SDN 응용 지원 OPENDAYLIGHT 컨트롤러 개발 및 활용 방법

2015.11

임창규, 박세형

SDN기술연구실

Contents

OpenDaylight

- Linux Foundation의 오픈 소스 프로젝트
- SDN을 위한 새로운 산업 공통 프레임워크를 제공
  - Openflow에 제한된 프레임워크가 아니라 다양한 프로토콜이 독립적으로 사용될 수 있는 프레임워크
    - Openflow 1.0 southbound plugin 제공으로 부터 시작함
  - OSGi 프레임워크를 지원
  - Northbound API를 위해 REST 기능을 지원

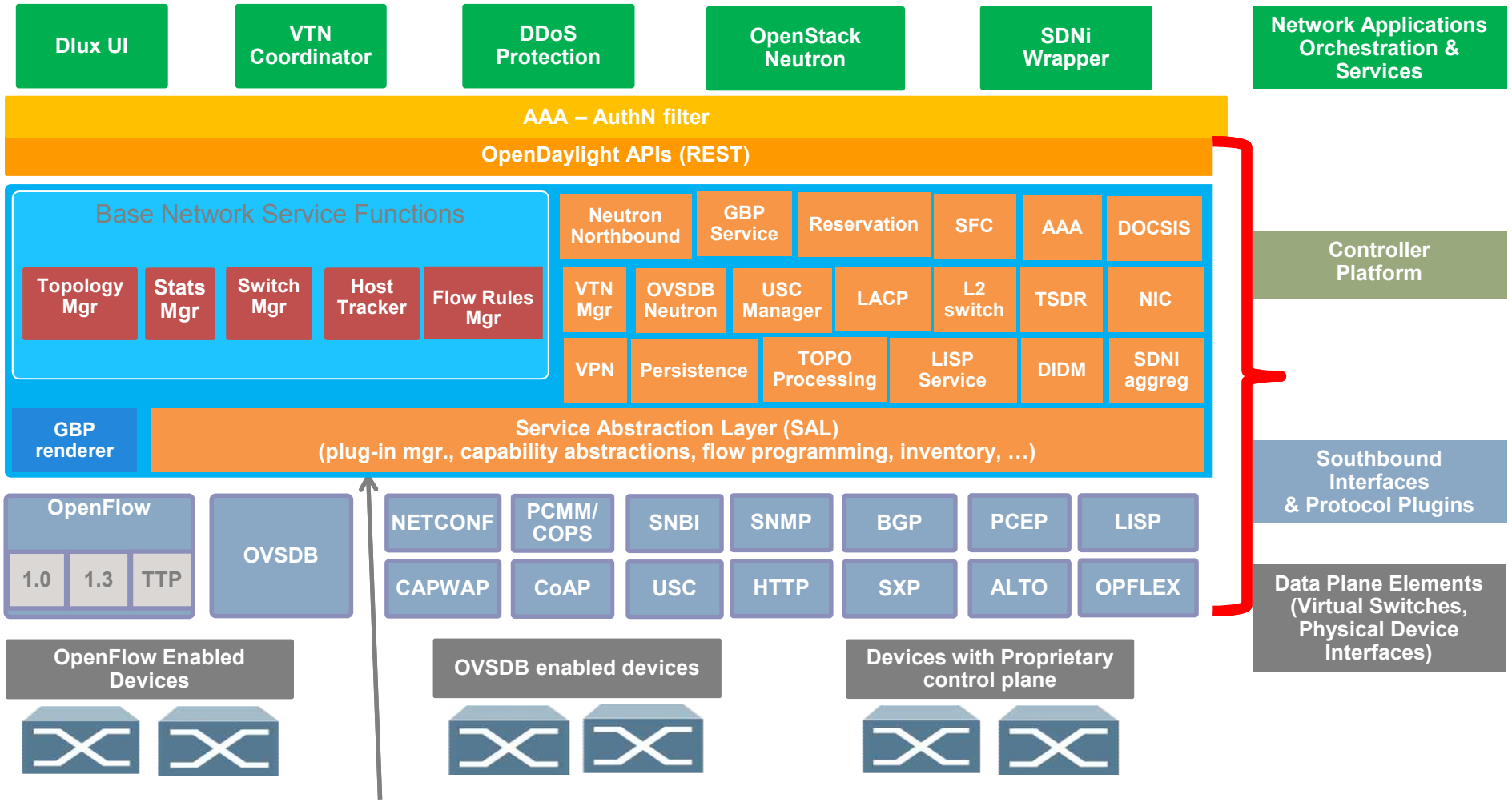
OpenDaylight Consortium

- 장비벤더 위주의 강력한 산업계 지원

OpenDaylight Release

- Hydrogen(Feb. 2014)
  - Base 1.0 — Feb. 2014
  - Virtualization 1.0 — Feb. 2014
  - Service provider 1.0 — Feb. 2014
- Helium(Sep. 2014)
  - SR1 — Nov. 2014
  - SR2 — Jan. 2015
  - SR3 — Mar. 2015
  - SR4 — Aug. 2015
- Lithium(Jun. 2015)
  - SR1 — Aug. 2015
  - SR2 — Oct. 2015

Lithium Release (June 2015)



Main difference from other
OpenFlow-centric controller platforms

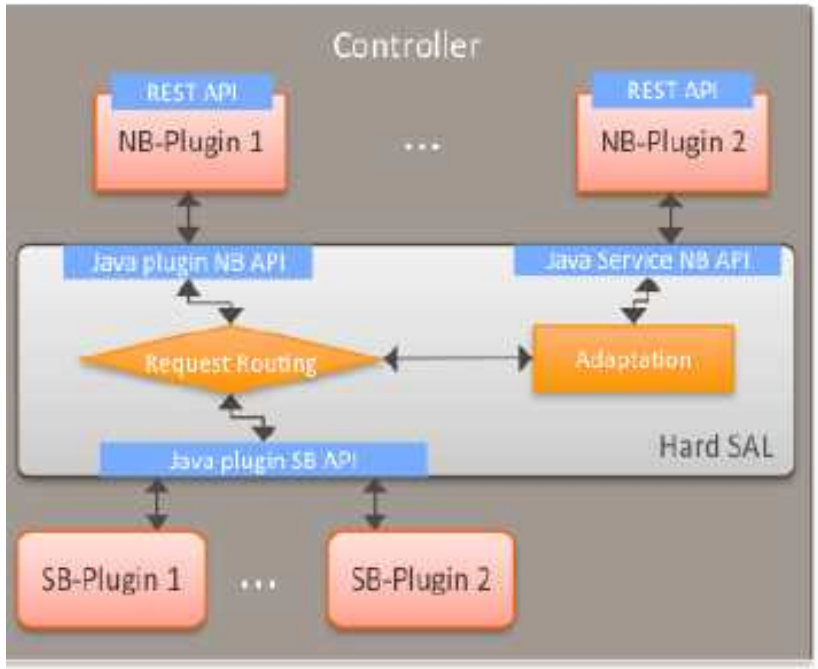
Lithium: List of Projects (Total: 40)

Project keyword	Description
aaa	AAA Project
alto*	Application-Layer Traffic Optimization
bgpcep	BGP/PCEP protocol library
capwap*	Control and Provisioning of Wireless Access Point
controller	OpenDaylight Controller
defense4all	Radware Defense4All
didm*	Device Identification and Drive Management
discovery	Discovery Service
dlux	OpenDaylight UI
docs	Documentation Project
groupbasedpolicy	Group Based Policy Plugin
integration	Integration Framework
iotdm*	IoT Data-centric Middleware
l2switch	Separate Layer 2 switching
lacp*	Link Aggregation Control Protocol
lispflowmapping	LISP Mapping Service
nic*	Network Intent Composition
openflowjava	OpenFlow Protocol Library
openflowplugin	OpenFlow Plugin
opflex*	OpFlex protocol
ovsdb	OVSDB protocol and OpenStack integration
packetcable	PacketCable PCMM/COPS
pss*	Persistence Store Service
plugin2oc	Southbound plugin to the OpenContrail platform
reservation*	Dynamic Resource Reservation project
sdninterfaceapp	SDNi Cross-controller interface
sfc	Service Function Chaining
snbi	Secure Network Bootstrap Infrastructure
snmp4sdn	SNMP4SDN Plugin
snmp*	SNMP Southbound Plugin
sxp*	Source-Group Tag eXchange Protocol
tcpmd5	TCP-MD5 Support library
toolkit	Quickstart Toolkit
tpf*	Topology Processing Framework
tsdr*	Time Series Data Repository
ttp	TTP Project
usc*	Unified Secure Channel
vtn	VTN (Virtual Tenant Network)
yangtools	YANG Tools

* → New in Lithium release

Service Abstraction Layer architecture

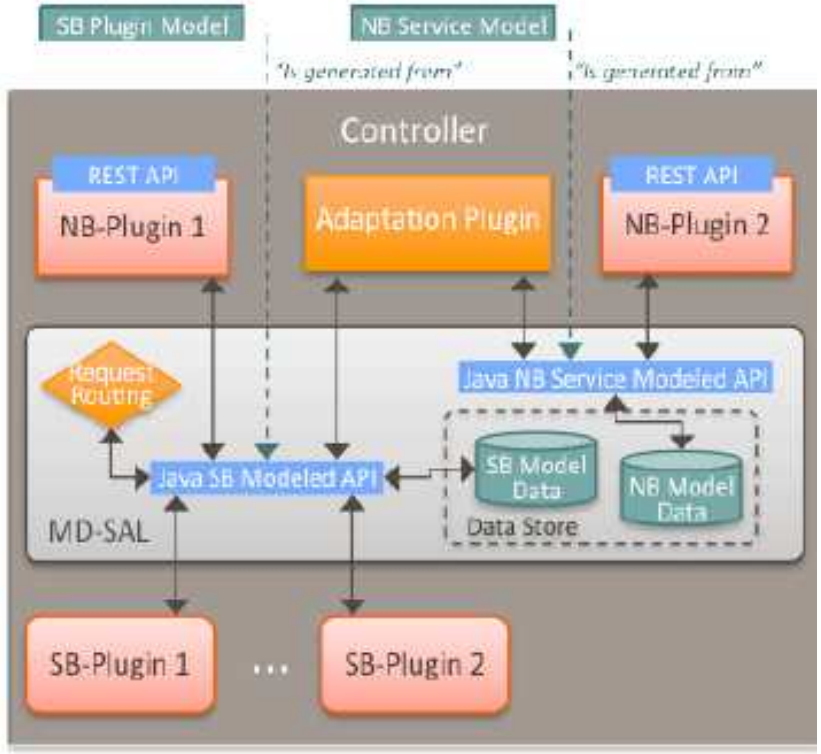
Hydrogen



Hard SAL

(API Driven SAL)

Helium



MD SAL

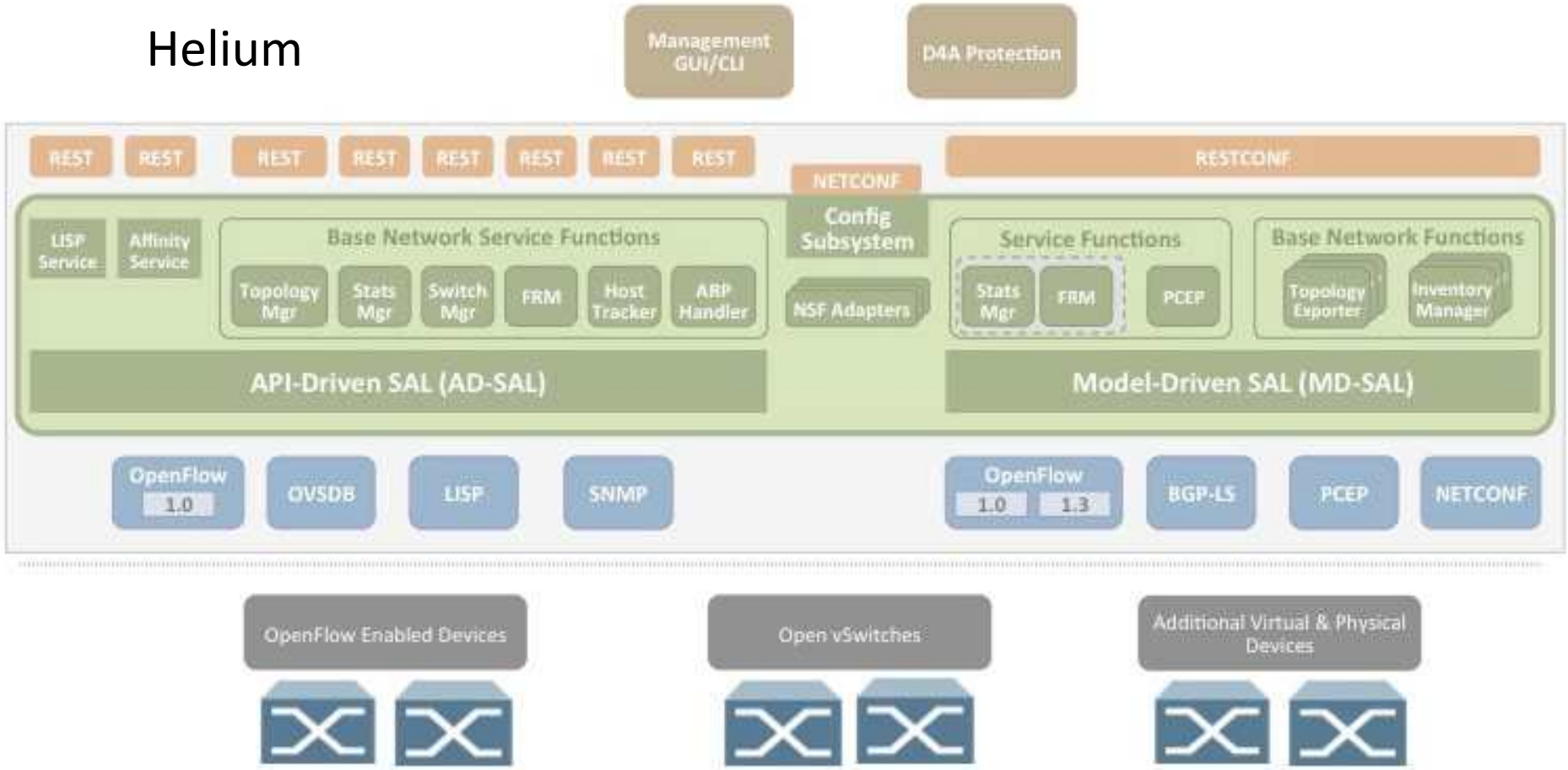
(Model Driven SAL)

Service Abstraction Layer architecture

- AD(API-Driven)-SAL
  - Plugins can be data providers or data consumers or both
  - SAL APIs request routing between consumers and providers, and data adaptations are all statically defined at compile/build time
  - Translation between SB plugin API and abstract NB API is done in the abstraction module in AD-SAL
  - AD-SAL has both NB and SB APIs
- MD(Model-Driven)-SAL
  - SAL APIs request routing between consumers and providers are defined from models, and data adaptation are provided by internal adaptation plugins
  - API code is generated from models when a plugin is compile
    - API code is loaded into the controller along with the rest of the plugin containing the model when the plugin OSGi bundle is loaded into the controller
  - Service adaptation is provided by plugin
    - An adaptation plugin is a regular plugin
    - Model to model translation between two APIs
  - Provider and consumer plugins can exchange data through the MD-SAL storage
  - MD-SAL allows both NB plugins and SB plugins to use the same API generated from a model

