TM FORUM OPEN API FUNDAMENTALS STUDENT NOTEBOOK

This document contains some key illustrations from the course to help you as you learn.

Please print this document and use it to take notes as you watch the videos. You should re-watch any material that you don't fully understand the first time through.

You should also study the TM Forum website where you will find documents that describe the TM Forum Open API Fundamentals in more detail.

The course and this document are not an exhaustive guide to the TM Forum Open APIs or the Framework.

Direct inquiries to the TM Forum office:
4 Century Drive
Suite 100
Parsippany, NJ 07054 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page www.tmforum.org

tmforum
connecting digital ecosystems

Module 1:
Introduction to REST APIs

Presented by: Joann O'Brien

Application Programming Interface

Provide interfaces to computer system for machine consumption

Allows applications (software programs) to talk to one another

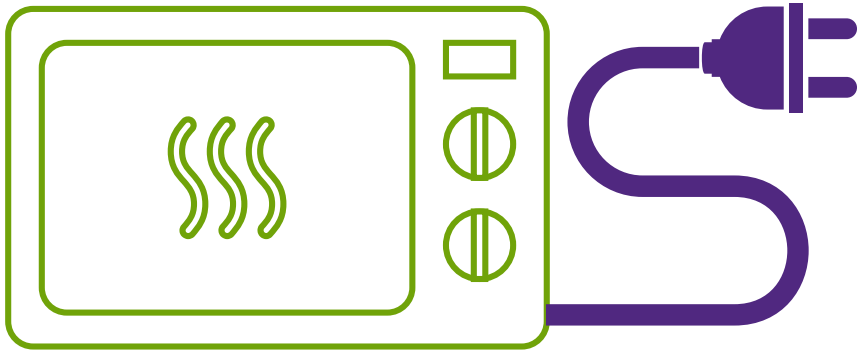
Airline Reservation System

GUI

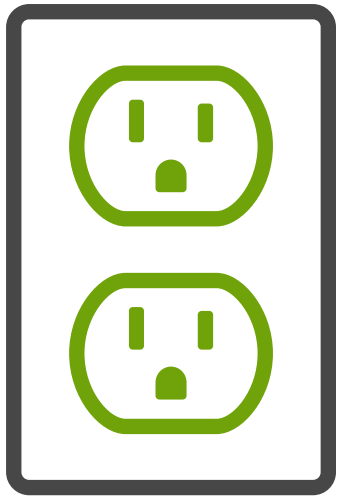
API

Machine – Application Software

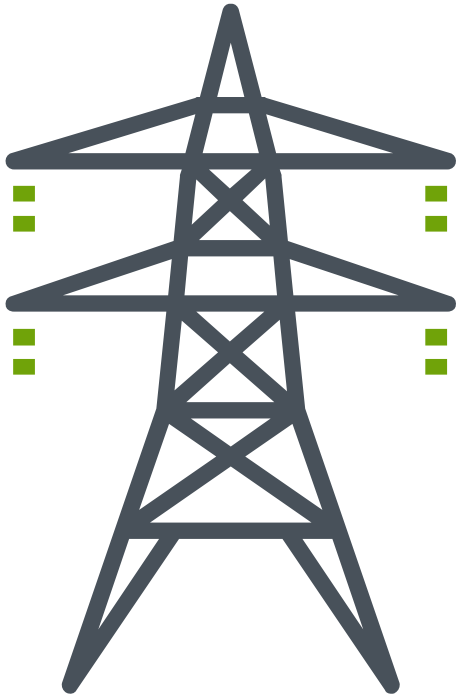
APIs are like Sockets!



Consumer

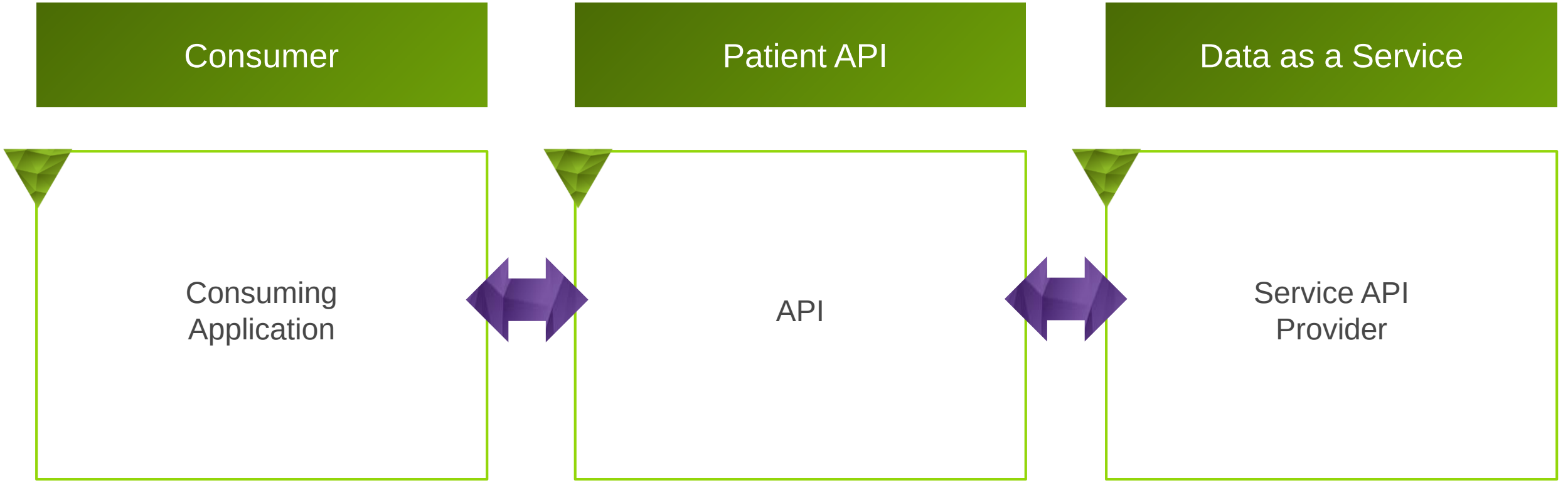


Wall Socket
Interface = API



Electricity
Service

APIs Enable Applications

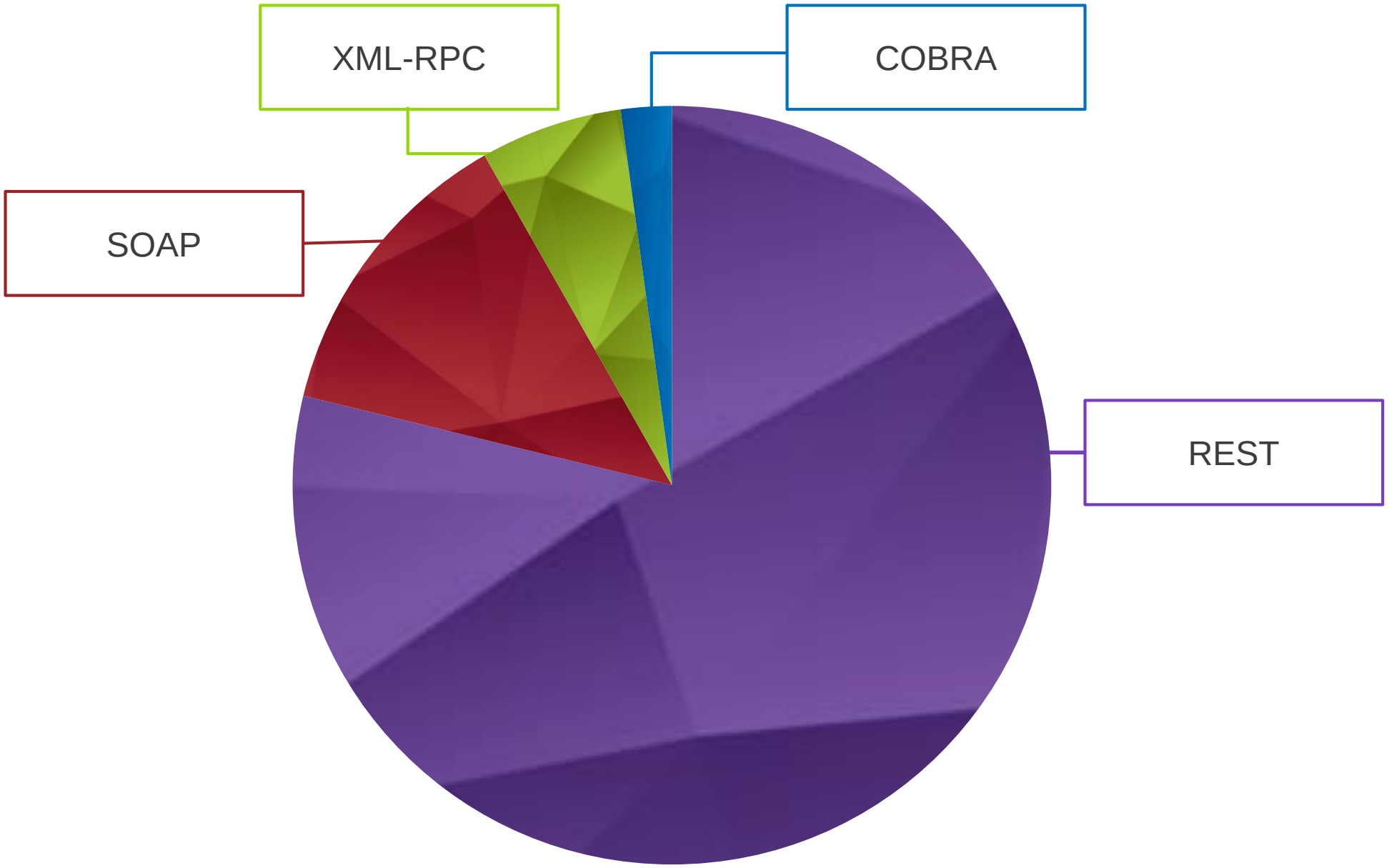



API
CONTRACT
A fair agreement with a team you can trust

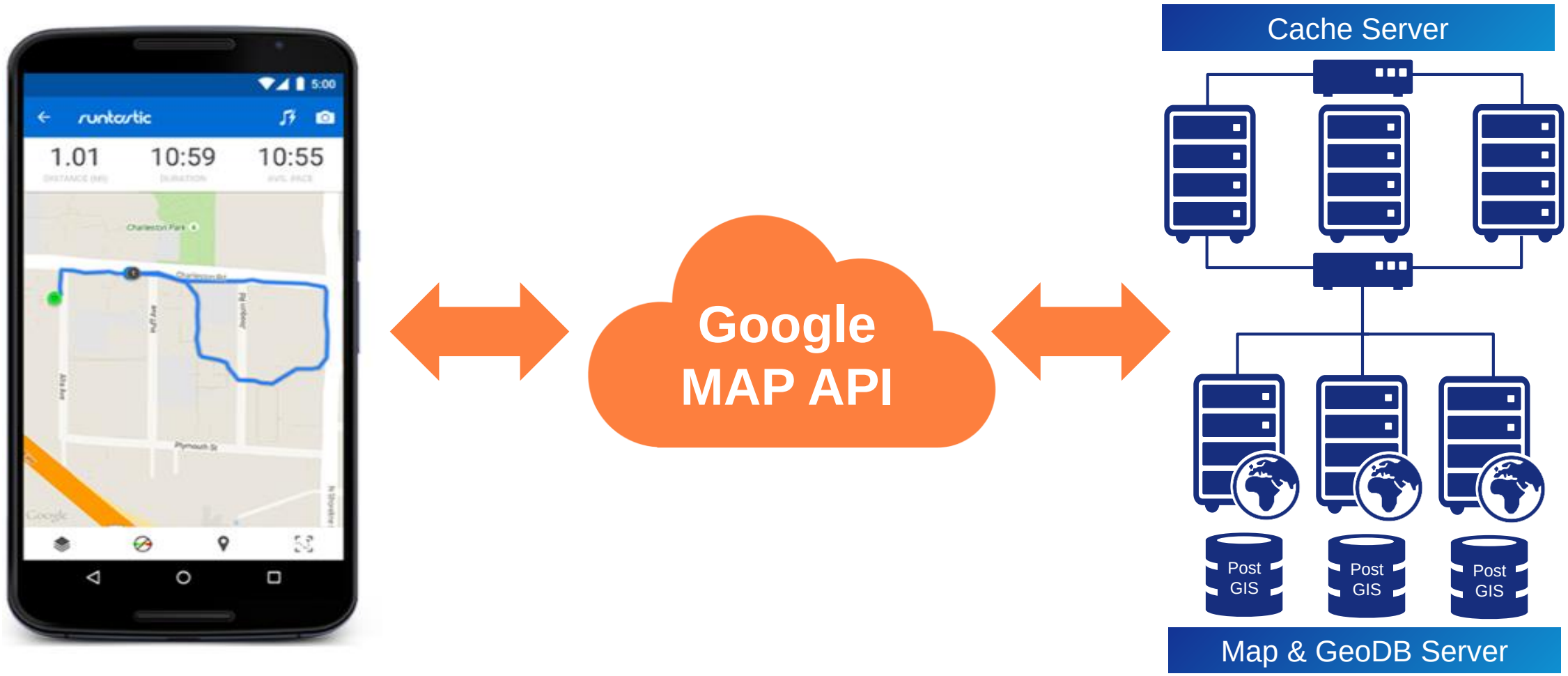
Web API

> A server-side web API is a programmatic interface consisting of one or more publicly exposed endpoints to a defined request - response message system, typically expressed in JSON or XML, which is exposed via the web - most commonly by means of an HTTP-based web server.
>
> *Wikipedia*

