ApplicationPattern

Implementation Documentation

Version	Date	Change details
1.0	Jan 31,2022	

Author	Prathiba Jeevan	prathiba.jeevan.external@telefonica.com
Reviewer	Thorsten Heinze	thorsten.heinze@telefonica.com

INDEX

1. Purpose

The purpose of this document is to outline the technical design of the ApplicationPattern generic modules in detail.
Its main purpose is to,

- Detail the functionality provided by each class/file in the modules
- Detail the functionality which will be provided by each component and shows how various components interacts in the design

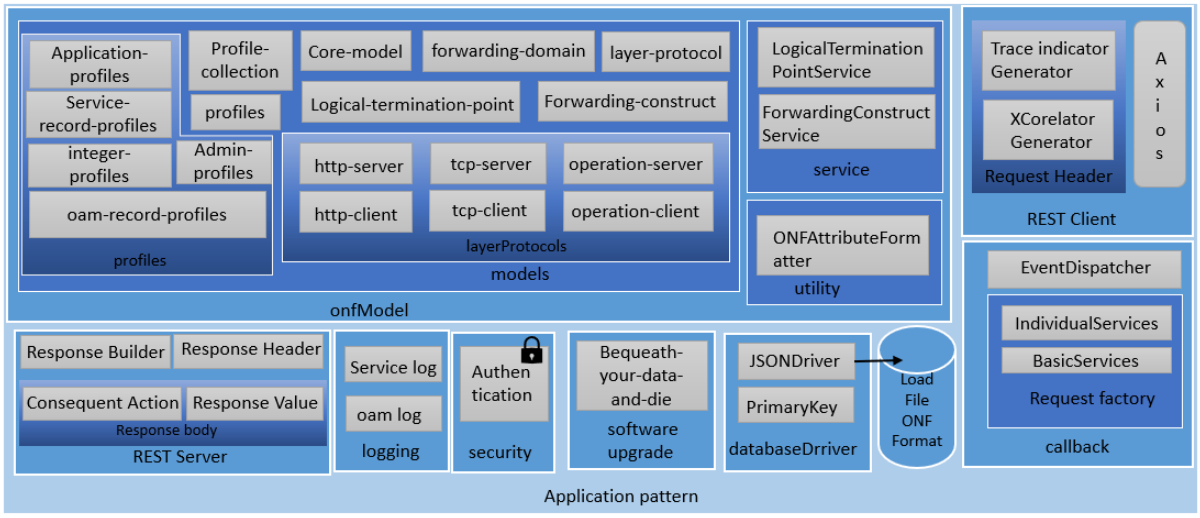
This document will be updated based on the changing requirement in the ApplicationPattern specification.

2. Scope

The ApplicationPattern design charted in this document is based on the scope defined in the requirement (OAS). This document is not intended to address the installation and configuration of the deployment.

3. Introduction

ApplicationPattern module is a ready-made framework that provides components and solutions that has generic functionalities specific to the SDN ApplicationPattern microservice specifications.



This framework provides APIs, …

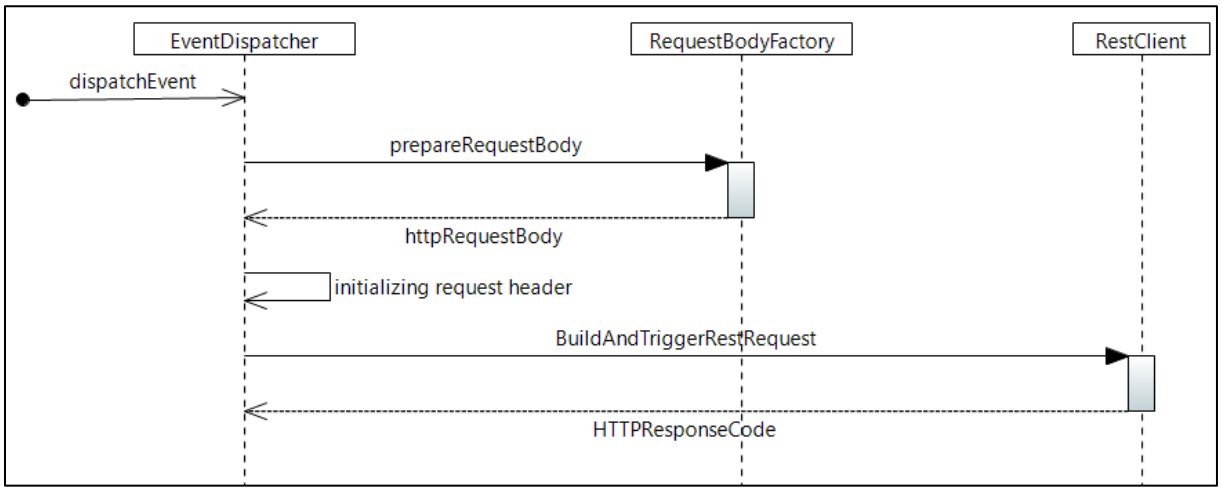
- to manipulate the LOADfile, which is in the ONF Core model.
- to configure and automate the forwardings between applications (please refer https://github.com/openBackhaul/ApplicationPattern/blob/tsi/doc/SpecifyingApplications/ForwardingList/ForwardingList.md to learn more about forwarding)
- to log the service and the OaM request to ExecutionAndTraceLog and OamLog application respectively.
- to validate authentication by interacting with the AdministratorAdministration application.

By including this already existing framework, one can focus on business logics rather than developing their own logic to embedding their application into the microservice architecture.

4. ApplicationPattern Modules

4.1 Callbacks

Callback module provides functionality to formulate and dispatch the HTTP request to a target REST server.



4.1.1 requestBodyFactory

4.1.1.1 basicservices.js

Provides functionality to formulate HTTP request body for the forwarding HTTP requests that are part of the basic services. This module is generic across all the application that are having core-model-1-4:control-construct as the root entity of their LOADfile.

Functions:

Method and description	Input parameters	Return type
prepareRequestBody This function formulates and returns the request body based on the operationName, clientApplicationName and the attribute list passed from the service layer.	{String} **clientApplicationName** name of the client application. {String} **operationName** name of the client operation that needs to be addressed. {String} **attributeList** list of attributes that needs to be included in the request body based on the operation name.	{Promise} **httpRequestBody** formulated JSON HTTP request body.

4.1.1.2 individualServices.js

Provides functionality to formulate HTTP request body for the forwarding HTTP requests that are part of the individual services. This file should be modified according to the individual service forwarding requirements.eventDispatcher.js

Functions:

Method and description	Input parameters	Return type

prepareRequestBody This function formulates and returns the request body based on the operationName, clientApplicationName and the attribute list passed from the service layer.	{String} **clientApplicationName** name of the client application. {String} **operationName** name of the client operation that needs to be addressed. {String} **attributeList** list of attributes that needs to be included in the request body based on the operation name.	{Promise} **httpRequestBody** formulated JSON HTTP request body.

4.1.1 eventDispatcher.js

This module provides functionalities to trigger and dispatch the HTTP REST request from this application to other applications.
This module associates with the requestBodyFactory to formulate the httpRequestBody in JSON format and interacts with the Axios REST client in the rest/client module to trigger the formulated HTTP request.

Functions:

Method and description	Input parameters	Return type
dispatchEvent This function formulates the HTTP request header, body and dispatches the event to the corresponding target REST server.	{string} **serviceType** provides "basic" if the service comes from the basicservice layer or else "individual". {String} **remoteIpAndPort** ip address, port of the client application in the format <ipaddress>:<port>. {String} **clientApplicationName** name of the client application to which we are going to send the request. {String} **operationName** name of the client operation that needs to be addressed. {String} **operationKey** operation key to access the service in the client application. {String} **attributeList** list of attributes that needs to be included in the request body based on the operation name. {String} **user** username of the request initiator. {String} **xCorrelator** UUID for the service execution flow that allows to correlate requests and responses. {String} **traceIndicator** Sequence number of the request. {String} **customerJourney** Holds information supporting customer's journey to which the execution applies.	{Promise} true if the operation is success.

4.2 database

This folder contains load.json file that consists of the ONF CoreModel representation of the application configuration.

