XOS

The Service Composition and Management Layer for CORD and SEBA

Scott Baker

Managing the service stack is a big space...

- Container lifecycle management
  - Deploy on hardware
  - Scheduling
  - Redeploy/migrate if hardware fails
- Service coordination
  - Service A needs to work with Service B
  - Abstractions may span A and B

Other tools handle container lifecycles

Docker

- Enforces compute isolation between services

Kubernetes

- Deploys containers
- Schedules containers to compute resources

Helm

- “The Kubernetes Package Manager”
- Manage dependencies between services
  - Chart A is comprised of charts B, C, and D...

XOS manages and coordinates services

Unify

- Provide coherent interface to collection of disaggregated components
- Tools to avoid NxM scenario (N northside masters and M components)

Coordinate (East/West)

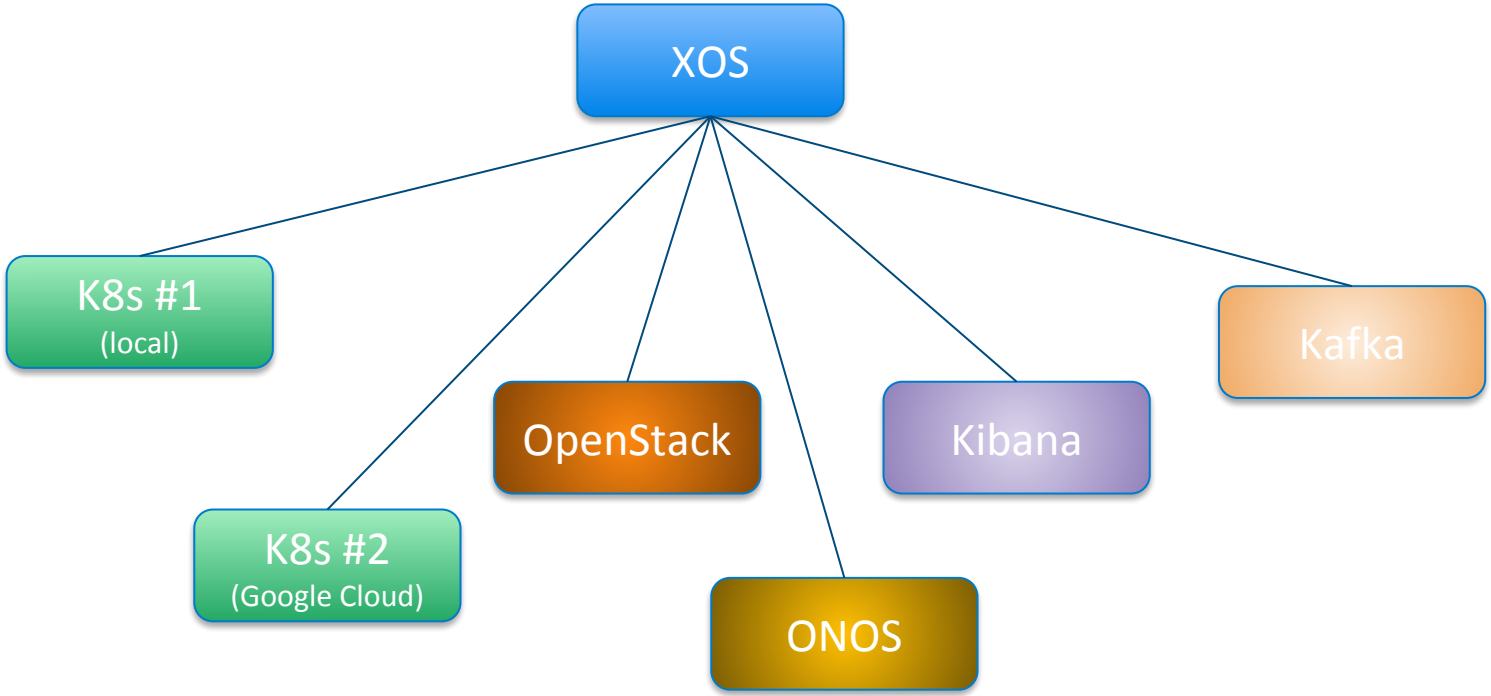
- Support low-latency/fine-grain interaction among internal components
- State needed to coordinate interdependent/adjacent components

Synchronize (North/South)

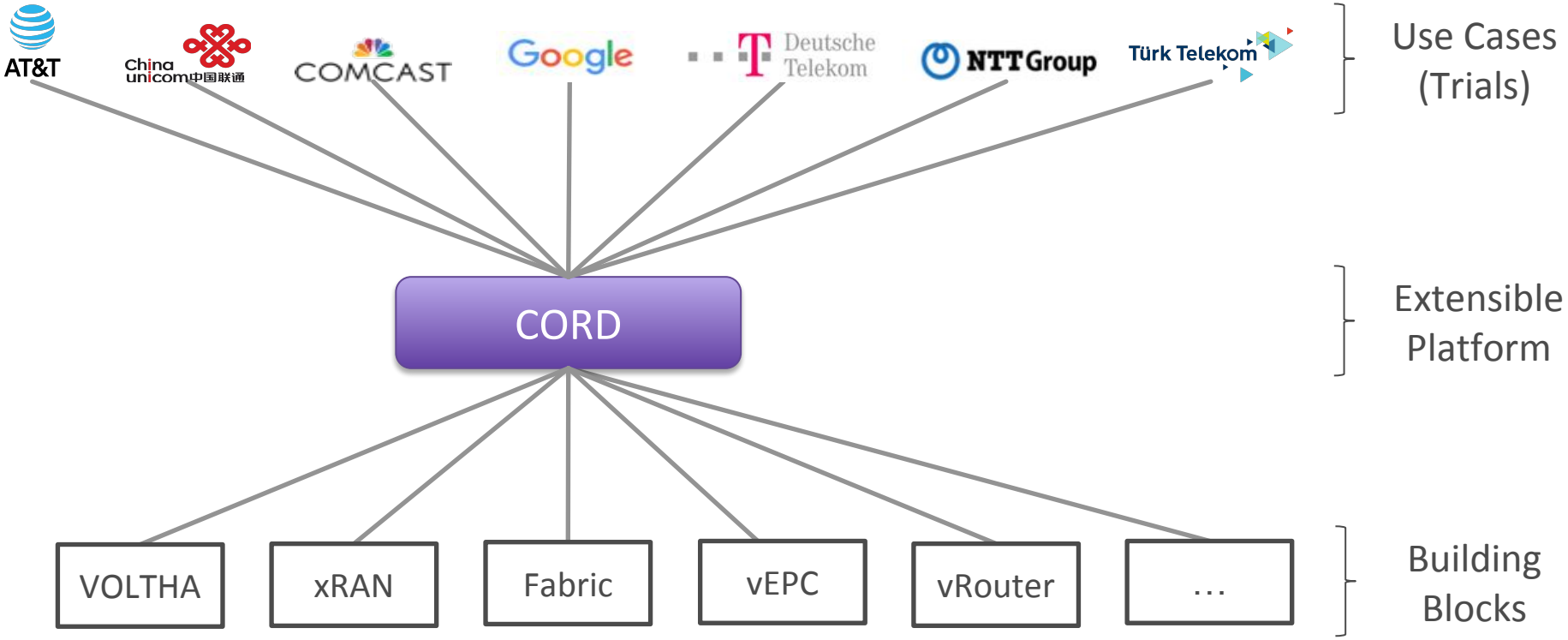
- Provide rendez-vous point for north-side directives / component activity
- State needed to synchronize both top-down and bottom-up workflows