Web APIs Protocols and Styles



Google Map API Example



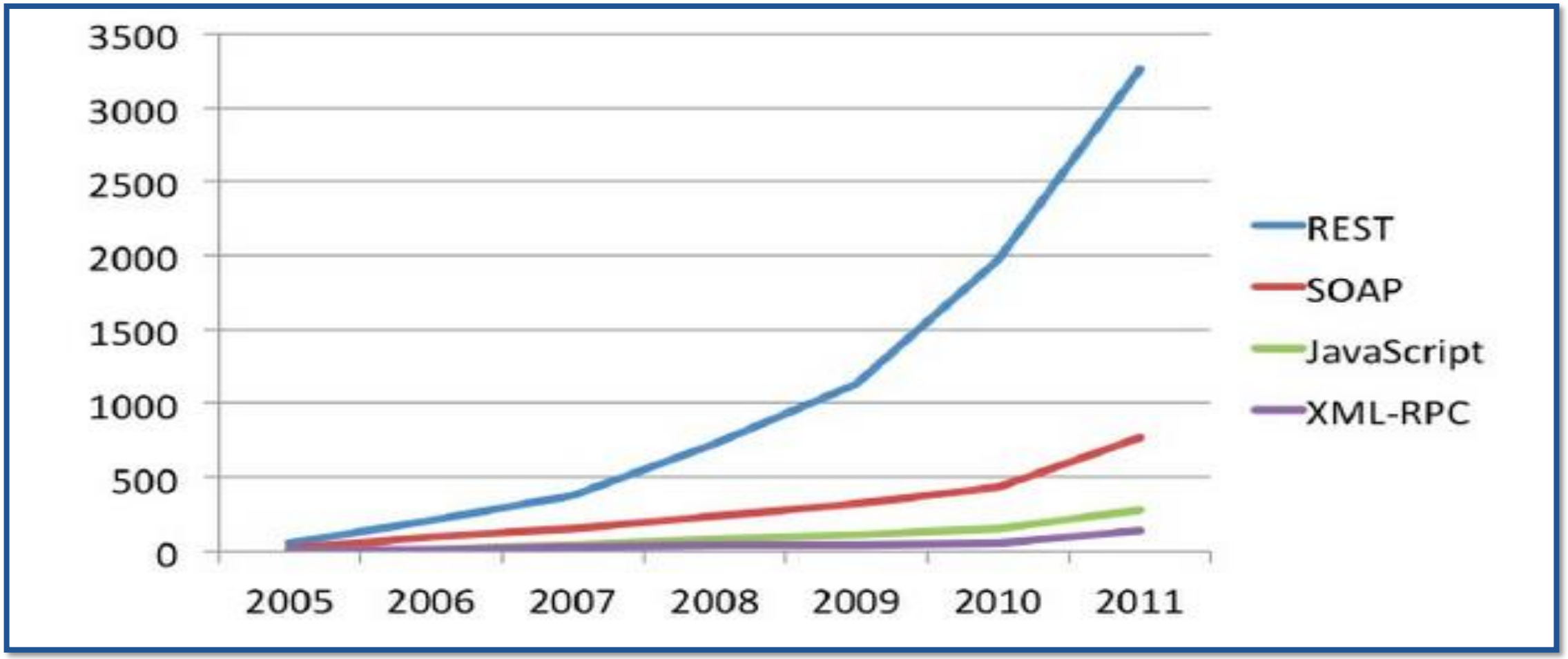
SOAP Web Services

- use XML messages that follow the Simple Object Access Protocol (SOAP) standard
- written in the Web Services Description Language (WSDL)
- TM Forum TIP, MTOSI and OSS/J APIs

REST Web Services

- better integrated with HTTP than SOAP-based services
- defined by Swagger or RAML
- payload in JSON (Java Scripting Object Notation)
- TM Forum Open APIs

API Protocols and styles



What is REST?

REpresentational **S**tate **T**ransfer

Architectural style of client-server applications not a technology

Centred around the transfer of representations of resources through requests and responses

Architecture is defined by REST constraints

Resources and Domain Entities

Resources represent the domain entities managed by the application

A resource can be a trouble Ticket, a Logical Port , an Order, a Task etc...

Resources are acted upon by the API operations

Using the Uniform Contract Verbs POST, GET, PUT, DELETE, PATCH, etc....

Operations on Resources

Affect the lifecycle and state of the corresponding entities

REST Constraints

Definition: REST constraints are design rules that are applied to establish the distinct characteristics of the REST architectural style.

- Client-Server
- Stateless
- Cache
- Interface/Uniform Contract
- Layered System
- Code-on-Demand

API REST Characteristics

- Everything is a Resource
- Resources can have multiple representations
- Suitable for CRUD (Create/Read/Update/Delete)
- Link things together (Hypermedia)
- Stateless Communication
- Cacheable

Uniform Contract Methods in HTTP

The high level methods provided by the uniform contract are used to access specific resources, each of which represents a custom function or logic that the rest service can provide.

HTTP set of generic methods

- GET, PUT, POST, PATCH, DELETE, HEAD and OPTIONS
- Pre-defined in the HTTP specification

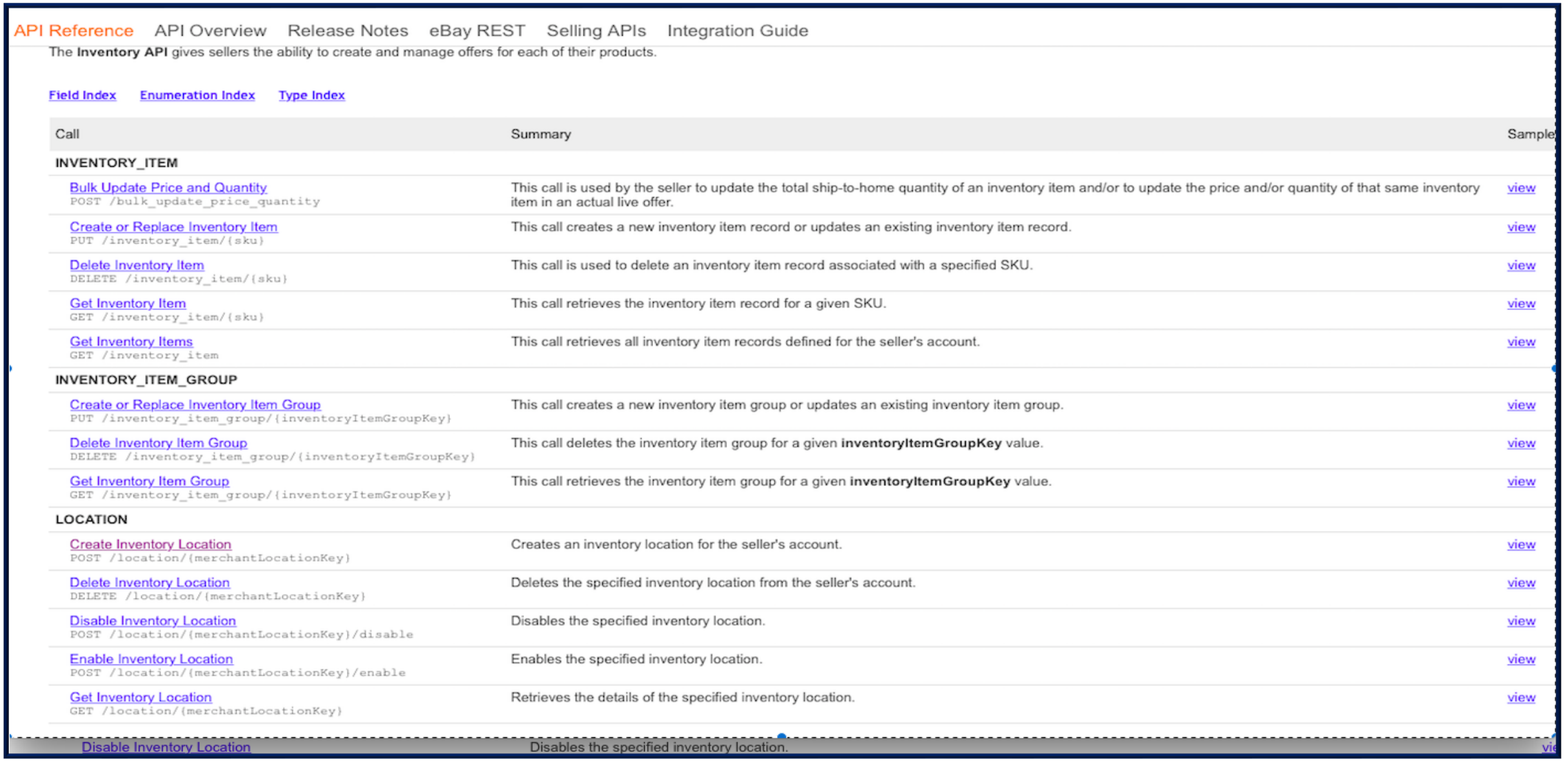
Complete protocol interactions

- A set of response codes
- A set of error codes
- Plus syntax for expressing various parameters that can be encoded in HTTP messages

REST API Operations

Operation on Entities	Uniform API Operation	Description
Query Entities	GET Resource	GET must be used to retrieve a representation of a resource
Create Entity	POST Resource	POST must be used to create a new resource
Partial Update of an Entity	PATCH Resource	PATCH must be used to partially update a resource
Complete Update of an Entity	PUT Resource	PUT must be used to completely update a resource identified by its resource URI
Remove an Entity	DELETE Resource	DELETE must be used to remove a resource
Execute an Action on an Entity	POST on TASK Resource	POST must be used to execute TASK resources
Other Request Methods	POST on TASK Resource	GET and POST must not be used to tunnel other request methods

The eBay APIs operates on Inventory Item and Location Resources using the well defined uniform contract verbs POST, PATCH, GET etc..



API Reference API Overview Release Notes eBay REST Selling APIs Integration Guide

The **Inventory API** gives sellers the ability to create and manage offers for each of their products.

