

# OOF Beijing (R2) Functional Testing planning

Table of Contents	<ul style="list-style-type: none"><li>• <a href="#">Test Planning for OOF Optimization Service Design Framework (OOF-OSDF)</a><ul style="list-style-type: none"><li>◦ <a href="#">Abbreviations Used</a></li><li>◦ <a href="#">OOF-OSDF Beijing Release CSIT Functional Test Cases</a></li><li>◦ <a href="#">Example Request/Response Payloads for OOF-OSDF Functional Test Cases</a></li></ul></li><li>• <a href="#">Test Planning for OOF Homing and Allocation Service (OOF-HAS)</a></li><li>• <a href="#">Appendix A: Overview of ONAP Testing Requirements</a></li><li>• <a href="#">Appendix B: Overview of OOF Scope</a><ul style="list-style-type: none"><li>◦ <a href="#">Overview of OOF-OSDF Scope</a><ul style="list-style-type: none"><li>▪ <a href="#">General Description</a></li><li>▪ <a href="#">Technical Description for OOF-OSDF Functionality related to OOF-HAS</a></li></ul></li><li>◦ <a href="#">Overview of OOF-HAS Scope</a><ul style="list-style-type: none"><li>▪ <a href="#">General Description</a></li><li>▪ <a href="#">Technical Description for OOF-HAS Functionality</a></li></ul></li></ul></li><li>• <a href="#">Appendix C: Resources and Links</a></li></ul>
-------------------	---

## Test Planning for OOF Optimization Service Design Framework (OOF-OSDF)

The test structure here has been adapted from [Policy Team's CSIT Functional Test Cases](#) created by [Pamela Dragosh](#)

### Abbreviations Used

The following abbreviations are used in the functional test case description below since there may be substantial repetition along with clarification notes associated with some terms:

Abbreviations:

- 1. **CHECK-REQ-OR-OPTIONAL**
  - a. Check if the test is required or optional. For instance, health checks for dependencies is likely optional because this will be captured in the tests for request/response
- 2. **EMULATORS-OR-SERVICES-ARE-UP**
  - a. Emulator or service should be up and running
  - b. Emulator or service configuration file should be available and loaded
  - c. **Notes:** For OOF internal components (e.g. OOF-OSDF connecting to OOF-HAS API), real services may be used when convenient
- 3. **HTTP-200-TRUE**
  - a. Component (or all components) should return health status as **"true"** (HTTP response code of 200, response content containing the string "true")
  - b. **Notes:** (a) Verify whether the external components also have standardized on "true" as the value
- 4. **SIMPLE-GET-HEALTH-CHECK-API**
  - a. API: healthcheck
  - b. HTTP Request Method: GET
  - c. HTTP Endpoint: http://<host>:<port>/healthcheck
  - d. **Notes:** (a) check whether https can/should be used, and whether mutual TLS is required when using OOM/K8S, and (b) verify if the health check is required for dependencies (it will help in quickly debugging but will add extra logic in our testing)
- 5. **SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES**
  - a. API: specific to each component
  - b. Endpoint: http://<host>:<port>/<specific-API>
  - c. Method - POST in most cases; GET in some cases
  - d. **Notes:** (a) check whether https can/should be used, and whether mutual TLS is required when using OOM/K8S

### OOF-OSDF Beijing Release CSIT Functional Test Cases

Id	Description	Pre-conditions	Test Steps	Expected Results
A: Health Checks for OOF-OSDF Components and Dependencies (Policy and OOF-HAS API)				
A.1	Perform health check for the OOF-OSDF components using Health Check API <ul style="list-style-type: none"><li>• OSDF Manager</li></ul>	[OSDF Manager] <b>EMULATORS-OR-SERVICES-ARE-UP</b>  Server and authentication details should be configured at <code>\$OOF_HOME/config/feature-healthcheck.properties</code>	<b>SIMPLE-GET-HEALTH-CHECK-API</b>	<b>HTTP-200-TRUE</b>