Why invent our own?

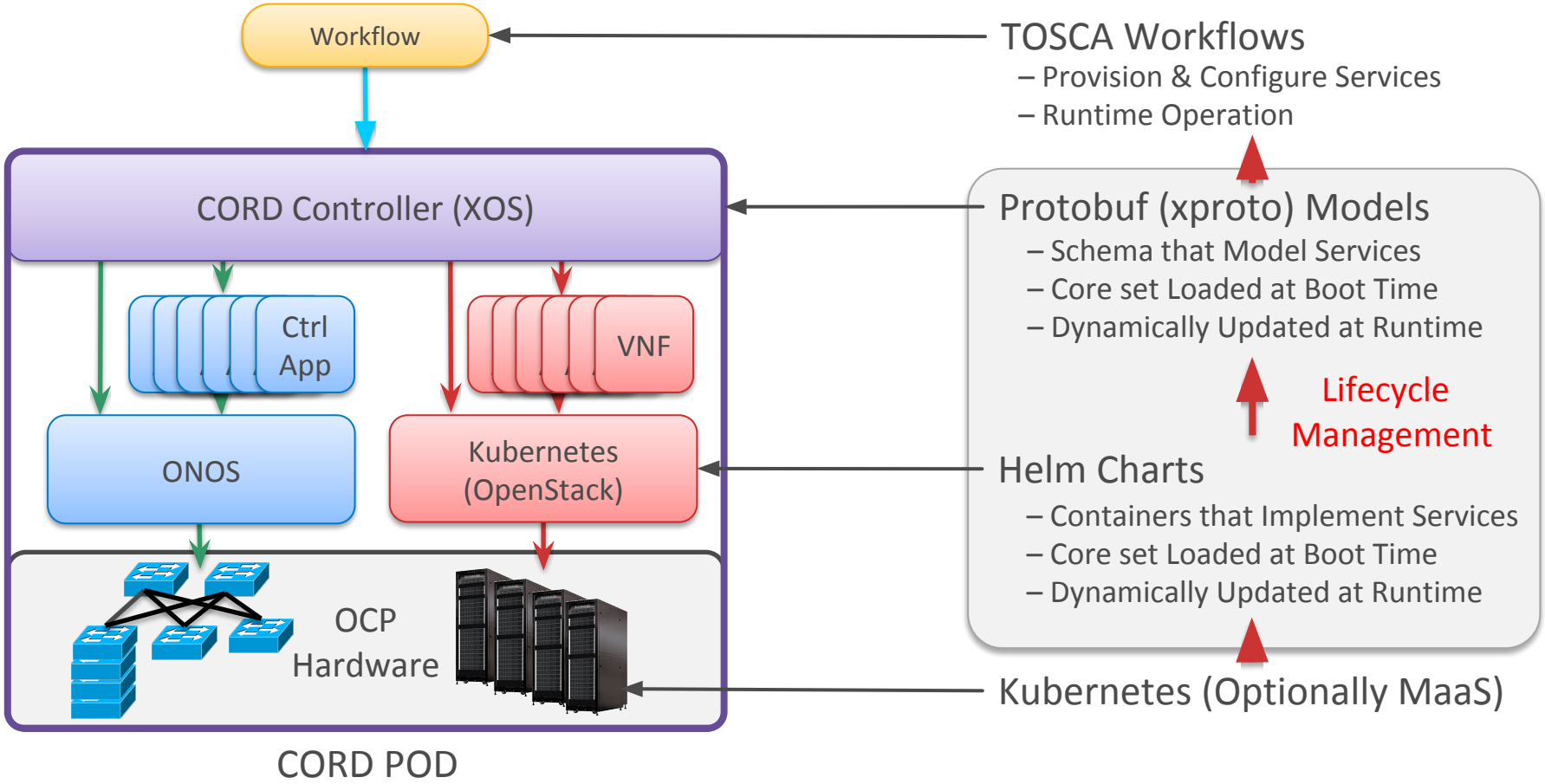
Create service meshes that span multiple disparate technologies and security domains:



