



FRINX

THE OPEN SOURCE NETWORK

FRINX VISION STATEMENT

“Deliver real and sustainable productivity gain by automating processes required to build, operate and grow communication networks.”

FRINX TYPICAL USE CASES

- Overall Network Device Automation, including but not limited to
 - LACP link bundles
 - BGP peering services
 - Business internet
 - EVPN
 - L2VPN (VLL, VPLS)
 - L3VPN
 - LLDP topology collection services
 - DOCSIS
 - Amazon VPCs & Direct connect (interface & peering)
- Network Inventory and Change Management
 - Network inventory (heterogeneous platform data transformed to common data model)
 - Operating software management
 - Device configuration
- Workflow management;
 - Network devices
 - Customer care tools
 - Subscriber provisioning tools
 - Billing systems

FRINX STRATEGY

- Develop a controller and agent to connect to customer network devices and provide an upstream network API for our customers.
- Use cloud native software architectures to provide workflow and inventory solutions that control one or many customers network APIs.
- Develop a thriving community to grow our open source device library supporting all device vendors.
- Partner with industry leaders to deliver a multi-tenant, massively scalable cloud based platform for communications and connectivity service providers.

CODE THAT WE WORK ON

- FRINX UniConfig, with FRINX ODL (for controllers) and Lighty.io (for agents) and deploy with large multi-national customers
- FRINX Open Source device library
- FRINX Machine, a cloud native workflow product based on Netflix Conductor and Elasticsearch
- FRINX contributions to the ODL project (NETCONF, GBP)

FRINX UNICONFIG

FRINX UNICONFIG ECOSYSTEM

FRINX



FRINX
UniConfig



Multivendor networks



CLI, NETCONF ...