A. 2	<del>CHECK-REQ-OR-OPTIONAL</del> Perform health check for the following external components and OOF components using Health Check API:  <ul style="list-style-type: none"> <li>Policy (external component)</li> <li>OOF HAS API (OOF component)</li> </ul>	[Policy Emulator] [OOF HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-HEALTH-CHECK-API</del>	<del>HTTP-200-TRUE</del>  NOTE: Per comment and discussion with Ramki, removing this cell  TODO: Retain this cell till 19 Feb 2018 and then remove it.
<b>B: Fetch Data from Emulators (valid and invalid data, via GET and POST)</b>				
B. 1	Retrieve response corresponding to "valid request" from HAS-API emulator  <ul style="list-style-type: none"> <li>OSDF HAS (POST data)</li> </ul>	[OOF-HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payload for OSDF-HAS request (based on SO-OOF/HAS Request Example below in section on payloads; <a href="#">An kitkumar Patel</a> to add the payload; <a href="#">Shankar anarayanan Puzhavakath Narayanan</a> to review)  TODO: Endpoint and ports	Should receive a "request accepted" type response, following which we query status and the status should be a valid one (translating, translated, solving, solved, solution not found, etc.)
B. 2	Retrieve response corresponding to "valid policy query" from Policy emulator  <ul style="list-style-type: none"> <li>OSDF Policy (POST query data)</li> </ul>	[Policy] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payloads and Endpoint	Should receive response for valid policy query  TODO: Payloads
B. 3	<del>CHECK-REQ-OR-OPTIONAL</del> <del>OSDF Policy (bad result)</del> <del>OSDF HAS (GET; bad statue)</del>  Moved to another cell <del>OSDF HAS (GET; solution found)</del>	[Policy Emulator] [OOF HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payloads and Endpoint	<del>Should receive corresponding responses</del>  <del>TODO: Payloads</del>  NOTE: Per comment and discussion with Ramki, removed tests for "malformed" requests (open to adding them later on as needed). Moved the remaining one to a separate cell.  TODO: Retain this cell till 19 Feb 2018 and then remove it.
B. 4	Retrieve response corresponding to a decision from Conductor (i.e. "done" with either a solution found or no solution found):  OSDF HAS (GET; solution found OR no solution found).  Since we cannot guarantee whether a solution can be found (it is dependent on dynamic state of the cloud instance), it may be better to merge it to a "solution found OR no solution found" – i.e. Conductor is done processing and gave a decision	[Policy Emulator] [OOF-HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payloads and Endpoint	TODO: Check format of response and valid status messages
<b>C: Run Complete Requests for Different Applications</b>				
C. 1	SO OSDF HAS (well formatted request)	[Policy Emulator] [OOF-HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payloads, Endpoint, and Call-Back URL	Should receive a valid Conductor reponse  TODO: Payloads
C. 2	<del>CHECK-REQ-OR-OPTIONAL</del> <del>SO-OSDF HAS (mal-formatted request or a data error so that the request goes through OSDF but fails at Conductor)</del>	[Policy Emulator] [OOF HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	<del>SIMPLE-GET-POST-TO-EMULATORS-OR-SERVICES</del>  TODO: Payloads, Endpoint, and Call-Back URL	<del>Should receive a RequestError or an error from Conductor</del>  <del>TODO: Payloads</del>  NOTE: Per comment and discussion with Ramki, removing this cell  TODO: Retain this cell till 19 Feb 2018 and then remove it.
C. 3	SO OSDF HAS OSDF Call Back URL  A valid request sent from SO to OSDF, which results in a valid template sent from OSDF to HAS. OSDF will then poll HAS till a decision is made (i.e. "done" with either a solution found or no solution found; it is probably difficult to ensure a solution is guaranteed – it is great if a solution is found, and it is OK for testing purposes even if there if no solution in some cases)	[Policy Emulator] [OOF-HAS API – container or emulator] [SO-OOF-OSDF call-back receiver]  <del>EMULATORS-OR-SERVICES-ARE-UP</del>	[Policy Emulator] [OOF-HAS API – container or emulator] <del>EMULATORS-OR-SERVICES-ARE-UP</del>	Should receive a "done" type Conductor reponse (either successful in finding a solution or failed to find a solution, but Conductor made a decision either way)  TODO: Payloads