Unify – Integrate Across Components



XOS Operationalizes CORD



Unification using the XOS data model

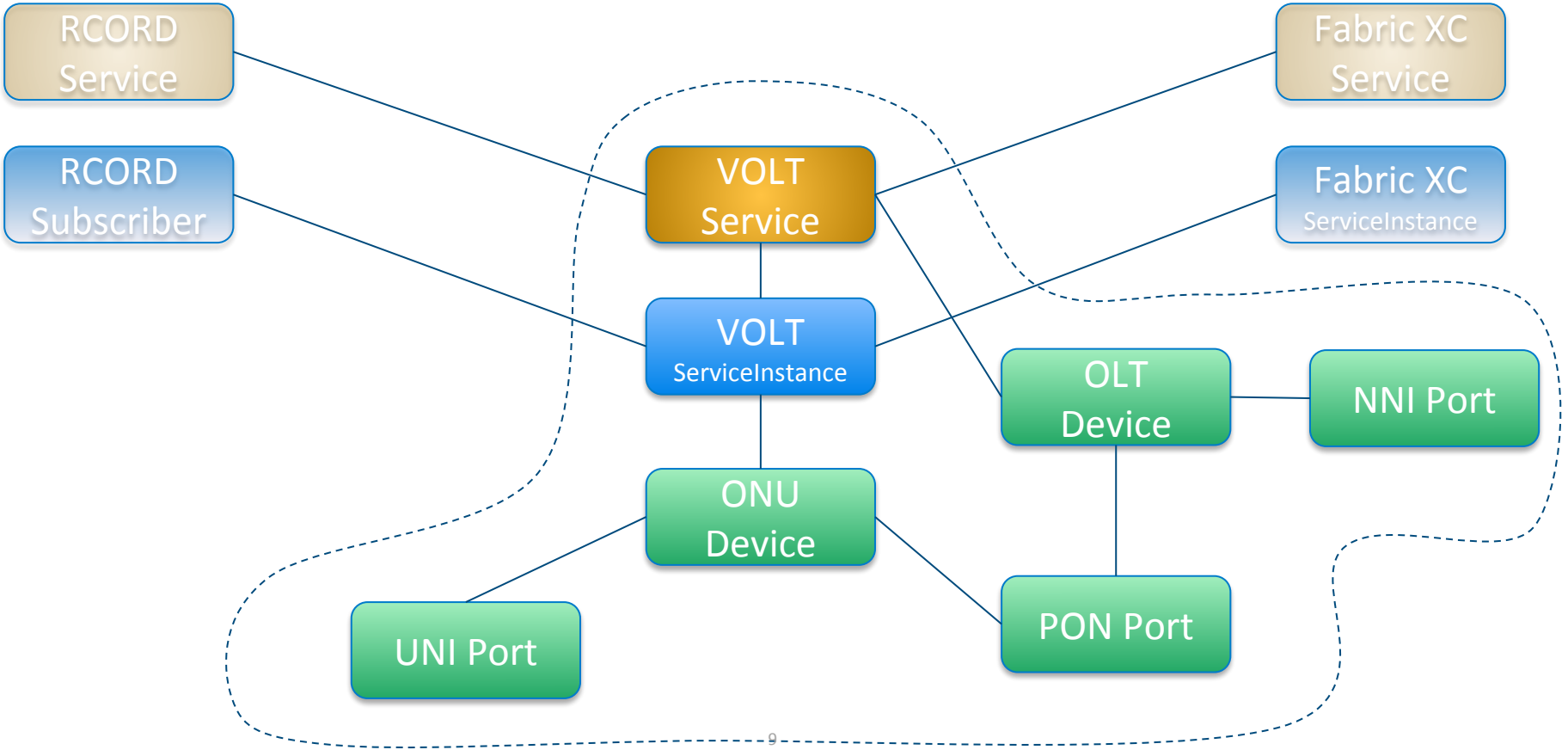
Base Models (stuff you get for free)

- Users and Permissions
- Compute and Network Resources
- Services, Tenancy, and Dependencies
- Chains

Extensibility (value you can add)

- Any service can add new models
- Service models can inherit from base models

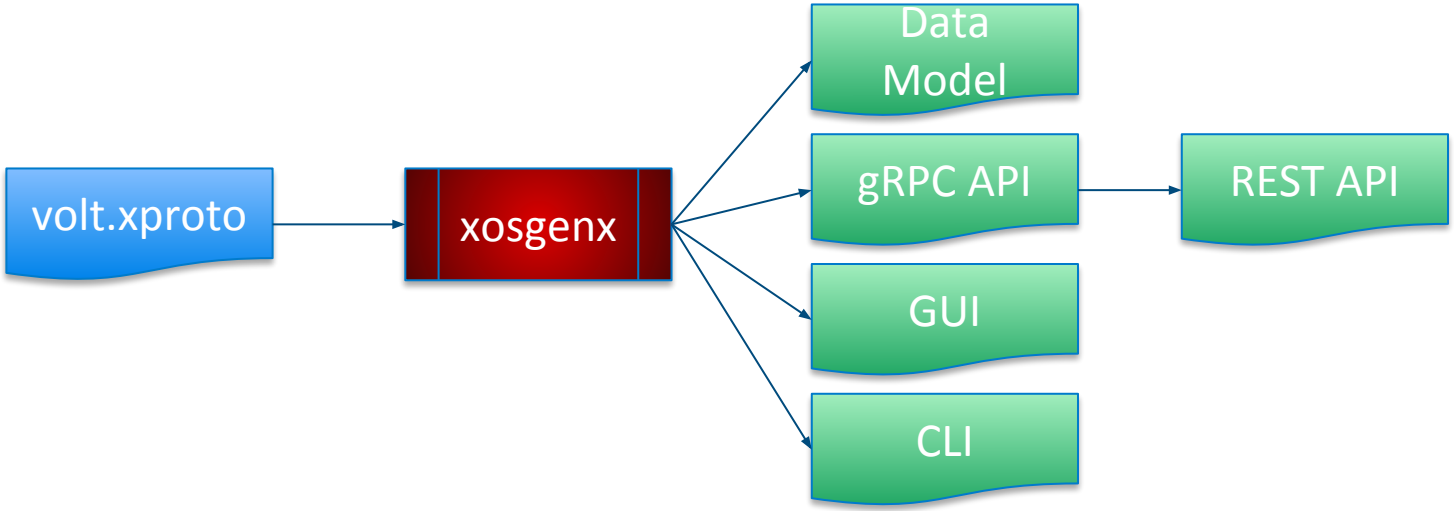
VOLT service example



Unification facilitates consistency

"xproto", the XOS data modeling language

- Based on protobuf, extended with relational features
- Used to autogenerate various targets (REST, GUI, etc)
- Make a change in one place, not six different places