FRINX UNICONFIG NETWORK AUTOMATION

FRINX

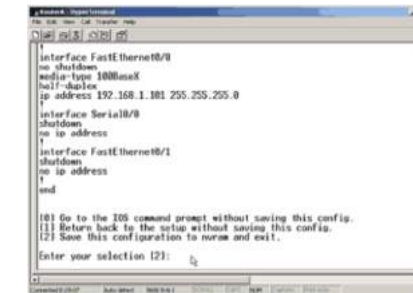


YANG

Any vendor YANG model



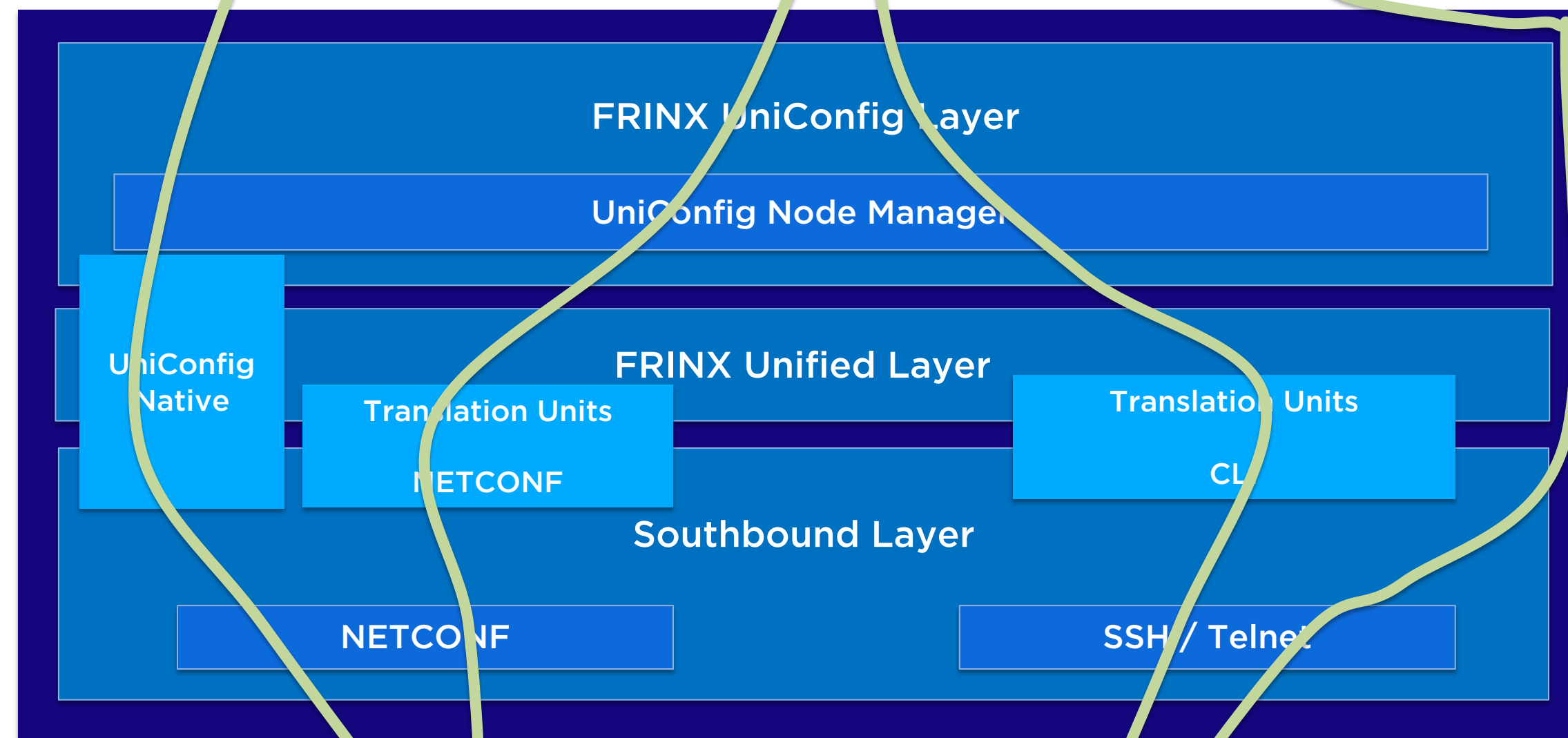
Unstructured data



FRINX
UniConfig



RESTconf or
NETCONF



Multivendor networks



CLI, NETCONF ...

FRINX MACHINE - NETWORK AUTOMATION

FRINX



FRINX Workflow & Inventory

REST Service &
Workflow APIs



FRINX UniConfig Layer

UniConfig Node Manager

UniConfig
Native

FRINX Unified Layer

Translation Units

NETCONF

Translation Units

CLI

Southbound Layer

NETCONF

SSH / Telnet



RESTconf or
NETCONF

Multivendor networks



CLI, NETCONF ...

FRINX UniConfig

Open Source Device Library

OSS DEVICE LIBRARY

- Support for stateful translation between CLI (semi-unstructured data) and YANG models
- CLI models are sequence aware. UniConfig service graph is implemented in Create, Read, Update & Delete operations for CLI configs. Required for rollback logic.
- Support for stateful translation between YANG models (e.g. private YANG models translated into OpenConfig and back)
- Stateful means that device configuration is stored in structured format (YANG) in operational and config data stores. This enables UniConfig operations (sync-from-network, diff between config and operational, commit to network, rollback & snapshots) on all device configurations that are mounted in UniConfig.
- Scales up to thousands of devices per controller (with 1000s of lines of config per device)