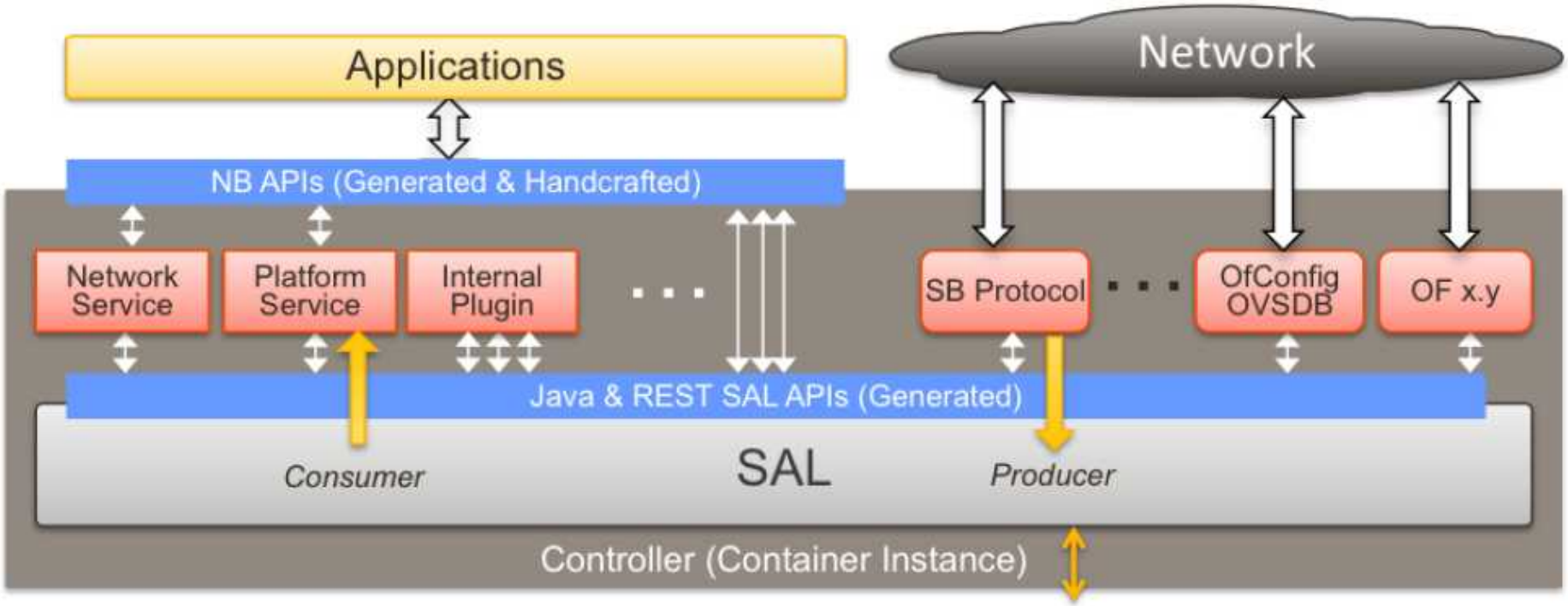
Service Abstraction Layer architecture

Helium

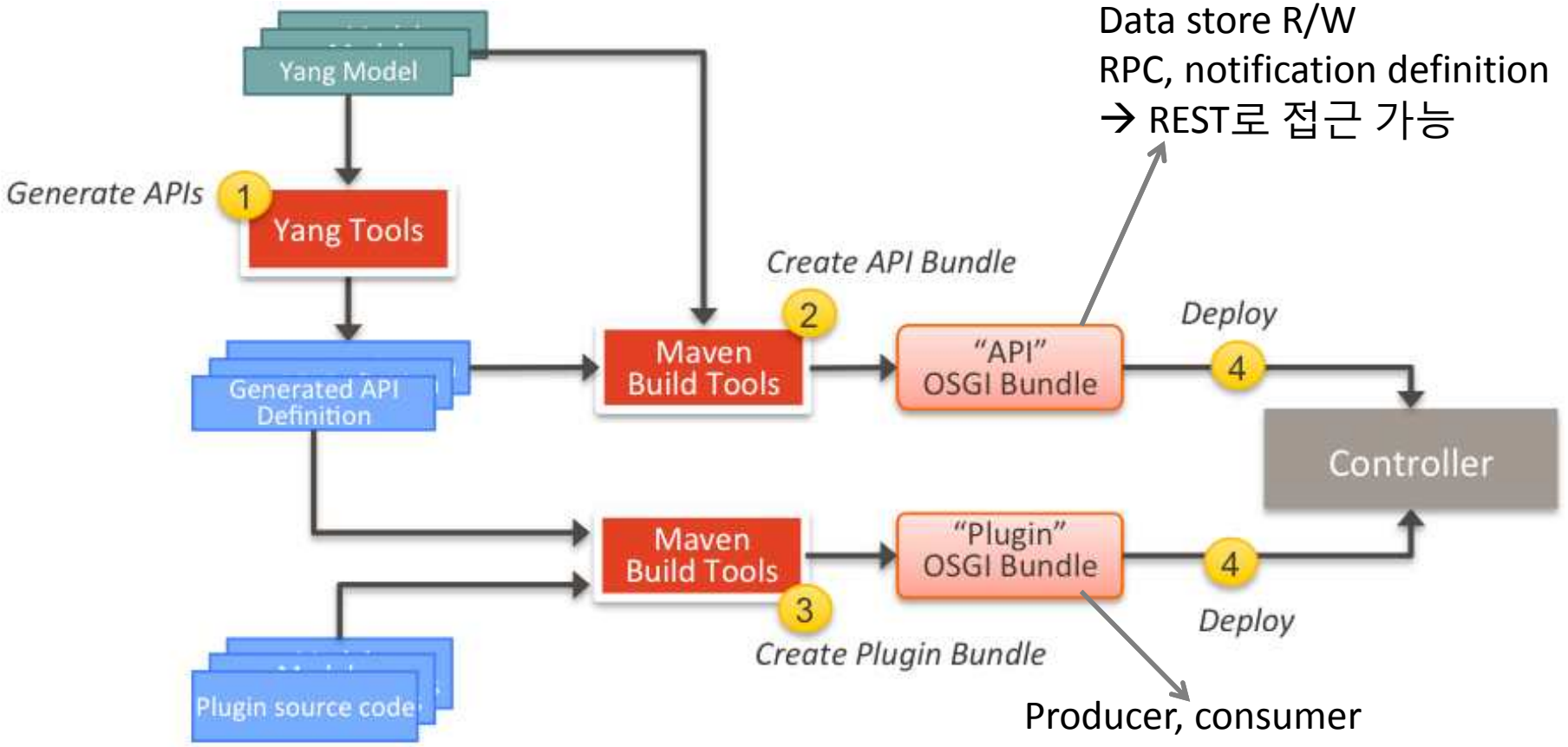


Consumer and producer(provider)

- Plugins: protocol, application, adaptation and others



API and plugin bundle



Data store, RPC, notification 사용

- API docs로 API bundle 사용

GET /config/toaster:toaster/

Request URL

```
http://localhost:8181/restconf/config/toaster:toaster/
```

Response Body

```
{
  "toaster": {
    "toasterModelNumber": "Model 1 - Binding Aware",
    "toasterManufacturer": "Opendaylight",
    "toasterStatus": "down"
  }
}
```

Response Code

```
200
```

Response Headers

```
{
  "Transfer-Encoding": "chunked",
  "Server": "Jetty(8.1.14.v20131031)",
  "Content-Type": "application/json"
}
```

POST /operations/toaster:make-toast

Parameters

Parameter	Value	Description
(make-toast)input	{"toaster:input": { "toasterDoneness":"10","toasterToastType":"Wheat-bread" }}	

Parameter content type: application/json

Try it out! Hide Response

Request URL

```
http://localhost:8181/restconf/operations/toaster:make-toast
```

Response Body

```
no content
```

Response Code

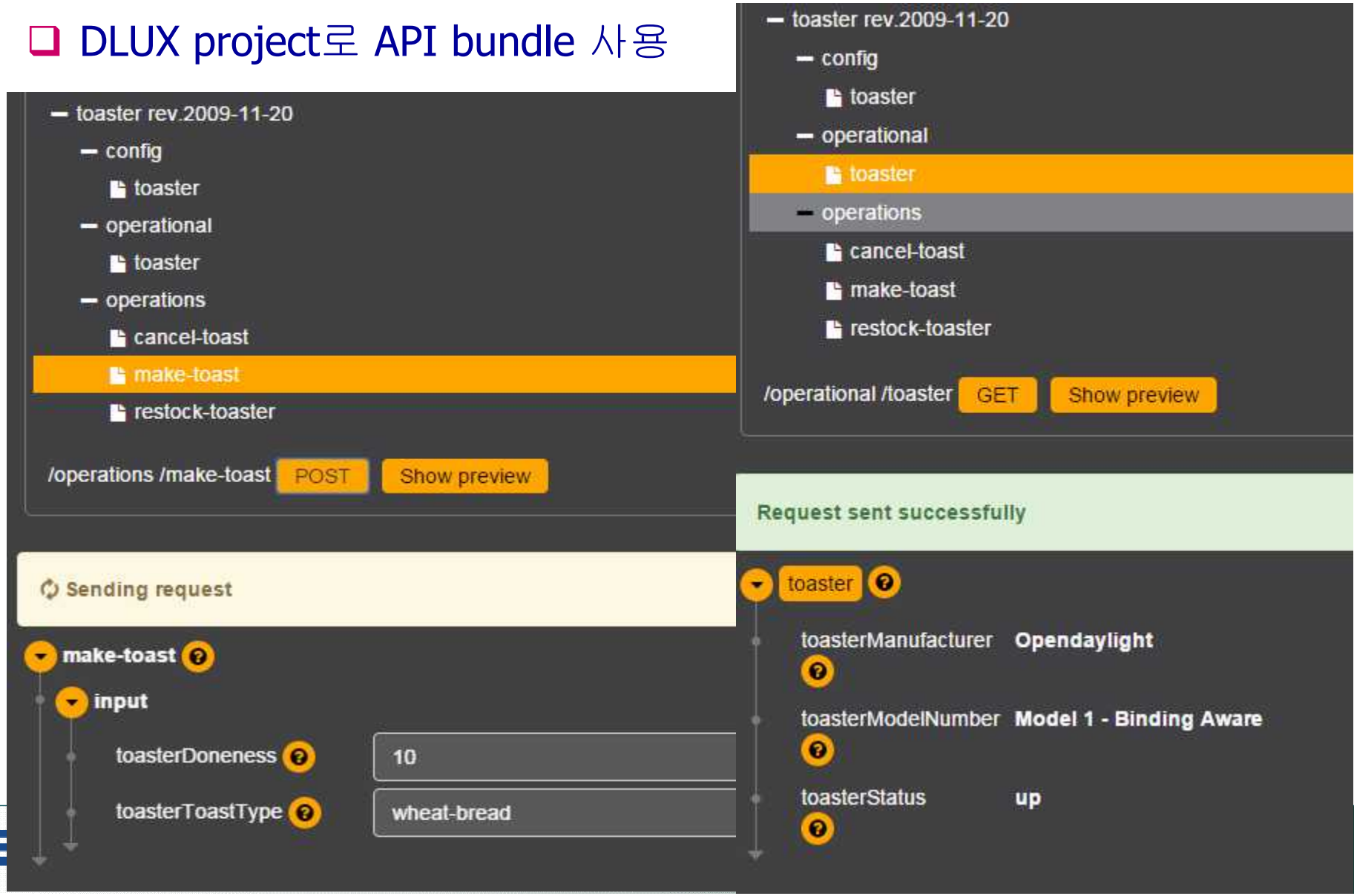
```
204
```

Response Headers

```
{
  "Access-Control-Allow-Origin": "http://localhost:8181",
  "Access-Control-Expose-Headers": "",
  "Access-Control-Allow-Credentials": "true",
  "Server": "Jetty(8.1.14.v20131031)",
  "Content-Type": "application/json"
}
```

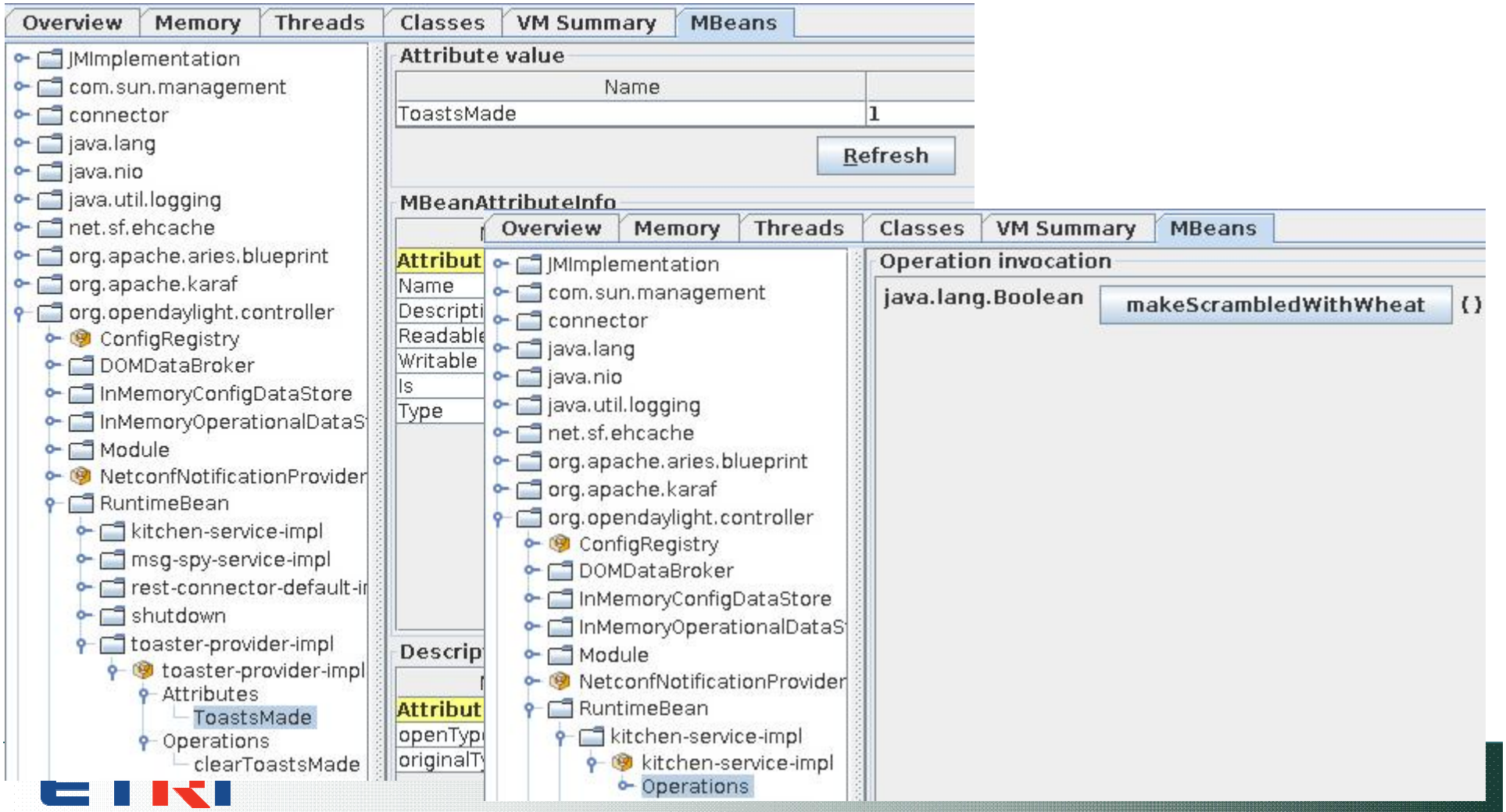
Data store, RPC, notification 사용

- DLUX project로 API bundle 사용



Producer, consumer 접근

- Jconsole로 JMX access



컨트롤러 개발 방법(모델링 및 plugin 개발)

