Overview of CDS source code

`ccsdk-apps/components`: common place for code with high reusability potential among mServices. All the blueprint data information.

`../components/core`: works with pure Tosca based definitions. Holds all the blueprint data information. Probably will move to controllerblueprints/modules in the near future.

`../core/../interfaces`: package that have the interfaces for all the major components of the blueprints processor: CatalogService, Enhancer, FunctionNode, RepoService, ScriptsService, Validator

`../core/../BluePrintValidator`: defines and provides all validation services, storing them in a Map by name. Also provides the doValidation method, that call the validate from all selected validators.

`../core/../BluePrintService`: defines the interface of a BluePrintService and provides the default implementation.

`../core/../BluePrintContext`: defines the context for the blueprint. It receives a ServiceTemplate as parameter and gets data from it

`../core/../BluePrintModel`: defines the blueprint model, based on the Tosca standard, currently version 1.2. Changes in the Tosca standards will cause updates in this file. Every bean in this file maps a Tosca definition.

`../core/../BluePrintTypes`: defines all the types of blueprint, alongside with their domain values

`../core/../CustomFunctions`: adds new functions to existing classes, even the ones from JDK. For example, String class gets some new functions, like *String.asJsonPrimitive()*

`../core/../BlueprintFunctionNode`: defines the contract that all Blueprint Functions that act upon a node must conform to. It is the base interface for all the capability components. It has:

- *prepareRequest* and *prepareResponse*, to help with the setting up of the request and response, respectively
- *process*, that executes the business logic
- *recover*, that gets any error that occurs, registers it and perform cleaning up

`../components/resource-dict`: resolves resources definitions during runtime. Meant for complete handling of resources dictionary. Expression resource definition and resource dictionary can be used interchangebly. It also provides information about the data representation, like type, range of acceptable values, size, etc.

`../components/definition-type`: types that will be loaded in the UI of CDS, to be reused

- *starter-type*: default reusable types that will be present in any instance created by a user.

- *starter-type/artifact-types*: defines the extensions of the files that are of interest. For example, it defines a velocity template (*artifact-template-velocity.json*) as a file that has the extension vtl. All artifact types derive from tosca.artifacts.Implementation, defined in this directory.

- *starter-type/data-types*: defines the generic reusable data-types.

- *starter-type/node-types*: defines data-types that will be used with nodes. This section defines what appears in the blueprint. For example, for Resource Resolution, there is the definition of the schema for component-resource-assignment. See components/model-catalog/definition-type/starter-type/node_type/component-resource-resolution.json and how it is used in line 108 of components/model-catalog/blueprint-model/test-blueprint/baseconfiguration/Definitions/activation-blueprint.json

`ccsdk-apps/model-catalog`: stores definitions and reusable blueprint models.

`ccsdk-apps/scripts`: reusable platform specific scripts, provide glue between the our framework and your jython scripts. It will be exported during the docker image creation and should be put in the path.

`ccsdk-apps/ms`: home of the microservices of blueprint processor

`../ms/controllerblueprints`: mService that is the backend of the CDS. The controller blueprint is the design time framework allowing the service designer to express what data needs to be resolved and the how to resolve this data. It uses a fully model-driven approach (declarative vs imperative) pushing for re-usability of artifacts. Resolved resources (along with user provided and defaults) gets stored in the MDSAL data-store of SDNC, under the GENERIC-RESOURCE-API sub-tree. (source).

`../application`: security and configuration related information

`../distribuition`: web packaging and dependency management

`../ms/blueprintsprocessor`: mService that provides the runtime processing of blueprints

Structure of a blueprint:
* metadata
* imports

* the words function and component can be used interchangeably in the CDS context

* all capability components must have a recovery plan and solution processing plan

* to create a new blueprint, duplicate starter-blueprint

* every blueprint has a TOSCA-metadata folder, with TOSCA.meta inside. Entry-definitions field on this file determines which template to call first, the actual controller blueprint.