FRINX UniConfig

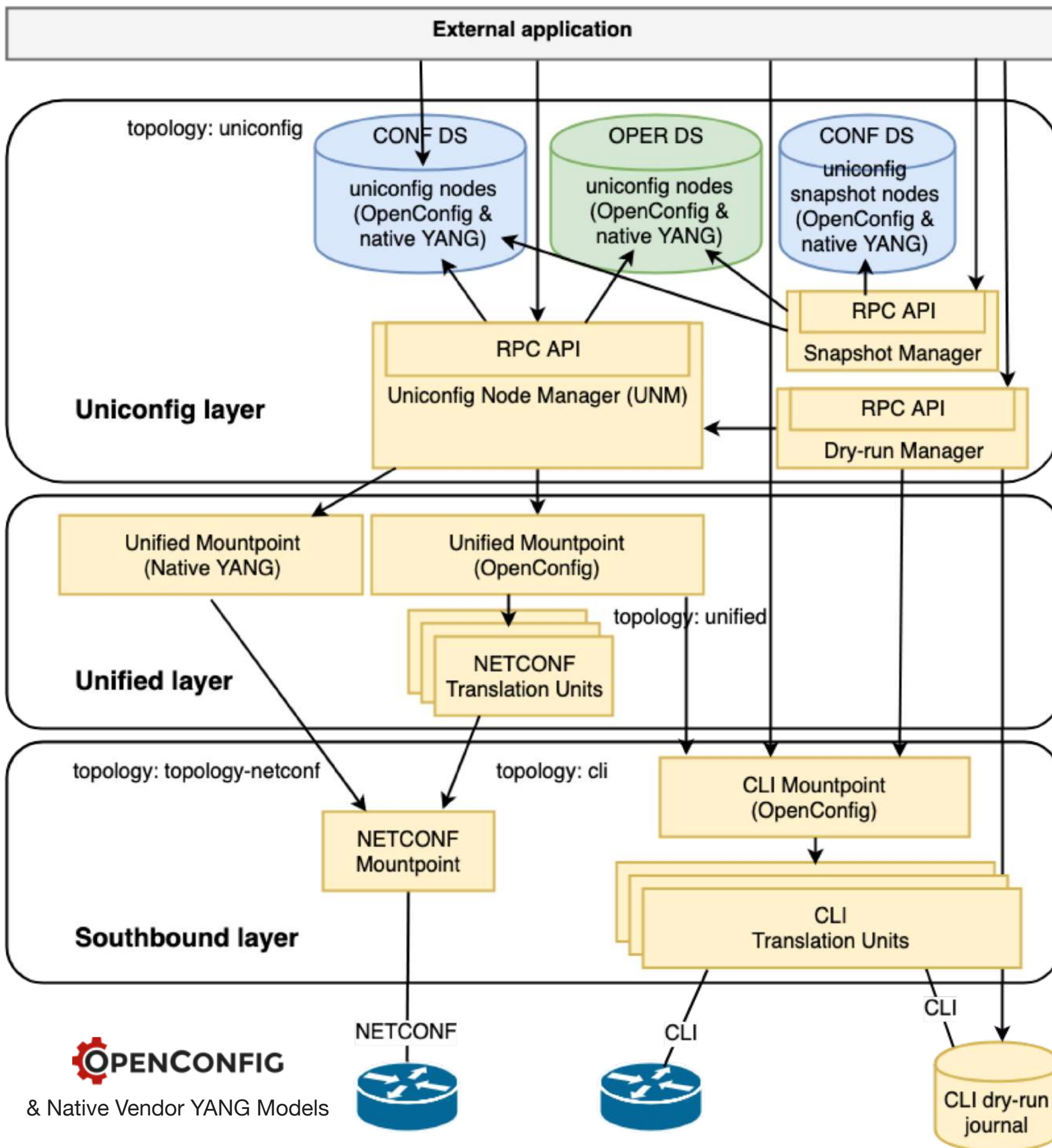
UniConfig Native

UNICONFIG NATIVE

- Read and write to and from devices using their native YANG data models (e.g. Cisco YANG models, JunOS Yang models, CableLabs YANG models ...)
- Use the same features on native YANG models as with regular UniConfig OpenConfig models (e.g. sync-from-network, commit, checked-commit, calculate-diff, replace-config-with-operational, rollback, create and load snapshots)
- Works along side UniConfig (some devices can be mounted as UniConfig native, while some devices can be mounted as UniConfig at the same time)
- Available starting with the FRINX ODL 4.2.0 release (April 2019)
- Loading and transformation of YANG models from devices happens on-the-fly. No pre-compilation required.
- Tested scale data point: 1120 devices with 4700 lines of configuration per device, require 3GB of heap

FRINX UniConfig

FRINX SOLUTIONS



FRINX UniConfig Framework – A Layered Architecture

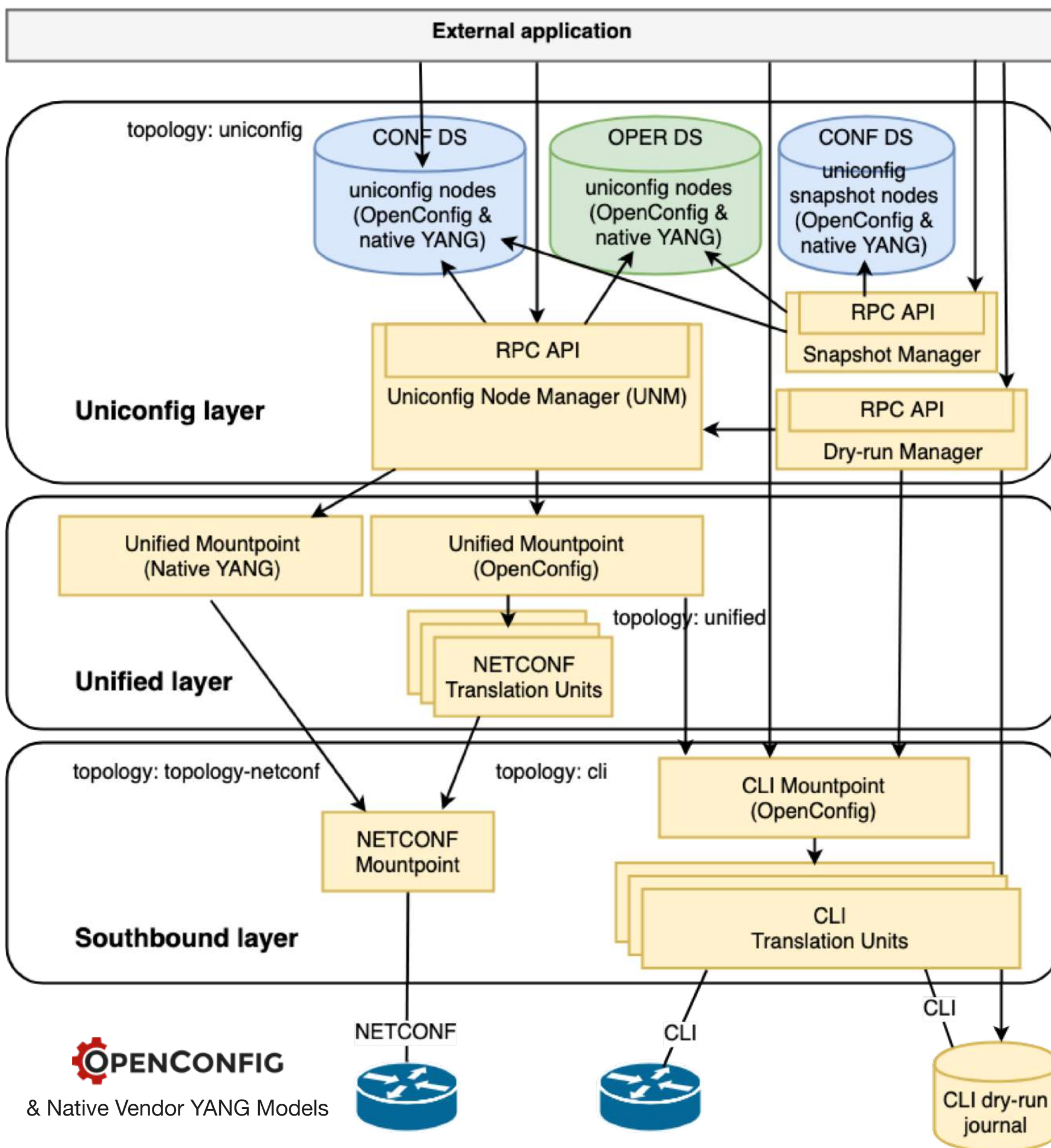
FRINX ODL service components use a layered design where the functionality of upper layers depends on the functionality of the layer underneath. Each layer thus provides a higher level of abstraction from the network elements. Applications are allowed to utilize any of the layers in the system. There are 3 main layers represented by these components (from top to bottom):