## Example Request/Response Payloads for OOF-OSDF Functional Test Cases

### SO-OOF/HAS Request Example

```
{
  "requestInfo": {
    "transactionId": "xxx-xxx-xxxx",
    "requestId": "yyy-yyy-yyyy",

```

```

"callbackUrl": "https://so:5000/callbackUrl",
"sourceId": "SO",
"requestType": "create",
"numSolutions": 1,
"optimizers": ["placement"],
"timeout": 600
},
"requestParameters": { "customerLatitude": 32.89748, "customerLongitude": -97.040443, "customerName": "xyz" },
"placementDemands": [
{
"resourceModuleName": "vGMuxInfra",
"serviceResourceId": "vGMuxInfra-xx",
"tenantId": "vGMuxInfra-tenant",
"resourceModelInfo": {
"modelInvariantId": "vGMuxInfra-modelInvariantId",
"modelVersionId": "vGMuxInfra-versionId",
"modelName": "vGMuxInfra-model",
"modelType": "resource",
"modelVersion": "1.0",
"modelCustomizationName": "vGMuxInfra-customeModelName"
},
"existingCandidates": { "identifierType": "service_instance_id", "identifiers": ["87257b49-9602-4ca1-9817-094e52bc873b"] },
"excludedCandidates": { "identifierType": "service_instance_id", "identifiers": ["1ac71fb8-ad43-4e16-9459-c3f372b8236d"] },
"requiredCandidates": { "identifierType": "service_instance_id", "identifiers": ["7e6c3e57-62cd-44f6-aa88-d0896998f7ec"] }
},
{
"resourceModuleName": "vG",
"serviceResourceId": "71d563e8-e714-4393-8f99-cc480144a05e",
"tenantId": "vG-tenant",
"resourceModelInfo": {
"modelInvariantId": "vG-modelInvariantId",
"modelVersionId": "vG-versionId",
"modelName": "vG-model",
"modelType": "resource",
"modelVersion": "1.0",
"modelCustomizationName": "vG-customeModelName"
},
"existingCandidates": { "identifierType": "service_instance_id", "identifiers": ["21d5f3e8-e714-4383-8f99-cc480144505a"] },
"excludedCandidates": { "identifierType": "service_instance_id", "identifiers": ["1ac71fb8-ad43-4e16-9459-c3f372b8236d"] },
"requiredCandidates": { "identifierType": "cloud_region_id", "identifiers": ["TXAUS219"] }
}
],
"serviceInfo": {
"serviceInstanceId": "d61b2543-5914-4b8f-8e81-81e38575b8ec",
"serviceModelInfo": {
"modelInvariantId": "vCPE-invariantId",
"modelVersionId": "vCPE-versionId",
"modelName": "vCPE-model",
"modelType": "service",
"modelVersion": "1.0",
"modelCustomizationName": "vCPE-customeModelName"
}
},
"licenseDemands": [
{
"resourceModuleName": "vGMuxInfra",
"serviceResourceId": "vGMuxInfra-xx",
"tenantId": "vGMuxInfra-tenant",
"resourceModelInfo": {
"modelInvariantId": "vGMuxInfra-modelInvariantId",
"modelVersionId": "vGMuxInfra-versionId",
"modelName": "vGMuxInfra-model",
"modelType": "resource",
"modelVersion": "1.0",
"modelCustomizationName": "vGMuxInfra-customeModelName"
}
},