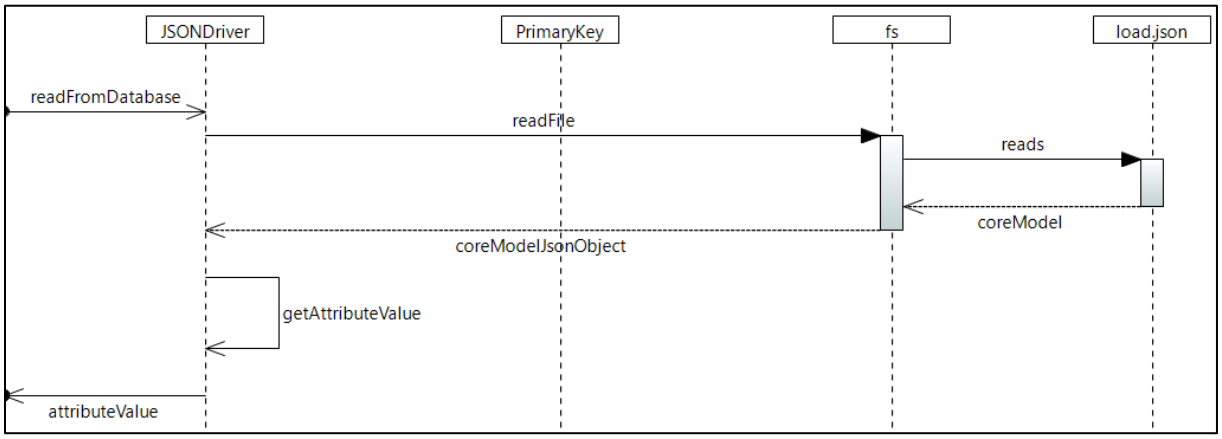
4.3 databaseDriver

This module consists of functionalities, which we can interact with the load.json file in ONF CoreModel.

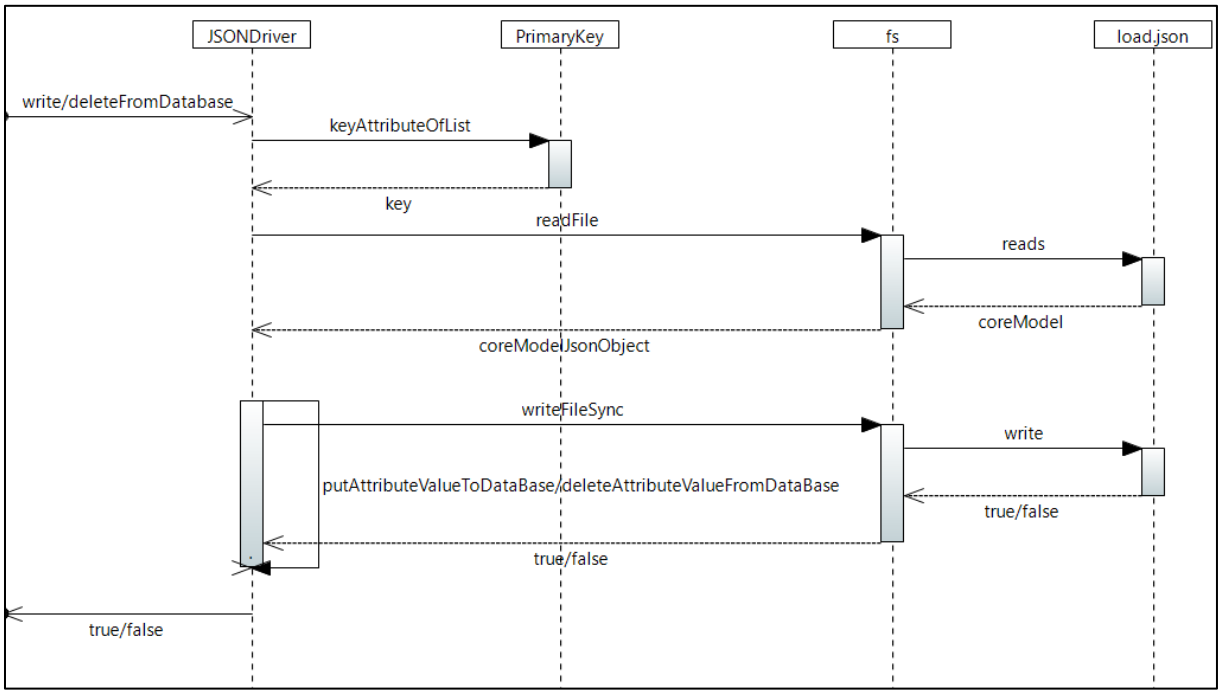
4.3.1 JSONDriver.js

This class provides functionality to perform CURD operation on the JSON database LOADfile, which is in the ONF CoreModel format.
The interaction with the file system will be performed with the use of 'fs module', which enables interacting with the file system in a way modeled on standard Portable Operating System Interface for UNIX(POSIX) functions.
Also, by using the 'path' module that provides utilities for working with files and directory paths. This module uses its own mechanism to traverse the ONF model JSON file based on the provided JSONPath.

Please refer the following flow for the ‘readFromDatabase’ operation,



Please refer the following flow for the ‘write/deleteFromDatabase’ operation,



Functions:

Method and description	Input parameters	Return type
readFromDatabase This function reads the requested JSONPath from the core-model.	{string} **JSONPath** JSON path that leads to the destined attribute.	{promise} return the requested value.
writeToDatabase This function updates an existing instance or creates a new instance in the LOADfile which is in ONF CoreModel based on the JSONPath and input parameters.	{string} **JSONPath** JSON path that leads to the destined attribute. {JSONObject|String} **valueToBeUpdated** value that needs to be updated. {boolean} **isAList** a boolean flag that represents whether the value to be updated is a list instance.	{promise} return true if the value is updated, otherwise returns false.
deletefromDatabase This function deletes the requested data in the JSONPath from the core-model.	{string} **JSONPath** JSON path that leads to the destined attribute. {JSONObject|String} **valueToBeDeleted** value that needs to be deleted. {boolean} **isAList** a boolean flag that represents whether the value to be deleted is a list.	{promise} return true if the value is deleted, otherwise returns false.

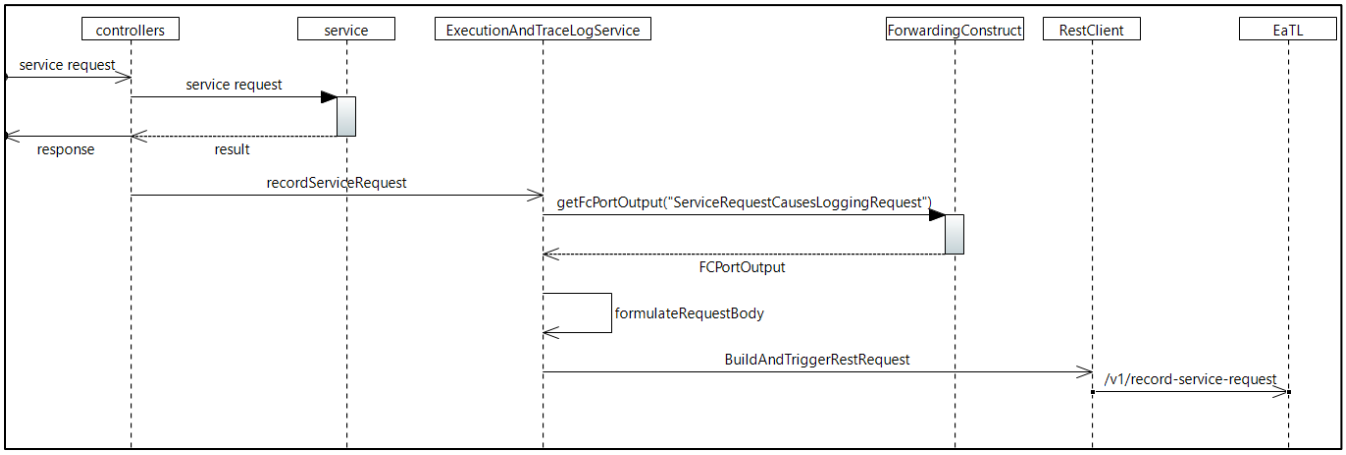
4.3.2 PrimaryKey.js

This file contains the primary key for the list attributes in core-model. If a new list is getting added, then the key attribute of the list should be updated in this file to make the JSONDriver functionalities to work as expected.

4.4 Logging

4.4.1 ExecutionAndTraceService.js

This class provides functionalities to log the Service request to the ExecutionAndTraceLog application in the SDN microservice architecture. A REST call will be initiated from the current application to the ExecutionAndTracelog application to record the transaction happened in the service layer.