Field Index Enumeration Index Type Index

Call	Summary	Sample
INVENTORY_ITEM		
Bulk Update Price and Quantity `POST /bulk_update_price_quantity`	This call is used by the seller to update the total ship-to-home quantity of an inventory item and/or to update the price and/or quantity of that same inventory item in an actual live offer.	view
Create or Replace Inventory Item `PUT /inventory_item/{sku}`	This call creates a new inventory item record or updates an existing inventory item record.	view
Delete Inventory Item `DELETE /inventory_item/{sku}`	This call is used to delete an inventory item record associated with a specified SKU.	view
Get Inventory Item `GET /inventory_item/{sku}`	This call retrieves the inventory item record for a given SKU.	view
Get Inventory Items `GET /inventory_item`	This call retrieves all inventory item records defined for the seller's account.	view
INVENTORY_ITEM_GROUP		
Create or Replace Inventory Item Group `PUT /inventory_item_group/{inventoryItemGroupKey}`	This call creates a new inventory item group or updates an existing inventory item group.	view
Delete Inventory Item Group `DELETE /inventory_item_group/{inventoryItemGroupKey}`	This call deletes the inventory item group for a given **inventoryItemGroupKey** value.	view
Get Inventory Item Group `GET /inventory_item_group/{inventoryItemGroupKey}`	This call retrieves the inventory item group for a given **inventoryItemGroupKey** value.	view
LOCATION		
Create Inventory Location `POST /location/{merchantLocationKey}`	Creates an inventory location for the seller's account.	view
Delete Inventory Location `DELETE /location/{merchantLocationKey}`	Deletes the specified inventory location from the seller's account.	view
Disable Inventory Location `POST /location/{merchantLocationKey}/disable`	Disables the specified inventory location.	view
Enable Inventory Location `POST /location/{merchantLocationKey}/enable`	Enables the specified inventory location.	view
Get Inventory Location `GET /location/{merchantLocationKey}`	Retrieves the details of the specified inventory location.	view

Disable Inventory Location Disables the specified inventory location. vie

eBay API Example with JSON

The SKU value associated with the inventory item to retrieve is input into the end of the call URI. In this case, the SKU value for the inventory item is `GP-Cam-01`.

URL format. See also the non-wrapped version of this URL.

```
GET https://api.ebay.com/sell/inventory/v1/inventory_item/GP-Cam-01
```

Output

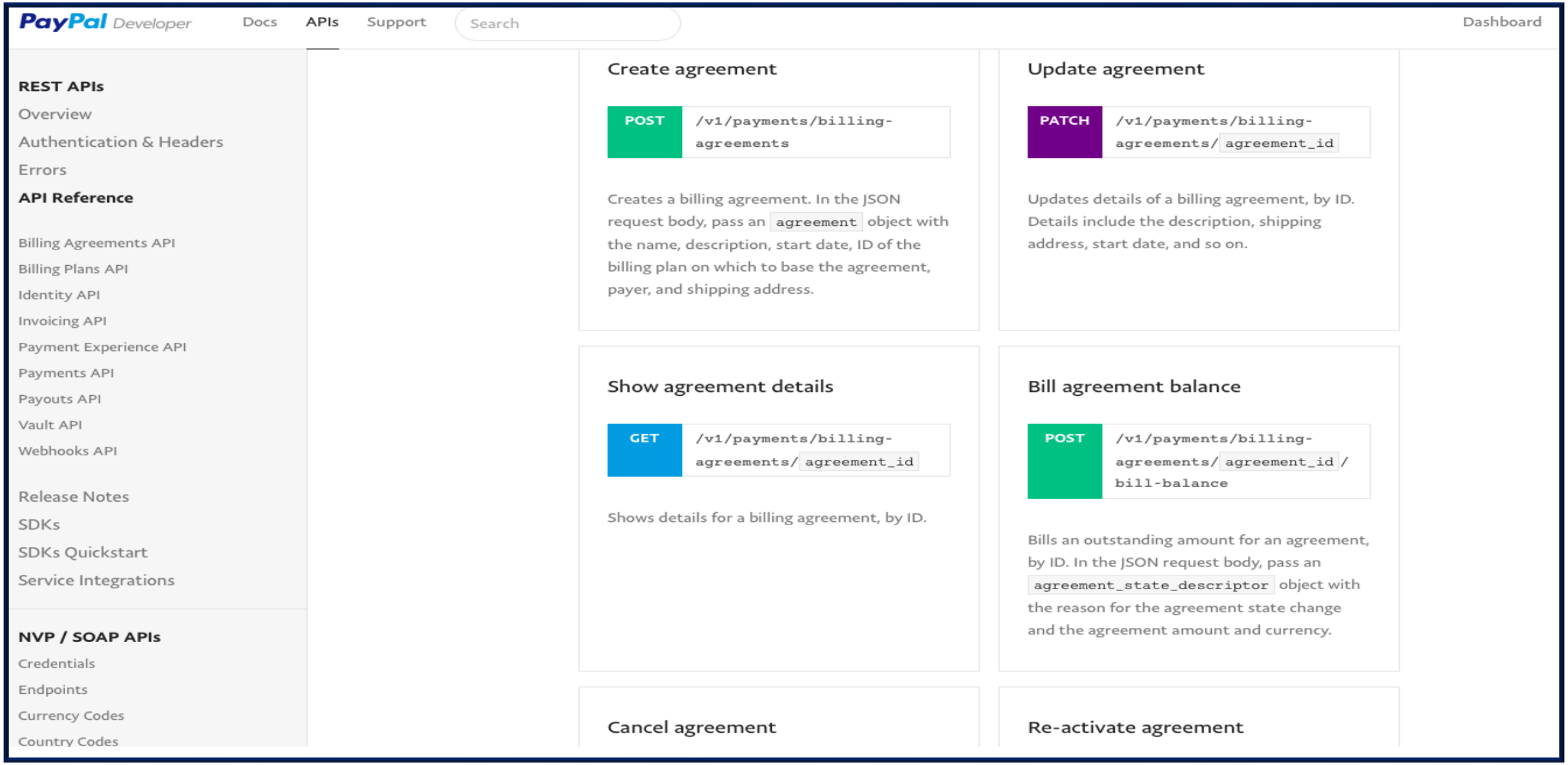
Based on the call response, the product identified as `GP-Cam-01`, has a total, ship-to-home quantity of `50`.

The condition of the product is `NEW` and there are three images associated with the inventory item. The **product** container also consists of the product's title, description, and item specifics.

JSON format.


```
{
"sku" : "GP-Cam-01",
"availability":
    {
    "shipToLocationAvailability":
        {
        "quantity": 50
        }
    },
"condition": "NEW",
"product":
    {
    "title": "GoPro Hero4 Helmet Cam"
    "description": "New GoPro Hero4 Helmet Cam. Unopened box.",
    "aspects": {
        "Brand" :["GoPro"],
        "Type" : ["Helmet/Action"],
        "Storage Type" : ["Removable"],
        "Recording Definition" : ["High Definition"],
        "Media Format" : ["Flash Drive (SSD)"],
        "Optical Zoom" : ["10x"]
      },
    "imageUrls": [
        "http://i.ebayimg.com/images/i/182196556219-0-1/s-l1000.jpg",
        "http://i.ebayimg.com/images/i/182196556219-0-1/s-l1001.jpg",
        "http://i.ebayimg.com/images/i/182196556219-0-1/s-l1002.jpg"
      ]
    }
}
```

PayPal API

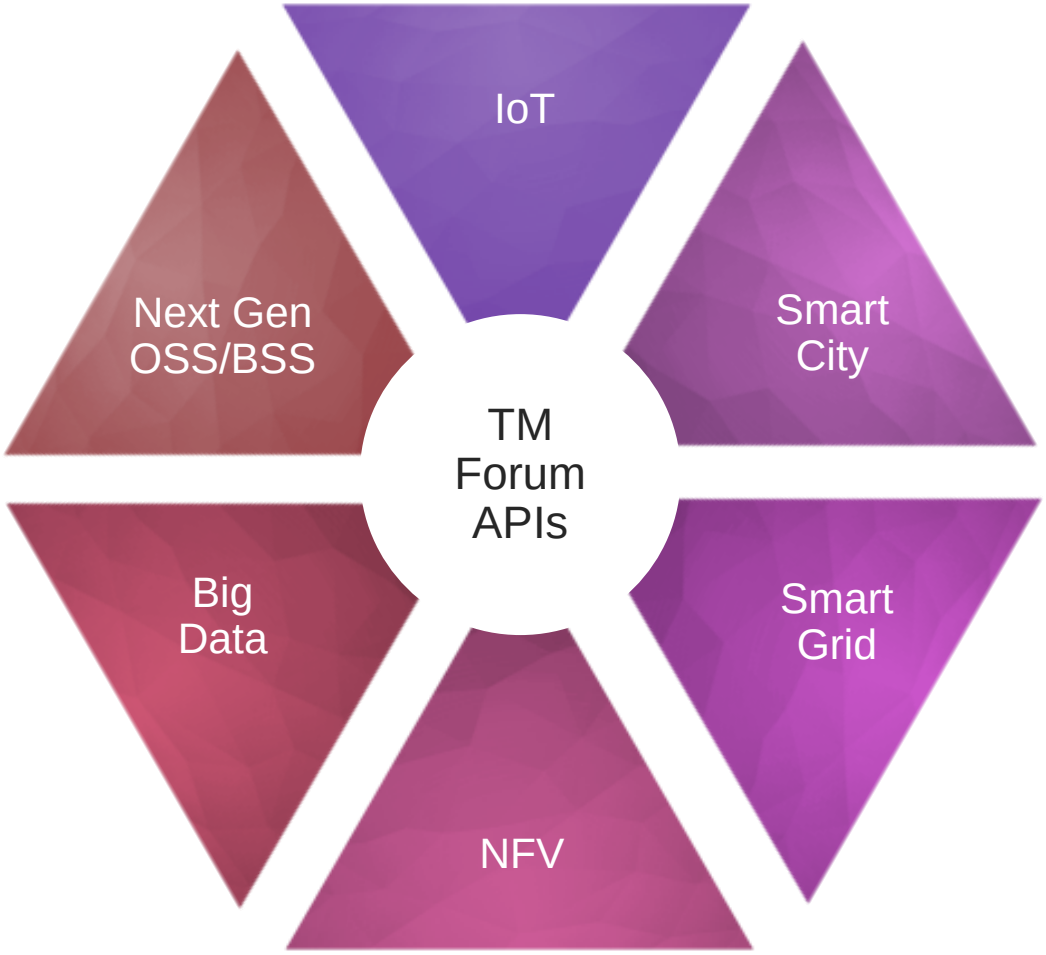

PayPal Developer
Docs
APIs
Support
Search
Dashboard
REST APIs
Overview
Authentication & Headers
Errors
API Reference
Billing Agreements API
Billing Plans API
Identity API
Invoicing API
Payment Experience API
Payments API
Payouts API
Vault API
Webhooks API
Release Notes
SDKs
SDKs Quickstart
Service Integrations
NVP / SOAP APIs
Credentials
Endpoints
Currency Codes
Country Codes
Create agreement
POST
/v1/payments/billing-agreements
Creates a billing agreement. In the JSON request body, pass an agreement object with the name, description, start date, ID of the billing plan on which to base the agreement, payer, and shipping address.
Update agreement
PATCH
/v1/payments/billing-agreements/agreement_id
Updates details of a billing agreement, by ID. Details include the description, shipping address, start date, and so on.
Show agreement details
GET
/v1/payments/billing-agreements/agreement_id
Shows details for a billing agreement, by ID.
Bill agreement balance
POST
/v1/payments/billing-agreements/agreement_id/bill-balance
Bills an outstanding amount for an agreement, by ID. In the JSON request body, pass an agreement_state_descriptor object with the reason for the agreement state change and the agreement amount and currency.
Cancel agreement
Re-activate agreement