OpenDaylight controller 개발 방법

- Data model 정의
- Provider plugin 구현
- Provider RPC 추가
- Provider configuration data store 변경
- Provider troubleshooting
  - JMX
- Consumer plugin 구현
- Notification

Data model 정의

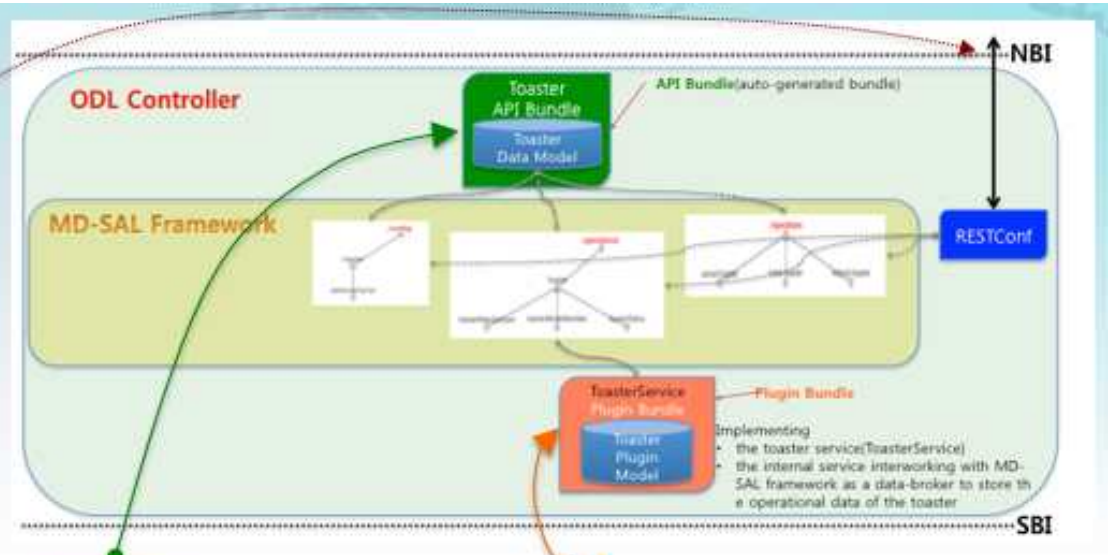
Real World

Consumer Side

Provider Side

- White-bread
- Wheat-bread

- Operation
  - *Make Toast with Doneness & Toast Type*
  - *Cancel Toast*
  - *Restock Toaster with Amount of bread*
- Operational Data
  - *Manufacturer*
  - *Model Number*
  - *Knob Status*
- Configuration Data
  - *Darkness Factor*
- Notification Data
  - *Out Of Bread*
  - *Bread Restocked*



toaster.yang

Modeling a Toaster

DataType, Attributes, RPC, Notification

ToasterService(java)

Implementing operational data for a real toaster to store MD-SAL's data store

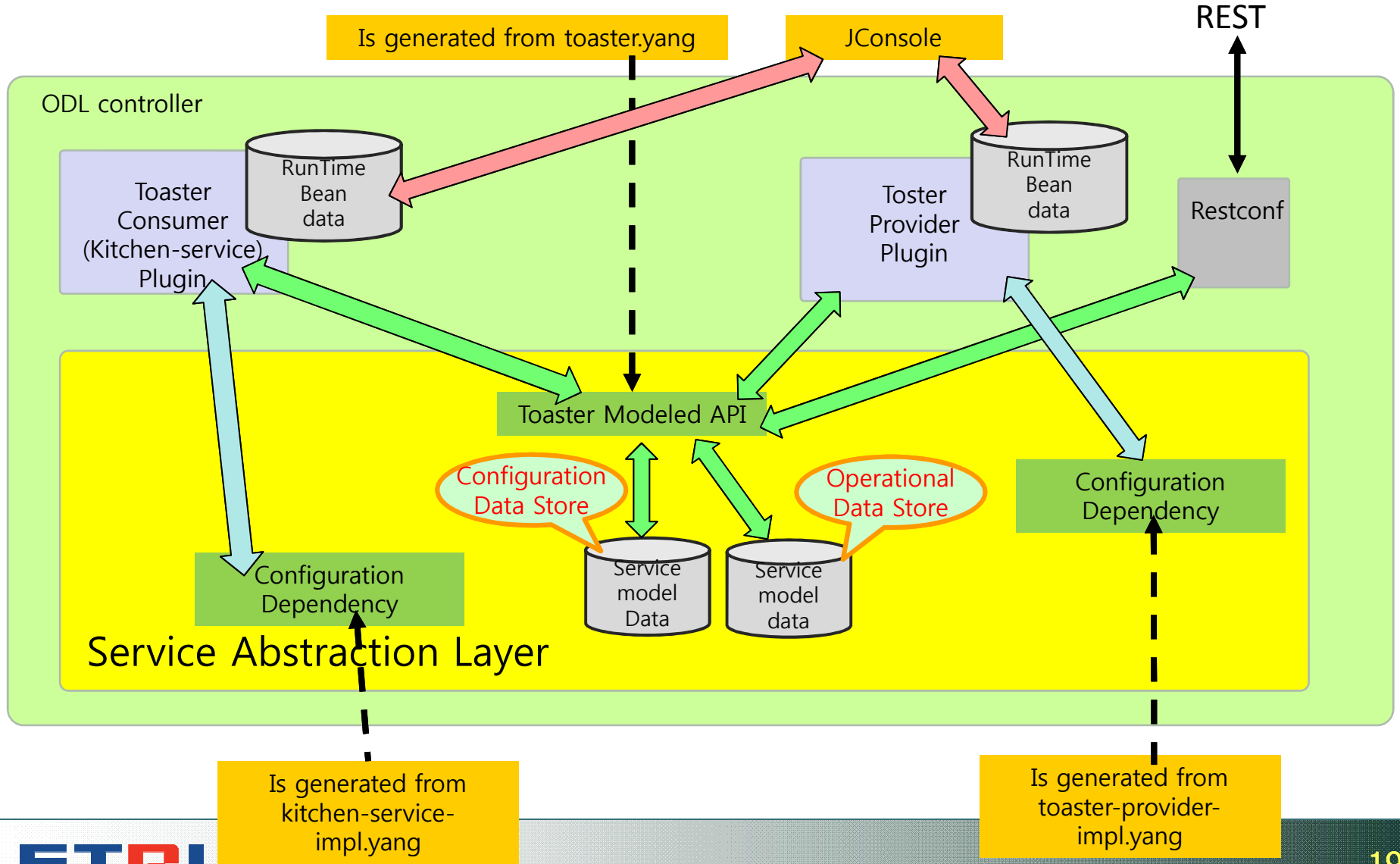
toaster-provider-impl.yang

Providing configuration data and services access for JMX, NET/RESTConf

YANG Modeling

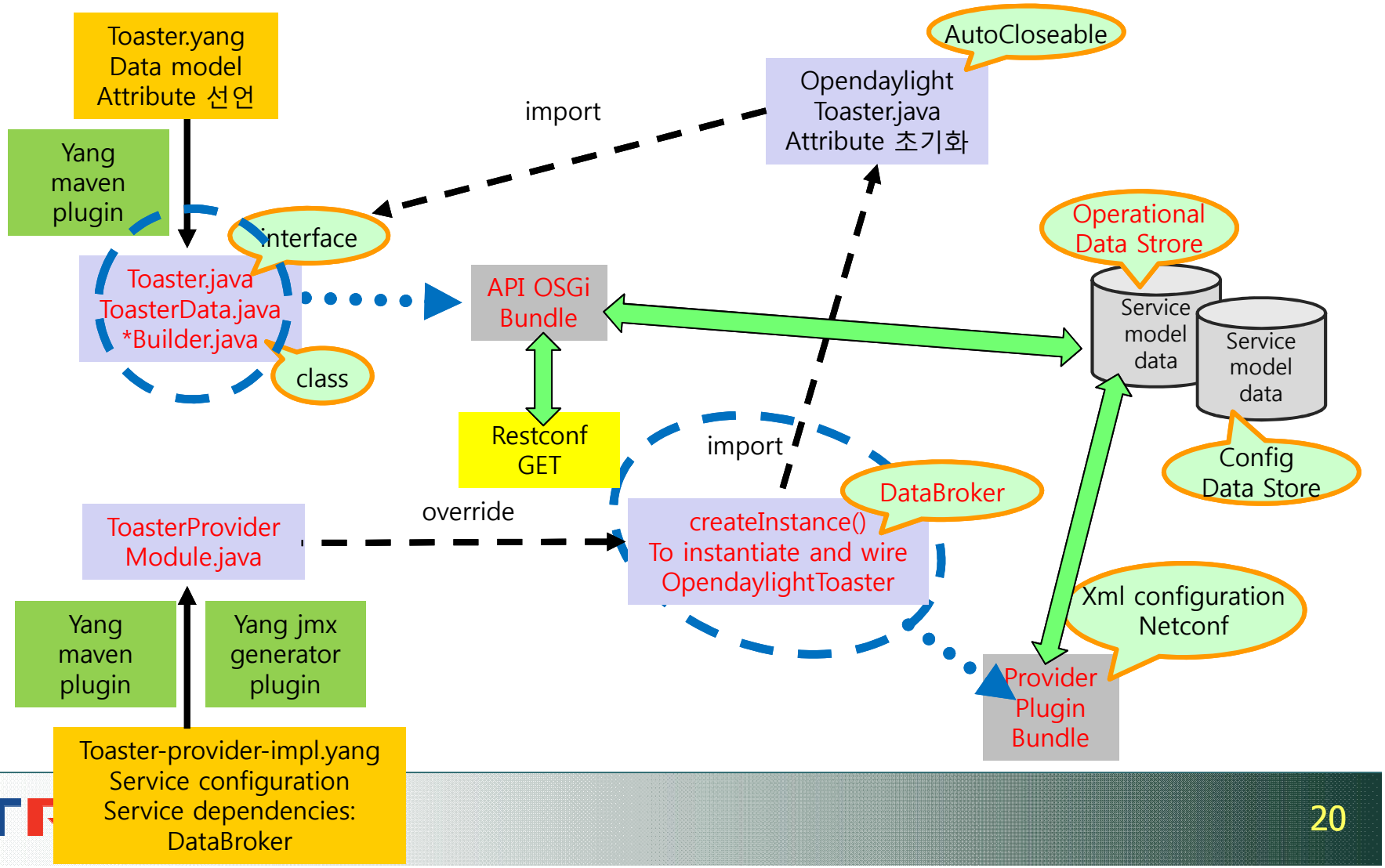
- Defining the north bound abstraction of the toaster data model for NBI Clients
  - Attributes
    - Configuration Data , Operational Data
  - RPC(Operation)
  - Notifications

OpenDaylight MD-SAL



ODL MD-SAL

- Toaster API bundle, Provider plugin bundle, Data broker 사용



Data model 정의

```
module toaster {

  //The yang version - today only 1 version exists. If omitted defaults to 1.
  yang-version 1;

  //a unique namespace for this toaster module, to uniquely identify it from other
  namespace
    "http://netconfcentral.org/ns/toaster";

  //a shorter prefix that represents the namespace for references used below
  prefix toast;

  //Defines the organization which defined / owns this .yang file.
  organization "Netconf Central";

  //defines the primary contact of this yang file.
  contact
    "Andy Bierman <andy@netconfcentral.org>";

  //provides a description of this .yang file.
  description
    "YANG version of the TOASTER-MIB.";

  //defines the dates of revisions for this yang file
  revision "2009-11-20" {
    description
      "Toaster module in progress.";
```

Data model 정의

```
container toaster {
  presence
    "Indicates the toaster service is available";
  description
    "Top-level container for all toaster database objects.";

  //Note in these three attributes that config = false. This indicates that t
  leaf toasterManufacturer {
    type DisplayString;
    config false;
    mandatory true;
    description
      "The name of the toaster's manufacturer. For instance, Microsoft Toaste
  }

  leaf toasterModelNumber {
    type DisplayString;
    config false;
    mandatory true;
    description
      "The name of the toaster's model. For instance, Radiant Automatic.";
  }

  leaf toasterStatus {
    type enumeration {
      enum "up" {
        value 1;
        description
          "The toaster knob position is up. No toast is being made now.";
      }
      enum "down" {
        value 2;
        description
          "The toaster knob position is down. Toast is being made now.";
      }
    }
    config false;
    mandatory true;
    description
      "This variable indicates the current state of  the toaster.";
```

Toaster.yang

- Source code
  - controller/opendaylight/md-sal/samples
    - toaster, toaster-provider, toaster-consumer 폴더 참고
- Generate the toaster yang data model source
  - Achieved by using yang-maven-plugin in pom.xml
    - Mvn clean install -DskipTests
  - toaster/src/main/yang-gen-sal/org/opendaylight/yang/gen/v1/http/netconfcentral/org/ns/toaster/rev091120
  - Data transfer object(DTO)
    - Toaster
  - DTO builder
    - ToasterBuilder

Toaster provider

- OpendaylightToaster.java
  - Be done manually
  - toaster-provider/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - Data model and implementation to be provided by different bundles
    - Thus, different bundles to define different implementations of the same data model
  - public class OpendaylightToaster implements AutoCloseable
- Wiring the OpendaylightToaster service
  - In order to access to the available toaster services and run time related data, the toaster provider and consumer plugins have been configured as subsystem modules
    - Done as a yang module definition per each plugin
    - Each configuration specifies the dependencies between our plugins and the rest of ODL
  - Define toaster provider service yang configuration
    - toaster-provider/src/main/yang/toaster-provider-impl.yang
    - We add a databroker container node
  - Yang-maven-plugin and yang-jmx-generator-plugin
    - ToasterProviderModule
      - createInstance() method provides the OpendaylightToaster instance
    - ToasterProviderModuleFactory
      - Instantiated internally by MD-SAL that creates ToasterProviderModule instances

Toaster provider

```
module toaster-provider-impl {
    yang-version 1;
    namespace "urn:opendaylight:params:xml:ns:yang:controller:config:toaster-provider:impl";
    prefix "toaster-provider-impl";

    import config { prefix config; revision-date 2013-04-05; }
    import opendaylight-md-sal-binding { prefix mdsal; revision-date 2013-10-28; }

    description
        "This module contains the base YANG definitions for toaster-provider impl implementation.";

    revision "2014-01-31" {
        description
            "Initial revision.";
    }

    // This is the definition of the service implementation as a module identity
    identity toaster-provider-impl {
            base config:module-type;

            // Specifies the prefix for generated java classes.
            config:java-name-prefix ToasterProvider;
    }

    // Augments the 'configuration' choice node under modules/module.
    augment "/config:modules/config:module/config:configuration" {
        case toaster-provider-impl {
            when "/config:modules/config:module/config:type = 'toaster-provider-impl'";

            //wires in the data-broker service
            container data-broker {
                uses config:service-ref {
                    refine type {
                        mandatory false;
                        config:required-identity mdsal:binding-async-data-broker;
```

Toaster provider

- Wiring the OpendaylightToaster service
  - ToasterProviderModule.java
    - Provides the wiring with the rest of the ODL through the config-subsystem
    - toaster-provider/src/main/java/org/opendaylight/controller/config/yang/config/toaster_provider/impl
    - Generated from toaster-provider-impl.yang
      - Defines an implementation of toaster service and services it needs from MD-SAL framework
    - public final class ToasterProviderModule extends org.opendaylight.controller.config.yang.config.toaster_provider.impl.AbstractToasterProviderModule
      - Be mostly complete from code generation
      - Only the createInstance() method needs to be implemented to instantiate and wire the OpendaylightToaster
      - opendaylightToaster = new OpendaylightToaster();
      - DataBroker dataBrokerService= getDataBrokerDependency();
      - opendaylightToaster.setDataProvider(dataBrokerService);