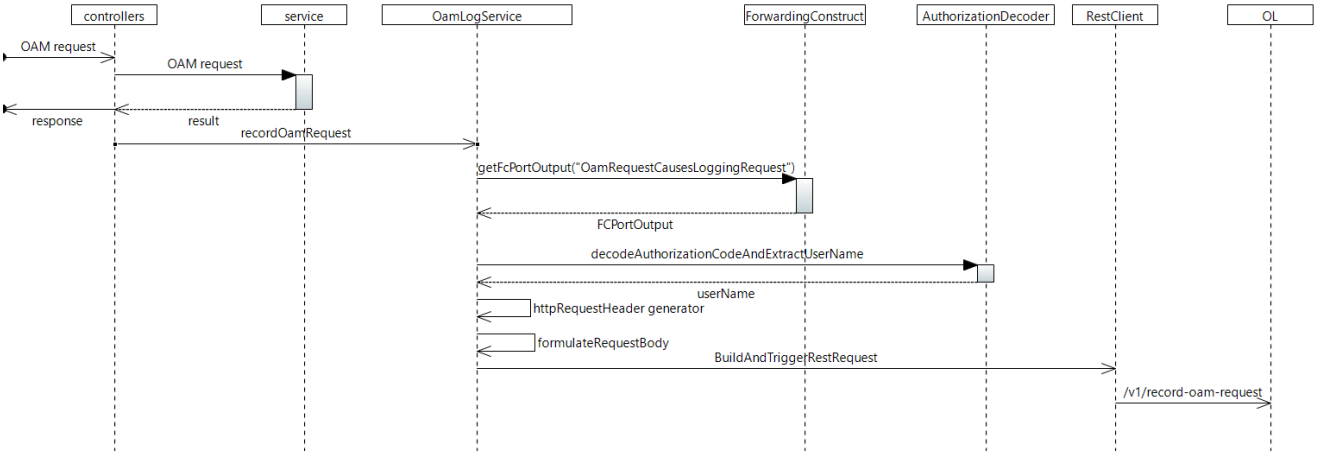
Function:

Method and description	Input parameters	Return type
recordServiceRequest This function formulates the request body with the required attributes that needs to be sent to record the service request in the ExecutionAndTraceLog application.	{string} **xCorrelator** correlation tag of the current execution. {string} **traceIndicator** sequence number of the execution. {string} **userName** name of the user who is accessed the service. {string} **originator** originator of the request. {string} **operationName** name of the called service. {string} **responseCode** response code of the REST call execution. {string} **requestBody** request body of the executed REST call. {string} **responseBody** response body of the executed REST call.	{Promise} return true if the operation is successful otherwise returns false.

4.4.2 OAMLogServices.js

This class provides functionality to log the OaM request to the OAMLog application in the SDN Microservice architecture.

A REST call will be initiated from this application to the OAMLog application to record the transaction happened in the OAM layer.



Function:

Method and description	Input parameters	Return type
recordOamRequest This function formulates the request body with the required attributes that needs to be sent to record the OaM request in the OamLog application.	{string} **oamPath** oam path that is accessed during the request. {string} **requestBody** incase if it is a put request, then the request body of the request. {string} **responseCode** response code of the REST call execution. {string} **authorizationCode** authorization code used to access the oam layer. This will then be decoded to find out the username. {string} **method** HTTP method of the OAM layer call. It can be PUT, GET.	{promise} return the requested value.

4.5 onfModel

4.5.1 model

4.5.1.1 CoreModel.js

This class provides a stub for ONF core-model. This class consolidates the technology specific extensions and provides functionality to manipulate the attributes in the core-model.

Field summary:

Type	Field
String	uuid
JSONarray of LogicalTerminationPoint	logicalTerminationPointList
JSONarray of forwardingDomain	forwardingDomainList
JSON object of profileCollection	profileCollection

Method summary:

Method and description	Input parameters	Return type
getUuid This function returns the uuid of core-model instance.		{promise} returns uuid.
createLogicalTerminationPoint This function adds a new logical-termination-point instance to the logical-termination-point list.	{JSONObject} **logicalTerminationPoint** an instance of the logical-termination-point.	{promise} returns true if the instance is added successfully to the logical-termination-point list.
deleteLogicalTerminationPoint This function deletes an instance from the logical-termination-point list.	{String} **uuid** of the logical-termination-point instance that needs to be deleted.	{promise} returns true if the instance is deleted successfully from the logical termination point list.
getLogicalTerminationPoint This function returns an instance from the logical-termination-point list for the provided uuid.	{String} **uuid** of the logical-termination-point instance that needs to be retrieved.	{promise} returns the logical-termination-point instance.
getLogicalTerminationPointList This function returns the list of logical-termination-point instances for the provided layer-protocol-name.	{String} **layerProtocolName** protocol name of the layer.	{promise} returns logical-termination-point instance List.
getForwardingDomainList This function returns the entire forwarding-domain list inside the core-model.		{promise} returns forwarding-domain list.

4.5.1.2 LogicalTerminationPoint.js

The LogicalTerminationPoint (LTP) class encapsulates the termination and adaptation functions of one or more technology specific layers represented by instances of LayerProtocol. This class provides a stub to instantiate and generate a JSON object for a LogicalTerminationPoint.

Field summary:

Type	Field
String	uuid
ltpDirectionEnum (SINK, SOURCE)	ltpDirection
Array of uuid	clientLTP
Array of uuid	serverLTP
JSON object of layerProtocol	layerProtocol

Constructor summary:

Constructor and description	parameters
This instantiates a new logicalTerminationPoint instance.	{String} **uuid** unified resource identifier for the httpClient. {String} ltpDirection direction of the LTP, it will be SINK for clients and SOURCE for servers. {String} **clientLTP** client LTPs ((operation-client/server) associated with http-client/server, ((http-client/server) associated with tcp-client/server)).

	{String} **serverLTP** server LTPs ((tcp-client/server) associated with http-client/server, ((http-client/server) associated with operation-client/server)). {String} **layerProtocol** an instance of the LayerProtocol class.

Method summary:

Method and description	Input parameters	Return type
getServerLtpList This function returns the server-ltp list for the given logical-termination-point uuid.	{String} **uuid** of the logical-termination-point.	{promise} returns the server-ltp list of the LTP.
getClientLtpList This function returns the client-ltp list for the given logical-termination-point uuid.	{String} **uuid** of the logical-termination-point.	{promise} returns the client-ltp list of the LTP.
setClientLtpList This function modifies the client-ltp list for the given logical-termination-point uuid.	{String} **uuid** of the logical-termination-point. {array} **clientUuidList** array of client uuids that needs to be updated.	{promise} returns true if the value is updated otherwise false.
setServerLtpList This function modifies the server-ltp list for the given logical-termination-point uuid.	{String} **uuid** of the logical-termination-point. {array} **serverUuidList** array of client uuids that needs to be updated.	{promise} returns true if the value is updated otherwise false.
getUuidListForTheProtocol This function returns the list of logical-termination-point uuid for the provided layer-protocol-name.	{String} **layerProtocolName** protocol name of the layer.	{promise} returns logical-termination-point uuid List.

4.5.1.3 LayerProtocols.js

The projection of an LTP into each technology specific layer is represented by a LayerProtocol (LP) instance. This class provides a stub to instantiate and generate a JSON object for a LayerProtocol.

Field summary:

Type	Field
String	localId
String (OPERATION_CLIENT, HTTP_CLIENT, TCP_CLIENT, OPERATION_SERVER, HTTP_SERVER, TCP_SERVER)	layerProtocolName

Constructor summary:

Constructor and description	parameters
This instantiates a new layerProtocolName instance.	{String} **localId** local identifier for the layerProtocol. {String} **layerProtocolName** name of the layer protocol (it can be tcp-server, tcp-client, http-server, http-client, operation-server, operation-client).

Method summary:

Method and description	Input parameters	Return type
getLayerProtocolName This function returns the layer-protocol-name for the given logical-termination-point uuid.	{String} **uuid** of the logical-termination-point.	{promise} returns the layerProtocolName of the LTP.

4.5.1.4 ForwardingDomain.js

The ForwardingDomain (FD) class models the component that represents a forwarding capability that provides the opportunity to enable forwarding (of specific transport characteristic information at one or more protocol layers) between points.