Module 2:
TM Forum Open API Introduction

Presented by: Joann O'Brien

What are the TM Forum Open APIs

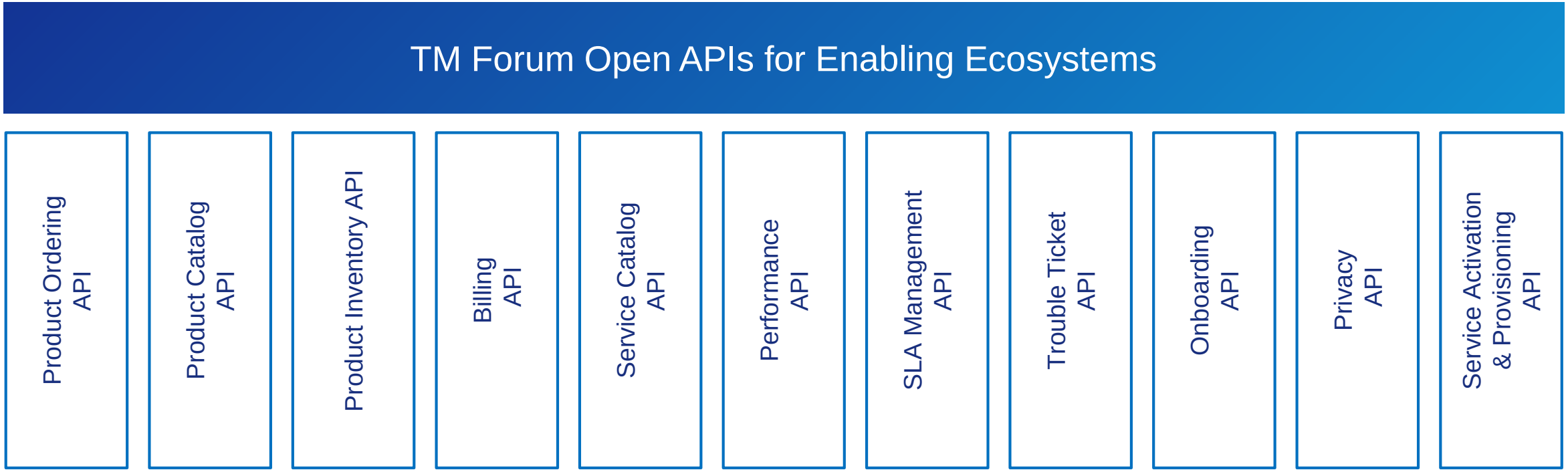
- A suite of APIs making it easier to
  - create
  - build
  - and operate
- complex, innovative services
- REST based



TM Forum Open APIs Key Facts

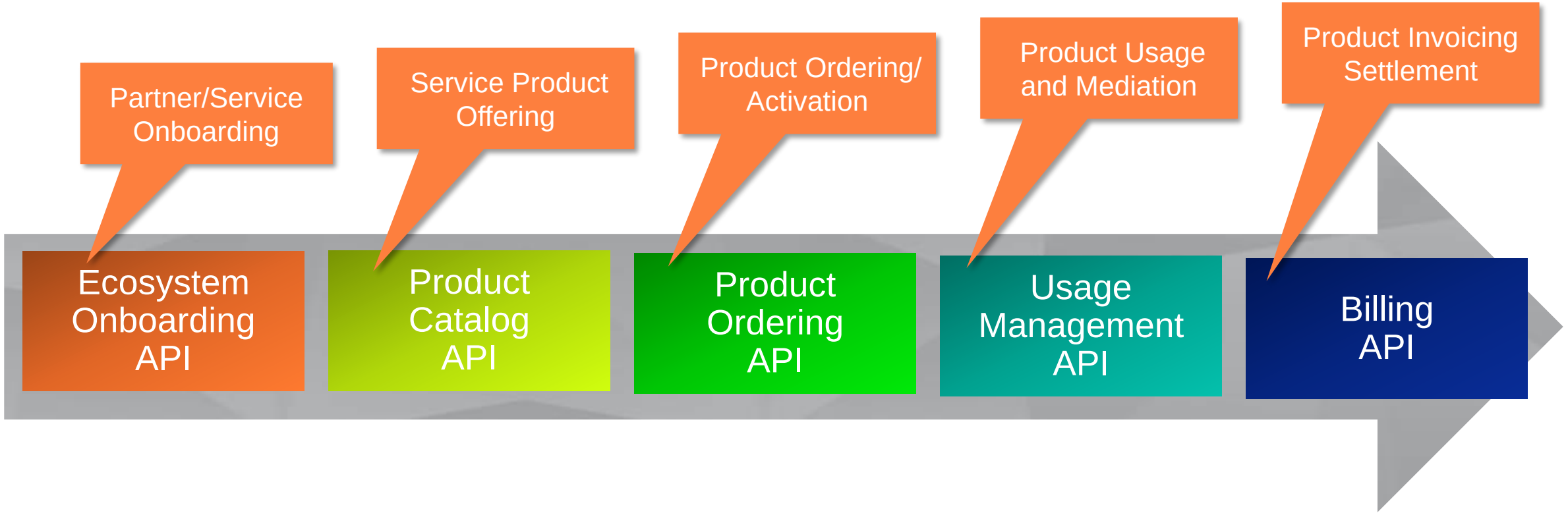
- TM Forum Open APIs are a suite of APIs:
  - that enables services to be managed end to end throughout their lifecycle
  - in an environment where multiple partners are involved in the service delivery
- The APIs allow simple and coherent management of any given service
- The APIs are based on industry strength key design patterns to enable their rapid implementation.

Examples of Open APIs Currently Available

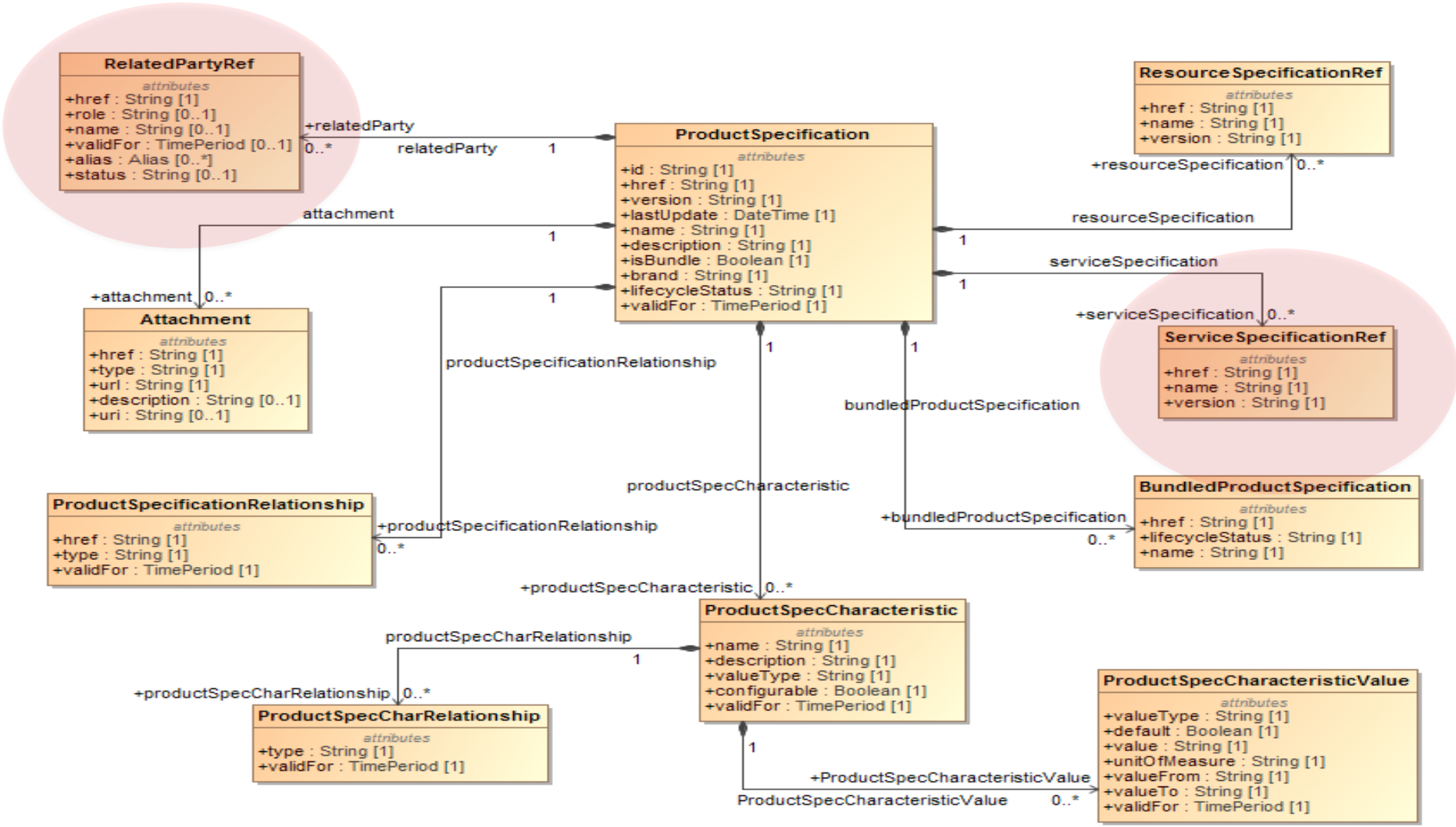


- From IoT device management to complex B2B value fabrics
- Managing partner arrangements across any business boundary
- Onboarding, SLA management, policy management and revenue sharing & settlement

End to End Management



Product Catalog API



- Open APIs are generic framework APIs which can be used for any type of Product, Service and Resources.

- Open APIs are generic framework APIs which can be used In the communications industry as well as other industries.
  - Network Connectivity Service
  - IOT Data Service

- Any Product, Service and Resources deployed in the TM Forum API Ecosystem can be instrumented with a set of generic APIs.

TM Forum Open APIs Product Example

Any Product can be Ordered via the Product Ordering API

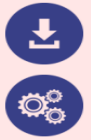
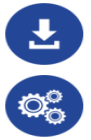
Any Product can be Offered in a Product Catalog via Product Catalog API

Product underlying Services can be Activated and Provisioned via Service Activation and Provisioning API

Products can be Hyperlinked to Service Specifications in Service Catalogs via Product Catalog API

Products can be Hyperlinked to Service Level Agreement in SLA Management System supporting SLA APIs

Open APIs Supported by Full Specification and Test Kits

Released TM Forum Open APIs	API Specification	Postman Collection	Swagger Swagger_UI	Reference Implementation Code	Download Specification	CTK Profile
Trouble Ticket API Provides a standardized client interface to Trouble Ticket Management Systems for creating, tracking and managing trouble tickets among partners as a result of an issue or problem identified by a customer or another system. Examples of Trouble Ticket API clients include CRM applications, network management or fault management systems, or other trouble ticket management systems (e.g. B2B).						
Customer Management API Provides a standardized mechanism for customer and customer account management, such as creation, update, retrieval, deletion and notification of events.						
Product Catalog Management API Provides a standardized solution for rapidly adding partners' products to an existing Catalog. It brings the capability for Service Providers to directly feed partners systems with the technical description of the products they propose to them.						
Product Inventory Management API Provides standardized mechanism for product inventory management such as creation, partial or full update and retrieval of the representation of a product in the inventory. It also allows the notification of events related to product lifecycle.						

http://www.tmforum.org/open-apis/

http://projects.tmforum.org/wiki/display/API/Open+API+Table

TM Forum Open APIs Benefits

Business agility and interoperability	By supporting an architecture based on modularity and reuse where capabilities are exposed through industry agreed Open APIs
Partnering for new services	Enable new products and services to be created by providing services to, and consuming services from, other business units, subsidiaries and external 3rd parties from telco and increasingly other verticals
Simplification of the IT estate	By enabling systems rationalisation
Reduced time to market	By enabling the configuration and reconfiguration of systems and capabilities through Open APIs to deliver new business solutions

Open APIs enabling Business Agility & Interoperability

1 Year Since Program Launch

31 Open API specifications

28 Member Companies Signed Open API Manifesto committing to deploy TM Forum Open APIs

14 Service Provider Markets deployed & 27 by end of 2017

600+ Member Companies & 4000+ Professionals downloading & using the TM Forum Open APIs

7 Open Hacks 400+ Hack Participants 55 Innovative solutions

21,000+ Member Professionals

Signees of TM Forum's Open API Manifesto

Open API Manifesto signed by 28 major companies since May 2016 demonstrating strategic alignment with TM Forum Open APIs

Current APIs

Open API Roadmap	Available	Planned Update

Open API Roadmap	Available	Planned Update

Crowdsourced

TM Forum Open API Governance Activities

Management of the portfolio of APIs

Planning the development of new services and updating current services, through service submission and change requests

Management of the individual API lifecycle

To ensure that service specification updates are consistent and interoperable through gated review and publication

Management of policies and standards

The rules to which all APIs must conform should themselves be governed

API Governance Group

Audience	**TM Forum API Programme Team, including the contributor/author of the API specification**
Input	All documents relevant to the API specification lifecycle milestone
Process	Consider the validity of the documents submitted against the principles and standards of the API programme, and relevant to the particular API development lifecycle milestone
Output	• Approval: The documents are complete and correct. Approved for this milestone without need of further change.
	• Conditional Approval: The documents are approved to progress pending minor remedial actions that do not require a resubmission.
	• Resubmission: The documents require design rework and a fresh review.
	• Dispensation: The documents are allowed to progress with ‘technical debit’ – i.e.: acknowledging that they are not correct.
	• Escalation: Agreement cannot be reached within the governance board and requires a higher/wider authority.