- Uniconfig layer
- Unified layer
- Southbound layer (NETCONF mountpoint, CLI mountpoint with translation units)

The Datastore is a component in ODL which stores structured data described by YANG models. There are two separate Datastores:

- Config datastore (CONF DS) - contains intended state (intended device configuration). This datastore is persistent and external (outside ODL) applications have read/write access.
- Operational datastore (OPER DS) - contains actual state (actual device configuration). OPER DS is not persistent and external applications have read only access.

Mountpoints in ODL represent communication interfaces with an external system. Mountpoints are usually registered under a node in a topology.



Business application using UniConfig

Device abstraction provides API to create, read, update and delete device configurations in common OpenConfig Format. Config data store contains intended configurations of all network devices, while operational data store contains all current configurations. Network transaction capabilities provide commit, snapshot and rollback functions across one or multiple devices.

Unified layer provides the ability to combine devices connected with different transport protocols and different models into one common representation. Includes open source device library (YANG <-> YANG) and the ability to interact with vendor YANG models (UniConfig native)

Southbound layer provides connectivity to devices via multiple protocols (NETCONF, SSH, Telnet, ...). Includes open source device library (YANG <-> CLI)

OPERATIONS

- UniConfig Manager
 - sync-from-network
 - commit
 - checked-commit
 - calculate-diff
 - replace-config-with-operational
- Snapshot Manager
 - create-snapshot
 - delete-snapshot
 - replace-config-with-snapshot
- Dry-run Manager
 - dry-run – works only with CLI nodes



FRINX *machine*

FRINX MACHINE - NETWORK AUTOMATION



Workflow & Inventory

REST Service &
Workflow APIs



RESTconf or
NETCONF

Multivendor networks



CLI, NETCONF ...