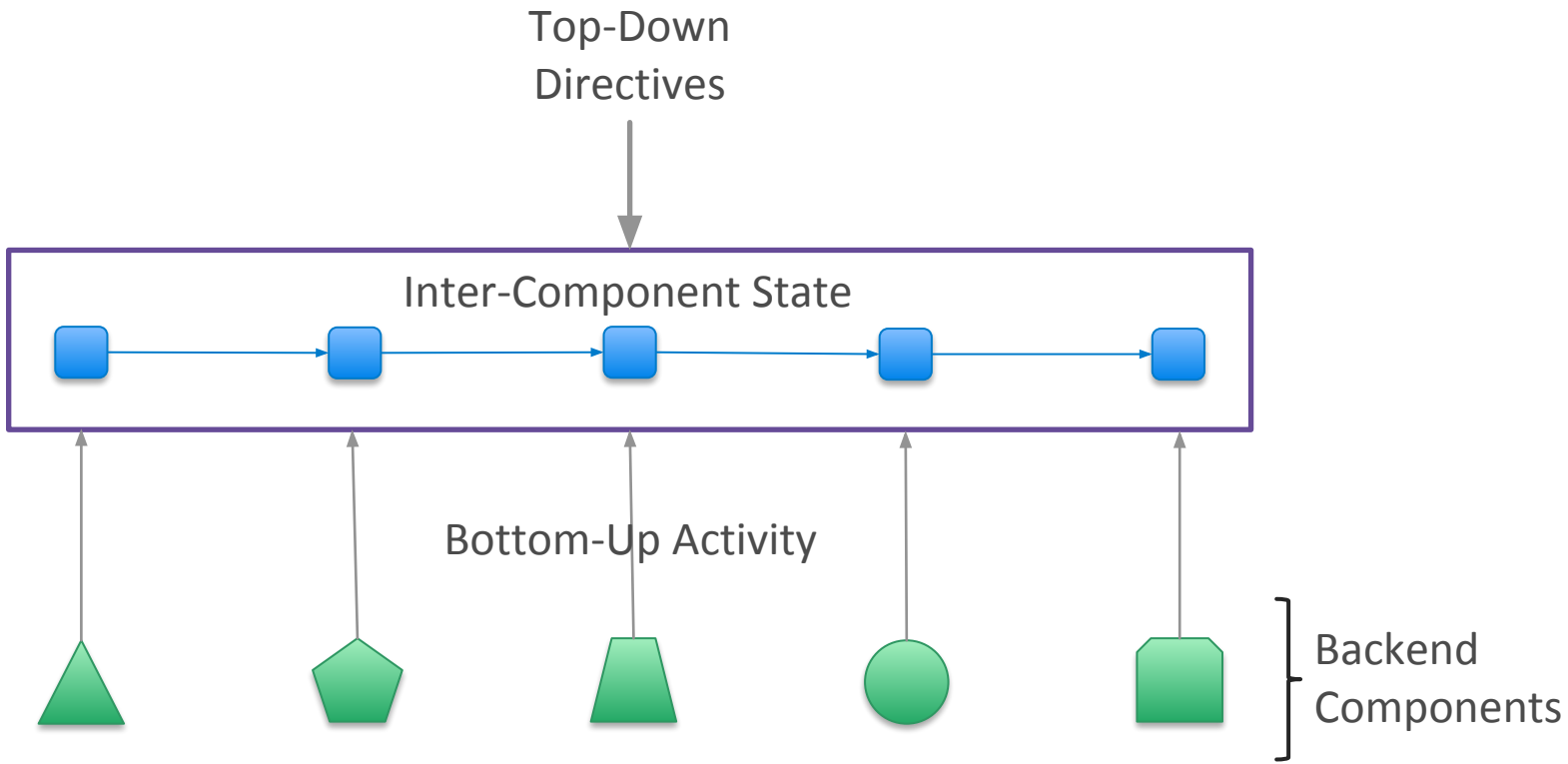
xproto example

```
message ONUDevice (XOSBase){
   option verbose_name = "ONU Device";
   option description = "Represents a physical ONU device";

   required manytoone pon_port->PONPort:onu_devices = 1:1001 [db_index = True];
   required string serial_number = 2 [max_length = 254, db_index = False, tosca_key=True, unique =
True];
   required string vendor = 3 [max_length = 254, db_index = False];
   required string device_type = 4 [help_text = "Device Type", default = "asfvolt16_olt",
max_length = 254, db_index = False];

   optional string device_id = 5 [max_length = 254, db_index = False, feedback_state = True];
   optional string admin_state = 6 [choices = "(('DISABLED', 'DISABLED'), ('ENABLED', 'ENABLED'))",
default="ENABLED", help_text = "admin_state", db_index = False];
   optional string oper_status = 7 [help_text = "oper_status", db_index = False, feedback_state =
True];
   optional string connect_status = 8 [help_text = "connect_status", db_index = False,
feedback_state = True];
}
```

From unification to coordination

Since we have a single unifying data model, services can use it to as a point of inter-service coordination…

Coordinate – East/West



Coordination Example

Authoritative state is held by different services

- RCORD Subscriber Service
  - Subscriber -> c_tag, s_tag, onu serial #, ip, mac, ...
- OLT Service
  - OLTDevice -> name, switch_datapath_id, switch_port, set of ONUs, …
  - ONUDevice -> onu serial #, admin_state, ...

