *org.openecomp.sdc.asdctool.servlets*

resources

- *config* – a configuration example for the asdctool.
- *scripts* – sh script wrappers that allow executing the main class.
- *es*-resources – Python scripts for the migration of Elastic Search data

catalog-be project

This project contains all the business logic of the application. It has other projects as dependencies, where each project provides a different functionality. The project contains the different servlets that expose the SDC's REST API's.

The project compiles into a WAR file.

The following table shows the project structure, and describes the included packages.

src

main

java

- *org.openecomp.sdc.be.auditing* – Contains the logic for auditing executed operations.
- *org.openecomp.sdc.be.components.clean* – Contains the task manager, which removes deleted components from the system

after they are no longer needed.

- *org.openecomp.sdc.be.components.distribution.engine* – Contains the logic of the distribution mechanism for sending notifications that services are available for distribution.
- *org.openecomp.sdc.be.components.impl* – Contains the logic for handling operations on the different SDC components, such as services, resources ,products, artifacts, and so on.
- *org.openecomp.sdc.be.components.lifecycle* – Manages the transition between different component lifecycle states available in SDC, such as: check-in, checkout, and certified.
- *org.openecomp.sdc.be.datamodel* – Contains utilities for working with data model objects.
- *org.openecomp.sdc.be.distribution* – Includes the servlets and business logic which handle requests from the SDC distribution client for distributing services and their artifacts.
- *org.openecomp.sdc.be.ecomp* – Contains the logic for managing SDC users using the OpenECOMP Portal.
- *org.openecomp.sdc.be.externalapi.servlet* – Contains servlets that expose SDC's APIs externally. These APIs are used by different applications to access and query the SDC catalog.
- *org.openecomp.sdc.be.filters* – Contains the following SDC filters:
    - an authentication filter used to validate the credentials of incoming requests to the external APIs.
    - a filter for logging incoming requests.
    - a filter to deny requests in case the application is unavailable.
- *org.openecomp.sdc.be.impl* – set of general utilities.
- *org.openecomp.sdc.be.info* – Includes the POJO (Plain Old Java Object) object used in the SDC for passing parameters.
- *org.openecomp.sdc.be.listen* – configuration listener used to pick up the configuration files.
- *org.openecomp.sdc.be.monitoring* – Contains a proxy servlet which passes requests to the Elastic Search.
- *org.openecomp.sdc.be.servlets* – Contains the internal API servlets which expose SDC functionality to the GUI.
- *org.openecomp.sdc.be.switchover.detector* – Contains the switchover mechanism which is activates if an SDC site becomes unavailable.
- *org.openecomp.sdc.be.tosca* – Contains the logic and model objects which generate the CSAR (Cloud Service ARchive file).

		• *org.openecomp.sdc.be.user* – Contains the logic for user management in SDC. • *org.openecomp.sdc.common.transaction* – an infrastructure for transaction management across the different data storages, currently not in use.
	resources	
		• *config* – the configuration file for the SDC back-end application • *import.tosca* – the normative types which need to be preloaded into the system. • *Scripts*– a set of Python scripts dedicated to the import of the base entities needed for the system. • *Swagger* – the Swagger UI • *webapps* – the WAR configuration and the Swagger UI static content.
test		
	Java	The different Junit tests separated by packages according to the logic they are testing.
	resources	Different file examples used in Junit and configurations used for testing.

catalog-dao project

The catalog-dao (DAO stands for Data Access Object) project contains all the functionality needed to manage the different persistence layers used in the SDC. SDC uses three types of persistent layers:

1. Titan Graph DB: the Titan Graph DB allows us to store data as a graph. Our data model entities are represented as a collection of vertices and edges. The vertices represent data and the edges represent the connection between the data. The Titan DB itself is a client application and uses the Cassandra DB for the actual persistency.
2. Cassandra: Cassandra is used for storing auditing events registered during the system operation, artifact files as binary data and we use Cassandra as a cache for our data model , we de-serialize our objects and store them as binary date in the Cassandra for fast retrieval.
3. Elastic Search: we use Elastic Search as a secondary storage site for our auditing events. By storing them in here we are able to use Kibana to analyze the operations that are performed on the system.