FRINX MACHINE - NETWORK AUTOMATION



Workflow & Inventory

REST Service & Workflow APIs



RESTconf or NETCONF

μ-services

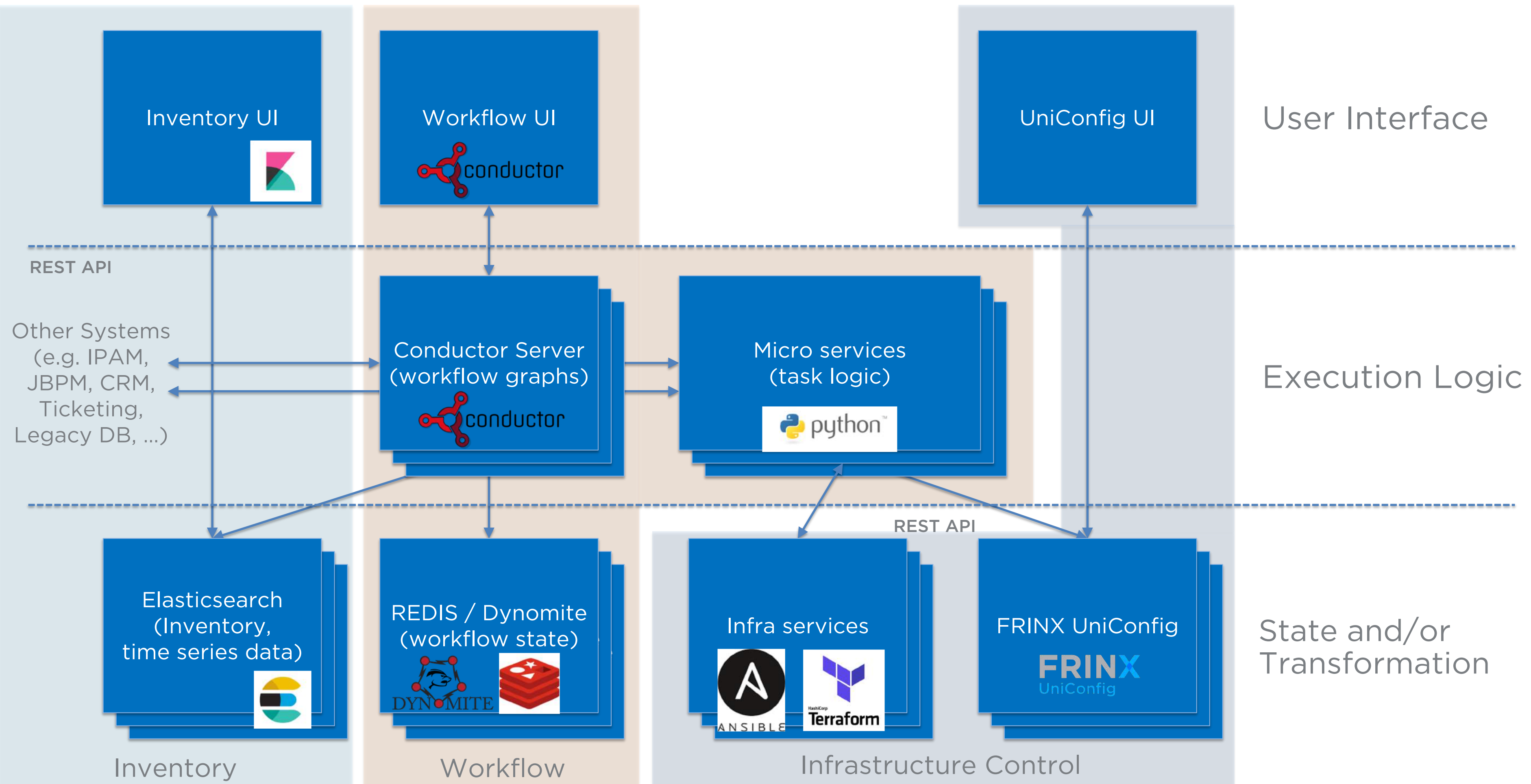
Multivendor networks



CLI, NETCONF ...

FRINX MACHINE – COMPONENTS

FRINX



Key Solution Tenets

- Use existing solutions wherever available (OpenDaylight, Elasticsearch, Conductor, Ansible, Terraform, ...)
- Provide stateful and stateless interaction with network infrastructure
- Provide a framework for components and how they interact
- Provide out-of-the-box workflows & services
- Provide open source device library
- Provide customer access to all source code
- Provide solution support for enterprises and operators
- FRINX Machine fits in 6 GB RAM / 30GB disk and installs and starts in a few minutes

FRINX MACHINE

User Interface

FRINX

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Scheduled

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

Task Queues

Poll Data

Create_L2VPN_P2P_OC_uniconfig723654 / 1 **RUNNING** Terminate Pause

Workflow ID	Owner App	Total Time (sec)	Start/End Time
e8ed94d5-4b8e-4e77-aac8-fe903e26d9ee			03/15/2019, 17:35:50:047 - 01/01/19

Execution Flow

Task Details

Input/Output

JSON

Edit input

```
graph TD; start((start)) --> task1[UNICONFIG_write_structured_device_data  
(UNICONFIG_write_structured_device_data_on_first_node)]; task1 --> task2[UNICONFIG_write_structured_device_data  
(UNICONFIG_write_structured_device_data_on_second_node)]; task2 --> task3[UNICONFIG_commit  
(UNICONFIG_commit)]; task3 --> decide1{decide1}; decide1 -- fail --> task4[http_get_generic  
(http_get_generic_instance_fail)]; decide1 -- complete --> task5[http_get_generic  
(http_get_generic_instance_complete)]; task4 --> end[ ]; task5 --> end[ ];
```

- Users can build on a library of workflows that ship with FRINX Machine to create their own automation workflows.
- Workflows can be started via REST interface or GUI.
- Workflows can be created without writing code
- Workflows can integrate with other systems (Ticketing, E-mail, Slack, ...)