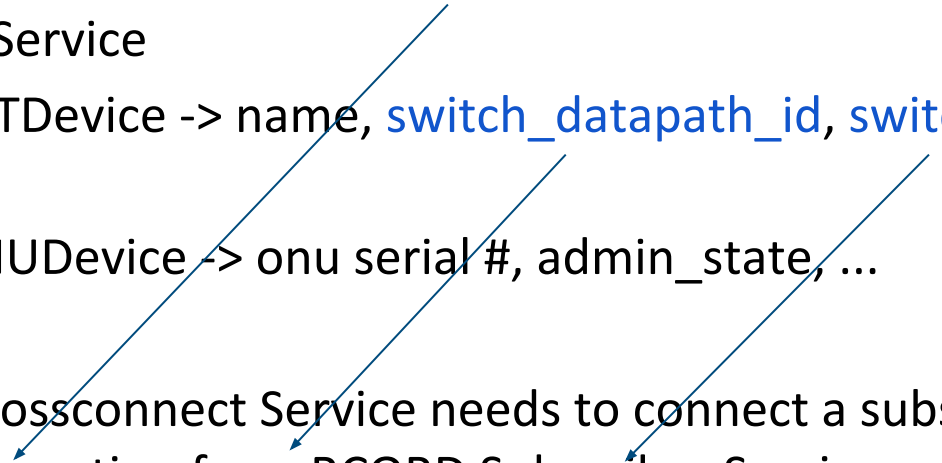
Fabric Crossconnect Service needs to connect a subscriber to the Internet, uses information from RCORD Subscriber Service and OLT Service

(s_tag, switch_datapath_id, src_port, dest_port)

Coordination Example

Authoritative state is held by different services

- RCORD Subscriber Service
  - Subscriber -> c_tag, s_tag, onu serial #, ip, mac, ...
- OLT Service
  - OLTDevice -> name, switch_datapath_id, switch_port, set of ONUs, …
  - ONUDevice -> onu serial #, admin_state, ...



Fabric Crossconnect Service needs to connect a subscriber to the Internet, uses information from RCORD Subscriber Service and OLT Service

(s_tag, switch_datapath_id, src_port, dest_port)

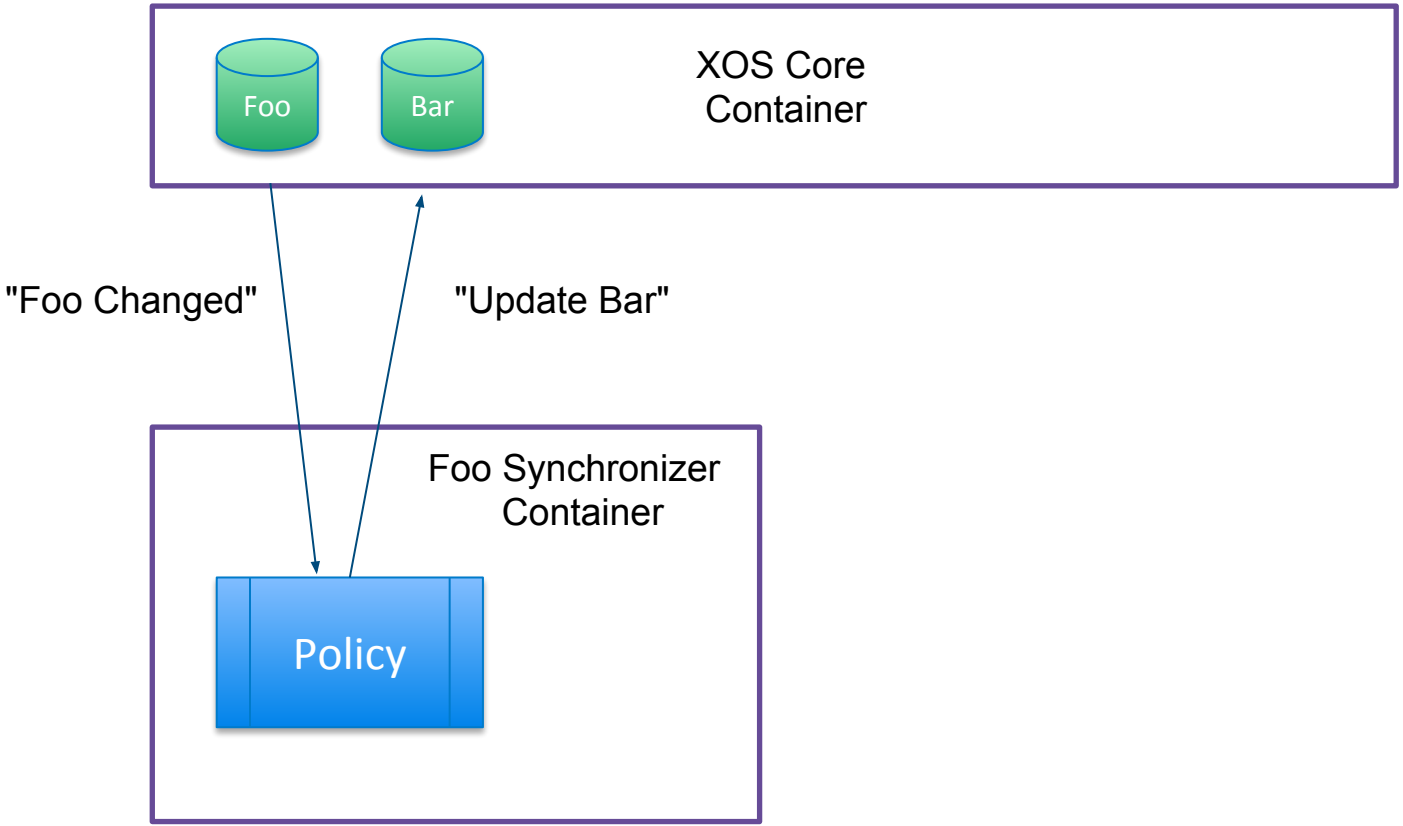
Coordination often occurs in Model Policies

Properties of model policies

- Triggered when models are created, updated, or deleted
- Causes changes to occur elsewhere in the data model
- Does not directly cause changes to occur in underlying services