Field summary:

Type	Field
String	uuid
JSONArray of ForwardingConstruct	forwardingConstructList

Constructor summary:

Constructor and description	parameters
This instantiates a new ForwardingDomain instance.	{String} **uuid** unique identifier of the forwarding-domain. {String} **forwardingConstructList** list of forwarding-construct.

Method summary:

Method and description	Input parameters	Return type
getForwardingConstructForTheUuid This function returns the forwarding-construct instance for the given forwarding-construct uuid.	{string} **forwardingConstructUuid** forwarding-construct uuid in the forwarding-construct list in forwarding-domain.	{promise} returns forwarding-construct instance.
getForwardingConstructList This function returns the entire list of forwarding-construct instances inside all forwarding domains.		{promise} returns all forwarding-construct instance list.
getForwardingConstructForTheFCName This function returns the forwarding-construct instance that matches the forwarding-construct name.	{string} **forwardingName** forwardingName of the forwarding-construct.	{promise} returns forwarding-construct instance.
getForwardingConstructListForTheFcPortManagementDirection This function returns the forwarding-construct instance list for the fc-port management direction.	{string} **FcPortManagementDirectionUuid** fc-port management direction logical-termination-point attribute value.	{promise} returns forwarding-construct instance list.
getForwardingConstructListForTheFcPortOutputDirection This function returns the forwarding-construct instance list for the fc-port output direction.	{string} **FcPortOutputDirectionUuid** fc-port output direction logical-termination-point attribute value.	{promise} returns forwarding-construct instance list.

4.5.1.5 ForwardingConstruct.js

The ForwardingConstruct class (FC) represents enabled constrained potential for forwarding between two or more FcPorts at a particular specific layerProtocol.

Field summary:

Type	Field
String	uuid
JSONArray of key value pair	nameList
JSONArray of FCPort	fcPortList

Constructor summary:

Constructor and description	parameters
This instantiates a new ForwardingConstruct instance.	{String} **uuid** unified resource identifier for the forwarding-construct. {String} **nameList** name list that holds the forwardingName and forwardingKind details. {String} **fcPortList fcPort** instance list.

Inner class summary:

Inner class and description	Field summary	Constructor summary
name This class provides stub for the name list.	valueName (String) name (String)	constructor (valueName, name)
FcPort This class provides stub to instantiate a fc-port.	localId (String) portDirection (MANAGEMENT, INPUT, OUTPUT) logicalTerminationPoint (String)	constructor (localId, portDirection, logicalTerminationPoint)

Method summary:

Method and description	Input parameters	Return type
getForwardingNameForTheUuid This function returns the forwarding-construct/name/value-name=ForwardingName instance for the given forwarding construct uuid.	{string} **forwardingConstructUuid** forwarding-construct uuid.	{promise} returns ForwardingName of the matched forwarding-construct.
getForwardingKindForTheUuid This function returns the forwarding-construct/name/value-name=ForwardingKind instance for the given forwarding construct uuid.	{string} **forwardingConstructUuid** forwarding-construct uuid.	{promise} returns ForwardingKind of the matched forwarding-construct.
getFcPortOutputDirectionLogicalTerminationPointListForTheForwardingName This function returns the logical-termination-point(uuid) list of the fc-port in the output direction for the forwardingName.	{string} **forwardingConstructName** forwarding construct name as in forwarding-domain/forwarding-construct/name/value-name.	{Promise} return the logical-termination-point(uuid) list of the fc-port in the output direction for the forwardingName.

getFcPortOutputDirectionLogicalTerminationPointListForTheFcPortInputDirection This function returns the logical-termination-point(uuid) list of the fc-port in the output direction for the input fcport.	{string} **fcPortLogicalTerminationPoint** logical-termination-point of the fc-port input direction. {string} **context** if we want to filter the output fc-port for a specific application (for example to perform /embed-yourself only for the specific application).	{Promise} return the logical-termination-point(uuid) list of the fc-port in the output direction for the input fcport.
getFcPortOutputDirectionLogicalTerminationPointListForTheUuid This function returns the logical-termination-point(uuid) list of the fc-port in the output direction for the forwarding-construct uuid.	{string} **forwardingConstructUuid** forwarding-construct uuid.	{Promise} return the logical-termination-point(uuid) list of the fc-port in the output direction for the forwarding-construct uuid.
generateNextFcPortLocalId This function returns the next available uuid for the fc-port based on the provided forwarding-construct uuid.	{String} **forwardingConstructUuid** uuid of the forwarding-construct.	{promise} returns the next free uuid instance that can be used for the fc-port creation.
modifyFcPortLogicalTerminationPointUuid This function updates the logical-termination-point attribute of the fc-port.	{String} **forwardingConstructUuid** uuid of the forwarding-construct. {String} **fcPortLocalId** local-id of the fc-port. {String} **fcPortNewLogicalTerminationPoint** new logical-termination-point that needs to be updated.	{promise} returns true if the value is updated.
isFcPortExists This function returns true if a fc-port is already available for the provided logical-termination-point of an operation(client/server) Uuid	{String} **forwardingConstructUuid** uuid of the forwarding-construct {String} **operationUuid** logical-termination-point of an operation(client/server) Uuid	{promise} returns true if a fc-port is already available
getFcPortLocalId This function returns the fc-port local-id for the provided logical-termination-point of an operation	{String} **forwardingConstructUuid** uuid of the forwarding-construct {String} **operationUuid** logical-termination-point of an operation(client/server)Uuid	{promise} returns the fc-port local-id
addFcPort This function adds a Fc port to the forwarding-construct	{String} **forwardingConstructUuid** uuid of the forwarding-construct {String} **fcPortLocalId** local-id of the fc-port {String} **fcPortDirection** direction of the fc-port {String} **fcPortLogicalTermincationPoint** logical-termination-point of the fc-port	{promise} returns true if the fc-port is added to the list

deleteFcPort This function deletes a Fc port from the forwarding-construct	{String} **forwardingConstructUuid** uuid of the forwarding-construct {String} **fcPortLocalId** fc-port local id	{promise} returns true if the fc-port is added to the list

4.5.1.6 ProfileCollection.js

The ProfileCollection class models the component that represents profiles collection in the CoreModel.

Field summary:

Type	Field
JSONArray of Profile	profileList

Constructor summary:

Constructor and description	parameters
This instantiates a new ProfileCollection instance.	{JSONArray} **profileList** list of profiles.

Method summary:

Method and description	Input parameters	Return type
getProfileInstanceForTheUuid This function returns an instance from the profile list for the provided uuid.	{String} **uuid** of the profile instance that needs to be retrieved.	{promise} returns profile instance.
isProfileExists This function returns true if the profile uuid exists in the profile list.	{String} **profileUuid** uuid of the profile instance that needs to be retrieved.	{promise} returns true if the profile uuid exists in the profile list.
getProfileList This function returns the profile list.		{promise} returns profile instance.
addProfile This function includes an instance to the profile list.	{Profile} **profileInstance** profile instance to be included.	{promise} returns true if the operation is success.
deleteProfile This function deletes a profile.	{String} profileUuid uuid of the profile.	{promise} returns true if the operation is successful.

4.5.1.7 Profile.js

The Profile class models the component that represents a profile. New profile types can extend this class.

Field summary:

Type	Field
String	uuid
String (APPLICATION_PROFILE, INTEGER_PROFILE, OAM_RECORD_PROFILE, SERVICE_RECORD_PROFILE, ADMIN_PROFILE)	profileName

Constructor summary:

Constructor and description	parameters
This instantiates a new profile instance.	{String} **uuid** unified resource identifier for the profile. {String} **profileName** name of the profile.

Method summary:

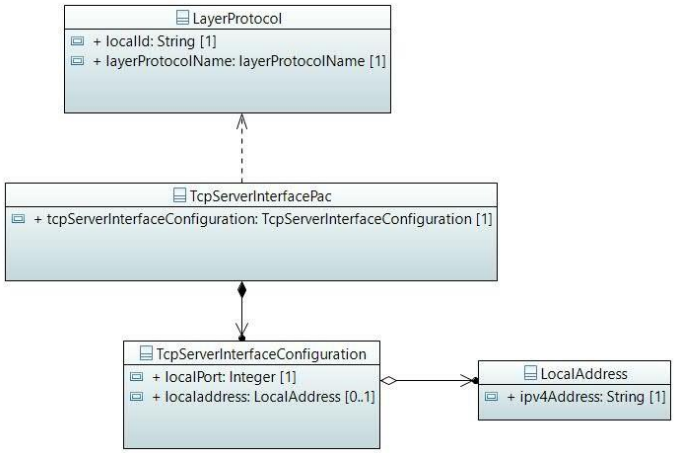
Method and description	Input parameters	Return type
getUuidListForTheProfileName This function returns the list of profile uuid for the provided profile-name.	{String} **profileNameType** name of the profile.	{promise} returns profile uuid List.