Workflows

Executions

- All
- Running
- Failed
- Timed Out
- Terminated
- Completed
- Scheduled

Metadata

- Workflow Defs
- Tasks

Workflow Events

- Event Handlers

Filter Workflows

Show All

Filter

Search

Search by Workflow keyword

Name/Version	Labels	Input Parameters
Execute_and_read_inventory / 1	SOUTHBOUND, CLI, EXECUTEANDREAD, INVENTORY	
Add_nested_field_to_inventory / 1	INVENTORY	["id[Unique identifier of a command in DB]", "field[Key (identifier) of a new field to add]"]
Create_L2VPN_P2P_connection / 1	OPENCONFIG, UNICONFIG, L2VPN, CREATE	["node01[First node of P2P connection][IOS01]", "interface01[Customer facing interface]"]
Add_cli_device_to_inventory / 1	INVENTORY, CLI	["id[Unique identifier of device across all systems]", "type[Type of device or platform]"]
Get_all_devices_as_json / 1	INVENTORY	["type[Optional filtering parameter which selects devices from inventory based on type]"]
Read_components_update_inventory / 1	OPENCONFIG, PLATFORM, UNIFIED, INVENTORY	
UNICONFIG_commit / 1	UNICONFIG	
Write_structured_device_data_in_uniconfig / 1	UNICONFIG, CONFIG	
Delete_loopback_interface_uniconfig / 1	OPENCONFIG, UNICONFIG, INTERFACES, LOOPBACK	

Executions

- All
- Running
- Failed
- Timed Out
- Terminated
- Completed
- Scheduled

Metadata

- Workflow Defs
- Tasks

Workflow Events

- Event Handlers

Task Queues

- Poll Data

Inputs of **Create_L2VPN_P2P_OC_uniconfig** workflow

Create P2P L2VPN in uniconfig - OPENCONFIG, UNICONFIG, L2VPN, CREATE

node01

IOS01

First node of P2P connection

interface01

GigabitEthernet1

Customer facing service interface on first node

vcid

444

Virtual Circuit Identifier (globally unique)

node02

IOS02

Second node of P2P connection

interface02

GigabitEthernet3

Customer facing service interface on second node

Schedule workflow +

Execute workflow



FRINX

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Scheduled

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

DiagramJSONInput

EditEscape

```
{
  "updateTime": 1550748145366,
  "name": "Create_L2VPN_P2P_OC_uniconfig",
  "description": "Create P2P L2VPN in uniconfig - OPENCONFIG, UNICONFIG, L2VPN, CREATE",
  "version": 1,
  "tasks": [
    {
      "name": "UNICONFIG_sync_from_network",
      "taskReferenceName": "UNICONFIG_sync_from_network",
      "type": "SIMPLE",
      "startDelay": 0
    },
    {
      "name": "UNICONFIG_replace_config_with_oper",
      "taskReferenceName": "UNICONFIG_replace_config_with_oper",
      "type": "SIMPLE",
      "startDelay": 0
    },
    {
      "name": "UNICONFIG_write_structured_device_data",
      "taskReferenceName": "UNICONFIG_write_structured_device_data_on_first_node",
      "inputParameters": {
        "node01": "${workflow.input.node01}",
        "id": "${workflow.input.node01}",
        "interface01": "${workflow.input.interface01}",
        "VCID": "${workflow.input.vcid}",
        "uri": "/frinx-openconfig-network-instance:network-instances/network-instance/conn1233",
        "template": "{
frinx-openconfig-network-instance:network-instance": [
  {
    "name": "conn1233",
    "config": {
      "name": "conn1233",
      "type": "frinx-openconfig-network-instance-types:L2P2P"
    },
    "connection-points": {
      "connection-point": [
```



FRINX

Diagram

JSON

Input

Cancel

Save

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Scheduled

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

{

"updateTime": 1550748145366,

"name": "Create L2VPN P2P OC uniconfig",

"description": "Create P2P L2VPN in uniconfig - OPENCONFIG, UNICONFIG, L2VPN, CREATE",

"version": 1,

"tasks": [

{

"name": "UNICONFIG

"taskReferenceName"

"type": "SIMPLE",

"startDelay": 0

},

{

"name": "UNICONFIG

"taskReferenceName"

"type": "SIMPLE",

"startDelay": 0

},

{

"name": "UNICONFIG

"taskReferenceName"

"inputParameters":

"node01": "\${work

"id": "\${workflow

"interface01": "\$

"VCID": "\${workfl

"uri": "/frinx-op

"template": "\${\r\n

"params": "{}"

},

"type": "SIMPLE",

"startDelay": 0

},

{

"name": "UNICONFIG_write_structured_device_data",

"taskReferenceName": "UNICONFIG_write_structured_device_data_on_second_node",

"inputParameters": {

{

"frinx-openconfig-network-instance:network-instance": [

{

"name": "conn1233",

"config": {

"name": "conn1233",

"type": "frinx-openconfig-network-instance-types:L2P2P"

},

"connection-points": {

"connection-point": [

{

"connection-point-id": "1",

"config": {

"connection-point-id": "1"