API Governance Group

Source	Description
API Governance Manifesto	12-point manifesto with very high-level guidance, and links to other material https://projects.tmforum.org/wiki/display/AP/Governance
GB990 API Data Model and Information Framework Mapping Guide Book R15.5.0	Describes SID->API Data Model mapping patterns (collapsing hierarchy, trimming relationships etc). https://projects.tmforum.org/wiki/display/AP/GB990+API+Data+Model+and+Information+Framework+Mapping+Guide+Book+R15.5.0
TR250 API REST Conformance Guidelines R15.5.0	"Self-Certification of TM Forum REST APIs". More relevant to API implementation https://projects.tmforum.org/wiki/display/AP/TR250+API+REST+Conformance+Guidelines+R15.5.0
API Governance	Wider Frameworx governance process, applied into API governance https://projects.tmforum.org/wiki/display/TP/API+Governance
TMF425 API Crowdsourcing Template R16.0.1	"the template for crowdsourcing APIs to TM Forum. It must be filled out prior to/in parallel with contributing a new API." https://projects.tmforum.org/wiki/display/PUB/TMF425+API+Crowdsourcing+Template+R16.0.1
TMF630 Conformance to REST API Design Guidelines	Part 1: Practical guidelines for RESTful API's naming, CRUD, filtering and notifications Part 2: Advanced guidelines for RESTful API's lifecycle management, polymorphism, common tasks https://projects.tmforum.org/wiki/display/API/API+Design+Guidelines

Open API Roadmap

Current APIs

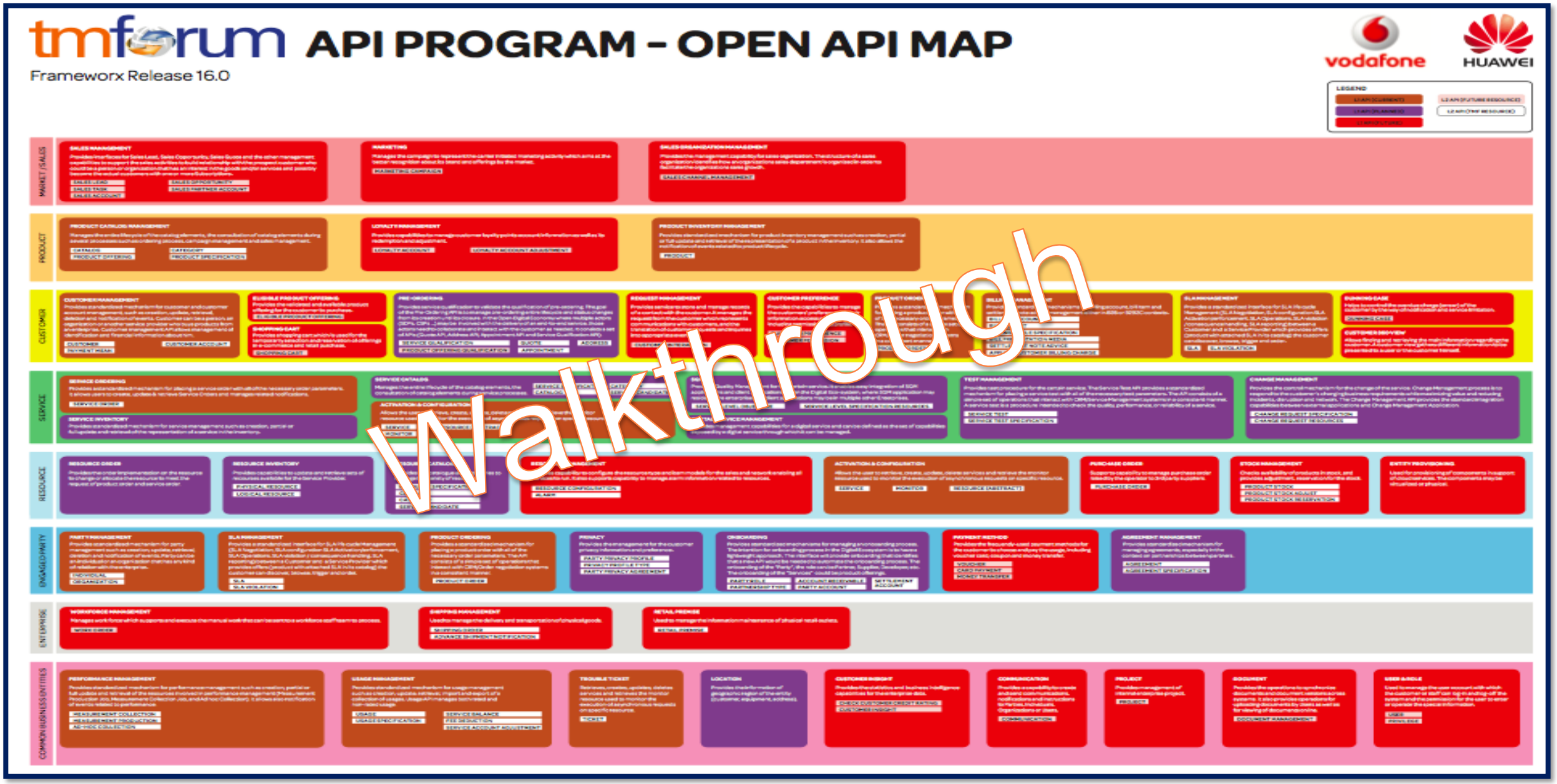
Open API Roadmap	Available	Planned Update
Activation & Configuration API	√	
Address API	√	Fx 17
Agreement API	√	
Appointment API	√	
Billing Management API	√	
Change Management API	√	Fx 17
Customer Management API	√	
Entity Provisioning API	√	
Loyalty Management API	√	Fx 17
Onboarding API	√	
Party Management API	√	
Performance Management API	√	Fx 17
Prepayment Balance Management API	√	
Privacy API	√	
Product Catalog Management API	√	

Open API Roadmap	Available	Planned Update
Product Inventory Management API	√	
Product Ordering API	√	
Quote API	√	
Resource Catalog API	√	
Resource Ordering API	√	
Service Catalog API	√	
Service Inventory API	√	
Service Ordering API	√	
Service Problem Management API (SPM)	√	
Service Quantification API	√	
Service Quality Management API	√	
Service Test Management API	√	
SLA Management API	√	
Trouble Ticket API	√	
Usage Management API	√	

