ALTO: Beryllium: Release Plan

Contents

Introduction

In the Beryllium release, the ALTO project is planning to accomplish the following tasks:

- Migrate all AD-SAL components to MD-SAL
- Rewrite the YANG model for ALTO services
- Introduce a new framework for *Endpoint Cost Service*
- Flexible service management and dynamic IRD generation
- (Optional) Implement practical services for OpenFlow scenario
- (Optional) Implement the *Routing Service Abstraction* service
- (Optional) Implement practical services using BGP information

Release Deliverables

Name	Description
alto-core	• Define the new YANG model for basic types used in ALTO • Define the models for ALTO services • Define the top-level container for ALTO services • Provide the RESTful API defined in RFC7285
alto-basic	• Provide the basic *Information Resource Directory* (IRD) service • Provide *Network Map* and *Cost Map* services that are manually configurable • Provide *Endpoint Cost Service* based on host tracker

Release Milestones

- **Offset:** 2

Milestone	Offset X Date	Deliverables	
		Name	**Description**
M1	August 6th, 2015	Release Plan	See Beryllium Release Plan for ALTO
		Deliverable Name	See Release Deliverables
M2	September 3th, 2015	**Name**	**Description**
		Release Plan	See Beryllium Release Plan for ALTO
		Deliverable Name	See Release Deliverables

M3	October 15th, 2015	**Name**	**Description**
		Feature Freeze	
		Deliverable Name	Deliverable Description
M4	November 12th, 2015	**Name**	**Description**
		API Freeze	
		Deliverable Name	Deliverable Description
M5	December 17th, 2015	**Name**	**Description**
		Code Freeze	
		Deliverable Name	Deliverable Description
RC0	TBD	**Name**	**Description**
		Deliverable Name	Deliverable Description
RC1	TBD	**Name**	**Description**
		Deliverable Name	Deliverable Description
RC2	TBD	**Name**	**Description**
		Release Review	Release Review Description
		Deliverable Name	Deliverable Description
Formal Release	TBD	**Name**	**Description**
		Deliverable Name	Deliverable Description

Externally Consumable APIs

- Please list and describe all externally consumable APIs or point to a document that does so.
- This need not be a list of all java interfaces and yang files, but should be a list of high-level functionality, e.g., flow programming.
  - For each such API, there should be a list of the bundles providing the API, but not necessarily the implementation.
- Each PROVISIONAL or TENTATIVE API must be listed as project deliverable
  - Must be declared as "IN" or "OUT" at the M3 Milestone

Short Name	Description	Type (at M2)	Type (at M3)	Type (release)	Contract	Supporting Code
API Name	Short Description	One of Provisional, Tentative, Stable, or Dropped as defined in the Beryrllium release plan definitions.			link to the Java interface, YANG file, WSDL description, etc. that defines the API	list of Karaf features, OSGi bundles, directories, etc. that provide the API

Expected Dependencies on Other Projects

Providing Project	Deliverable Name	Needed By	Acknowledged?	Description
L2Switch	l2switch-hosttracker	M4	Yes	Host Tracker from L2Switch project

Expected Incompatibilities with Other Projects

Please note any known or expected incompatibilities with other projects. For example, the different projects providing Neutron APIs have historically been incompatible with each other. For each incompatibility:

1. Note which expected features are incompatible if known
   a. Why
   b. Whether discussions occurred with the projects of the incompatible features as to how to become compatible.

i. The results of those discussion.

If the incompatibility is expected, but the features are not yet know, please provide as much as is known or expected.

Projects are encouraged to engage with other projects to discuss and explore ways this incompatibility can be avoided either during this release or in a future release.

Compatibility with Previous Releases

Since the northbound API follows the ALTO protocol defined in 7285, thus it is fully compatible with the one in the previous release.

All YANG models and interfaces

Removed APIs and/or Functionality

- Please include list of a APIs and/or functionality that existed the previous release and will be removed.
  - For each such API/functionality, discuss any suggestions for how those who are using it should adapt.
  - In order for an API/functionality to be removed, it must have been deprecated in a previous release.
    - Functionality and/or APIs that were correctly tagged with the @Deprecated annotation in Java before the Lithium release can be considered deprecated and thus removed in Beryllium.

Deprecated APIs and/or Functionality

- Please include list of a APIs and/or functionality that existed the previous release and will be deprecated.
  - For each such API/functionality, discuss any suggestions for how those who are using it should adapt.
  - If possible, e.g., for all human-generated Java, add the @Deprecated annotation
  - If not possible, please note clearly things which are deprecated in a clearly visible manner

Changed APIs and/or Functionality

- Please include list of a APIs and/or functionality that existed the previous release and will be changed.
  - For each such API/functionality, provide guidance about who will be affected and how they should adapt.
  - In general, project's should strive to be backward compatible with the previous release and note what functionality will be removed by deprecating it and noting that with the @Deprecated annotations wherever possible.

Themes and Priorities

Requests from Other Projects

For each API request, the requesting project should create an entry like the example below.

Requesting Project	API Name	Needed By	Acknowledged?	Description
XYZ Project	call_me	M4	No	This is an example to request API supported

After creating the entry, the requesting project should send an e-mail to release@lists.opendaylight.org, and both projects' dev lists using this template:

```
Subject: [REQUEST FOR NEW OR EXTENDED API] ${API name}

Note: This email is a request from ${requesting project} for a new or
extended API in ${providing project}.

API Name: ${API name}
Request: ${link to the request in the providing project's release plan}

Please let us know if you will be able to provide this new
functionality by the listed milestone. If you need clarifications or
help in providing the API, let us know so we can reach an agreement.

If you feel that providing this API is a bad idea regardless of where
the resources are coming from, please let us know why and ideally,
suggest and alternative.
```

Example Request

- **Requesting Project:**
- **Providing Project:** <should be this project, i.e., the project whose release plan this page is>
- **Requested Deliverable Name:**
- **Needed Milestone:** <e.g., M3, assumed to be at that milestone at the offset of the providing project unless otherwise stated>
- **Requested Deliverable Description:** <description of the change or new API needed>
- **Response:** <one of Yes, No, Tentative>

    - **Description:** <provides details of the response, e.g., "Yes, we can do that", "No, that's a bad idea", "No, we don't have the staff", "Tentaive, it's a good idea, but we don't know if we can">
    - **Resources From:** <should be either the providing project if they agree to do the work or the requesting project if the agree>
    - **Link to Section in Requesting Project Release Plan:** <if No or Tentative, link to the adjustments in the requesting project's release plan>
    - **Link to Section in Providing Project Release Plan:** <if Yes or Tentative, link to the deliverable in this release plan>
- **Negotiation:**
    - <this should be a back and fort until a response of Yes, No, or Tentative is reached>
    - <it should probably start with a response from the providing project that can be one of Yes, No, or Tentative, OR asking for clarification>
    - <after that it should go back and forth reaching a conclusion and then documented in the Response and Description above>
    - <if the negotiation fails or stalls, the TSC will do it's best to help>

Test Tools Requirements

- Please specify if the project will run System Test (ST) inside OpenDaylight cloud
- In case affirmative please enumerate any test tool (mininet, etc...) you think will be required for testing your project
    - The goal is to start test tools installation in rackspace as soon as possible
- In case negative be aware you will be required to provide System Test (ST) reports upon any release creation (weekly Release, Release Candidate, Formal Release, etc...)

Other