Toaster provider

- Getting the operational status of the toaster

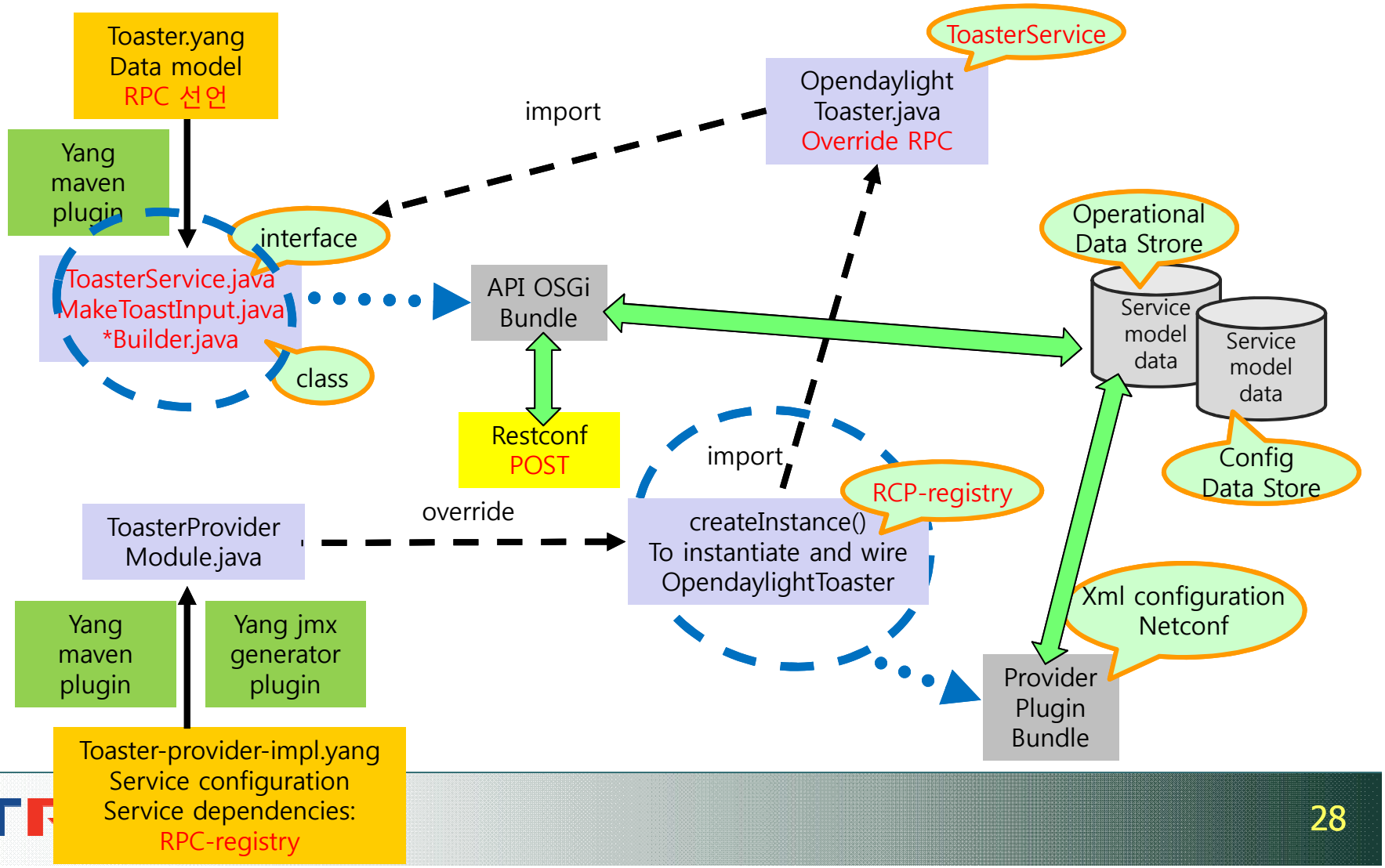
```
HTTP Method: GET
HTTP URL: http://localhost:8080/restconf/operational/toaster:toaster
```

You should see the following response:

```
{
    toaster: {
        toasterManufacturer: "Opendaylight"
        toasterModelNumber: "Model 1 - Binding Aware"
        toasterStatus: "Up"
    }
}
```

ODL MD-SAL

❑ RPC insertion



RPC 정의

```
module toaster {
    ...
//This defines a Remote Procedure Call (rpc). RPC provide the ability to initia
//on the data model. In this case the initating action takes two optional input
//QUESTION: Am I correct that the inputs are optional because they have default
rpc make-toast {
  description
    "Make some toast. The toastDone notification will be sent when the toast is
     An 'in-use' error will be returned if toast is already being made. A 'reso
     be returned if the toaster service is disabled.";

  input {
    leaf toasterDoneness {
      type uint32 {
        range "1 .. 10";
      }
      default '5';
      description
        "This variable controls how well-done is the ensuing toast. It should b
         Toast made at 10 generally is considered unfit for human consumption;
    }

    leaf toasterToastType {
      type identityref {
        base toast:toast-type;
      }
      default 'wheat-bread';
      description
        "This variable informs the toaster of the type of material that is bein
          combined with toasterDoneness, to compute for how long the material m
    }
  }
}  // rpc make-toast

// action to cancel making toast - takes no input parameters
rpc cancel-toast {
  description
    "Stop making toast, if any is being made.
       A 'resource-denied' error will be returned
       if the toaster service is disabled.";
}  // rpc cancel-toast
```

Toaster provider

- OpendaylightToaster.java
  - toaster-provider/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class OpendaylightToaster implements ToasterService, AutoCloseable
  - RPC overriding
    - makeToast(final MakeToastInput input)
    - CancelToast()
- Wiring the OpendaylightToaster service
  - Define toaster provider service yang configuration
    - toaster-provider/src/main/yang/toaster-provider-impl.yang

```
augment "/config:modules/config:module/config:configuration" {
    case toaster-provider-impl {
        when "/config:modules/config:module/config:type = 'toaster-provider-impl'";
        ...

        //Wires dependent services into this class - in this case the RPC registry service
        container rpc-registry {
            uses config:service-ref {
                refine type {
                    mandatory true;
                    config:required-identity mdsal:binding-rpc-registry;
                }
```

  - ToasterProviderModule.java
    - createInstance()
      - BindingAwareBroker.RpcRegistration<ToasterService> rpcRegistration = getRpcRegistryDependency() .addRpcImplementation(ToasterService.class, opendaylightToaster)

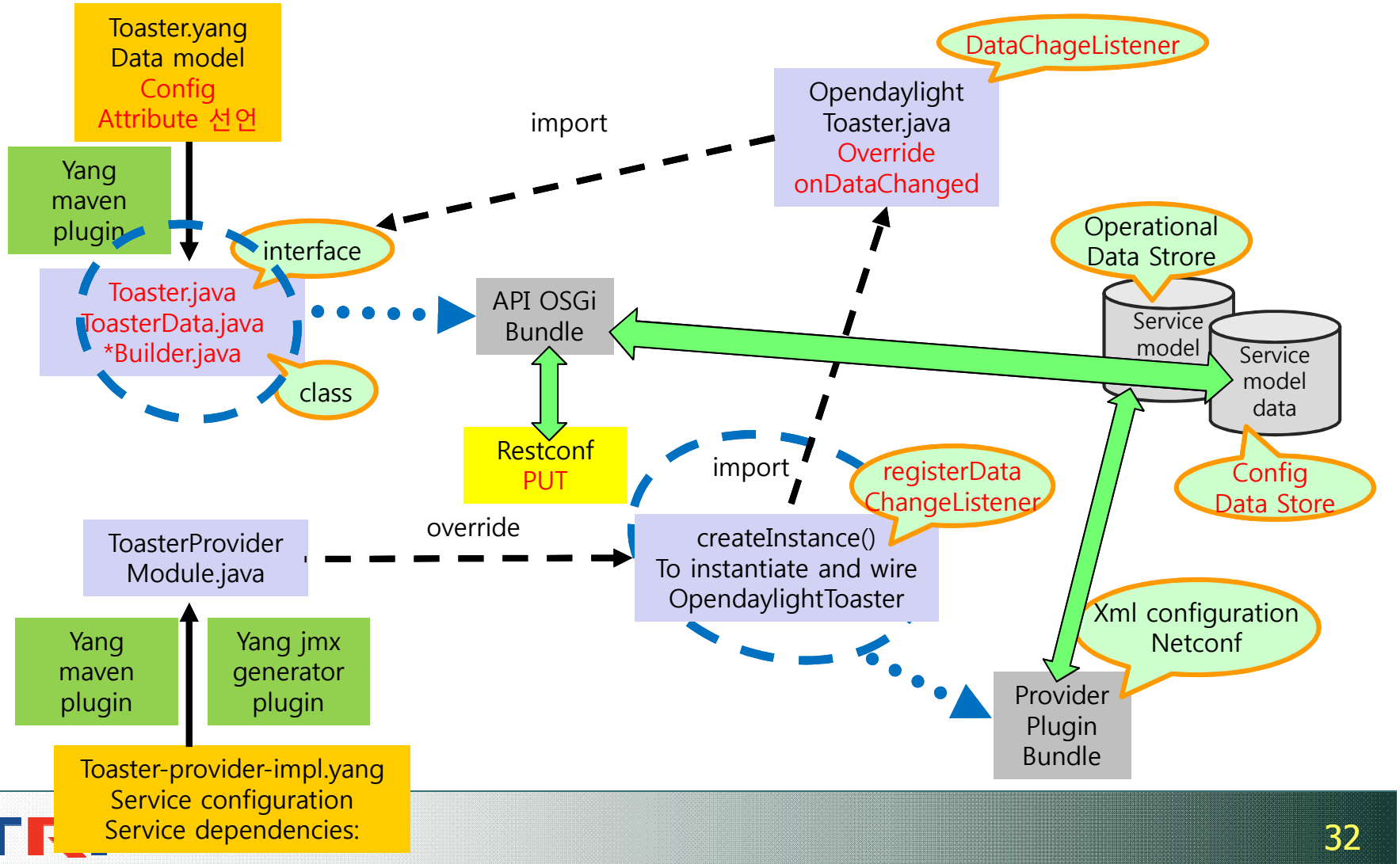
Toaster provider

- Invoke make-toast via Restconf

```
HTTP Method => POST
URL => http://localhost:8080/restconf/operations/toaster:make-toast
Header =>   Content-Type: application/yang.data+json
Body =>
{
  "input" :
  {
     "toaster:toasterDoneness" : "10",
     "toaster:toasterToastType":"wheat-bread"
  }
}
```

ODL MD-SAL

❑ Add configuration attribute



Configuration data 정의

- toaster/src/main/yang/toaster.yang

```
container toaster {
    ...

    leaf darknessFactor {
      type uint32;
      config true;
      default 1000;
      description
        "The darkness factor. Basically, the number of ms
    }

    ...
}
```

Toaster provider

- OpendaylightToaster.java
  - toaster-provider/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class OpendaylightToaster implements ToasterService, AutoCloseable, DataChangeListener
  - method overriding
    - onDataChanged(change)
- ToasterProviderModule.java
  - createInstance()
    - ListenerRegistration<DataChangeListener> dataChangeListenerRegistration = dataBrokerService .registerDataChangeListener(LogicalDatastoreType.CONFIGURATION, OpendaylightToaster.TOASTER_IID, opendaylightToaster, DataChangeScope.SUBTREE);

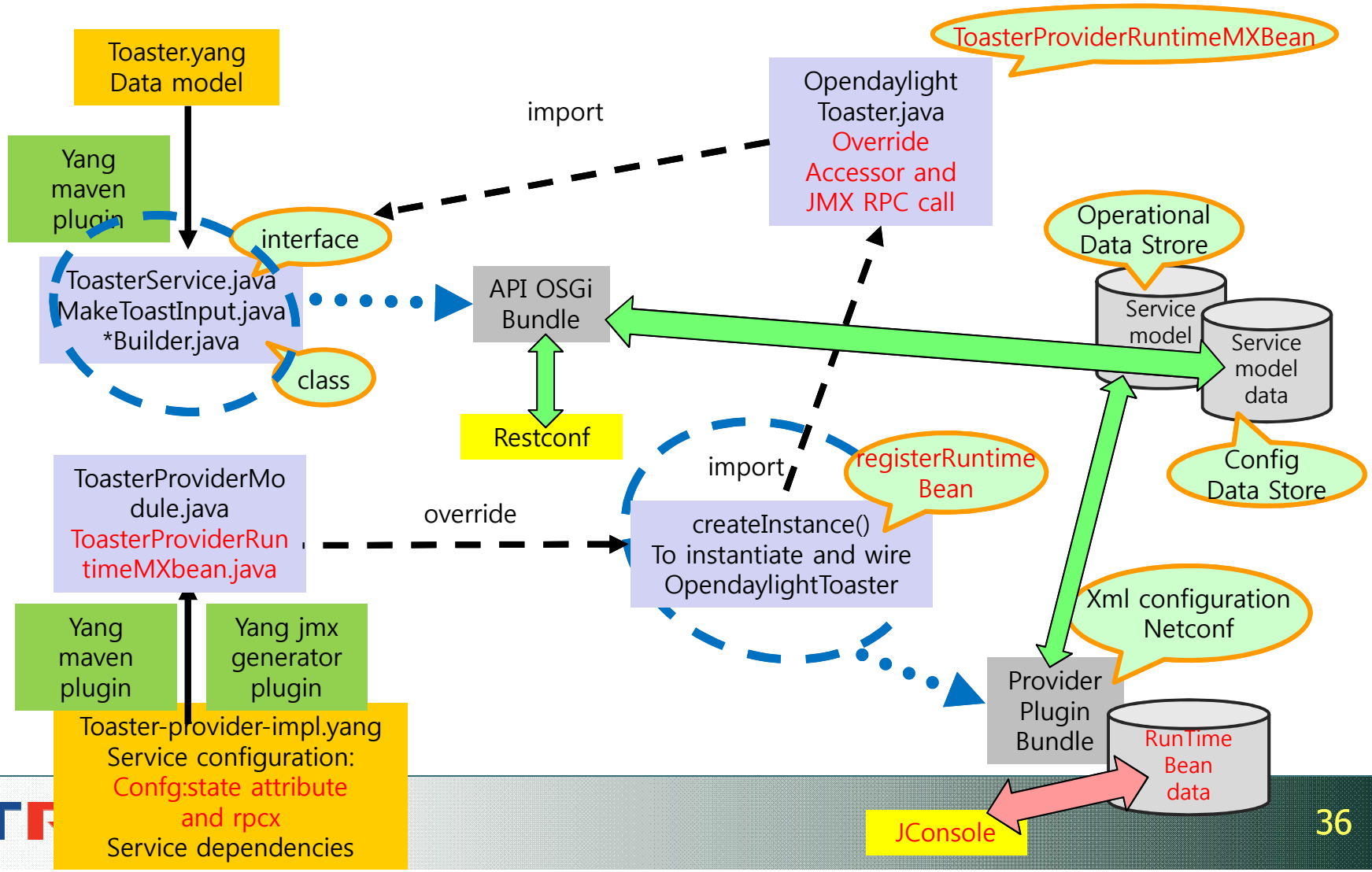
Toaster provider

- Changing the Darkness Factor

```
HTTP Method: PUT
URL:  http://localhost:8080/restconf/config/toaster:toaster
HEADER: content-type: application/yang.data+json
BODY:
{
  toaster:
  {
     darknessFactor: "2000"
  }
}
```