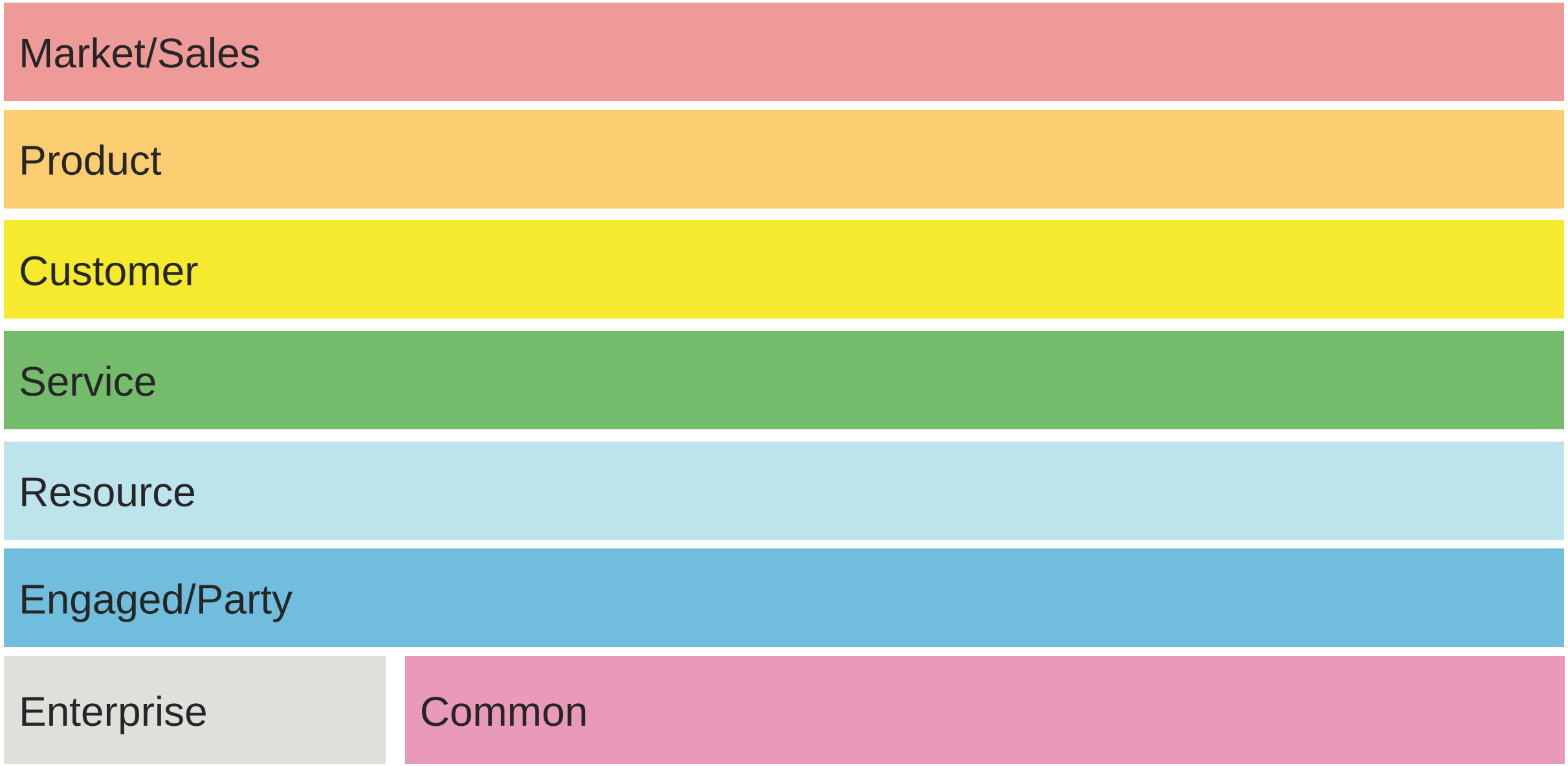
tmforum
connecting digital ecosystems

Module 3:
TM Forum Open API Structure

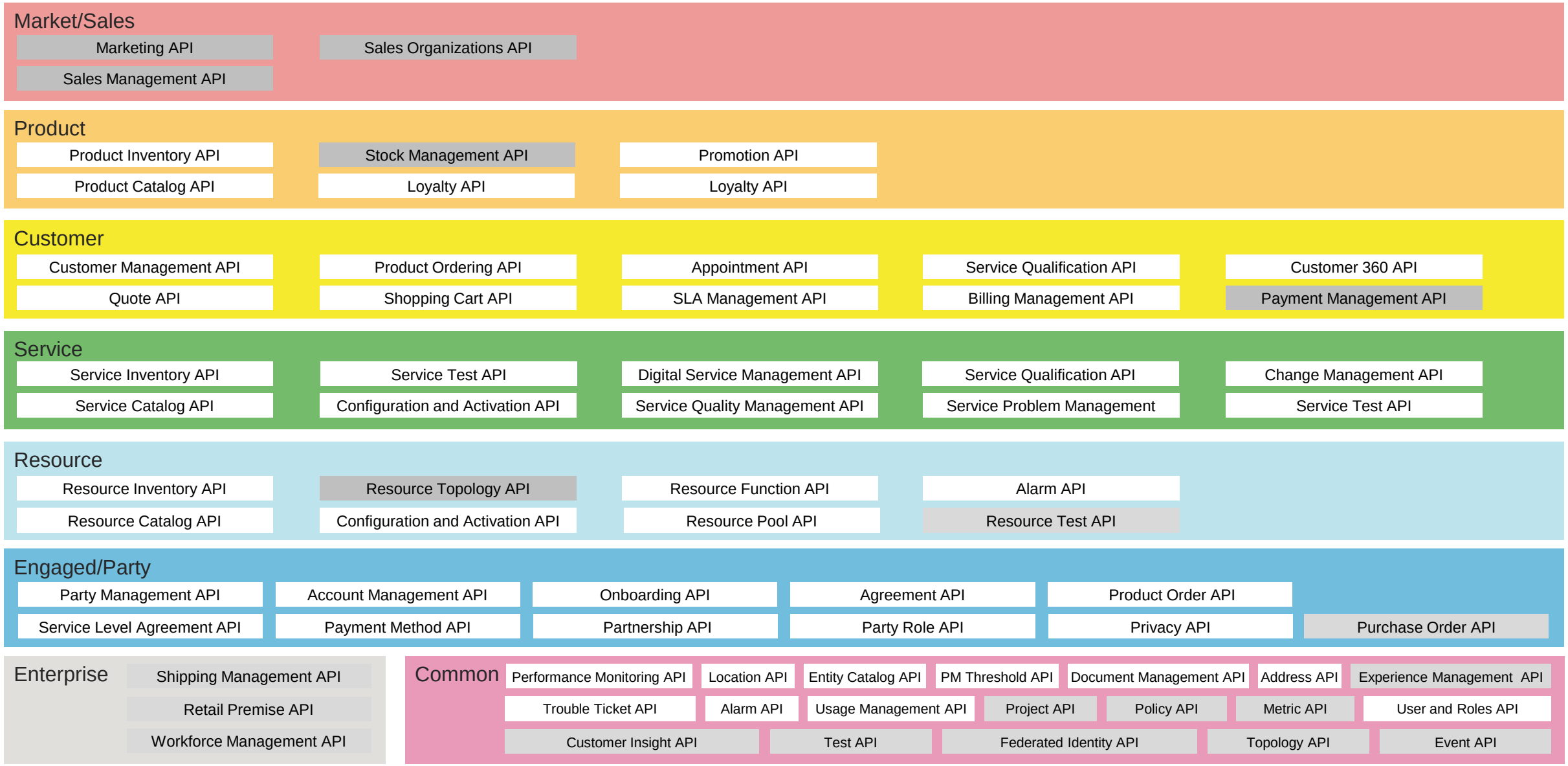
Presented by: Joann O'Brien


tmforum
API PROGRAM – OPEN API MAP
Frameworx Release 16.0
vodafone
HUAWEI
LEGEND
MARKET / SALES
PRODUCT
CUSTOMER
SERVICE
RESOURCE
ENGAGED PARTY
ENTERPRISE
COMMON BUSINESS ENTITIES
Walkthrough