4.5.1.8 LayerProtocol

This package consists of a list of sub classes for the layerProtocol class.

4.5.1.8.1 TcpServerInterface.js

This class provides a stub to instantiate and generate a JSON object for a tcpServerInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the tcpServerInterfacePac,



Constructor summary:

Constructor and description	parameters
This instantiates a new tcp server layer protocol.	{string} **localAddress** tcp server ipaddress where the application is hosted. {string} **localPort** tcp server port where the application is running.

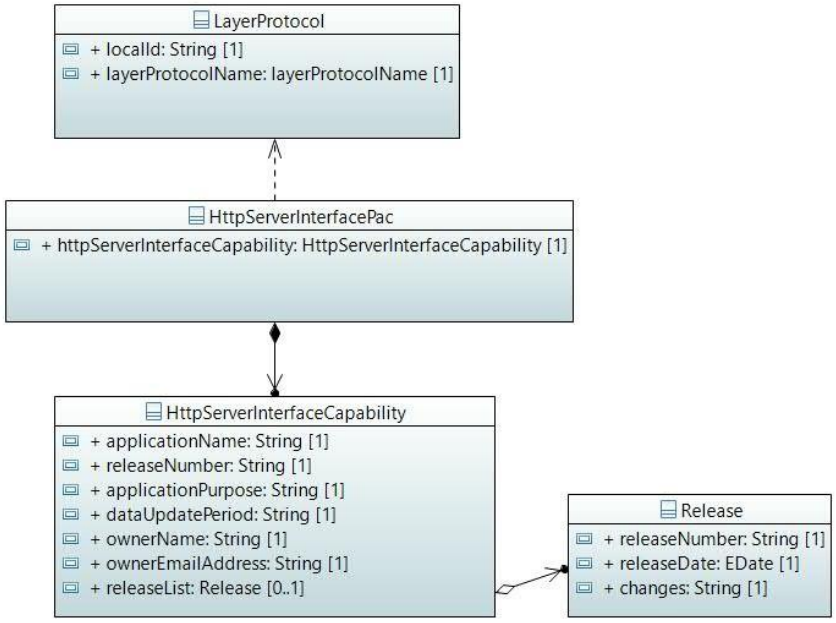
Method summary:

Method and description	Input parameters	Return type
getLocalAddress		{promise} returns ip address of the current application.

This function returns the IpV4 address of the current application.		
getLocalPort This function returns the port where the current application is running.		{promise} returns the port where the current application is running.

4.5.1.8.2 HttpServerInterface.js

This class provides a stub to instantiate and generate a JSON object for a httpServerInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the httpServerInterfacePac,



Constructor summary:

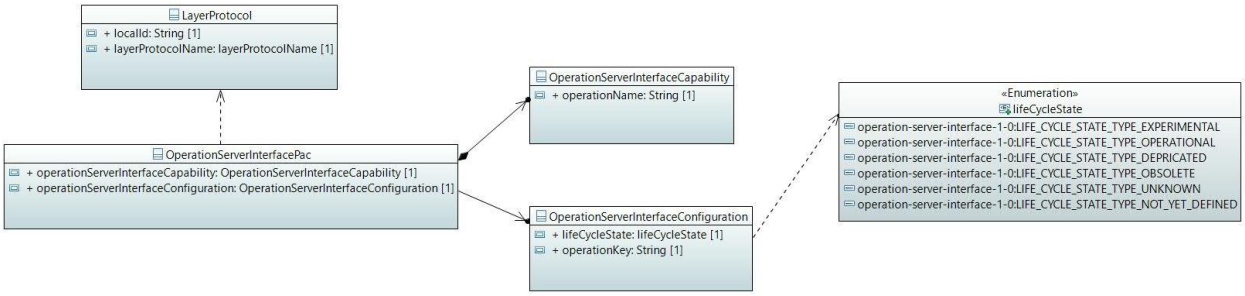
Constructor and description	parameters
This instantiates a new HTTP server layer protocol.	{string} **applicationName** name of the current application. {string} **releaseNumber** release number of the current application. {string} **applicationPurpose** purpose of the current application. {string} **dataUpdatePeriod** data update period can be 24hr, 1hr, manual or realtime. {string} **ownerName** name of the application owner. {string} **ownerEmailAddress** email address of the application owner. {string} **releaseList** release list of the application along with its history.

Method summary:

Method and description	Input parameters	Return type
getHttpServerCapability This function returns the HTTP server capability.		{promise} returns the capability of the HTTP server.
getApplicationName This function returns the name of the current application.		{promise} returns the name of current application.
getReleaseNumber This function returns the release number of the current application.		{promise} returns release number of current applications.
getReleaseList This function returns the list of releases for the application.		{promise} returns the release list of the application.

4.5.1.8.3 OperationServerInterface.js

This class provides a stub to instantiate and generate a JSON object for an operationServerInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the operationServerInterfacePac



Constructor summary:

Constructor and description	parameters
This instantiates a new operation server layer protocol.	{String} **operationName** name of the operation.

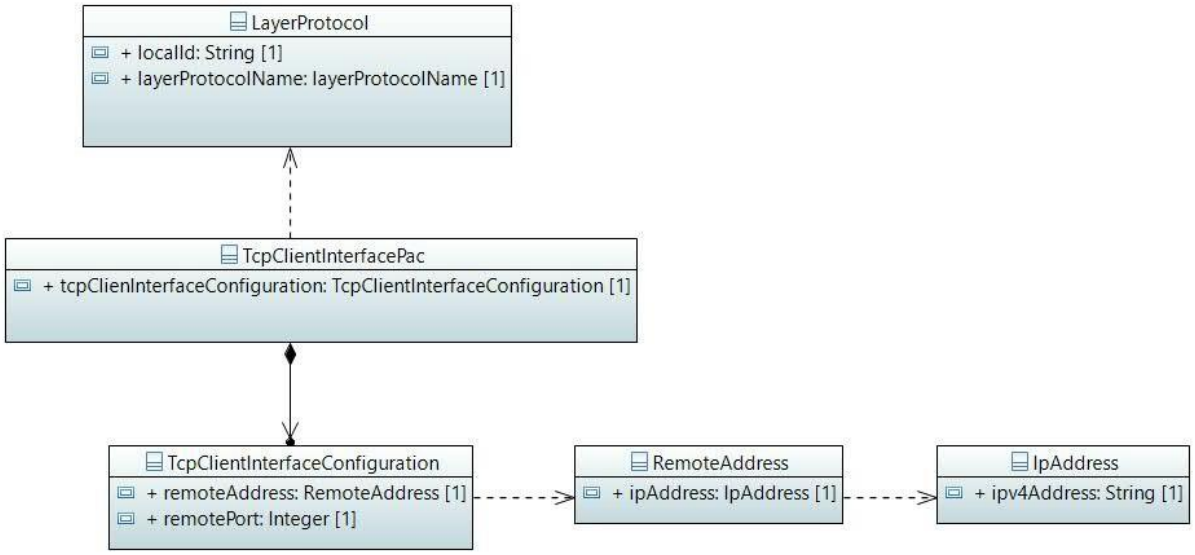
Method summary:

Method and description	Input parameters	Return type
getOperationName This function returns the operation name for the given operation server uuid.	{String} **operationServerUuid** uuid of the operation server instance.	{promise} returns operation name of the operation server instance.
getOperationKey This function returns the operation key of the operation server.	{String} **operationServerUuid** uuid of the operation server.	{promise} returns the operation key.
setOperationKey This function sets the operation key of the operation server.	{String} **operationServerUuid** uuid of the operation server {String} operationKey operation key that needs to be updated.	{promise} returns true if the operation is successful.
getLifeCycleState This function returns the life-cycle-state for the given operation server uuid.	{String} **operationServerUuid** uuid of the operation server instance.	{promise} returns life-cycle-state of

		the operation server instance.
getOperationServerUuidFor TheOperationName This function returns the operation server uuid for the given operation name.	{String} **operationName** operation name of the operation server.	{promise} returns operation server uuid.

