CCSDK-2870 DP: CPS REST API Documentation - PROPOSAL

Jira Ticket:

~~CPS-50~~ - DP: Agree and document proposed long-term REST API(s) CLOSED

Jira Backlog:

https://jira.onap.org/secure/RapidBoard.jspa?rapidView=223&view=planning.nodetail&selectedIssue=CCSDK-2912&issueLimit=100

Resources

- https://restfulapi.net/resource-naming/
- https://api.gov.au/standards/national_api_standards/naming-conventions.html
- https://opensource.zalando.com/restful-api-guidelines/

Assumptions

#	Description
1	Yang namespaces are globally unique (except for multiple revisions)

Open Issues/Decisions

#	Description	Details	Decisions
~~1~~	How to differentiate GET queries with different parameters. If we don't have named parameters how best to know what is meant	Alternatives 1. GET /anchorpoint/dataspace/{dataspace} versus /anchor-point/name/{dataspace} 2. GET /anchorpoint?dataspace={dataspace} versus /anchor-point?name={dataspace} 3. GET /dataspace/{dataspace-id}/anchorpoint versus /anchor-point/{name} 4. GET /dataspace/{dataspace-id}/anchorpoint versus /anchor-point/name/{name} 5. GET /dataspace/{dataspace-id}/anchorpoint versus /name/{name}/anchor-point 6. etc.	Team meeting Flavio,Tony,Toine,Bruno,Niamh, Rishi 8 Oct 2020: Use hierarchical resource urls including the concept of 'dataspaces' and 'anchors'. 'Fragments' will be replaced by 'Nodes' e.g. GET /dataspaces/{dataspace_id}/anchors returns Anchor-points GET /dataspaces/{dataspace_id}/anchors/{anchor-id}/nodes?xpath="..." returns Node(s) GET /dataspaces/{dataspace_id}/anchors/{anchor-id}/nodes?schema_node_identifier="..." returns Node(s) GET /dataspaces/{dataspace_id}/nodes&xpath="..." returns Node(s) GET /dataspaces/{dataspace_id}/modules returns Modules
~~2~~	Advance queries v Simple Queries (gets) for data fragments. Do we want/need differentiate on UTL?	1. GET /fragment/query/{xpath} 2. GET/fragment/{xpath} 3. GET /query/{xpath} Note . The xpath could be a full unambiguous xpath returning one object or it could be partial path using wildcards etc allowing for more advanced queries return 0 or more fragments (developed over many iterations)	Team meeting Flavio,Tony,Toine,Bruno,Niamh, Rishi 8 Oct 2020: Advanced queries will implemented as advanced xpath' (and or schema node identifiers) no need for separate URLs
~~3~~	We have no modules sets!	model files just arbitrarily group a few modules, is there a need to persist this grouping (We do have the concept of a dataspace as well)	Team meeting Flavio,Tony,Toine,Bruno,Niamh, Rishi 8 Oct 2020: The way modules are delivered i.e. grouped in files is not relevant for the CPS application. What needs to be stored for each 'anchor' is the namespace and revision of the model that describes the 'root' of that instance

~~4~~	Treat operations with xpath and schema_node_identifier as 'queries' using POST?	xpath and schema node identifiers aren't suitable for URLs. The suggestion is to create query URLs instead and pass this data in as part of the request body	Team meeting Niamh Core, Toine Siebelink and Rishi Chail10 Oct 2020 Agreed format, GET nodes resource with query details request body, applied for #11-#13 in the table below

API definitions

Note 1. This list of URL operations is not meant to be complete or final. It is mainly a starting point to establish a pattern and naming convention for the Configuration Persistence Service.

Note 2. All urls in below table will be prefixed with something like : <server>/api/cps/v1/

Note 3. This does not yet cover authorization.

Topic Area	#	Operation	Payload	Description
Modelling storage (Create/Read modules) Dataspace is a group (CPS term)	1	POST /dataspaces/{dataspace-id}/modules	File	Create (and validate) a module set (upload a model file) for the given dataspace. Payload is a file containing 1 or more yang modules. This operation will also create a dataspace.
	2	GET /dataspaces/{dataspace-id}/modules		Read all modules in the store for the given dataspace
	3	GET /dataspaces/{dataspace-id}/modules?namespace="namespace-id"		Read all modules in the store for the given dataspace and namespace
	4	GET /dataspaces/{dataspace-id}/modules?namespace="namespace-id"&revision="revision"		Read all modules in the store for the given dataspace, namespace and revision
	5	DELETE /dataspaces/{dataspace-id}/		Delete the given dataspace
Anchor persistence (Internal to CPS, associate data relates to a model set to an anchor. Associate data w/ models that give rules when someone queries it).	6	POST /dataspaces/{dataspace-id}/anchors	Json Object	Create a new anchor in the given dataspace (payload includes anchor name, module namespace and revision)
	7	GET /dataspaces/{dataspace-id}/anchors/{anchor-id}		Read an anchor and the associated attributes given a anchor ID and a dataspace.
	8	DELETE /dataspaces/{dataspace-id}/anchors/{anchor-id}		Delete an anchor given a anchor ID and a dataspace. This will delete the whole tree
	9	GET /dataspaces/{dataspace-id}/anchors		Read all anchors in the given a dataspace.
Node persistence (Create/Read of Node Instance Data) Yang Node.	10	POST /dataspaces/{dataspace-id}/nodes	File	Create a (root) node for a given anchor for the given dataspace, the node can have children. Their children will also be persisted as separate nodes in the system.
	11	GET /dataspaces/{dataspace-id}/anchors/{anchor-id}/nodes	Request body: { xpath: { xpath specification }}	Get a node given an anchor for the given dataspace (return just one level with just xpath references to its children)
	12	GET /dataspaces/{dataspace-id}/nodes	Request body: { xpath: { xpath specification }}	Get a node (under any anchor) given a Xpath expression for the given dataspace
	13	GET /dataspaces/{dataspace-id}/nodes	Request body: { schema_node_identifier: { schema_node_identifierspecification }}	Get all the relevant nodes given a schema node identifier for the given dataspace (not need to specify dataspace is schema-node-identifier is globally unique)