Policy code resides in service-specific synchronizers

Model Policies live in Synchronizer Containers



Example: VOLT Service

Western neighbors

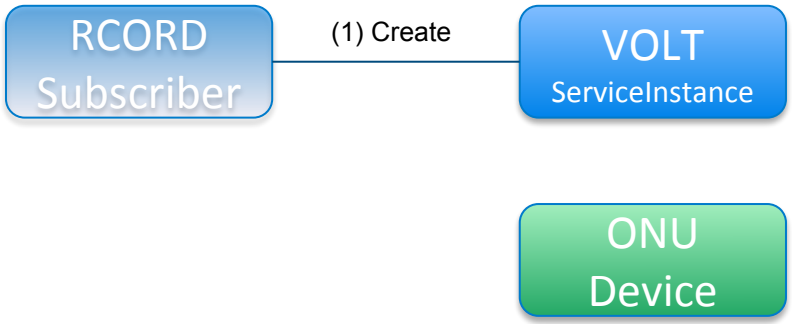
- RCORD Service - Manages subscriber state

Eastern neighbors

- Fabric Crossconnect Service - Connects OLT to BNG via switch

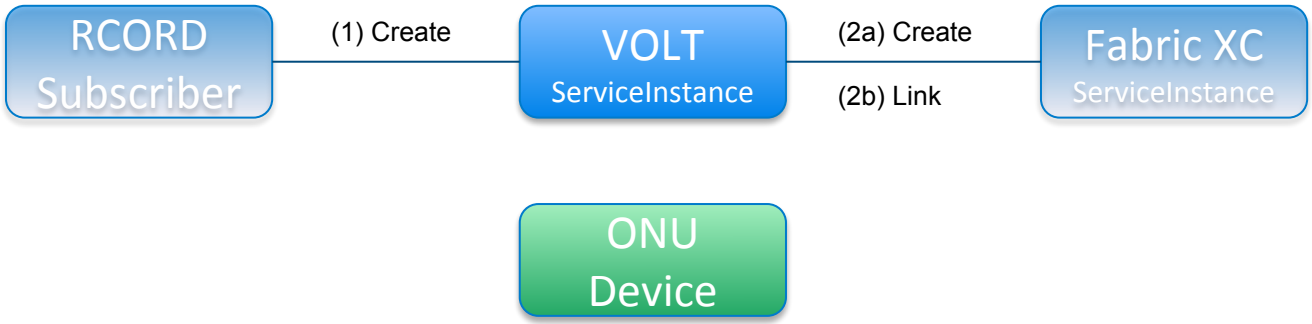
VOLTServiceInstance Create/Update Policy

1. VOLTServiceInstance created



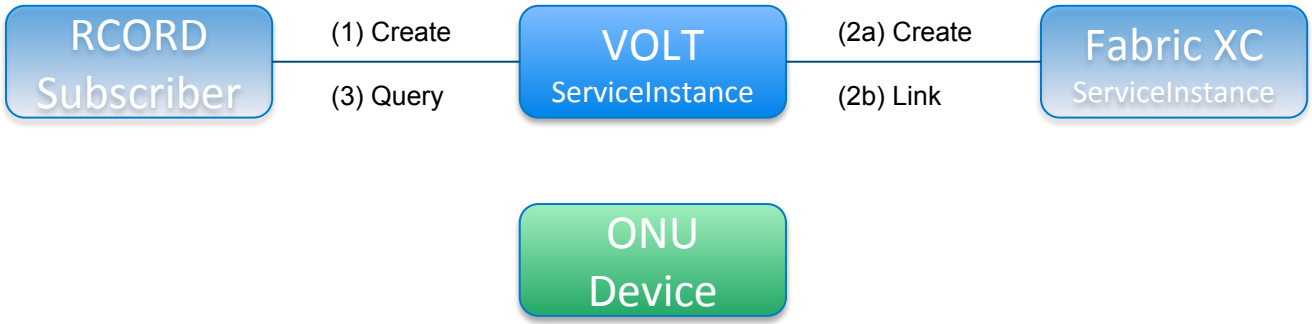
VOLTServiceInstance Create/Update Policy

1. VOLTServiceInstance created
2. FabricCrossconnectServiceInstance created and linked



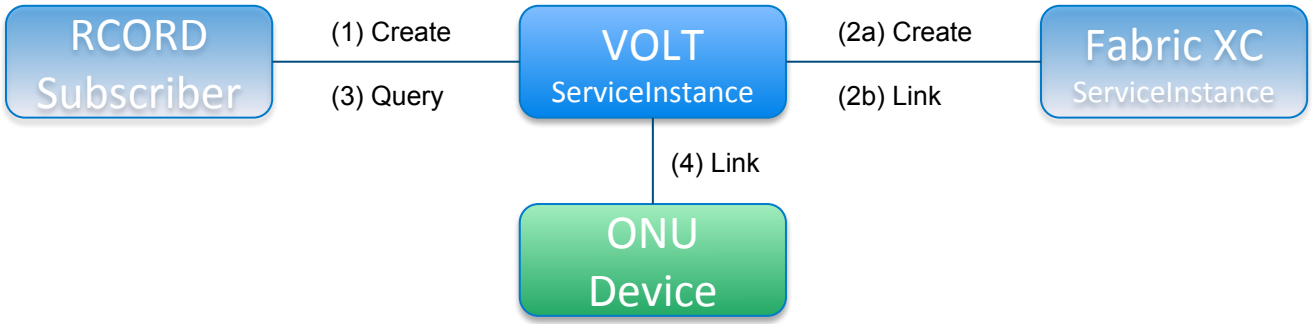
VOLTServiceInstance Create/Update Policy

1. VOLTServiceInstance created
2. FabricCrossconnectServiceInstance created and linked
3. Retrieve serial number from Subscriber



VOLTServiceInstance Create/Update Policy

1. VOLTServiceInstance created
2. FabricCrossconnectServiceInstance created and linked
3. Retrieve serial number from Subscriber
4. Attach ONUDevice matching serial number to VOLTServiceInstance