Note: catalog-dao contains code parts used for Neo4j persistence; Neo4j is a graph DB which in the future may replace Titan. In the current version Neo4j is not deployed and the code is not used.

The project compiles into a JAR file which is used in the creation of the SDC Backend WAR file.

The following table shows the project structure, and describes the included packages.

src

main

java

- org.openecomp.sdc.be.dao.api –contains the Elastic Search base for the different operations and base DAO for the No4j .
- org.openecomp.sdc.be.dao.cassandra –the package contains the Cassandra client used by the DAO and the DAO implementation for the different operations with the Cassandra.
- org.openecomp.sdc.be.dao.cassandra.schema –the package contains the logic and the date for the creation of the Cassandra keyspaces, tables and indexes used by SDC.
- org.openecomp.sdc.be.dao.es – the package contains the elastic search client used by the Elastic Search DAO.
- org.openecomp.sdc.be.dao.graph – the package continues enemas and objects representing graph entities.
- org.openecomp.sdc.be.dao.impl –the package contains the extended Elastic Search DAO and the Neo4j DAO.
- org.openecomp.sdc.be.dao.model –the package contains objects representing different Elastic Search results.
- org.openecomp.sdc.be.dao.neo4j –the package includes the Neo4j client and its modeling objects. The package also includes the enum describing the different properties on the graph, their type and index. On application startup the graph indexes are defined according to this enum.
- org.openecomp.sdc.be.dao.rest –the package contains the http client and its configuration object. The HTTP client is used by the Neo4j client for is communication with the server.
- org.openecomp.sdc.be.dao.titan –the package contains the Titan client and the Titan DAO used as a base for all operations on the graph.
- org.openecomp.sdc.be.dao.utils – the package contains the utilities and constants used by the different DAO’s.
- org.openecomp.sdc.be.resources.api – the package contents are deprecated and will be removed in future releases.
- org.openecomp.sdc.be.resources.data –the package includes all the data objects used for storing data in the different persistency layers. Using the ORM (Object-Relational Mapping) style we map the objects to the data store. The DAO layer uses this object throughout.

test

java The different Junit tests separated by packages according to the logic they are

	testing
resources	Different files used in Junit and configurations used for testing

catalog-fe project

This project contains the Frontend (FE) server logic. The Frontend is used as a proxy between the UI and the Backend server. A REST request sent from the UI is directed to the Frontend server and from there to the Backend.

The project compiles into a WAR file. During a full build process of SDC the UI minified source will be placed in the WAR file created for the FE.

The following table shows the project structure, and describes the included packages.

src			
	main		
		java	• org.openecomp.sdc.fe.client –the package contains a rest client that is deprecated and will be removed in future releases. • org.openecomp.sdc.fe.impl –the package contains model object used by the rest client, the code is deprecated and will be removed in future releases. • org.openecomp.sdc.fe.listen –the package contains the configuration listening responsible for reading the configuration on the startup of the server. • org.openecomp.sdc.fe.servlets – the package contains the servlets responsible for the proxy functionality, health check and access to index.html etc. The package also contains the logic responsible for managing the application health check.
		resources	
			config - the folder contains all the configuration needed for the Frontend server execution.
			scripts – the content of the folder is not in use and will be removed in next release.
			webapps – the folder contains the servlet mapping and the context definition for the deployed WAR file. On a full build the UI content will be placed here in order to allow the web server to supply it.
		jasmine-standalone-2.0.0	The folder contains the framework used by the Frontend EAM (Enterprise Asset Management) to develop unit testing.
		java	The different Junit tests separated by packages according to the logic they are testing

resources	Different files used in Junit and configurations used for testing
spec	The folder content is not in use and will be removed in future releases.
testScripts	The folder content is not in use and will be removed in future releases

catalog-model project

This project is the connection layer between the business logic and the Dao.

The project contains the data model objects and the operations used to interact with them.

An operation is a set of logic that manipulates the persistent data according to the changes made to the model.

The project compiles into a JAR file which is used in the creation of the SDC Backend WAR file.