},

"endpoints": {

"endpoint": [

{

"endpoint-id": "default",

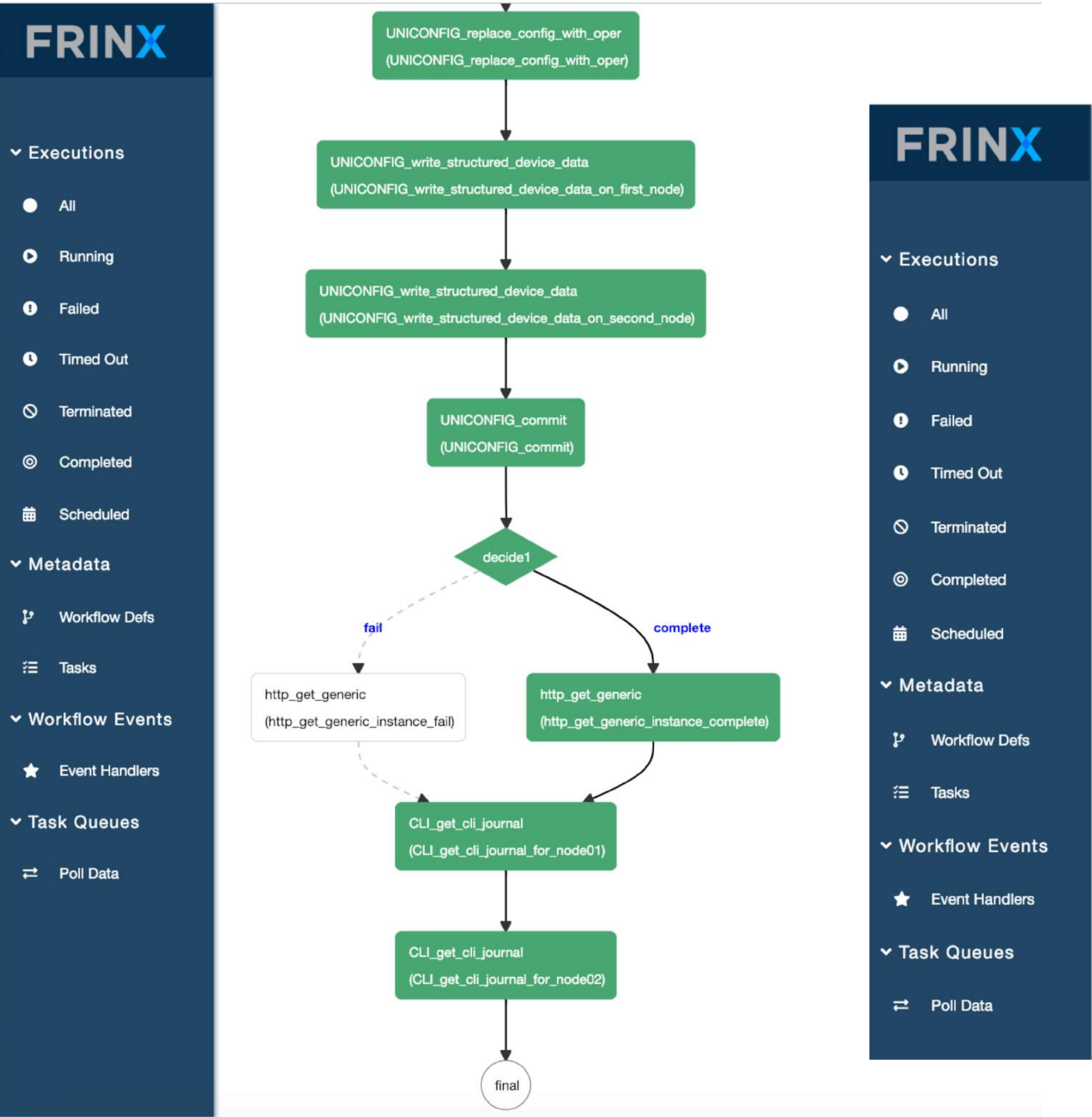
Cancel

Save

Server: http://conductor-server:8080/api/

Version: 1.10.10 | Build Date: 2018-07-14_02:31:26

FRINX Machine Workflows – Detailed Execution Information



FRINX

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Scheduled

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

Task Queues

Poll Data

Create_L2VPN_P2P_OC_uniconfig / 1 **COMPLETED**

Restart

Workflow ID	Owner App	Total Time (sec)	Start/End Time	Correlation ID
ed897a30-7e50-462c-8571-bc83b3c9f975		12.263	02/11/2019, 16:43:26:806 - 02/11/2019, 16:43:39:069	

Execution Flow

Task Details

Input/Output

JSON

Edit input

Workflow Input

Unescape

```
{  "node01": "IOS01",  "interface01": "GigabitEthernet1",  "vcid": "444",  "node02": "IOS02",  "interface02": "GigabitEthernet3"}
```