```

```

    "existingLicenses": {
      "entitlementPoolUUID": [ "87257b49-9602-4ca1-9817-094e52bc873b", "43257b49-9602-4fe5-9337-094e52bc9435" ],
      "licenseKeyGroupUUID": [ "87257b49-9602-4ca1-9817-094e52bc873b", "43257b49-9602-4fe5-9337-094e52bc9435" ]
    }
  }
}

```

### SO-OOF/HAS Response Example

```

{
  "transactionId": "xxx-xxx-xxxx",
  "requestId": "yyy-yyy-yyy",
  "requestState": "completed",
  "statusMessage": "Success!",
  "solutions": {
    "placementSolutions": [
      {
        "resourceModuleName": "vGMuxInfra",
        "serviceResourceId": "some_resource_id",
        "identifierType": "service_instance_id",
        "identifier": "1ac71fb8-ad43-4e16-9459-c3f372b8236d",
        "assignmentInfo": [
          { "key": "cloudOwner", "value": "amazon" },
          { "key": "vnfHostName", "value": "ahr344gh" },
          { "key": "isRehome", "value": "False" },
          { "key": "cloud_region_id", "value": "1ac71fb8-ad43-4e16-9459-c3f372b8236d" }
        ]
      },
      {
        "resourceModuleName": "vG",
        "serviceResourceId": "some_resource_id",
        "identifierType": "cloud_region_id",
        "identifier": "2ac71fb8-ad43-4e16-9459-c3f372b8236d",
        "assignmentInfo": [
          { "key": "cloudOwner", "value": "amazon" },
          { "key": "cloud_region_id", "value": "1ac71fb8-ad43-4e16-9459-c3f372b8236d" }
        ]
      }
    ],
    "licenseSolutions": [
      {
        "resourceModuleName": "vGMuxInfra",
        "serviceResourceId": "some_resource_id",
        "entitlementPoolUUID": [ "1ac71fb8-ad43-4e16-9459-c3f372b8236d", "834fc71fb8-ad43-4fh7-9459-c3f372b8236f" ],
        "licenseKeyGroupUUID": [ "1ac71fb8-ad43-4e16-9459-c3f372b8236d", "834fc71fb8-ad43-4fh7-9459-c3f372b8236f" ],
        "entitlementPoolInvariantUUID": [ "1ac71fb8-ad43-4e16-9459-c3f372b8236d", "834fc71fb8-ad43-4fh7-9459-c3f372b8236f" ],
        "licenseKeyGroupInvariantUUID": [ "1ac71fb8-ad43-4e16-9459-c3f372b8236d", "834fc71fb8-ad43-4fh7-9459-c3f372b8236f" ]
      }
    ]
  }
}

```

## Test Planning for OOF Homing and Allocation Service (OOF-HAS)

All Functional Test Cases described here below will be automatized in the CSIT ONAP integration environment. OOF-HAS is a data driven component, this means that test cases and related results have dependency on A&AI network database content. For this reason OOF-HAS Functional Test cases divided in 2 groups according to OOF-HAS functionality definition status and dependency from A&AI: Test cases marked in Green are those that will be delivered as first set of Functional Test cases as they have limited dependencies on A&AI data set, those marked in white will be scoped by ONAP Beijing delivery final test steps where all components have better stability in the scope of Beijing release.

Note: for the moment we consider the whole OOF component as the contribution of 2 Docker Containers:

- OSDF : handling “R” interface, which is invoked by SO
- OOF-HAS: handling the internal “R” interface, which in invoked by OSDF