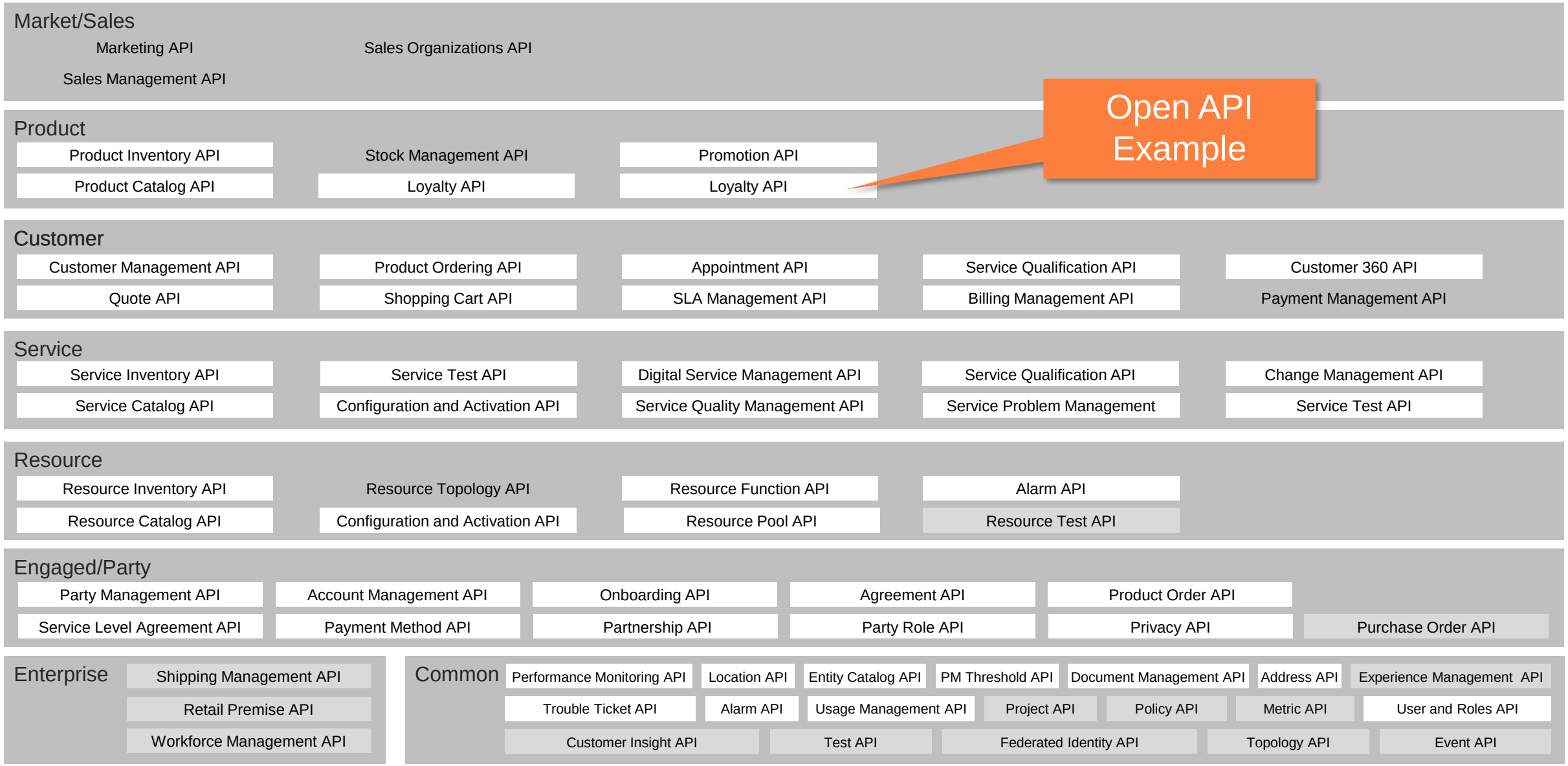
Open API Structure – Level 0



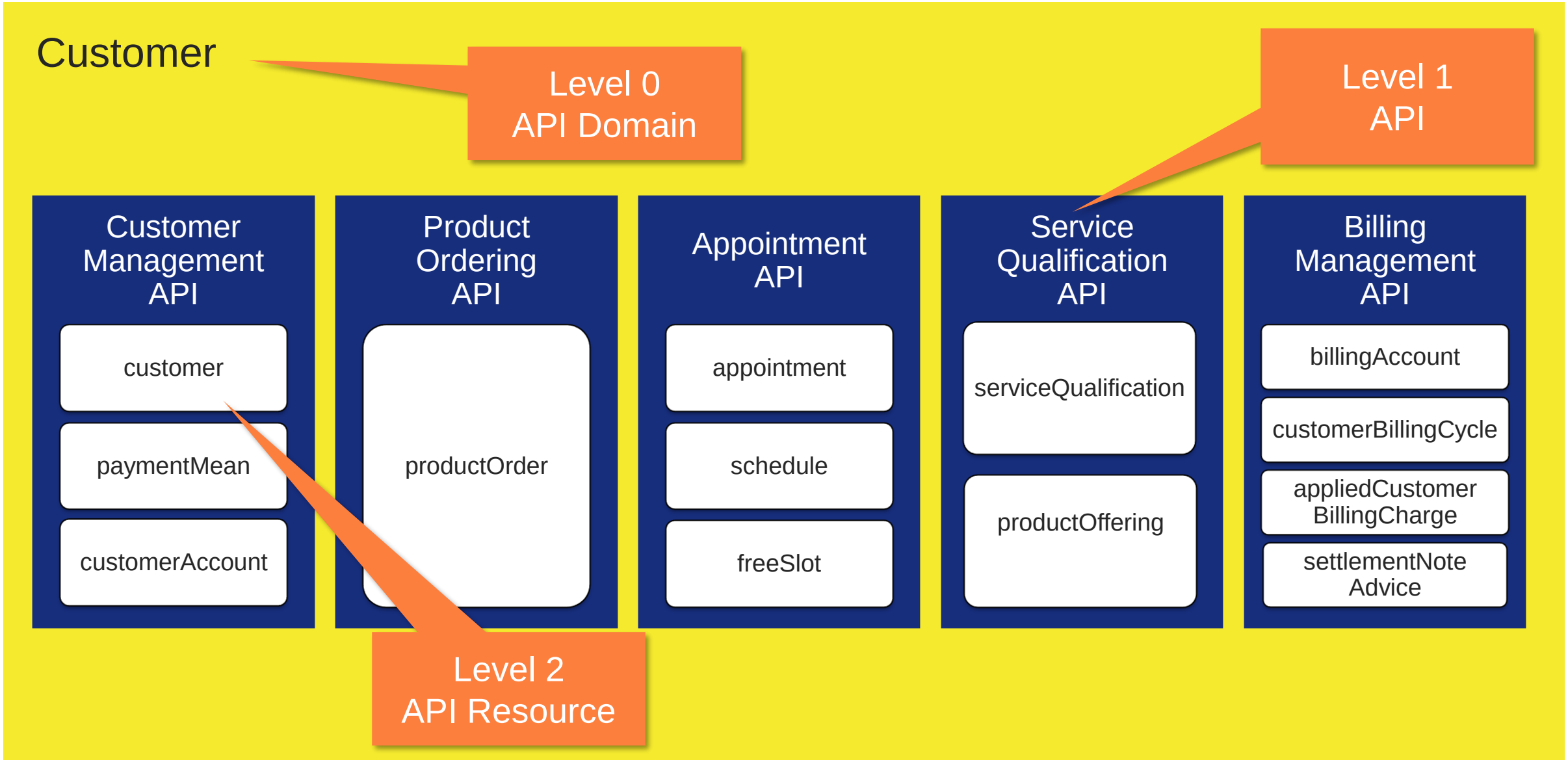
Open API Structure – Level 1



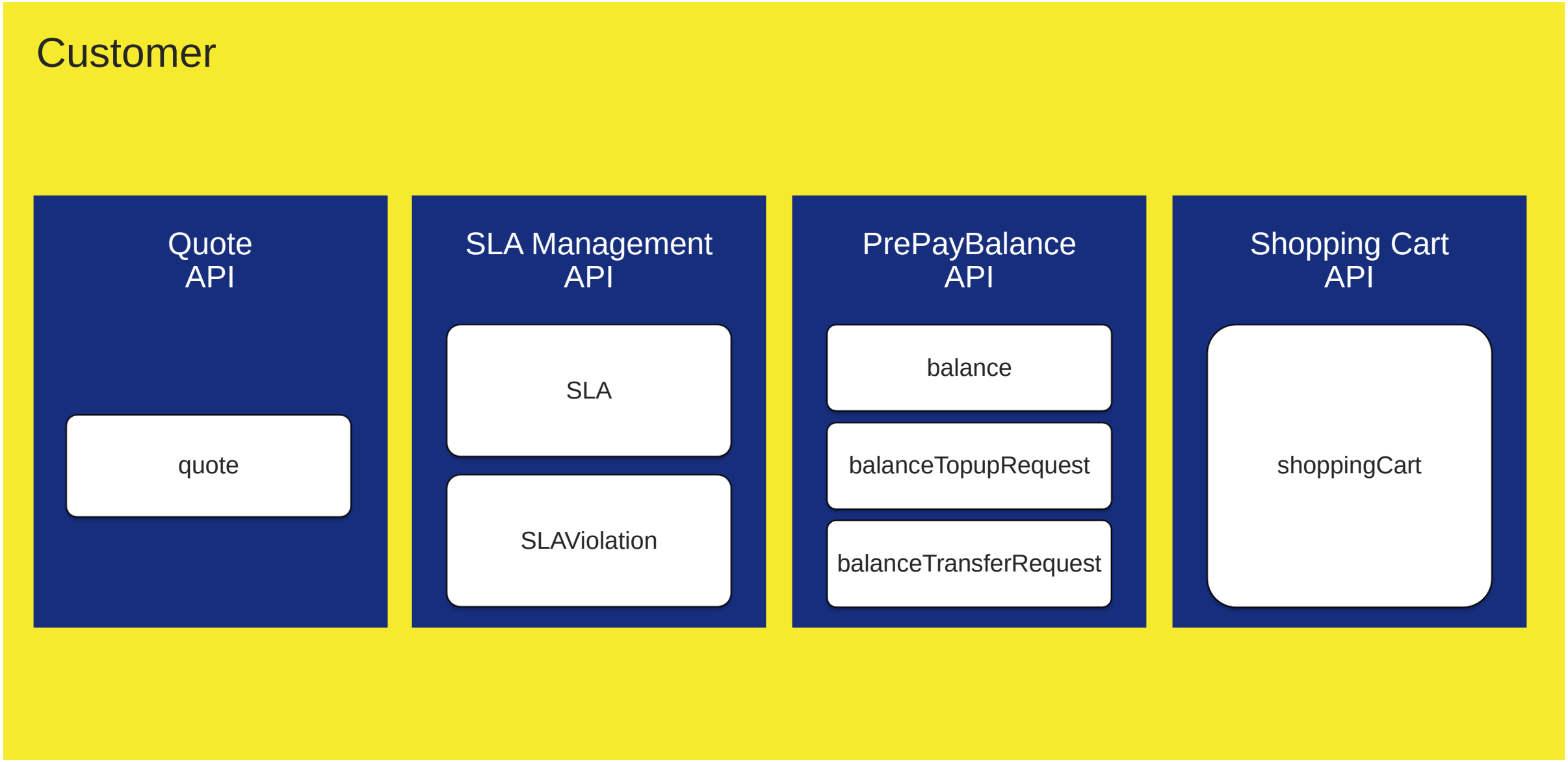
Open API Structure – Level 1



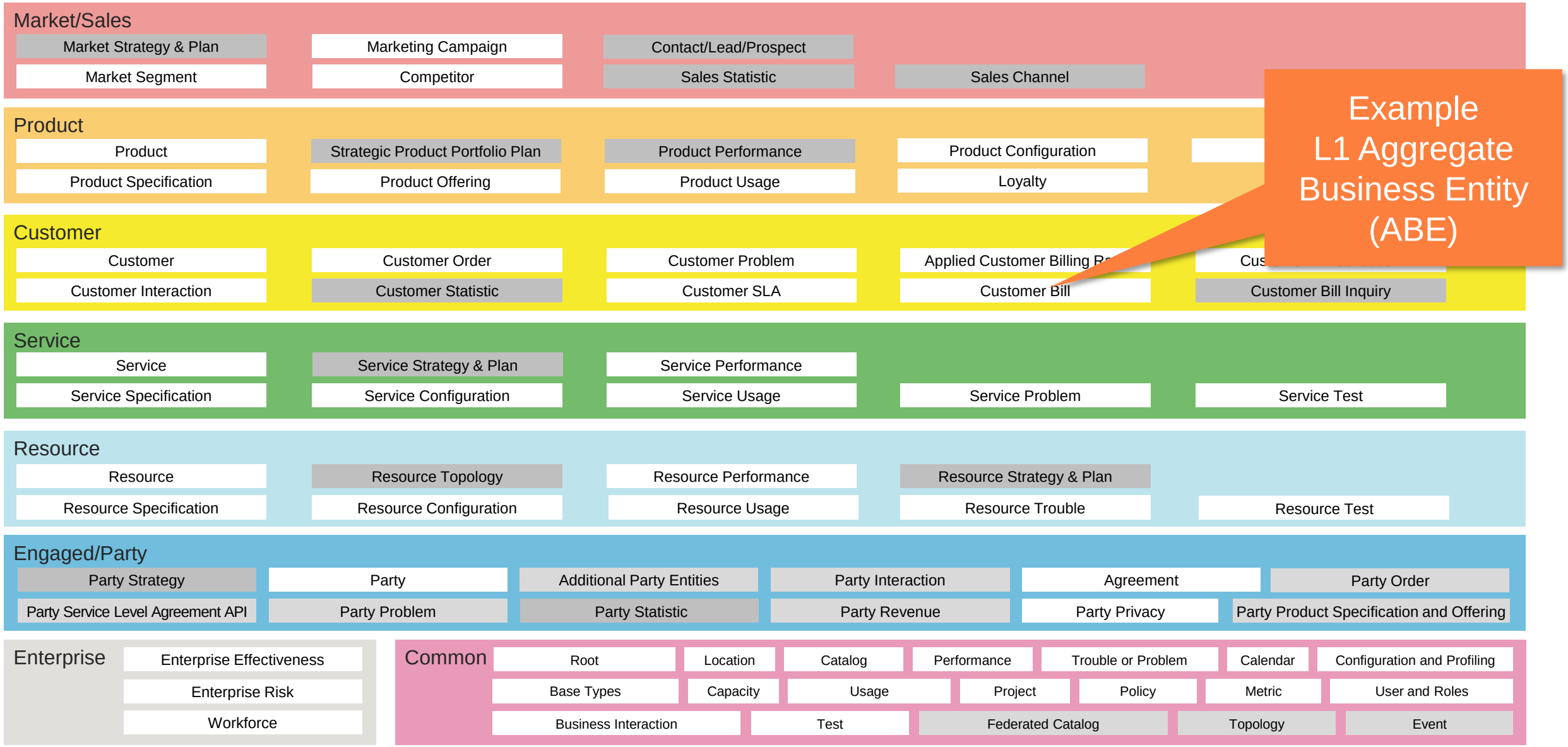
Open API Structure - Level 2 Customer Domain



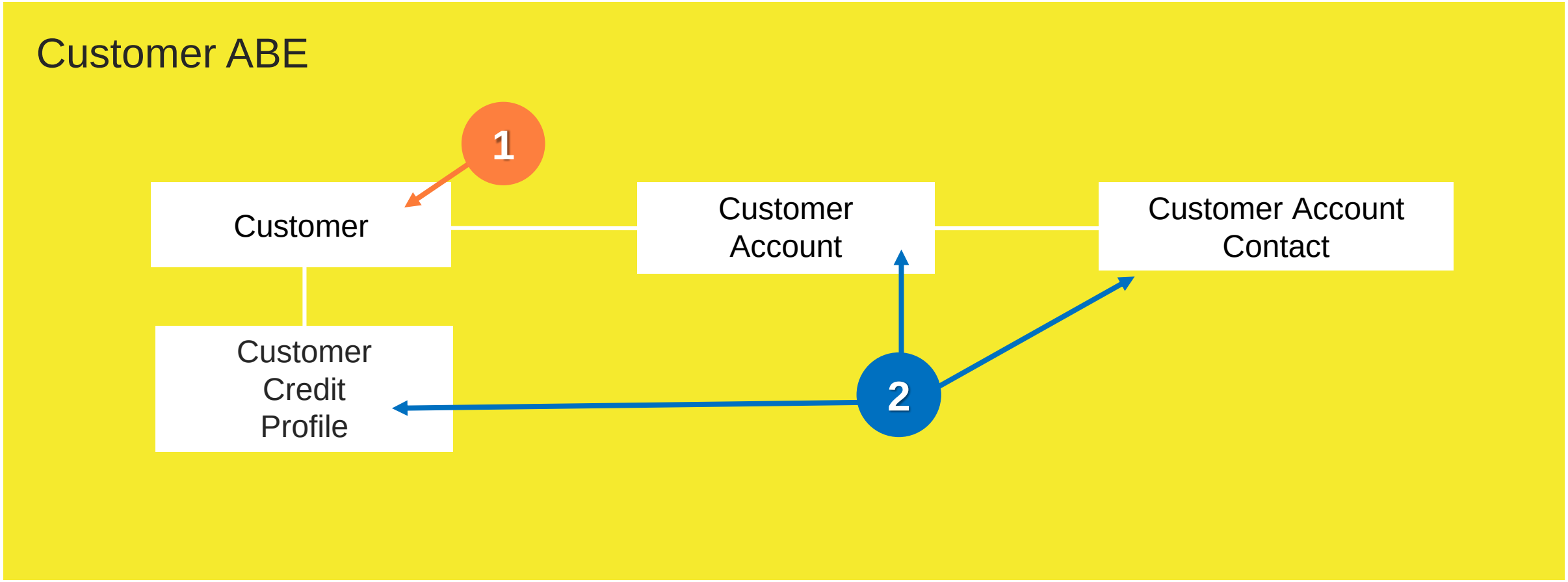
Open API Structure - Level 2 Customer Domain



Information Framework Structure

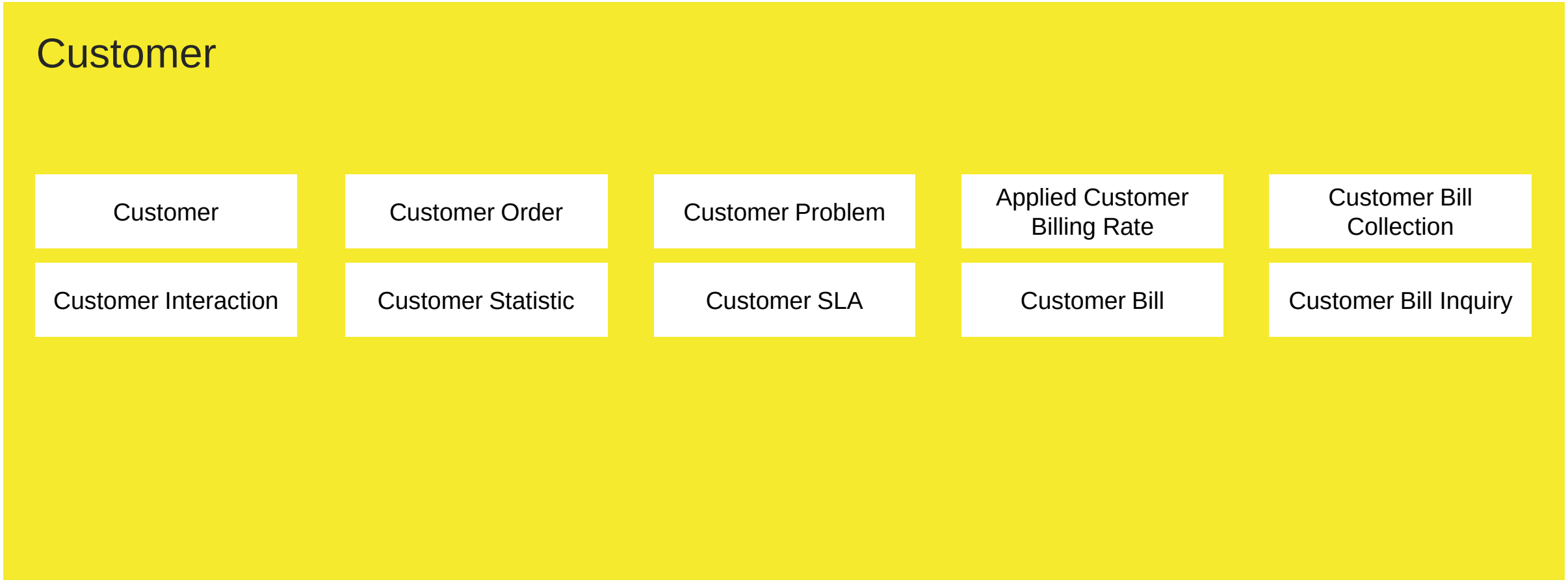


Structure of an ABE



Structure of a Domain

All ABEs that completely define a key management area are grouped together.