Workflow Output

Escape

```
{  "response_body": "Journal for IOS01:  {output={journal=2019-02-11T15:32:51.26: show running-config 2019-02-11T15:33:21.35: show history all | include Configured from 2019-02-11T15:33:21.446: show running-config 2019-02-11T15:35:00.07: show history all | include Configured from 2019-02-11T15:35:00.087: show running-config 2019-02-11T15:35:07.778: configure terminal pseudowire-class conn1233 encapsulation mpls ~~~~~"
```

FRINX Workflow UI provides detailed task information (input and output data per task, failure reason, time stats)



FRINX

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

Task Queues

Poll Data

INVENTORY_add_cli_device / 1 **COMPLETED**

Workflow ID

ef92d596-75a9-42c2-8049-ace6194c4954

Execution Flow

Task Details

Input/Output

JSON

#	Task Type	Task Ref. Name	Schedule
1	INVENTORY_add_device	INVENTORY_add_device	08/3

Task Details

Task Input

```
{  "id": "IOSXR",  "type": "ios xr",  "version": "6.0",  "host": "192.168.1.112",  "protocol": "ssh",  "port": "22",  "username": "cisco",  "password": "cisco"}
```

Task Output

```
{  "url": "http://elasticsearch:9200/inventory/device/IOSXR",  "response_body": {    "_type": "device",    "_shards": {      "successful": 1,      "failed": 0,      "total": 2    },    "_index": "inventory",    "_version": 1,    "created": true,    "result": "created",    "_id": "IOSXR"  },  "response_code": 201}
```

Task Failure Reason (if any)

relation ID

Details

Input/Output

9d1419c5620a

Server: http://conductor-server:8080/api/

-07-14_02:31:26

50

FRINX Workflow UI provides detailed stats about task and workflow execution and status



FRINX

Executions

All

Running

Failed

Timed Out

Terminated

Completed

Metadata

Workflow Defs

Tasks

Workflow Events

Event Handlers

Task Queues

Poll Data

INVENTORY_add_cli_device / 1 **COMPLETED**

Restart

Workflow ID	Owner App	Total Time (sec)	Start/End Time	Correlation ID
ef92d596-75a9-42c2-8049-ace6194c4954		1.007	08/31/2018, 18:30:21:122 - 08/31/2018, 18:30:22:129	

Execution Flow

Task Details

Input/Output

JSON

Edit input

{

"createTime": 1535733021122,

"updateTime": 1535733022129,

"status": "COMPLETED",

"endTime": 1535733022129,

"workflowId": "ef92d596-75a9-42c2-8049-ace6194c4954",

"tasks": [

{

"taskType": "INVENTORY_add_device",

"status": "COMPLETED",

"inputData": {

"id": "IOSXR",

"type": "ios xr",

"version": "6.0",

"host": "192.168.1.112",

"protocol": "ssh",

"port": "22",

"username": "cisco",

"password": "cisco"

}

},

"referenceTaskName": "INVENTORY_add_device",

"retryCount": 0,

"seq": 1,

"pollCount": 1,

"taskDefName": "INVENTORY_add_device",

"scheduledTime": 1535733021216,

"startTime": 1535733021762,

"endTime": 1535733022214,

"updateTime": 1535733022214,

"startDelayInSeconds": 0,

"retried": false,

"executed": true,

"callbackFromWorker": true,

"responseTimeoutSeconds": 10,

"workflowInstanceId": "ef92d596-75a9-42c2-8049-ace6194c4954",

"workflowType": "INVENTORY_add_cli_device",

"taskId": "8005a521-e54f-4835-885f-a3f935daea31",

"callbackAfterSeconds": 0

}

Server: http://conductor-server:8080/api/

Version: 1.10.10 | Build Date: 2018-07-14_02:31:26

32



Good afternoon, **Gerhard**
gwieser@frinx.io



```
frinxit$ show odl version
Success. Status code: 200
{
  "output": {
    "versions": {
      "controller-version": "3.1.7.frinx"
    }
  }
}
frinxit$ show uniconfig calculate-diff
Success. Status code: 200
No diffs between config and operational datastore.
frinxit$
frinxit$
frinxit$ show cli operational
Success. Status code: 200
Node ID           Host IP           Host Status
ASR01_CA01_SJ04   sample-topology   connected
ASR01_NJ03_ISL01   sample-topology   connected
ASR01_NJ03_ISL02   sample-topology   connected
ASR01_NJ03_MPLW07   sample-topology   connected
ASR01_NY08_BKLN01   sample-topology   connected
ASR01_NY08_BKLN02   sample-topology   connected
IOS01             sample-topology   connected
IOS02             sample-topology   connected
frinxit$ show cli operational | grep NY
ASR01_NY08_BKLN01   sample-topology   connected
ASR01_NY08_BKLN02   sample-topology   connected
frinxit$
```



For more details about Frinx please
contact frinx@frinx.io