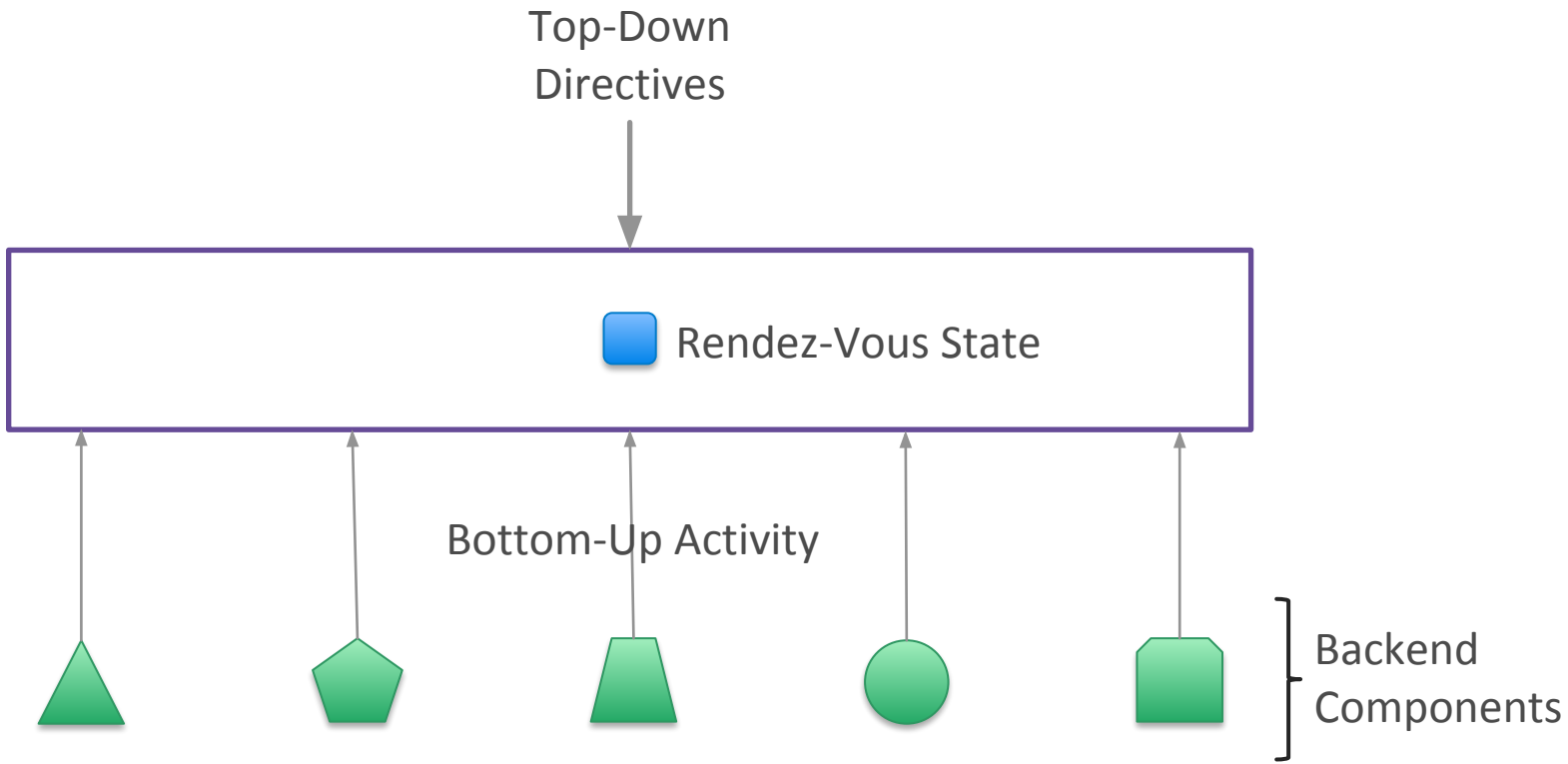


From coordination to Synchronization

Model policies operate within the data model.

Synchronizer containers also handle coordinating state between the data model and external services.