The following table shows the project structure, and describes the included packages.

src

main

java

- org.openecomp.sdc.be.model - the package contains all the model objects used through the SDC.
- org.openecomp.sdc.be.model.cache – the package contains the cache implementation,cache model objects and the async workers responsible for handling requests to the cache. We use a cache to store the data model objects in their de-serialized form for fast retrieval.
- org.openecomp.sdc.be.model.cache.jobs – the package contains the different jobs that can be given to the workers working with the cache. Each job represents a different operation (update, delete add etc).
- org.openecomp.sdc.be.model.cache.workers – the package contains the workers implementation. The workers are async task runners.
- org.openecomp.sdc.be.model.category – the package contains the model objects representing the different categories used to describe resources/services/products.
- org.openecomp.sdc.be.model.heat – the package contains the enum mapping the different heat types and there validators and converters.
- org.openecomp.sdc.be.model.operations.api – the package

contains all the interfaces for the operations.

- org.openecomp.sdc.be.model.operations.impl – the package contains the implementation of all the different operations in sdc. The operation class is to aggregate all the operations needed to be performed on the specific data model.
- org.openecomp.sdc.be.model.operations.utils – the package contains utilities used by the different operations.
- org.openecomp.sdc.be.model.tosca.constraints – the package contains the constraints objects and exceptions used on constraints validation errors. Which are used to set constraints on properties.
- org.openecomp.sdc.be.model.tosca.converters – the package contains converters for converting property values into objects.
- org.openecomp.sdc.be.model.tosca.validators
- org.openecomp.sdc.be.model.tosca.version
- org.openecomp.sdc.be.unittests.utils

test

java	The different Junit tests separated by packages according to the logic they are testing.
resources	Different files used in Junit and configurations used for testing

common-app-api project

This project contains a set of general utilities and shared logic that is needed in the Frontend and Backend servers.

The project compiles into a JAR file that is referenced by both servers during their WAR file creation.

src

main

java

- org.openecomp.sdc.be.config – the package contains the logic for reading and managing the different configuration files used in the Backend server. The configuration is mapped in to objects on startup. The objects are accessible anywhere in the application.
- org.openecomp.sdc.be.monitoring – the package holds the Backend service responsible for collecting matrices regarding the system and sending them to the monitoring end point for logging into Elastic Search.
- org.openecomp.sdc.common.api – the package contains a number of

general purpose enum’s and POJO objects used in the SDC.

- org.openecomp.sdc.common.config – the package contains enum’s and logic used for logging OpenECOMP notifications which occur in the SDC.
- org.openecomp.sdc.common.datastructure –the package contains general purpose data structures.
- org.openecomp.sdc.common.impl – the package contains the logic used for reading the configuration stored as YAML files, in addition the package contains the implementation of the configuration change listener that in the future will allow for making changes to the SDC configuration at run time.
- org.openecomp.sdc.common.kpi.api –the package contains the KPI implementation. For the current release the KPI is not collected but the API is already called at the needed points.
- org.openecomp.sdc.be.model.operations.impl – the package contains the implementation of all the different operations in SDC. The operation class is used to aggregate all the operations needed to be performed on the specific data model.
- org.openecomp.sdc.common.listener –the package contains the context listener used for communication between the business logic and the servlets.
- org.openecomp.sdc.common.monitoring – the package contains logic for retrieving the monitoring data and storing the data in a POJO sent to the monitoring end point.
- org.openecomp.sdc.common.rest – the package contains an HTTP client and a wrapper layer for easier use of the client.
- org.openecomp.sdc.common.servlets – the package contains a base implementation of the servlet used in the Frontend and Backend server.
- org.openecomp.sdc.common.util – the package contains utilities used in the comman-app-api package and in the other projects.
- org.openecomp.sdc.exception- the package contains common exceptions used in both SDC servers.
- org.openecomp.sdc.fe.config – the package contains the logic for reading and managing the different configuration files used in the Frontend server. The configuration is mapped in to objects on startup. The objects are accessible anywhere in the application.
- org.openecomp.sdc.fe.monitoring – the package holds the Frontend service responsible for collecting matrices regarding the system and sending them to the monitoring end point for logging into Elastic Search.

test		
	java	The different Junit tests separated by packages according to the logic they are testing.
	resources	Different files used in Junit and configurations used for testing