4.5.1.8.4 TcpClientInterface.js

This class provides a stub to instantiate and generate a JSON object for a tcpClientInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the tcpClientInterfacePac



Constructor summary:

Constructor and description	**parameters**
This instantiates a new tcp client layer protocol.	{string} **remoteAddress** tcp ipaddress where the application is hosted. {string} **remotePort** tcp port where the application is running.

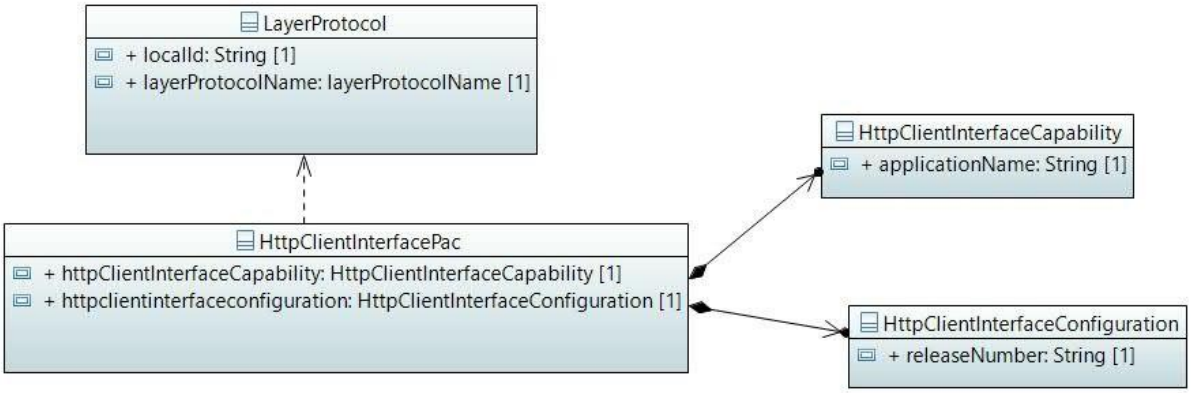
Method summary:

Method and description	**Input parameters**	**Return type**
getTcpIpAddressAndPort This function returns the tcp ip address and port (in the format <ipaddress>:<port>) where the client application is running.	{String} **tcpClientUuid** uuid of the tcp client.	{promise} returns tcp ip address and port (in the format <ipaddress>:<port>).
getRemoteAddress This function returns the tcp ip address where the client application is running.	{String} **tcpClientUuid** uuid of the tcp client.	{promise} returns tcp ip address.
getRemotePort This function returns the tcp port where the client application is running.	{String} **tcpClientUuid** uuid of the tcp client.	{promise} returns tcp port.

generateNextUuid This function generates the tcp-client uuid for the given http-client uuid.	{String} **httpClientUuid** uuid of the http-client-interface logical-termination-point.	{promise} returns the tcp-client uuid generated for the given http-client uuid.
createTcpClientInterfaceAnd AddtoLogicalTerminationPoint This function creates a new tcp-client-interface and update the created instance to the logical-termination-point list.	{String} **httpClientUuid** http-client uuid for the application for which we are going to create the tcp-client-interface. {String} **tcpClientUuid** tcp-client uuid to create the new tcp-client instance. {String} **ipv4Address** ipaddress where the application is hosted. {String} **port** where the application is running.	{promise} returns true if the tcp-client interface is created.
setTcpRemoteAddressAndPortForTh eUuid This function modifies the tcp-client remote-address and remote-port for the provided tcp client uuid.	{String} **tcpClientUuid** uuid of the tcp-client. {String} **remoteAddress** that needs to be modified. {String} **remotePort** that needs to be modified.	{promise} returns true if the value is updated or return false.

4.5.1.8.5 HttpClientInterface.js

This class provides a stub to instantiate and generate a JSON object for a httpClientInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the httpClientInterfacePac



Constructor summary:

Constructor and description	parameters
This instantiates a new HTTP client layer protocol.	{string} **applicationName** name of the client application. {string} **releaseNumber** release number of the client application.

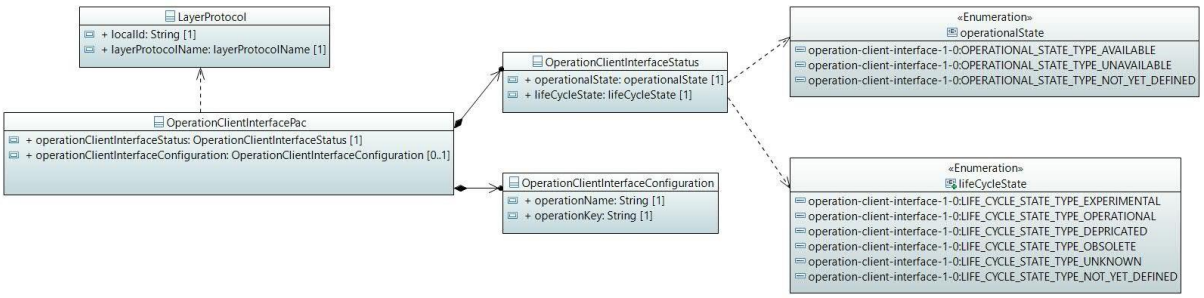
Method summary:

Method and description	Input parameters	Return type

getApplicationName This function returns the application name for the HTTP client uuid.	{String} **httpClientUuid** uuid of the http-client-interface.	{promise} returns the application name.
getReleaseNumber This function returns the release number for the HTTP client uuid.	{String} **httpClientUuid** uuid of the http-client-interface.	returns {promise} returns the release number.
setReleaseNumber This function sets the release number for the HTTP client uuid.	{String} **httpClientUuid** uuid of the http-client-interface. {String} **newReleaseNumber** new release number of the http-client-interface.	{promise} returns true if the value is set.
generateNextUuid This function returns the next available uuid for the http-client-interface.		returns {promise} returns the next free uuid instance that can be used for the http-client-interface ltp creation.
getHttpClientUuidForTheApplicationAndReleaseNumber This function returns the uuid of the http-client-interface for the application-name and release-number.	{String} **applicationName** name of the application. {String} **releaseNumber** release number of the application.	{promise} returns HTTP logical-termination-point uuid or undefined incase if there is no match found.
getHttpClientUuidForTheApplicationName This function returns the uuid of the http-client-interface for the application-name and release-number.	{String} **applicationName** name of the application. {String} **releaseNumber** release number of the application.	{promise} returns HTTP logical-termination-point uuid or undefined incase if there is no match found.
createHttpClientInterfaceAndAddto LogicalTerminationPoint This function creates a new http-client-interface and update the created instance to the logical-termination-point list	{String} **httpClientUuid** HTTP client unique identifier for the new application. {String} **operationClientUuidList** associated services for the application. {String} **tcpClientUuid** tcp client uuid that provides information about the ip address and port number of the application. {String} **applicationName** name of the application. {String} **releaseNumber** release number of the application.	{promise} returns true if the http-client interface is created.

4.5.1.8.6 OperationClientInterface.js

This class provides a stub to instantiate and generate a JSON object for an operationClientIInterface layer protocol. This class is a sub class for LayerProtocol. This class has the following model that represents the operationClientInterfacePac



Constructor summary:

Constructor and description	parameters
This instantiates a new operation client layer protocol.	{string} **operationName** operation name of the client that needs to be called back.

Method summary:

Method and description	Input parameters	Return type
getOperationName This function returns the operation name of the operation client.	{String} **operationClientUuid** uuid of the operation client.	{promise} returns the operation name.
getOperationKey This function returns the operation key of the operation client.	{String} **operationClientUuid** uuid of the operation client.	{promise} returns the operation key.
setOperationKey This function sets the operation key of the operation client.	{String} **operationClientUuid** uuid of the operation client. {String} operationKey operation key that needs to be updated.	{promise} returns true if the operation is successful.
getTcpIpAddressAndPortFor TheOperationClient This function returns the tcp ip address and port where the application that provides the operation-client operation is running.	{String} **operationClientUuid** uuid of the operation.	{promise} returns the tcp ip address and port where the application that provides the operation-client operation is running.
getOperationClientUuidFor TheOperationName This function returns the operation client uuid information for the given http-client uuid and operation name.	{String} **httpClientUuid** uuid of the http-client. {String} **operationName** name of the operation.	{promise} returns the operation client uuid for the operation name.
generateNextUuid This function generates the operation client uuid for the given HTTP client uuid and operation name.	{String} **httpClientUuid** uuid of the HTTP client logical termination point. {String} **operationName** operation name of the operation client.	{promise} returns the operation client uuid generated for the given HTTP uuid.
createOperationClientInterfaceAndAddto LogicalTerminationPoint This function creates a new http-client-interface and update the created instance to the logical-termination-point list.	{String} **httpClientUuid** http-client unique identifier for the new application in which the operation exists.	{promise} returns true if the operation-client interface is created.