Synchronize – North/South

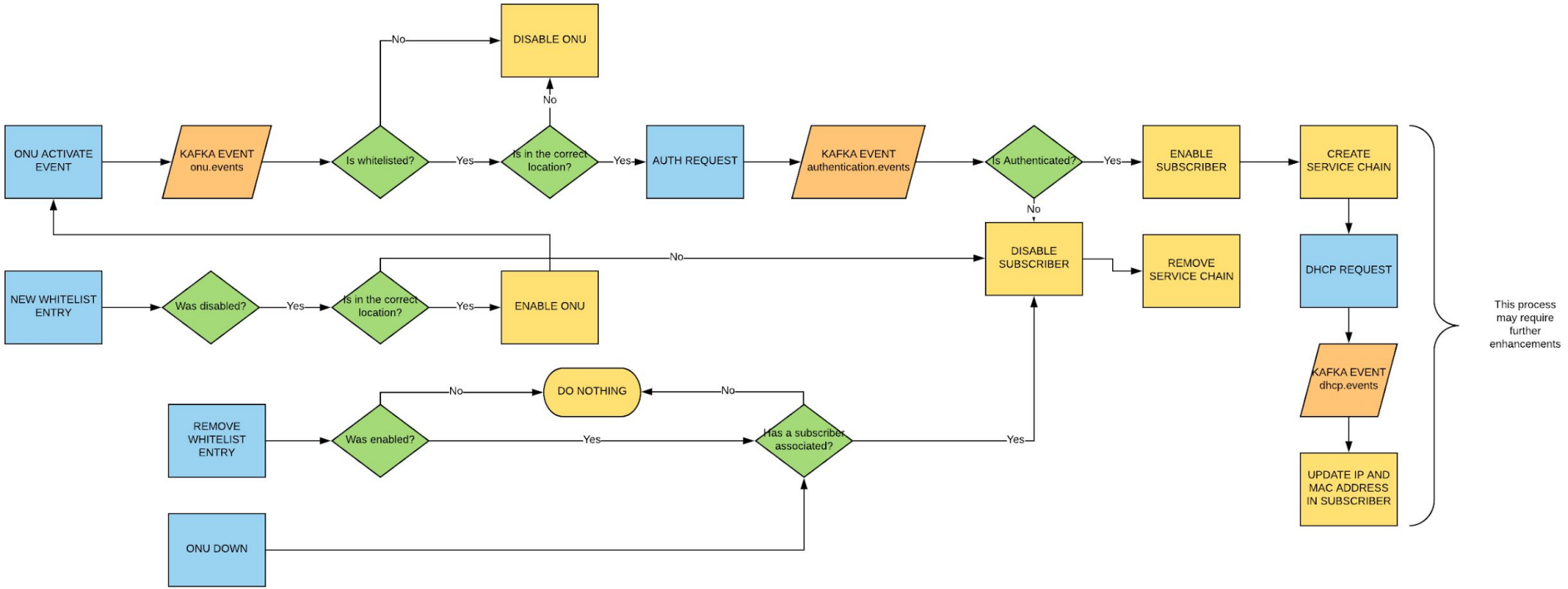


Example - Authorize an ONU

Operator uses API to add an ONU Serial number to whitelist

- Create AttWorkflowDriverWhiteListEntry in data model
- Check to see if ONU is on the wrong PONPort:
  - Set onu_state="DISABLED", authentication_state="AWAITING"
  - Make REST call to Voltha to disable ONU
- Else:
  - Make REST call to Voltha to enable ONU
  - Wait for ONU activate event
  - Wait for ONU authenticate event
  - Set authentication_state="APPROVED", subscriber.status="ENABLED"
  - … move on to DHCP state machine ...

Example - Authorize an ONU



Types of Synchronizer "Steps"

Sync Step

- Apply data model state to an underlying service
- Keeps running until the job is done
- Example: OLTDevice has been created in XOS, so make REST call to create the OLT in Voltha

Delete Step

- Delete an object in an underlying service
- Example: OLTDevice has been deleted in XOS, so make REST call to delete the OLT in Voltha

Types of Synchronizer "Steps"

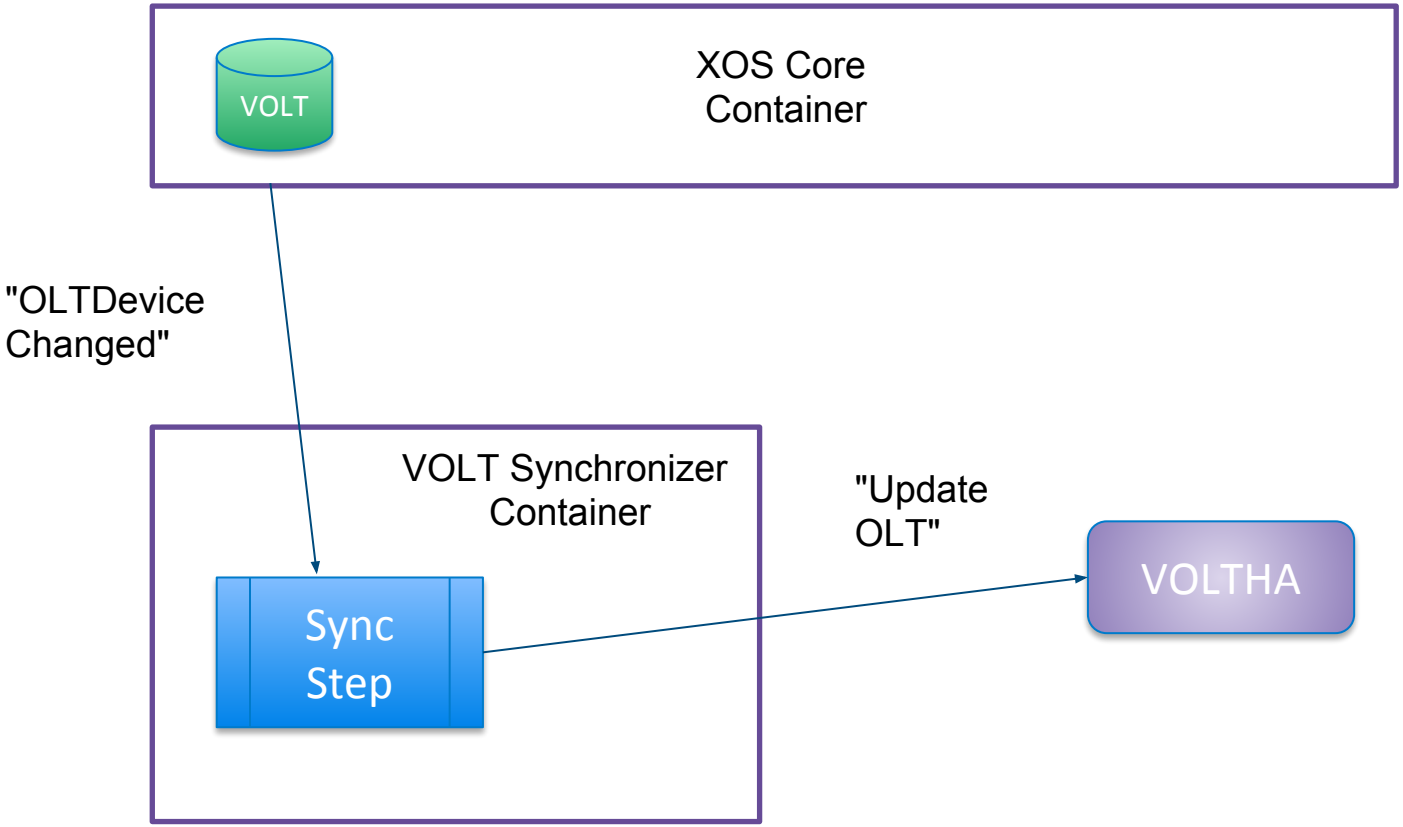
Pull Step

- Poll and underlying service for state, apply to data model
- Example: Query Voltha via REST for ONUs, create ONUDevices in XOS

Event Step

- Watch kafka for events
- Example: ONU connectivity status has changed, update ONUDevice in XOS

Steps Also live in Synchronizer Containers



Ways to get involved

- Integrate existing services
  - Cassandra
  - Nagios
  - CDN
  - ...
- Invent new services
  - Firewall VNF
  - Parental control VNF
  - ...