ODL MD-SAL

- JMX access for internal statistical purposes and troubleshooting



Toaster provider

❑ Wiring the OpendaylightToaster service

- Define toaster provider service yang configuration

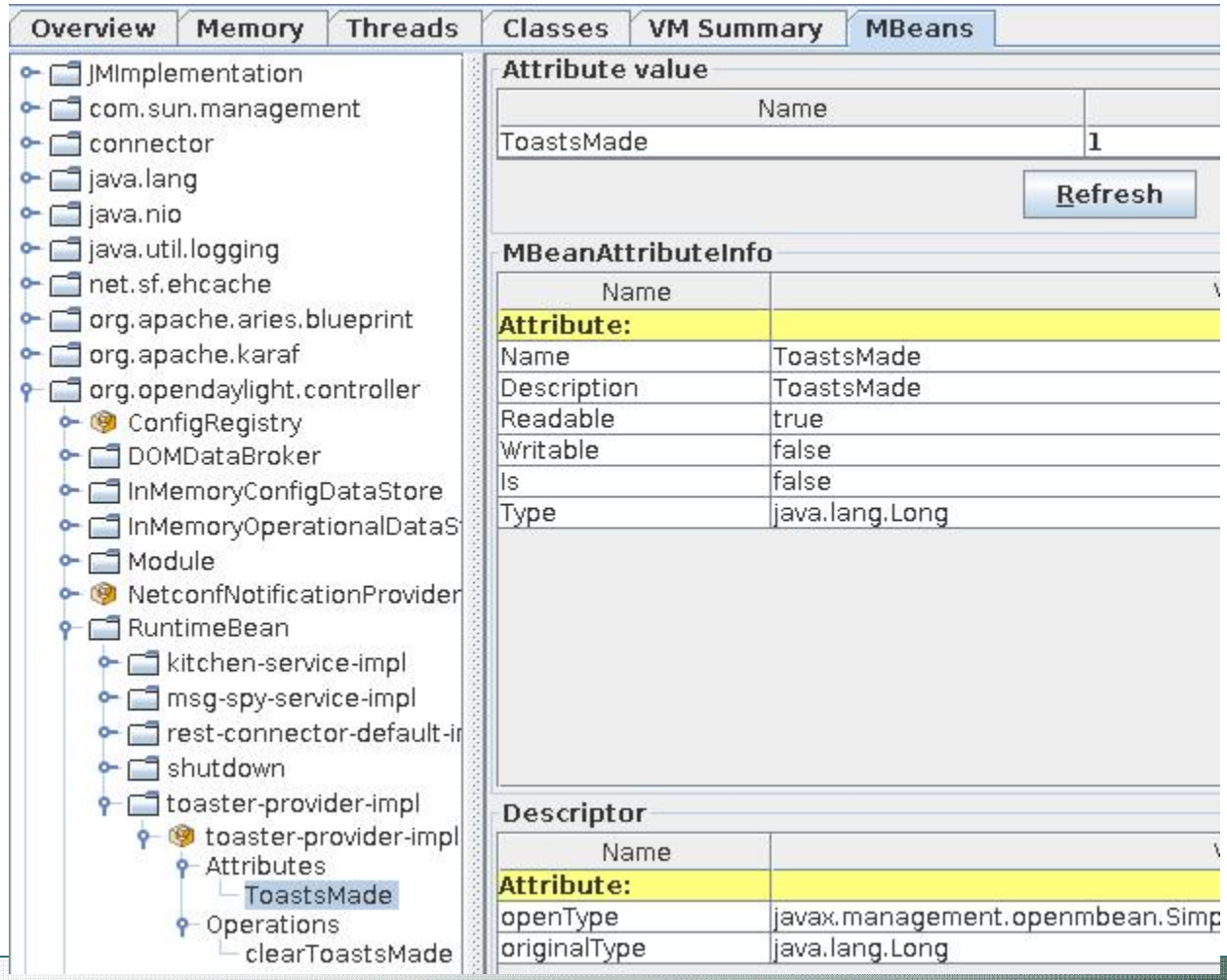
```
import rpc-context { prefix rpcx; revision-date 2013-06-17; }
...
augment "/config:modules/config:module/config:state" {
    case toaster-provider-impl {
        when "/config:modules/config:module/config:type = 'toaster-provider-impl'";

        leaf toasts-made {
            type uint32;
        }

        rpcx:rpc-context-instance "clear-toasts-made-rpc";
    }
}

identity clear-toasts-made-rpc;

rpc clear-toasts-made  {
    description
      "JMX call to clear the toasts-made counter.";

    input {
        uses rpcx:rpc-context-ref {
            refine context-instance {
                rpcx:rpc-context-instance clear-toasts-made-rpc;
            }
```

Toaster provider

- Wiring the OpendaylightToaster service
  - ToasterProviderRuntimeMXBean.java
    - toaster-provider/src/main/yang-gen-config/org/opendaylight/controller/config/yang/config/toaster_provider/impl
    - Generated from toaster-provider-impl.yang
    - interface ToasterProviderRuntimeMXBean extends org.opendaylight.controller.config.api.runtime.RuntimeBean
      - JMX bean interface that defines getToastsMade() to provide access to toasts-mode attribute and clearToastsMade() RPC
- OpendaylightToaster.java
  - toaster-provider/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class OpendaylightToaster implements ToasterService, AutoCloseable, DataChangeListener, ToasterProviderRuntimeMXBean
  - JMX RPC overriding
    - Void clearToastsMade()
  - Accessor method overriding
    - getToastsMade()
- ToasterProviderModule.java
  - createInstance()
    - ToasterProviderRuntimeRegistration runtimeReg = getRootRuntimeBeanRegistratorWrapper().register(opendaylightToaster);

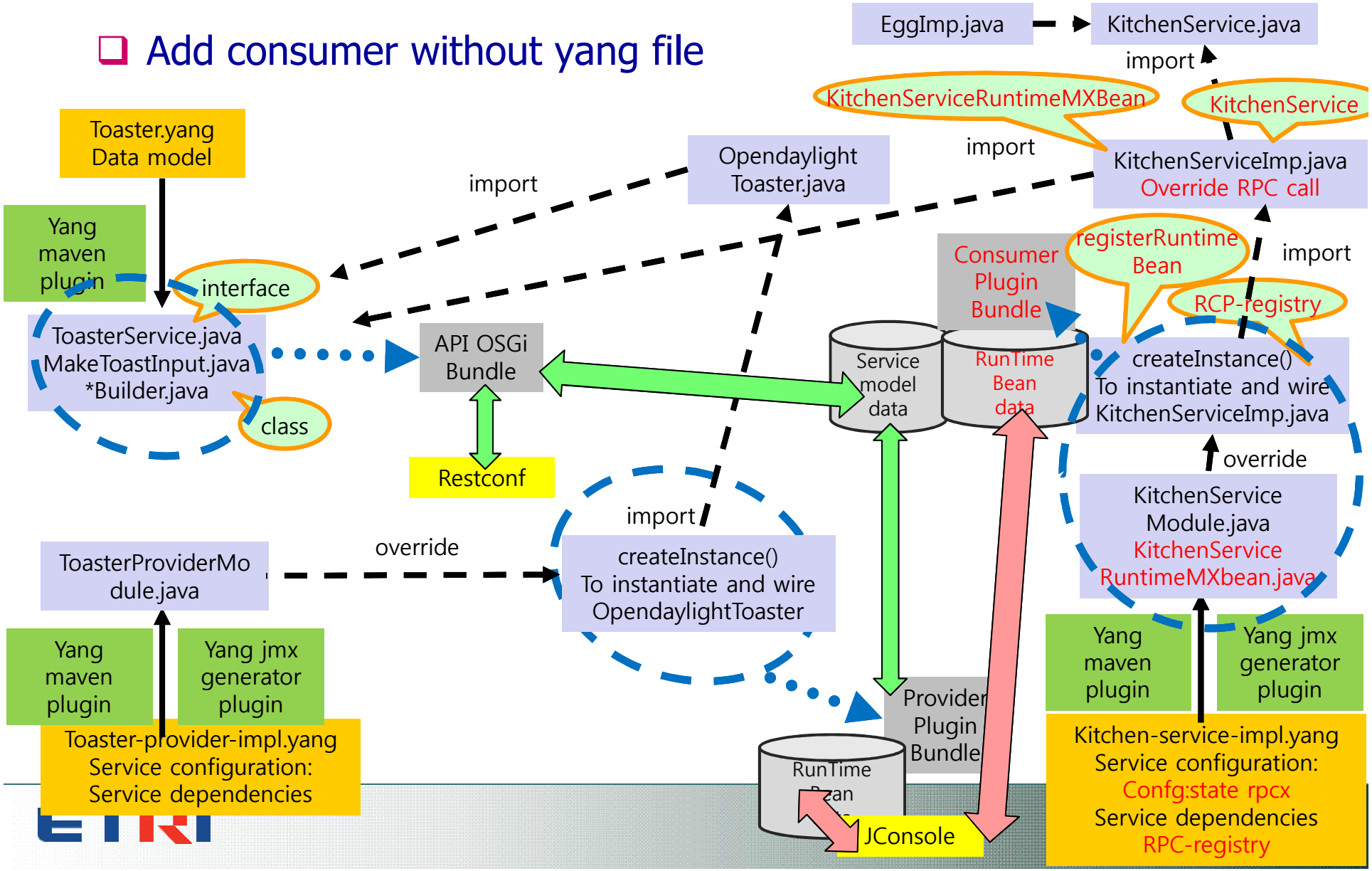
Toaster provider

- Jconsole: JMX access



ODL MD-SAL

- Add consumer without yang file



Toaster consumer

- Consumer without yang file
  - KitchenService.java
    - toaster-consumer/src/main/java/org/opendaylight/controller/sample/kitchen/api
    - public interface KitchenService
      - makeBreakfast
- KitchenServiceImpl.java
  - Be done manually
  - toaster-consumer/src/main/java/org/opendaylight/controller/sample/kitchen/impl
  - public class KitchenServiceImpl implements KitchenService
  - RPC overriding
    - makeBreakfast()
      - Using the makeToast() Rpc

Toaster consumer

```
module kitchen-service-impl {

    yang-version 1;
    namespace "urn:opendaylight:params:xml:ns:yang:controller:config:kitchen-service:impl";
    prefix "kitchen-service-impl";

    import config { prefix config; revision-date 2013-04-05; }
    import rpc-context { prefix rpcx; revision-date 2013-06-17; }

    import opendaylight-md-sal-binding { prefix mdsal; revision-date 2013-10-28; }

    description
        "This module contains the base YANG definitions for
        kitchen-service impl implementation.";

    revision "2014-01-31" {
        description
            "Initial revision.";
    }

    // This is the definition of kitchen service interface identity.
    identity kitchen-service {
        base "config:service-type";
        config:java-class "org.opendaylight.controller.sample.kitchen.api.KitchenService";
    }

    // This is the definition of kitchen service implementation module identity.
    identity kitchen-service-impl {
            base config:module-type;
            config:provided-service kitchen-service;
            config:java-name-prefix KitchenService;
    }

    augment "/config:modules/config:module/config:configuration" {
        case kitchen-service-impl {
            when "/config:modules/config:module/config:type = 'kitchen-service-impl'";

            container rpc-registry {
                uses config:service-ref {
                    refine type {
                        mandatory true;
                        config:required-identity mdsal:binding-rpc-registry;
```

Toaster consumer

- Wiring the KitchenService service
  - Provides the wiring with the rest of the ODL through the config-subsystem
    - KitchenServiceModule.java
    - This file is generated once and further completed manually
  - toaster-consumer/src/main/yang/kitchen-service-impl.yang
  - KitchenServiceModule.java
    - toaster-consumer/src/main/java/org/opendaylight/controller/config/yang/config/kitchen_service/impl
    - Generated from kitchen-service-impl.yang
    - public final class KitchenServiceModule extends AbstractKitchenServiceModule
      - Be mostly complete from code generation
    - Only the createInstance() method needs to be implemented to instantiate and wire the KitchenService
      - ToasterService toasterService = getRpcRegistryDependency().getRpcService(ToasterService.class);
      - kitchenService = new KitchenServiceImpl(toasterService);

Toaster consumer

Wiring the KitchenService service

- Define toaster consumer service yang configuration

```
augment "/config:modules/config:module/config:state" {
    case kitchen-service-impl {
        when "/config:modules/config:module/config:type = 'kitchen-service-impl'";

        rpcx:rpc-context-instance "make-scrambled-with-wheat-rpc";
    }
}

identity make-scrambled-with-wheat-rpc;

rpc make-scrambled-with-wheat  {
    description
      "Shortcut JMX call to make breakfast with scrambled eggs and wheat toast for testing.";

    input {
        uses rpcx:rpc-context-ref {
            refine context-instance {
                rpcx:rpc-context-instance make-scrambled-with-wheat-rpc;
            }
        }
    }
    output {
        leaf result {
            type boolean;
```

Toaster consumer

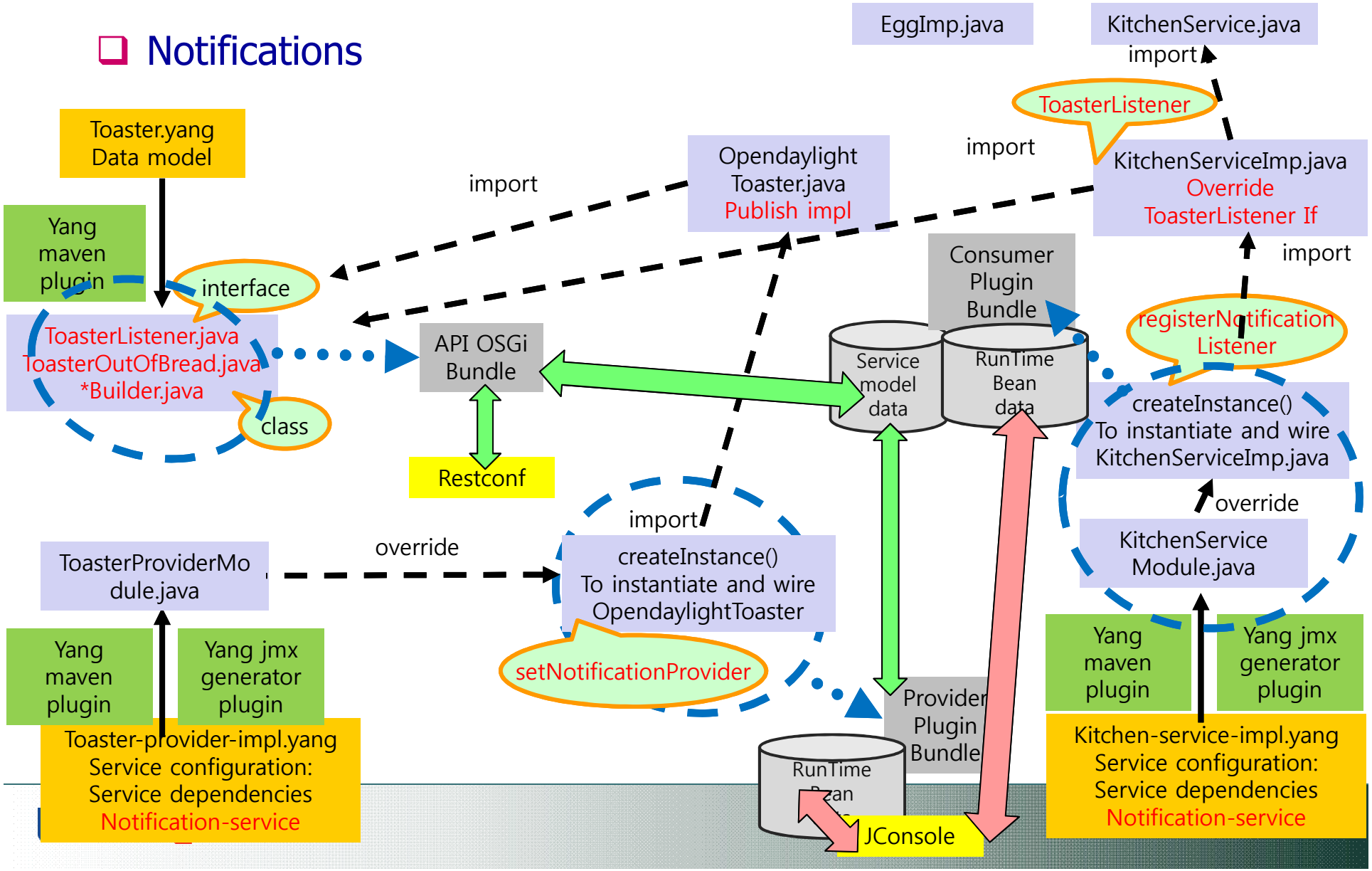
- KitchenServiceImpl.java
  - toaster-consumer/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class OpendaylightToaster implements KitchenService, KitchenServiceRuntimeMXBean
  - JMX RPC overriding
    - Void makeScrambledWithWheat()
- KitchenServiceModule.java
  - createInstance()
    - KitchenServiceRuntimeRegistration runtimeReg = getRootRuntimeBeanRegistratorWrapper().register(kitchenService);