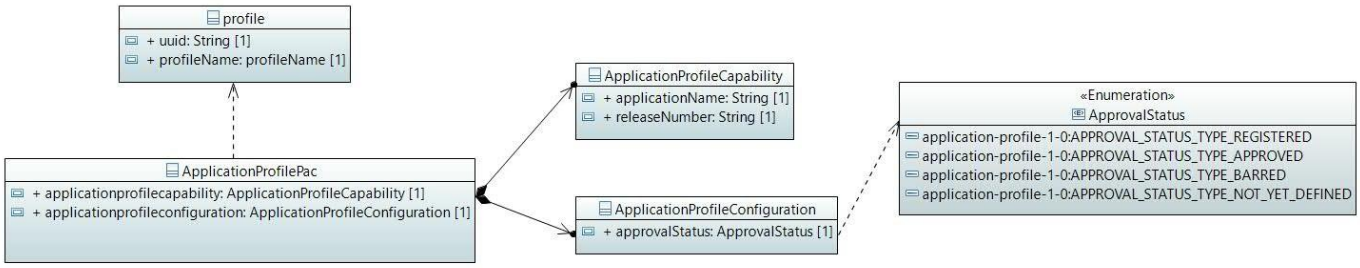
	{String} operationClientUuid operation-client uuid for the new operation. {String} **operationName** name of the operation.	
setOperationNameForTheUuid This function modifies the operation name for the provided operation client uuid.	{String} **operationClientUuid** uuid of the operation-client. {String} **operationName** name of the operation.	{promise} returns true if the value is updated or return false.

4.5.1.9 profiles

This package consists of a list of sub classes for the profile class

4.5.1.9.1 ApplicationProfile.js

This class provides a stub to instantiate and generate a JSON object for a ApplicationProfile. This class is a sub class for profile. This Application profile is being utilized by TypeApprovalApplication to store the application's approval status. This class has the following model that represents the applicationProfilePac.



Constructor summary:

Constructor and description	parameters
This instantiates a new application profile	{string} applicationName name of the client application. {string} releaseNumber release number of the client application. {string} approvalStatus approval status of the client application.

Method summary:

Method and description	Input parameters	Return type
getApprovalStatusForTheUuid This function returns the approval status for the provided application profile uuid.	{String} **uuid** of the application profile.	{promise} returns the approval status.
getApplicationNameForTheUuid This function returns the application name for the provided application profile uuid.	{String} **uuid** of the application profile.	{promise} returns the application name.
getApplicationReleaseNumberForTheUuid	{String} **uuid** of the application profile.	{promise} returns the application release number.

This function returns the application release number for the provided application profile uuid.		
getApprovalStatus This function returns the approval status for the provided application and release number.	{String} **applicationName** name of the application {String} releaseNumber release number of the application.	{promise} returns the approval status.
getProfileUuid This function returns the approval status for the provided application and release number.	{String} **applicationName** name of the application {String} releaseNumber release number of the application.	{promise} returns the approval status.
isProfileExists This function returns true if a profile exists for the provided application and release number.	{String} **applicationName** name of the application. {String} **releaseNumber** release number of the application.	{promise} returns true if the profile exists.
setApprovalStatus This function sets the approval-status for the provided application-name and release-number.	{String} **applicationName** name of the application. {String} **releaseNumber** release number of the application. {String} **approvalStatus** approval status of the application.	{promise} returns true if the value is set.
createProfile This function creates a new application profile	{String} profileName name of the profile {array} profileAttributes list of attributes for the profile creation	{promise} returns uuid of the created profile
generateNextUuid This function returns the next available uuid of the application-profile.		{promise} returns the next free uuid instance that can be used for the application profile creation.

4.5.2 Services

4.5.2.1 LogicalTerminationPointService.js

This module provides functionality to manipulate the logical termination point. For example, to instantiate client instances for a new application in the LOADfile, this module provides a service called "createLogicalTerminationPointInstanceGroup" which will instantiate the tcp, http, operation client instances for the new application and updates it to the logical-termination-point list.

Function:

Method and description	Input parameters	Return type
createLogicalTerminationPoint InstanceGroup This function creates the tcp, http, operation client instances (if it doesn't exist) and link them together.	{String} **applicationName** name of the client application. {String} **releaseNumber** release of the client application.	{object} operationClientUuid InformationInstance returns the generated

	{String} **ipv4Address** ip address of the client application. {String} **port** of the client application. {array} **operationList** list of operation client that needs to be created.	operation client information.
updateLogicalTerminationPoint InstanceGroup This function updates the tcp, http, operation client instances that linked together with the new values provided in the input	{String} **applicationName** name of the client application {String} **releaseNumber** release of the client application {String} **ipv4Address** ip address of the client application {String} **port** of the client application {array} **operationList** list of operation client that needs to be created	{promise} return true if the value is updated, otherwise returns false
deleteLogicalTerminationPoint InstanceGroup This function deletes the tcp, http, operation client for the provided application and release number	{String} **applicationName** name of the client application {String} **releaseNumber** release of the client application	{Promise} returns the deleted OperationClientLists associated to the application

4.5.2.2 ForwardingConstructServices.js

This module provides functionality to configure, unconfigure and automate the ForwardingConstruct.

Function:

Method and description	Input parameters	Return type
configureAndAutomateForwardingConstruct This function configures the forwarding construct based on the provided new operation client information and automates the already existing forwarding construct.	{String} **serviceType** service type can be basic or individual. {String} **operationServerUuid** operation server uuid of the request url. {String} **forwardingConstructConfigurationList** list of operation uuid along with the forwarding name that needs to be modified. {list} **attributeList** list of attributes required during forwarding construct automation (to send in the request body). {String} **user** who initiates this request. {string} **xCorrelator** flow id of this request. {string} **traceIndicator** trace indicator of the request. {string} **customerJourney** customer journey of the request.	

unConfigureAndAutomateForwardingConstruct This function removes the configured operation clients in the forwarding construct based on the provided operation client information and automates the forwarding construct.	{String} **serviceType** service type can be basic or individual. {String} **operationServerUuid** operation server uuid of the request url. {String} **operationClientUuidLists** list of operation client uuids that needs to be deleted. {list} **attributeList** list of attributes required during forwarding construct automation (to send in the request body). {String} **user** who initiates this request. {string} **xCorrelator** flow id of this request. {string} **traceIndicator** trace indicator of the request. {string} **customerJourney** customer journey of the request.	

4.5.3 Utility

4.5.3.1 ONfAttributeFormatter.js

This module provides functionalities that converts the attributes to ONF CoreModel format.

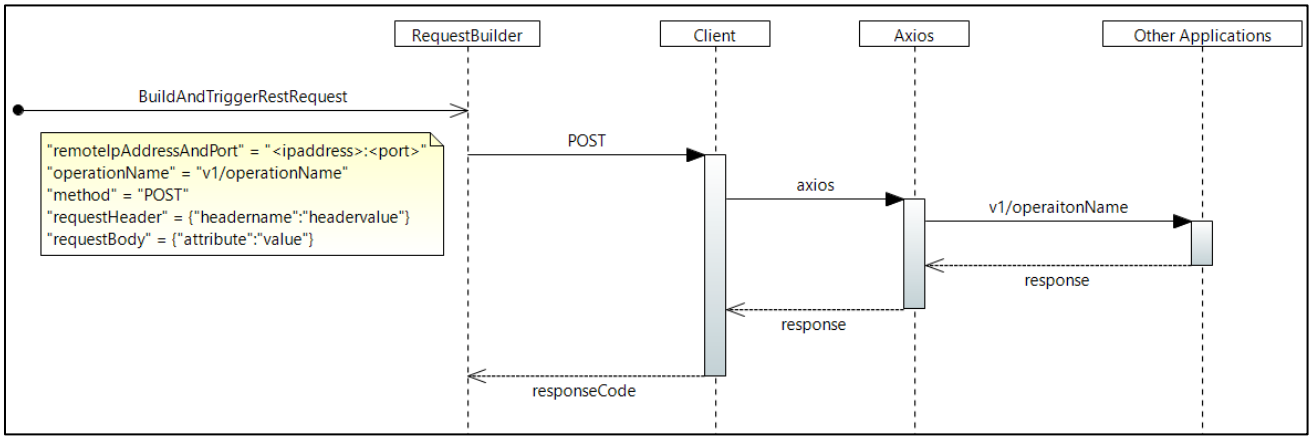
Function:

Method and description	Input parameters	Return type
modifyJSONObjectKeysToKebabCase This function modifies the JSON object keys from lower camelCase to kebabCase	{Object} **JSONObject** JSON object for which the keys need to be modified to kebabCase.	{Object} modified JSON object in kebabcase.

