### Six System Level Layer-2 Service Provisioning Tests

AW-SDX software system consists of multiple distributed components across three layers: OXP, middleware, and user interface. The distributed and microservice based system architecture allows the three layers to evolve independently. The end-to-end test workflow in the testing environment consists of three layer level tests, two cross-layer integration tests, and the final system test, as depicted in the following figure.



Before new deployment, the AW-SDX software system has to validate the services on an end-to-end basis that are specified in the next section. The whole test suite consists of six types of tests in three categories.

#### 1.1 Individual Layer tests:

##### 1.1.1 OXP Layer test:

Using the hand crafted inputs (OXP Test Input) to the OXP provisioning system interface (Rest API, etc) to validate the end-to-end services in the data plane.

FIU input: the vlan ranges on the two ports on an inter-domain link should be the same (pre-agreed upon by the admin). So (1) need a validation check when adding topologies on this; (2) vlan translation happens in a domain. vlan assignment becomes simpler after the path is obtained.  
An manual configuration example over 3 domains (2 inter-domain links) from Mert/Itlo: the vlan path: 201-202-202-203-203-201.

Amlight domain

curl -H 'Content-type: application/json' -X POST $AMLIGHT/api/kytos/mef\_eline/v2/evc -d '{"name": "AMLIGHT\_vlan\_201\_202\_Ampath\_Tenet", "dynamic\_backup\_path": true, "uni\_a": {"tag": {"value": 201, "tag\_type": 1}, "interface\_id": "aa:00:00:00:00:00:00:03:50"}, "uni\_z": {"tag": {"value": 202, "tag\_type": 1}, "interface\_id": "aa:00:00:00:00:00:00:01:40"}}'

SAX domain

curl -H 'Content-type: application/json' -X POST $SAX/api/kytos/mef\_eline/v2/evc -d '{"name": "SAX\_vlan\_202\_203\_Ampath\_Tenet", "dynamic\_backup\_path": true, "uni\_a": {"tag": {"value": 202, "tag\_type": 1}, "interface\_id": "dd:00:00:00:00:00:00:04:40"}, "uni\_z": {"tag": {"value": 203, "tag\_type": 1}, "interface\_id": "dd:00:00:00:00:00:00:05:41"}}'

TENET domain

curl -H 'Content-type: application/json' -X POST $TENET/api/kytos/mef\_eline/v2/evc -d '{"name": "TENET\_vlan\_201\_203\_Ampath\_Tenet", "dynamic\_backup\_path": true, "uni\_a": {"tag": {"value": 203, "tag\_type": 1}, "interface\_id": "cc:00:00:00:00:00:00:07:41"}, "uni\_z": {"tag": {"value": 201, "tag\_type": 1}, "interface\_id": "cc:00:00:00:00:00:00:08:50"}}'

##### 1.1.2 Middleware Layer test:

Using the AW-SDX Service data model (Service Test Input in the format of JSON) to the SDX-Controller service endpoint to validate if the middleware can satisfy the service request and generate the breakdowns as the input to the mock OXP systems.

<https://github.com/atlanticwave-sdx/sdx-controller/tree/main/tests/data>

##### 1.1.3 UI Layer Test:

Using the Meican GUI to validate if it can generate the service data model as the Input to the mock SD-Controller.

#### 

#### 1.2 Cross-layer Integration Tests

##### 1.2.1 Middleware-OXP cross-layer test.

###### 1.2.1.1 Topology publication and update.

The supported OXP system, according to the data model specification, needs to be able to (1) publish the original OXP topology to SDX-LC and (2) update the topology, in JSON format via the SDX-LC APIs, to SDX-LC, who will publish the information into the AW-SDX Message Queue, which will be received by the AW-SDX Controller.

###### 1.2.1.2 Service provisioning

The user is able to send a service request, in JSON, to the AW-SDX API, who will compute the path(s), break down the results, and send the per-OXP segments to the corresponding SDX-LC, who will format the per-OXP requests to the OXP provisioning system APIs.

##### 1.2.2 Middleware-UI cross-layer test

###### 1.2.2.1 Topology publication and update.

Meican is able to pull and render the OXP system topology via the AW-SDX Controller Rest APIs.

###### 1.2.2.2 Service provisioning

Meican is able to format the send the service request to the AW-SDX Controller Rest APIs.

#### 1.3 End-to-end System Test

See real traffic exchanges in the simulated data plane network in Mininet.

### Test Cases: Service Data Model as Input to AW-SDX

#### 2.1 Layer 2 VPN Service Model

##### 2.1.1 Point-to-point Virtual Private Wire Services (VPWSs)

##### 2.1.2 Multipoint Virtual Private LAN Services (VPLSs)

#### 2.2 Layer 3 IP VPN Service Model

A collection of sites that are authorized to exchange traffic between each other over a shared IP infrastructure.

### Test Environment.

The AW-SDX Testing Environment is created in RENCI Cloud where multiple VMs are provisioned to emulate the hosts of the distributed AW-SDX OXP middleware components, the supported OXP provisioning systems, and a mininet setting simulating the data plane.