Toaster consumer

- Jconsole: JMX access

ODL MD-SAL

❑ Notifications



Notification 정의

```
module toaster {
  ...
  rpc restock-toaster {
      description
        "Restocks the toaster with the amount of bread specified.";


      input {
          leaf amountOfBreadToStock {
              type uint32;
              description
                "Indicates the amount of bread to re-stock";
          }
      }
  }


  notification toasterOutOfBread {
    description
      "Indicates that the toaster has run of out bread.";
  }  // notification toasterOutOfStock


  notification toasterRestocked {
    description
      "Indicates that the toaster has run of out bread.";
    leaf amountOfBread {
      type uint32;
      description
        "Indicates the amount of bread that was re-stocked";
    }
  }  // notification toasterRestocked

}  // module toaster
```

Toaster provider

- OpendaylightToaster.java
  - toaster-provider/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class OpendaylightToaster implements ToasterService, AutoCloseable,DataChangeListener,ToasterProviderRuntimeMXBean
  - Notification publishing
    - private NotificationProviderService notificationProvider;
    - notificationProvider.publish(reStockedNotification);
    - notificationProvider.publish(new ToasterOutOfBreadBuilder().build());

Toaster provider

Wiring the OpendaylightToaster service

- Define toaster provider service yang configuration
  - toaster-provider/src/main/yang/toaster-provider-impl.yang

```
augment "/config:modules/config:module/config:configuration" {
   case toaster-consumer-impl {
       when "/config:modules/config:module/config:type = 'toaster-consumer-impl'";
       ...


       container notification-service {
           uses config:service-ref {
               refine type {
                   mandatory true;
                   config:required-identity mdsal:binding-notification-service;
               }
           }
       }
   }
}
```

- ToasterProviderModule.java
  - createInstance()
    - opendaylightToaster.setNotificationProvider(getNotificationServiceDependency());

Toaster consumer

- KitchenServiceImpl.java
  - toaster-consumer/src/main/java/org/opendaylight/controller/sample/toaster/provider
  - public class KitchenServiceImpl implements KitchenService, KitchenServiceRuntimeMXBean, ToasterListener
  - ToasterListener overriding
    - void onToasterOutOfBread(ToasterOutOfBread notification)
    - void onToasterRestocked(ToasterRestocked notification)

Toaster consumer

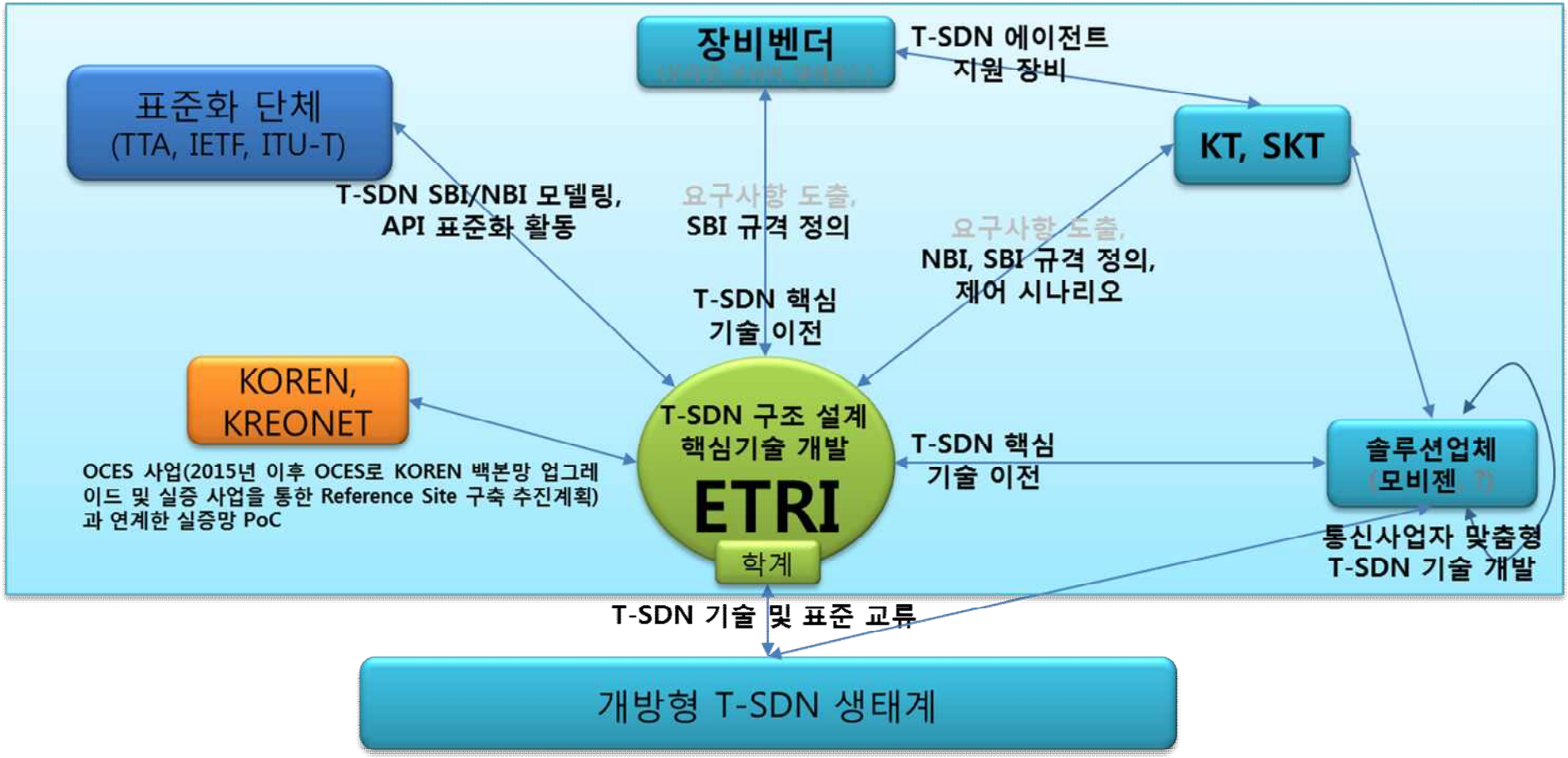
- Wiring the KitchenService service
  - Define toaster consumer service yang configuration

```
augment "/config:modules/config:module/config:configuration" {
    case kitchen-service-impl {
        when "/config:modules/config:module/config:type = 'kitchen-service-impl'";
        ...


        container notification-service {
            uses config:service-ref {
                refine type {
                    mandatory true;
                    config:required-identity mdsal:binding-notification-service;
                }
            }
        }
    }
}
```

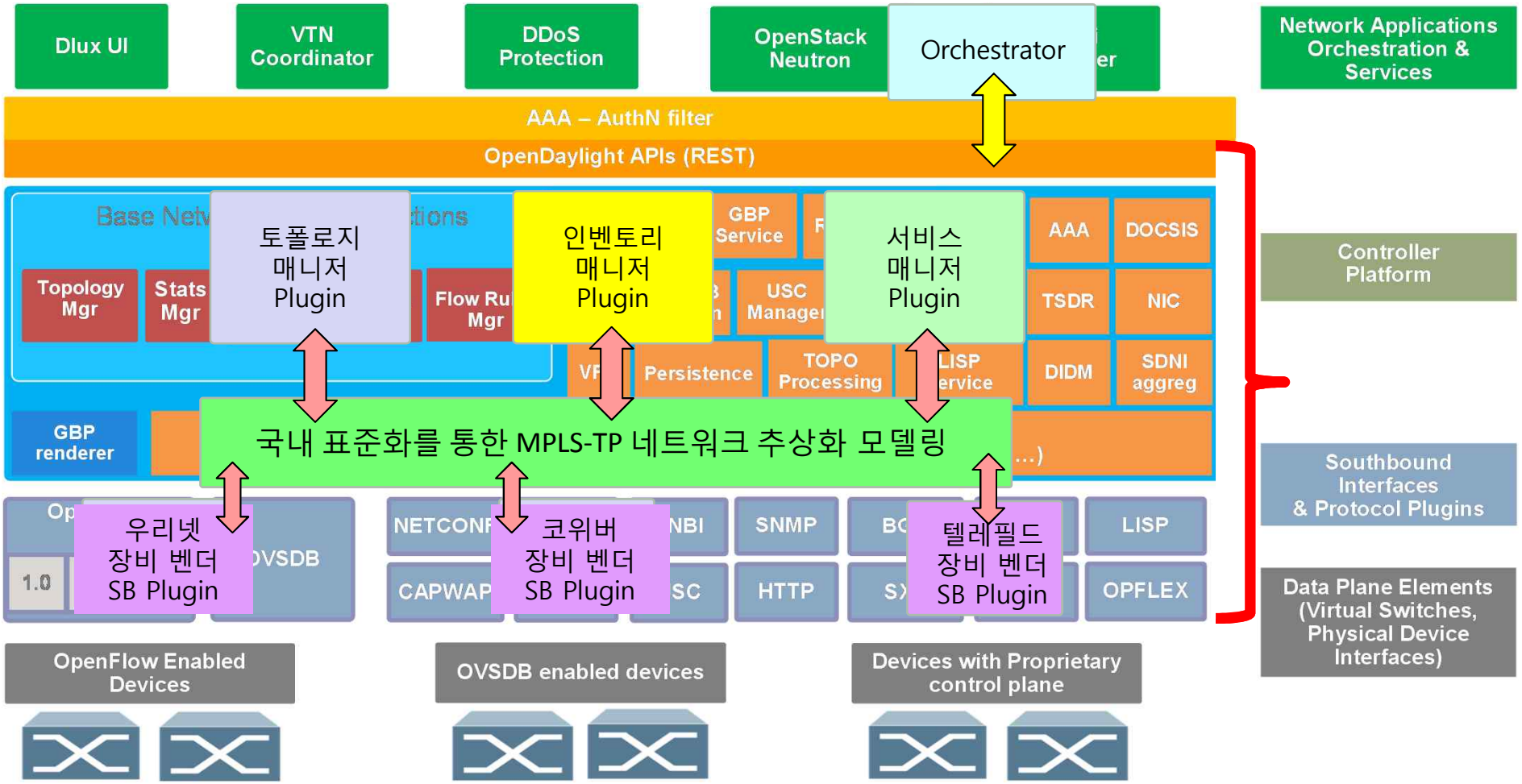
  - KitchenServiceModule.java
    - createInstance()
      - ListenerRegistration<NotificationListener> toasterListenerReg = getNotificationServiceDependency().registerNotificationListener(kitchenService);

ODL 기반 개발 사례: Transport SDN, MPLS-TP

국책사업: 스마트 네트워킹 핵심 기술 개발 사업 생태계



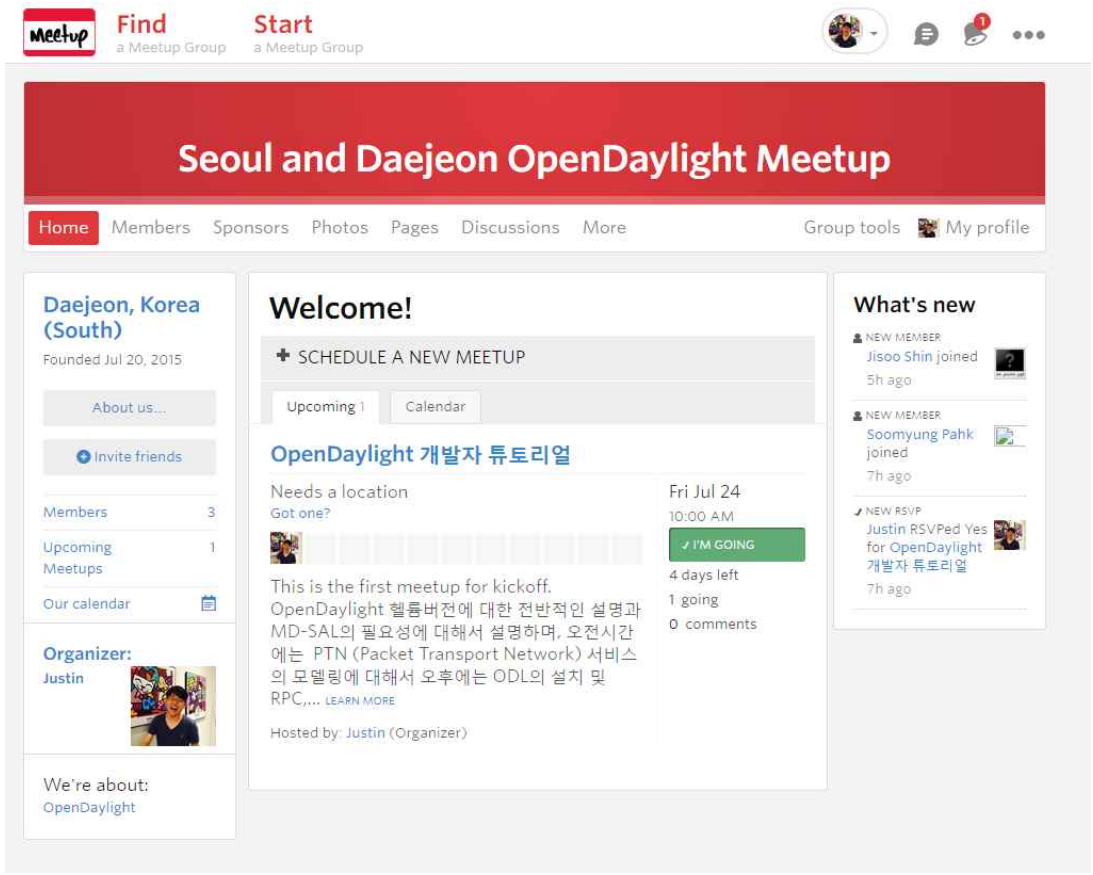
ODL 기반 개발 사례: Transport SDN, MPLS-TP



컨트롤러 활용 실습

ODLUG Seoul/Daejeon Meetup

- http://www.meetup.com/OpenDaylight-Korean-User-Group/

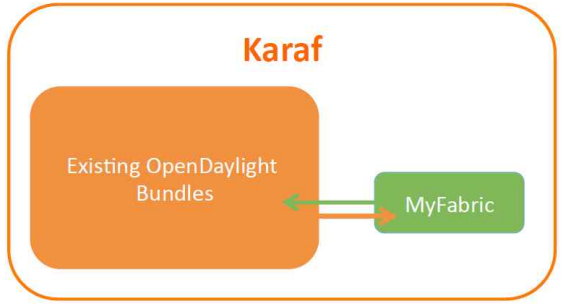


ODL 기반 프로그래밍 이란?