Id	Description	Pre-conditions	Test Steps	Expected Results	Status
N. 1	<b>Name: Verify docker Containers are up and running</b>	1. MUSIC (real ONAP) docker image is up and running 2. OOF-HAS docker image is up and running	Robot Framework is checking with "docker ps" command that all needed docker containers are up and in execution	N. 4 Docker Containers for music are Up and running (music-db, music-zk, music-war, music-tomcat) N.5 Docker Containers for OPTF-HAS are up and running (cond-api, cond.solv, cond-cont, cond-data, cond-resv)	Implemented
N. 2	<b>Name: OOF-HAS Get root</b> Interface (R'). Perform GET on root "/" url	1. OOF-HAS docker image is up and running	Robot Framework is sending a REST call to OOF-HAS API – "/"  Method - GET  <b>Endpoint: http://\$(hostname):8091/</b>	OOF-HAS should respond with HTTP 200 and body containing "true"	Implemented
N. 3	<b>Name: OOF-HAS Healthcheck</b> Interface (R').  Perform healthcheck for OOF-HAS using <b>Healthcheck</b> REST API	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. Music is prepopulated with Healthcheck row	Robot Framework is sending a Rest Call to MUSIC to Inject a Plan named "healthcheck"  Method - PUT  Endpoint: /MUSIC/rest/v2/ /keyspaces/conductor/tables/plans/ /rows?id=healthcheck	MUSIC should respond with HTTP 200	Implemented
			Robot Framework is sending a REST call to OOF-HAS API – healthcheck  Method - GET  <b>Endpoint: http://\$(hostname):8091/v1/plans/healthcheck</b>	OOF-HAS should respond with HTTP 200 and body containing "true"	Implemented
N. 4	<b>Name: OOF-HAS Wrong Version</b> Interface (R').  Perform sanity sending a plan with <b>wrong Version</b>	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. Music is prepopulated with Healthcheck row	Robot Framework is sending a REST call to OOF-HAS API – to Post a Plan  Method - POST  Endpoint: http://\$(hostname):8091/v1/plans	OOF-HAS should respond with HTTP 201 and body containing the plan acceptance (i.e. the plan is in "template" status and a unique identifier <planid> is returned)	Implemented
			Robot Framework is sending a REST call to OOF-HAS API – to GET a final recommendations  Method - GET  Endpoint: http://\$(hostname):8091/v1/plans/<planId>	OOF-HAS should respond with HTTP 200 and body containing "the error reason"	Implemented
N. 5	<b>Name: OOF-HAS Missing Demand Section</b> Interface (R').  Perform Sanity sending a plan with <b>missing Demand Section</b>	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. Music is prepopulated with Healthcheck row	Robot Framework is sending a REST call to OOF-HAS API – to Post a Plan  Method - POST  Endpoint: http://\$(hostname):8091/v1/plans	OOF-HAS should respond with HTTP 201 and body containing the plan acceptance (i.e. the plan is in "template" status and a unique identifier <planid> is returned)	Implemented
			Robot Framework is sending a REST call to OOF-HAS API – to GET a final recommendations  Method - GET  Endpoint: http://\$(hostname):8091/v1/plans/<planId>	OOF-HAS should respond with HTTP 200 and body containing "the error reason"	Implemented
N. 6	<b>Name: OOF-HAS Wrong Constraint</b> Interface (R').  Perform sanity sending a plan with <b>wrong Constraints</b>	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. A&AI simulator docker image is up and running and it is populated in such a way that OOF cache can be built	Robot Framework is sending a REST call to OOF-HAS API – to Post a Plan  Method - POST  Endpoint: http://\$(hostname):8091/v1/plans	OOF-HAS should respond with HTTP 201 and body containing the plan acceptance (i.e. the plan is in "template" status and a unique identifier <planid> is returned)	Implemented