4.6 REST

4.5.2 client

The module in this package provides a REST client using which we can communicate with the REST server in other applications.



4.5.2.1 Client.js

This module provides functionality to perform HTTP request to client application. This module uses the AXIOS package as a REST client.

Function:

Method and description	Input parameters	Return type
post This function performs HTTP POST method.	{object} **request** object that consists of the httpRequestBody, httpRequestHeader, url.	{promise} return the response code.

4.5.2.2 RequestBuilder.js

This module provides functionality to construct a REST request

Function:

Method and description	Input parameters	Return type
BuildAndTriggerRESTRequest This function triggers a REST request by calling the RESTClient	{string} **remoteIpAddressAndPort** ip address, port of the client application in the format <ipaddress>:<port>. {string} **operationName** service that needs to be addressed in the client application. {string} **method** HTTP method for the REST request. {string} **requestHeader** HTTP request header for the REST call. {string} **requestBody** request body for the REST call	{promise} returns the HTTP response received

4.5.2.3 RequestHeader.js

This class provides functionality to create a HTTP request header.

Field summary:

Type	Field
String	user
String	originator
String	xCorrelator
String	traceIndicator
String	customerJourney
String	operationKey
String	contentType

Constructor summary:

Constructor and description	parameters

This instantiates a new Request header instance.	{String} **user** identifier from the system starting the service call. If not available, originator value will be copied to this attribute. {String} **xCorrelator** UUID for the service execution flow that allows to correlate requests and responses. {String} **traceIndicator** Sequence of request numbers along the flow, if it is empty, set it to 1. {String} **customerJourney** Holds information supporting customer's journey to which the execution applies. {String} **operationKey** operation key to access the service in the client application.

Method summary:

Method and description	Input parameters	Return type
xCorrelatorGenerator This function generates a xCorrelator based on the regular expression provided in the specification.		{promise} return the xCorrelator.

4.5.2 server

The modules and classes in this package provide functionality to support the REST server to formulate the HTTP response (header, body, response code) as per the requirement of the "ApplicationPattern specification".

4.5.2.1 ResponseBuilder.js

This module provides functionality to build the HTTP response object
Function:

Method and description	Input parameters	Return type
buildResponse This function builds the HTTP response object.	{JSONObject} **response** HTTP response object. {String} **responseCode** HTTP response code. {JSONObject} **responseBody** HTTP response body. {JSONObject} **responseHeader** HTTP response header.	

4.5.2.2 RequestHeader.js

This class provides functionality to create a HTTP response header.
Field summary:

Type	Field
String	xCorrelator
String	execTime
String	backendTime
String	lifeCycleState
String	contentType

Constructor summary:

Constructor and description	parameters
This instantiates a new response header instance.	{String} **xCorrelator** User identifier from the system starting the service call. If not available, originator value will be copied to this attribute. {String} **execTime** Identification for the system consuming the API, name of the current application. {String} **backendTime** UUID for the service execution flow that allows to correlate requests and responses. {String} **lifeCycleState** Sequence of request numbers along the flow, if it is empty, set it to 1.

Method summary:

Method and description	Input parameters	Return type
xCorrelatorGenerator This function generates a xCorrelator based on the regular expression provided in the specification.		{promise} return the xCorrelator.
executionTimeInMilliseconds This function calculates the execution time for processing the request.	{JSONObject} startTime start time of the request.	{promise} return the execution time in milli seconds.
createResponseHeader This function creates response header based on the provided input values.	{JSONObject} **xCorrelator** of the request. {JSONObject} **startTime** start time of the request. {JSONObject} **operationName** of the request.	{promise} return the response header.

4.5.2.3 responseBody

This package contains classes that are used to represent generic response bodies.

4.5.2.3.1 ConsequentAction.js

This class provides a stub for the consequent action list.

Field summary:

Type	Field
String	label
String	request
String	displayInNewBrowserWindow

Constructor summary:

Constructor and description	parameters
This instantiates a new consequent action list	{String} **label** of the consequent action. {String} **request** url that needs to be addressed to perform the consequent action.

	{String} **displayInNewBrowserWindow** should be true if the consequent action needs to be displayed in a new tab.

4.5.2.3.2 ResponseValue.js

This class provides a stub for the consequent action list

Field summary:

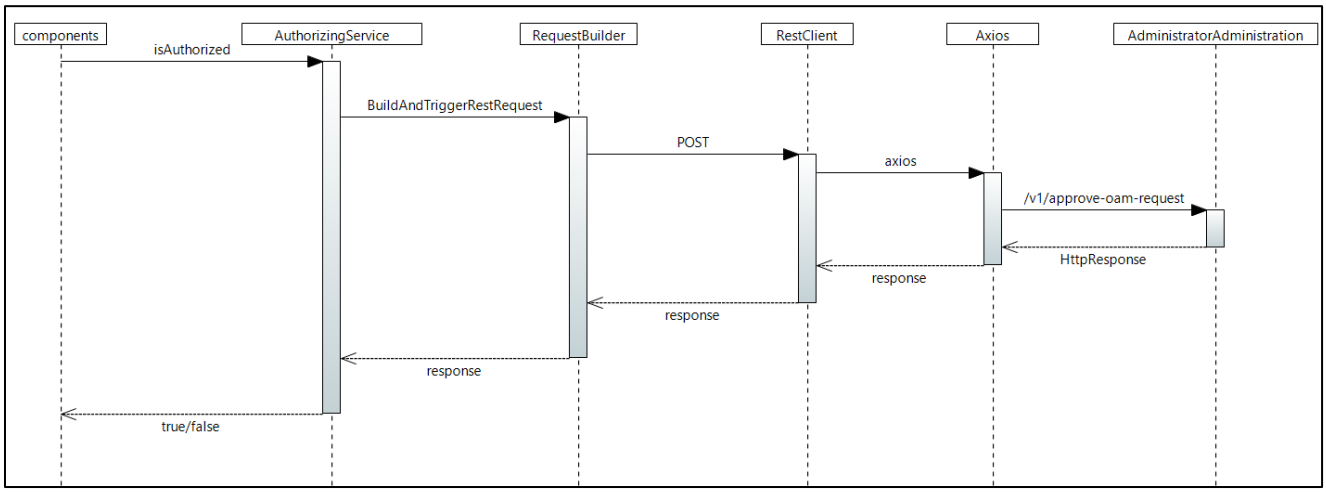
Type	Field
String	fieldName
String	value
String	datatype

Constructor summary:

Constructor and description	parameters
This instantiates a new ResponseValue object	{String} **fieldName** field name of the response value list. {String} **value** of the field name. {String} **datatype** data type of the value.

4.7 Security

This package contains modules that provides authorizing service.



4.7.1 AuthorizingService.js

This module provides functionality to authenticate an OAM layer request by getting an approval from the AdministratorAdministration.

Function:

Method and description	Input parameters	Return type

isAuthorized This function authorizes the user credentials.	{string} **authorizationCode** authorization code received from the header. {string} **method** is the https method name.	{boolean} return the authorization result.

4.7.2 AuthorizationDecoder.js

This module provides functionality to decode an authorization code.

Function:

Method and description	Input parameters	Return type
decodeAuthorizationCodeAndExtractUserName To decode base64 authorization code from authorization header.	{string} **authorizationCode** base64 encoded authorization code.	{Promise} returns user name based on the decoded authorization code.

4.8 softwareUpgrade

4.8.1 BequeathYourDataAndDie.js

This module provides functionality to migrate data from the current version to the next version. This file should be modified according to the individual service forwarding requirements