1. Model-View-Control 기법에 익숙해 지는 것
   - YANG Model 은 데이터, RPC, Notification 을 모델링
   - RESTconf는 View를 Yang모델을 기반으로 자동 생성
   - 실제로 Java를 이용하여 필요한 Code를 구현하는 부분

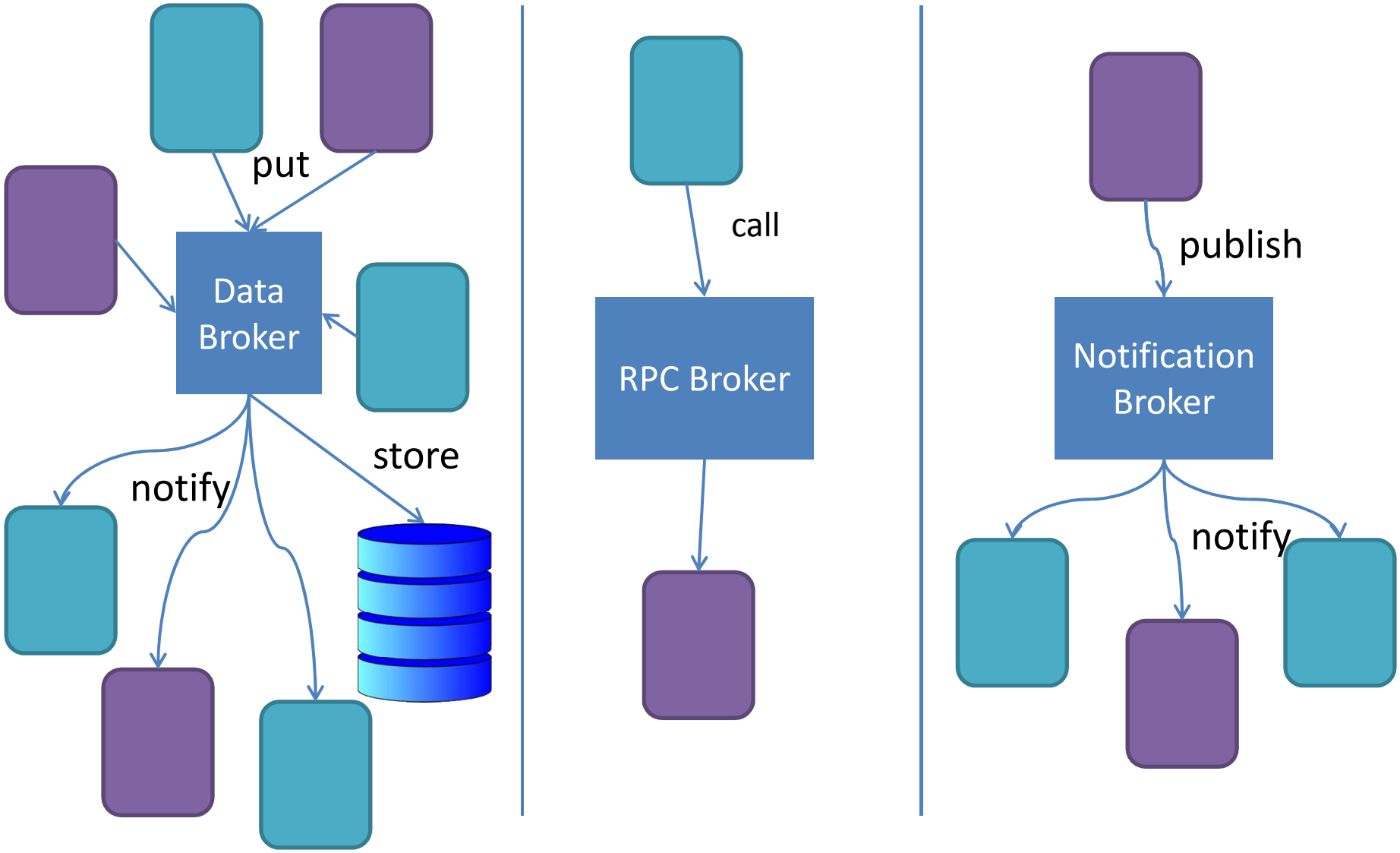
2. 플랫폼에 꼭 필요한 유틸에 익숙해 지는 것 (maven, git, Callable Interface, dependencies) and useful tools

3. 기존 프로젝트와 모듈의 재사용을 배우는 것
   - 다른 프로젝트의 코드를 고쳐서 사용하는 것이 아니라 바이너리 번들로 로딩하여 사용함



3 가지가 본 실습의 목적이다~

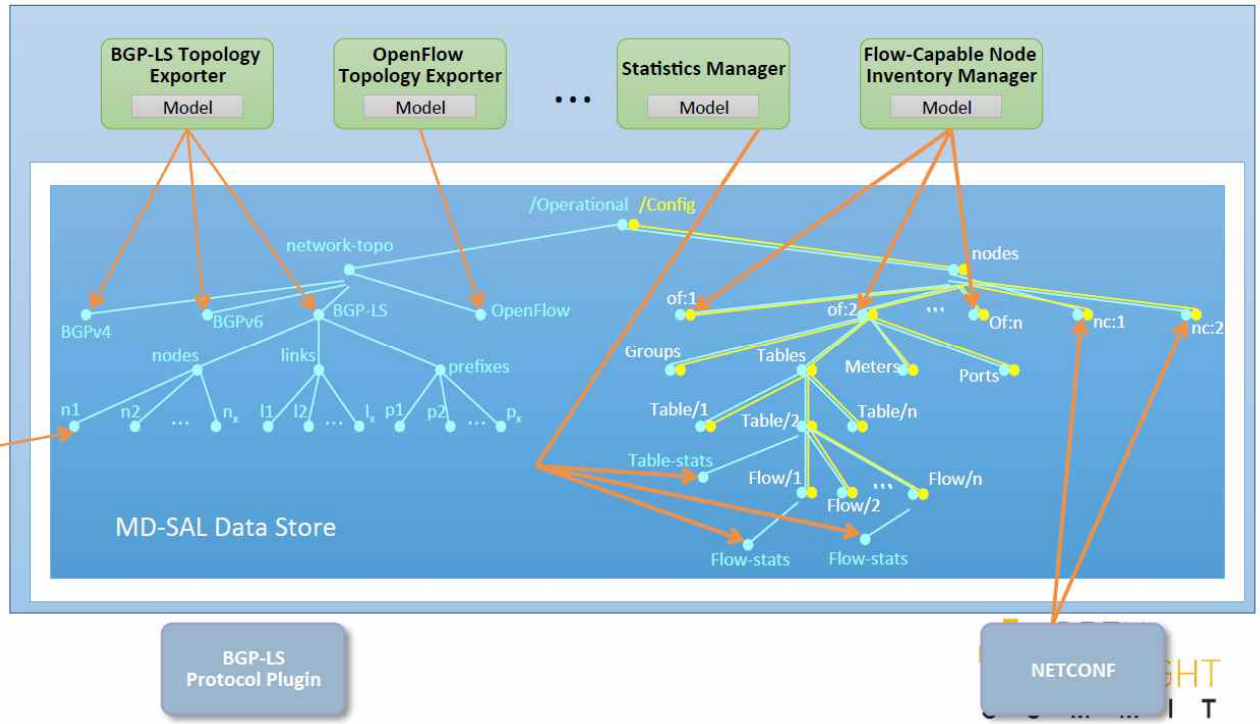
MD-SAL – 3 Brokers



data Broker

- MD-SAL은 동일 모델에 대하여 2 종류의 트리 존재
  - Config tree: 주로 응용개발자에 의해서 input 값으로 사용된다. RESTConf로 통해서 입력할 수 있다.
  - Operational tree: 주로 운영상에서 발생하는 값들을 기록하는 용도로 사용된다. 이 값은 RESTconf를 통해서 변경될 수 없다.

- Yang data is a tree
- Two Logical Data Stores:
  - Config
  - Operational
- Unified View
- InstanceIdentifier:
  - Pointer to a node



YANG 모델링

YANG

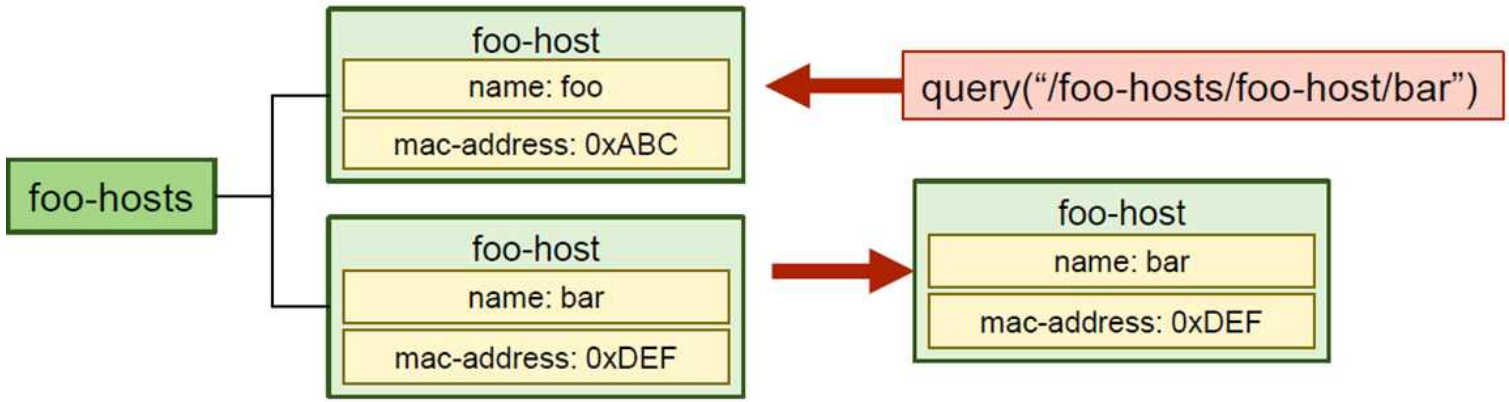
- 데이터 모델링 언어이며 NETCONF 설정 프로토콜
- 참고자료:
  - YANG introductory tutorial
  - RFC 6020 - YANG - A data modeling language for NETCONF

```
module hello {
     yang-version 1;
     namespace "urn:opendaylight:params:xml:ns:yang:hello";
     prefix "hello";
     revision "2015-01-05" {
          description "Initial revision of hello model";
     }
     rpc hello-world {
          input {
               leaf strin {
                    type string;
               }
     }    output {
               leaf greating {
                    type string;
               }
     }
}
```

package kr.re.etri.tsdn.yang.gen.v1. urn.opendaylight.params.xml.ns.yang.hello.rev150105;

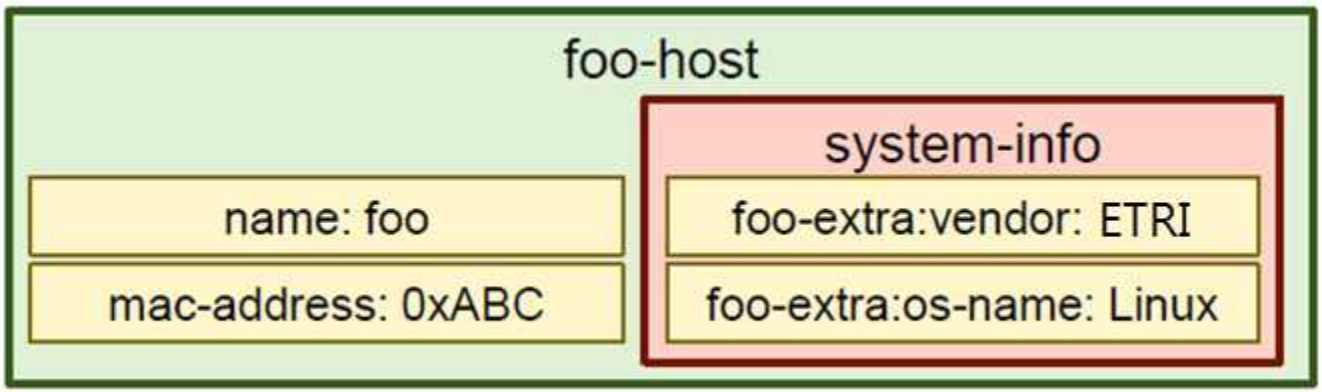
Yang to model 1/2

```
container foo-hosts {
      list foo-host {
            key “name”;

            leaf name {
                  type string;
            }
            leaf mac-address {
                  type yang:mac-address;
            }
      }
}
```



Yang to model 2/2

```
Module foo-extra {
      yang-version 1;
      namespace “urn:opendaylight:odlug:food:extra”;
      prefix fooext;

      import yang-ext { prefix etx; revision-date 2013-07-09; }
      import foo-model { prefix foo; revision-date 2015-01-24; }
      revision 2015-01-24 {}

      augment “/food:foo-hosts/foo:foo-host” {
      // yang-ext
      ext:augment-identifier “system-info”;

      leaf vendor {
            type string;
      }
      leaf os-name {
            type string;
      }
}
```



Yang: Java Bindings (1)

1) CamelCase

```
typedef foo-info {
  ......
}
```

```
public class FooInfo{
  ......
}
```

1) getter/setter

```
leaf foo-value {
  type string;
}
```

```
public String getFooValue() {
  ......
}
```

Yang: Java Bindings (2)

3) Grouping

```
grouping foo-host-info {
  leaf name {
    type string;
  }
  leaf mac-address {
    type yang:mac-address;
  }
}
```

```
package org.opendaylight.yang.gen.v1.urn.opendaylight.
  odlug.foo.rev150124;

import org.opendaylight.yangtools.yang.binding.DataObject;
import org.opendaylight.yang.gen.v1.urn.ietf.params.xml.ns.
  yang.ietf.yang.types.rev100924.MacAddress;

public interface FooHostInfo extends DataObject {
  HostName getName();
  MacAddress getMacAddress();
}
```

Yang: Java Bindings (3)

3) Container

```
container foo-hosts {
  list foo-host {
    key "name";
    uses foo-host-info;
  }
}
```

```
import org.opendaylight.yang.gen.v1.urn.opendaylight.odlug.foo.
  rev150124.foo.hosts.FooHost;
import org.opendaylight.yang.gen.v1.urn.opendaylight.odlug.foo.
  rev150124.foo.hosts.FooHostBuilder;

// Builder
HostName name = new HostName("foo");
MacAddress mac = new MacAddress("00:00:00:aa:bb:cc");
FooHostBuilder builder = new FooHostBuilder().
  setName(name).setMacAddress(mac);

// FooHost
FooHost host = builder.build();
```

Yang: Java Bindings (4)

```
rpc add-int32 {
  input {
    leaf augend { type int32; }
    leaf addend { type int32; }
  }
  output {
    leaf sum { type int32; }
  }
}
```

```
package org.opendaylight.yang.gen.v1.urn.opendaylight.odlug.
  foo.rev150124;

import java.util.concurrent.Future;
import org.opendaylight.yangtools.yang.binding.RpcService;
import org.opendaylight.yangtools.yang.common.RpcResult;

public interface FooModuleService extends RpcService {
  Future<RpcResult<AddInt32Output>> addInt32(AddInt32Input input);
}
```

Yang: Java Bindings (5)

```
notification add-int32-called {
   leaf augend { type int32; }
   leaf addend { type int32; }
   leaf sum { type int32; }
}
```

```
package org.opendaylight.yang.gen.v1.urn.opendaylight.odlug.foo.
  rev150124;

import org.opendaylight.yangtools.yang.binding.NotificationListener;

public interface FooModuleListener extends NotificationListener {
  void onAddInt32Called(AddInt32Called notification);
}
```