			Robot Framework is sending a REST call to OOF-HAS API – to GET a final recommendations  Method - GET  Endpoint: http://\$(hostname):8091/v1/plans/<planId>	OOF-HAS should respond with HTTP 200 and body containing "the error reason"	Implemented
N. 7	<b>Name: OOF-HAS Correct plan no result</b>  Interface (R').  Send a correct plan <b>requiring Optimization request</b> for a set of Candidates and constraints that cannot be satisfied	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. A&AI simulator docker image is up and running and it is populated in such a way that OOF cache can be built and that a set of recommendations can be returned	Robot Framework is sending a REST call to OOF-HAS API – to Post a Plan  Method - POST  Endpoint: http://\$(hostname):8091/v1/plans	OOF-HAS should respond with HTTP 201 and body containing the plan acceptance (i.e. the plan is in "template" status and a unique identifier <planId> is returned)	Implemented
			Robot Framework is sending a REST call to OOF-HAS API – to GET a final recommendations  Method - GET  Endpoint: http://\$(hostname):8091/v1/plans/<planId>	OOF-HAS should respond with HTTP 200 and body containing NO recommendations (i.e. the plan is in "not found" status and no resources are returned back)	Implemented
N. 8	<b>Name: Correct Plan with recommendations</b>  Interface (R').  Send a plan <b>requiring Optimization request</b> for a set of Candidates	1. OOF-HAS docker image is up and running 2. OOF-HAS configuration is performed 3. MUSIC (real ONAP) docker image is up and running 4. A&AI simulator docker image is up and running and it is populated in such a way that OOF cache can be built and that a set of recommendations can be returned	Robot Framework is sending a REST call to OOF-HAS API – to Post a Plan  Method - POST  Endpoint: http://\$(hostname):8091/v1/plans	OOF-HAS should respond with HTTP 201 and body containing the plan acceptance (i.e. the plan is in "template" status and a unique identifier <planId> is returned)	Implemented
			Robot Framework is sending a REST call to OOF-HAS API – to GET a final recommendations  Method - GET  Endpoint: http://\$(hostname):8091/v1/plans/<planId>	OOF-HAS should respond with HTTP 200 and body containing the plan recommendations (i.e. the plan is in "done" status and a set of recommendations are returned to the caller )	Implemented

## Appendix A: Overview of ONAP Testing Requirements

TODO

## Appendix B: Overview of OOF Scope

TODO

### Overview of OOF-OSDF Scope

#### General Description

The OOF-OSDF is meant to provide an environment for creating policy-driven optimization applications in a declarative manner easily. It also provides an execution environment for these models to be interpreted and run. Additionally, it supports external, custom optimizers such as the HAS application by providing various levels of functionality to the optimization applications. For example, the OSDF may fetch and translate policies for HAS, or it may fetch policies and data for another application.

#### Technical Description for OOF-OSDF Functionality related to OOF-HAS

The OOF-OSDF is provides the following functionality to support OOF-HAS:

1. Provide an end point for SO to make homing requests
2. Ensure authentication and validate the incoming request payload based on a model (Python Schematics model based on the SO-OOF API)
3. Fetch policies relevant to the SO's request (e.g. based on specific use case such as vCPE) and ensure that the policies are valid (well formed and contain required attributes)
4. Send response to SO that the request is accepted and is in processing (or send an error response)
5. Create a "template" (request payload) for OOF-HAS and submit the request to OOF-HAS
6. Periodically poll OOF-HAS for request processing status and optimization solution (with a configurable timeout) and validate the response based on a model (Python Schematics model)
7. Post the optimization solution to the call-back URL specified in the request from SO in the format defined by SO-OOF API (or send an error response)

### Overview of OOF-HAS Scope

## General Description

TODO

## Technical Description for OOF-HAS Functionality

TODO

## Appendix C: Resources and Links

1. Notes on creating a CSIT test script: <https://wiki.onap.org/display/DW/Creating+a+CSIT+Test>
2. [Policy Team's CSIT Functional Test Cases](#) by Pamela Dragosh. The OOF-OSDF test cases are adapted from that page.
3. Slides on Platform Maturity Requirements for Beijing Release: <https://wiki.onap.org/download/attachments/16002054/Platform%20Maturity%20Level%20proposal%2013Dec2017v2.pdf?version=1&modificationDate=1513625784000&api=v2>
4. Current Individual Project Commitment for supporting Platform Maturity Requirements for Beijing Release: <https://wiki.onap.org/display/DW/Beijing+Release+Platform+Maturity>
5. ONAP 4 level CI/CD architecture: [Integration \(5/11/2017\)](#)