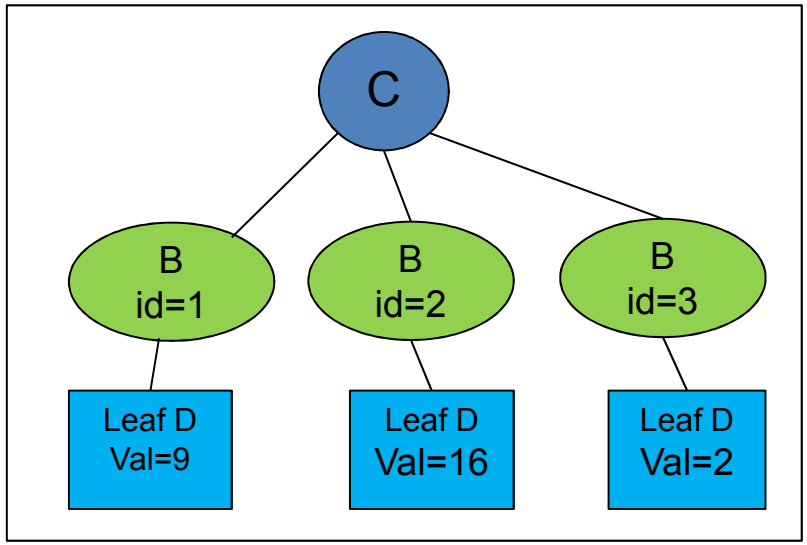
YANG

```
module hello {
      yang-version 1;
     namespace
"urn:opendaylight:params:xml:ns:yang:hello";
      prefix "hello";
      revision "2015-01-05" {
            description "Initial revision of
hello model";
     }
     rpc hello-world {
            input {
                  leaf strin {
                        type string;
                  }
      }    output {
                  leaf greating {
                        type string;
            }
      }
      container helloWorld {
            leaf counter {
                  type uint32;
                  config true;
            default 100;
            }
            leaf value {
                  type string;
                  config false;
            mandatory false;                  }
    }
}
```

```
module model1 {

    namespace "urn:model1";
    prefix model1;
    yang-version 1;

    revision 2015-04-06 {
        description "Initial revision";
    }

    grouping A {
        list B {
            key id;
            leaf id {
                type uint32;
            }
            leaf D {
                type uint32;
            }
        }
    }

    container C {
        uses A;
    }
}
```

MD-SAL Data Access

- Model-driven SAL은 OpenDaylight의 커널
- Model-driven SAL은 번들간의 모든 상태 교환을 중앙 집중적으로 관여함.
- YANG 모델을 런타임 시 로딩하여 트리 구조를 만듦



MD-SAL Data Access (contd.)

- 해당 모델을 maven으로 컴파일하면 다음과 같은 클래스들이 생성 된다. Model1Data. java, B.java, and C.java

- InstanceIdentifier 자식을 가리키는 포인터로 아래의 InstanceIdentifier는 첫 번째 노드를 가리킴

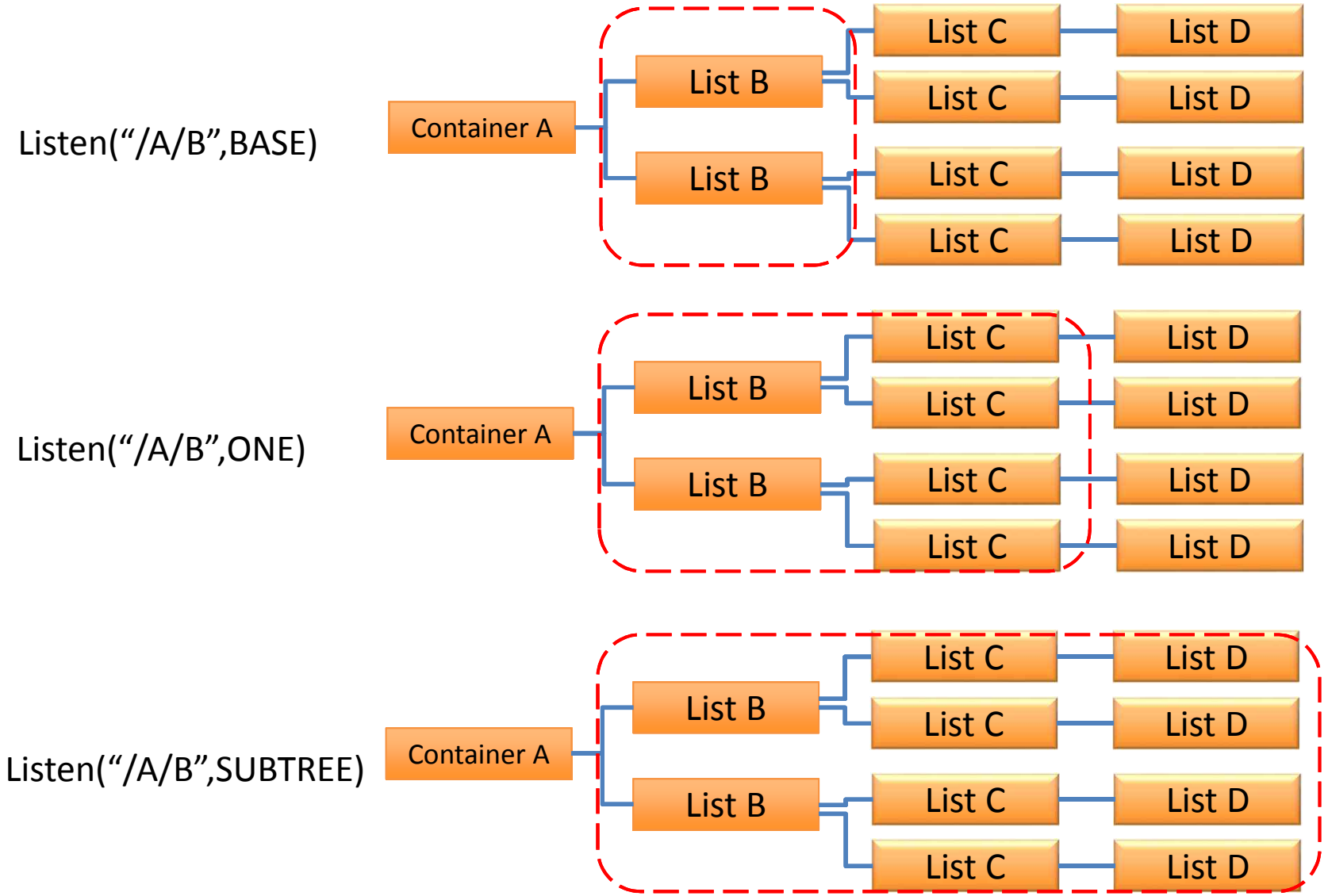
```
InstanceIdentifier iid = InstanceIdentifier.builder(C.class)
     .child(B.class, new BKey((long)1))
     .build();
```

- 저장소를 읽고 쓸 때는 ReadOnlyTransaction 또는 WriteTransaction 필요

```
B updatedB = new BBuilder().setD((long)19).build();
WriteTransaction modification = dataBroker.newWriteOnlyTransaction();
modification.merge(LogicalDataStoreType.CONFIGURATION, iid, updatedB, true);
modification.submit();
```

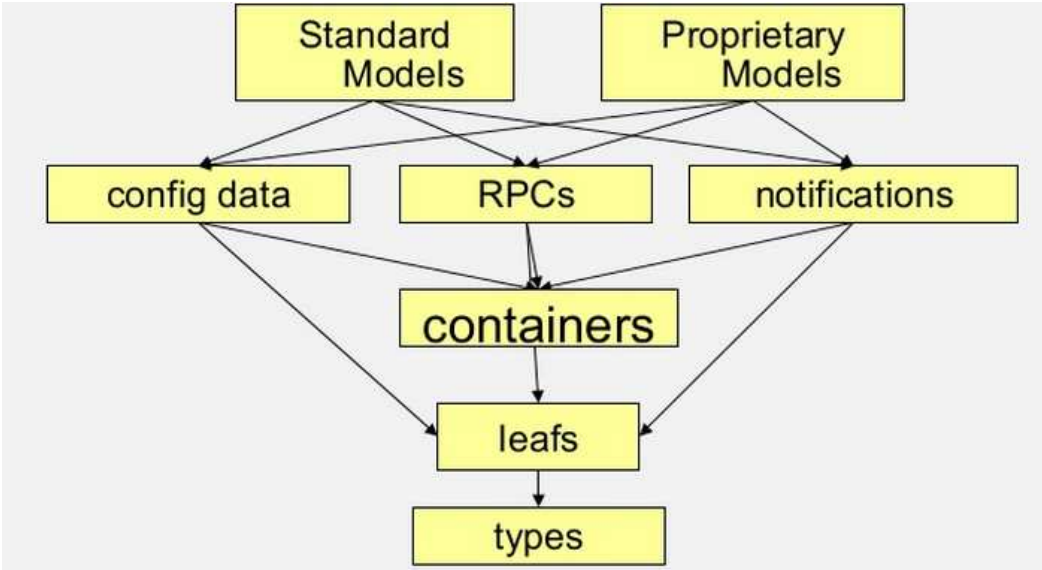
- Transaction은 배치처리 가능함

Data Broker DataChangeListener



YANG not restricted to Just Data Store

- Notifications:
  - Publish one or more notifications to registered listeners
- RPC:
  - Perform procedure call with input/output,
    without worrying about actual provider for that procedure



Using Notifications and RPCs

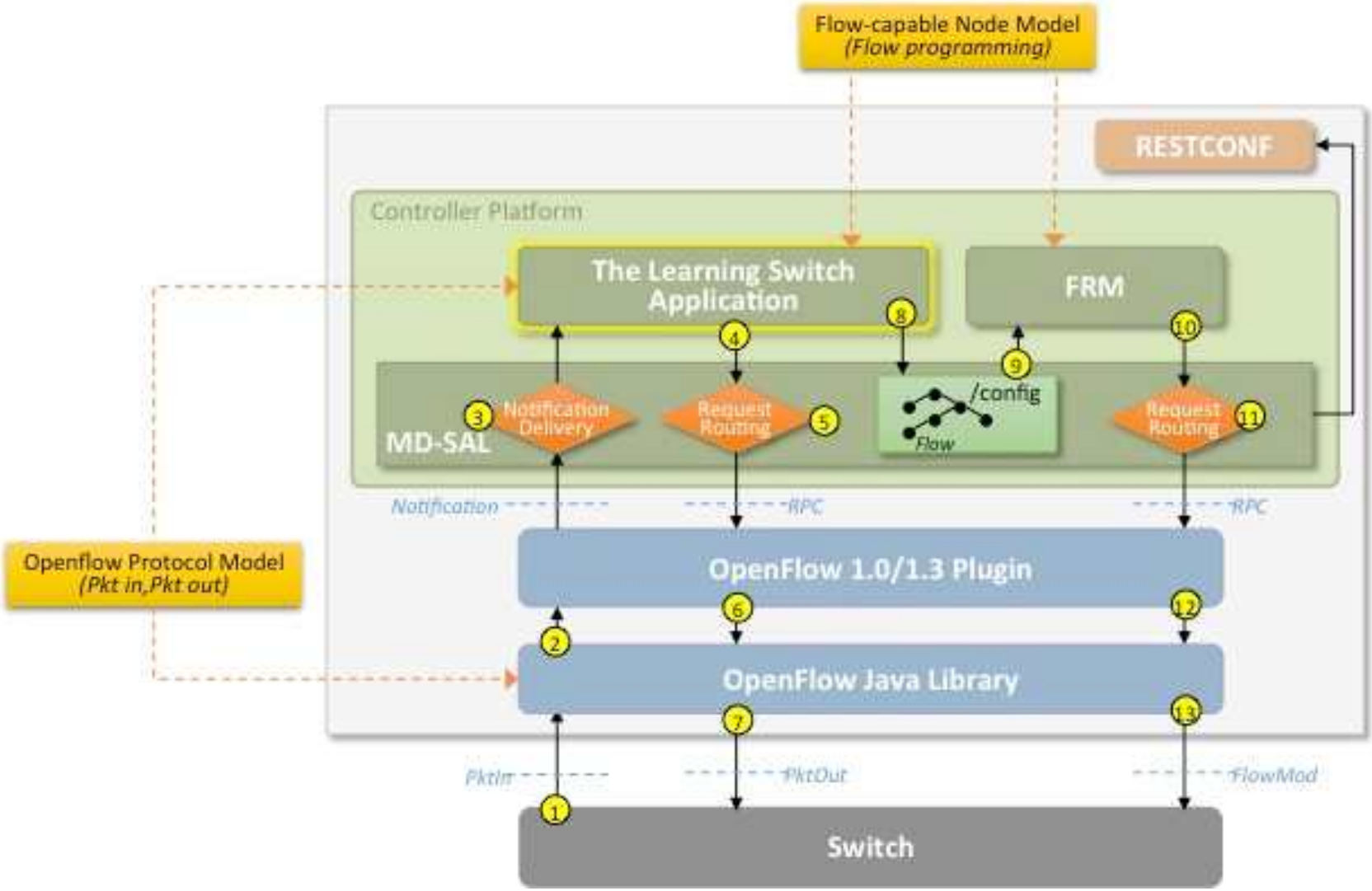
- Application developer usage for receiving notification and making RPC calls.

```
public class ConsumerImpl implements OpendaylightInventoryListener
    public ConsumerImpl(..) {
            notificationService.registerNotificationListener(this);
            this.salFlowService = rpcProviderRegistry.getRpcService(SalFlowService.class)
    }

    @Override
    public void onNodeUpdated(NodeUpdated nodeUpdated) {
            RemoveFlowInputBuilder flowBuilder = new RemoveFlowInputBuilder();
            flowBuilder.setBarrier(true);
            flowBuilder.setNode(NodeUtils.createNodeRef(nodeUpdated.getId()));
            salFlowService.removeFlow(flowBuilder.build());
    }
}
```

- **Note**: Whenever there is a change in the MD-SAL data store, you can receive a notification similar to the YANG defined notifications by implementing the DataChangeListener interface in a provider module

Example of Learning Switch

(Uses Data + Notifications + RPC)



OpenDaylight hands-on!

http://www.meetup.com/OpenDaylight-Korean-User-Group/

환경설정

- Ubuntu 15.04 또는 14.04 LTS 추천
- Java 1.7 설치
- Maven 3.31
- 쉘설치
  - https://github.com/t-sdn/
- OpenDaylight Gerrit 계정 생성 및 repository 설정

- Eclipse 설치
  - https://wiki.opendaylight.org/view/GettingStarted:_Eclipse
- 실습 Google docs
  - http://bit.ly/1HG1QcO

Useful ODL sites

- MD-SAL API
- https://developer.cisco.com/site/openSDN/documents/java-extension-services-api/sal-binding-api/#
- twitts
- https://twitter.com/OpenDaylightSDN
- gerrit
- https://git.opendaylight.org/gerrit/
- ask
- http://ask.opendaylight.org
- Bugzila
- https://bugs.opendaylight.org/buglist.cgi?bug_status=__open__&content=&no_redirect=1&order=Importance&product=&query_format=specific
- Wiki
- https://wiki.opendaylight.org/view/Main_Page
- ODLUG
- http://www.meetup.com/OpenDaylight-Korean-User-Group/
- IRC
- http://webchat.freenode.net/?channels=opendaylight
- Bitly
- https://bitly.com/

Reference

- http://www.slideshare.net/sdnhub/opendaylight-app-development-tutorial
- https://wiki.opendaylight.org/view/Main_Page
- Model Driven Service Abstraction Layer (MD-SAL) by NEC