Mapping to SID ABE

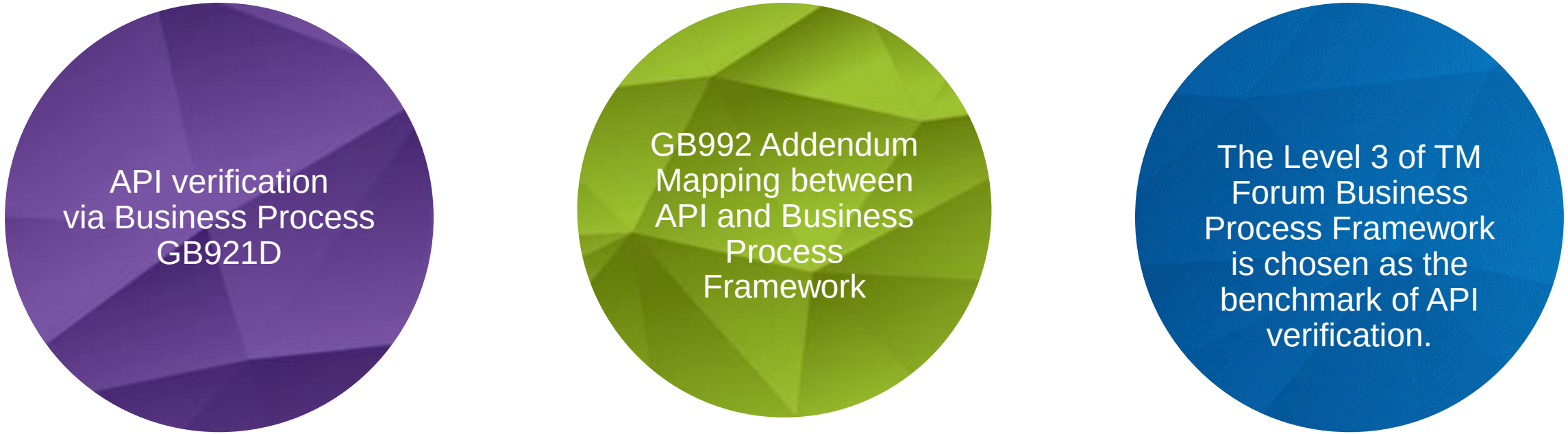
#	API NAME (API)	Resource (API)	Information Framework Domain::L1 ABE-	API L0 Domain
1	Product Catalog Management API	Catalog	Common Business Entities Domain::Catalog ABE	Product
		Category	N/A	
		Product Offering	Product Domain::Product Offering ABE	
		Product Specification	Product Domain::Product Specification ABE	
2	Product Inventory Management API	Product	Product Domain::Product ABE	Product
3	Billing Management API	Billing Account	Customer Domain::Customer ABE Engaged Party Domain::Additional Party Entities ABE	Customer

Relationship with Frameworks

The TM Forum APIs are related to all other Forum Frameworks, in that they:

- Are typically entity-centric, manipulating a primary resource that can be found in TM Forum Information Framework (SID), an Information view of the APIs
- Play a part in (potentially a number of) business processes from TM Forum Business Process Framework (eTOM), a Business Process view of the APIs
- Can be exposed by an normalized endpoint application or application-domain described by TM Forum Application Framework (TAM), an Application view of the APIs

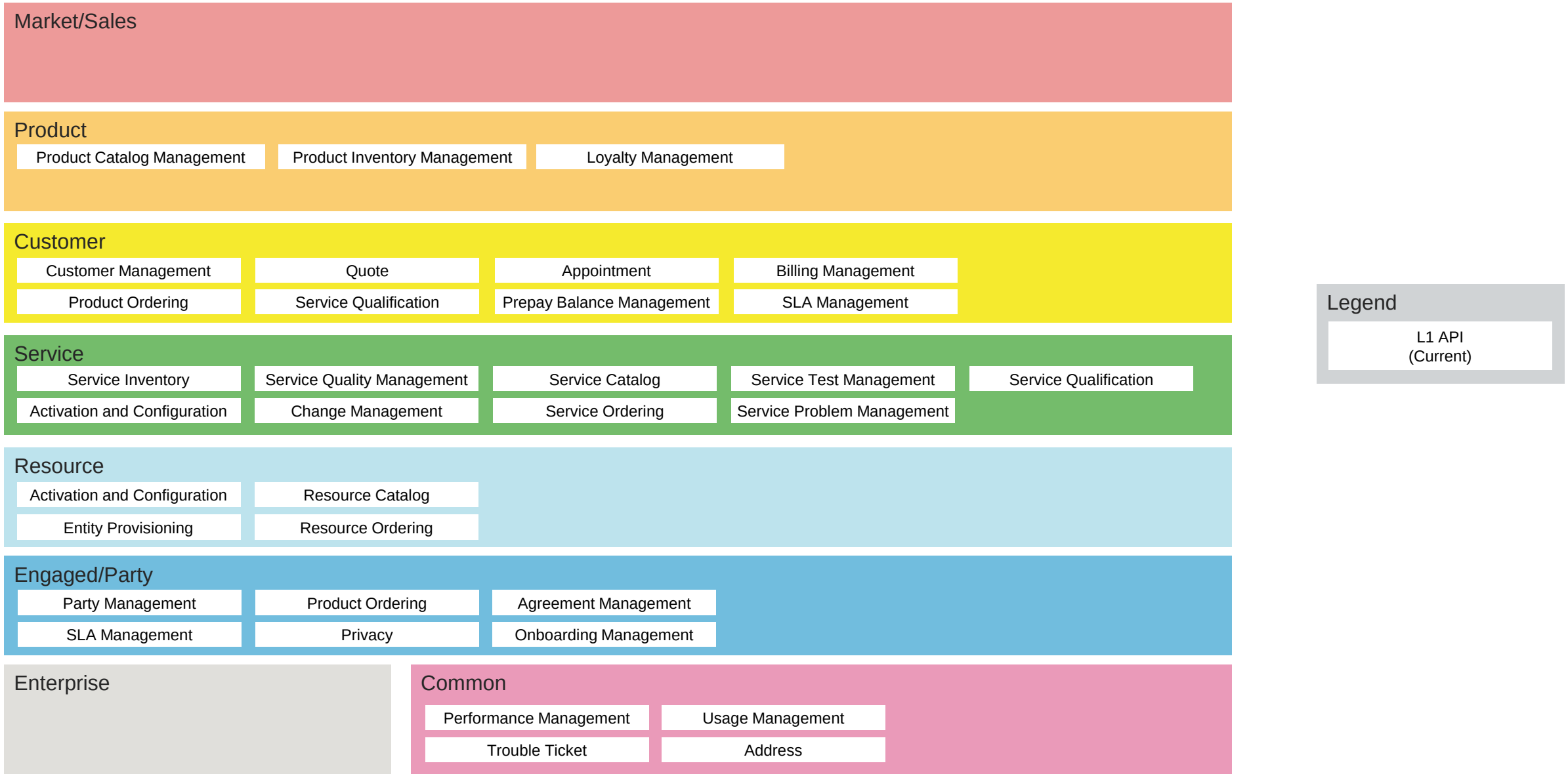
API verification via Business Process

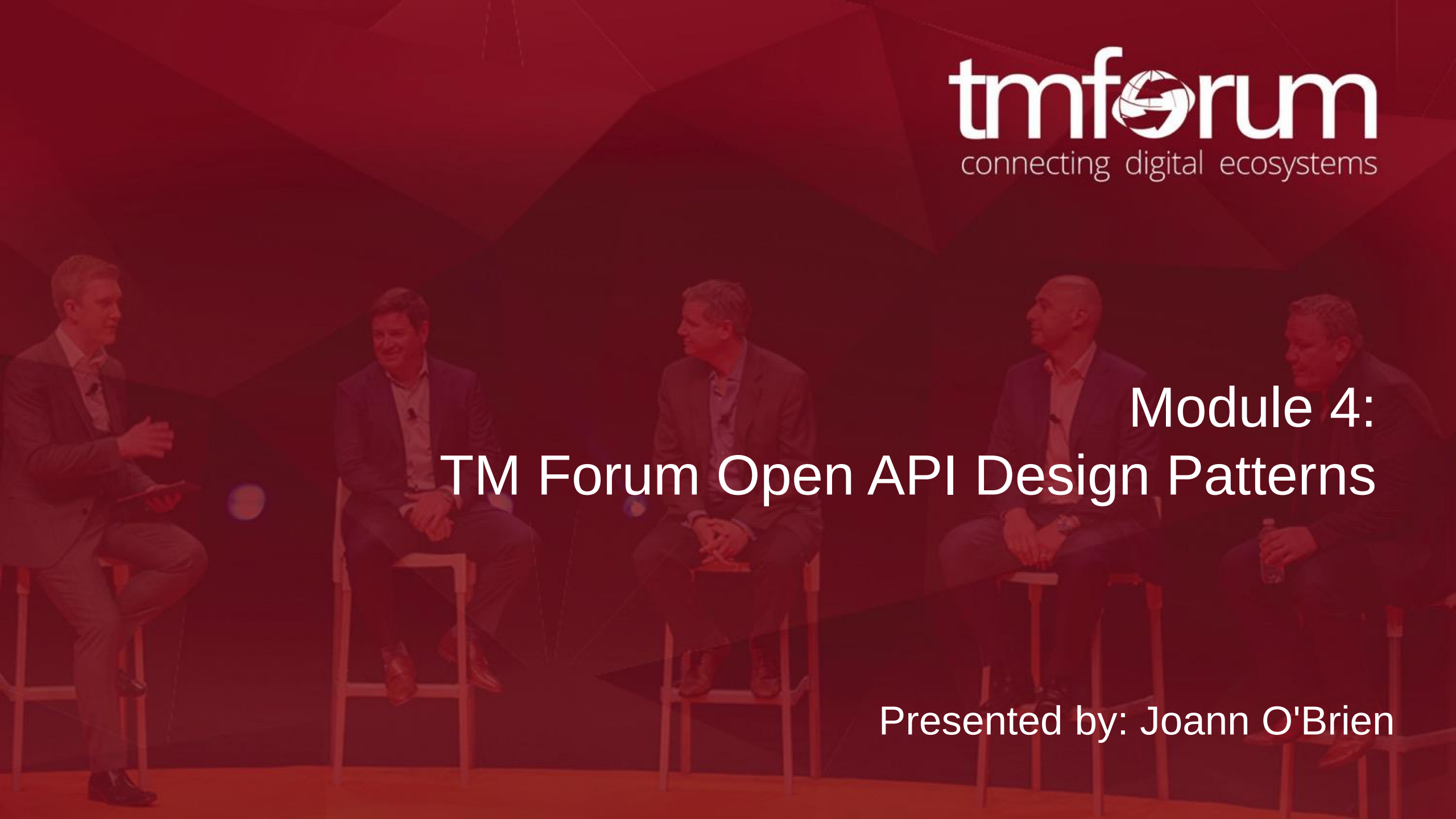


APIs and Business Process Framework

Frameworx Process	Level 3 Category	Process Identifier	Brief Description	TMF API Alignment	Process as API consumer or producer	TMF API Resource Name
Product Specification Development & Retirement	(3) eTOM Process Type	1.2.7.1	Develop and deliver new product specifications as well as enhancements and new features, ready for use by other processes, including Product Offering	Product Catalog Management Product Inventory Mgmt	Producer	Product Offering, Product
Product Offering Development & Retirement	(3) eTOM Process Type	1.2.7.2	Develop and deliver new product offerings, their pricing, as well as catalogs that contain both.	Product Catalog Management Product Inventory Mgmt	Producer	Product Offering, Product
Develop Product Capacity	(3) eTOM Process Type	1.2.8.1	Perform all applicable generalized capacity management processes associated with product capacity.	Product Catalog Management Product Inventory Mgmt	Producer	Product Offering, Product
Develop Product Capacity Demand	(3) eTOM Process Type	1.2.8.2	Perform all applicable generalized capacity management processes associated with product capacity demand.	Product Catalog Management Product Inventory Mgmt	Producer	Product Offering, Product
Support Customer Interface Management	(3) eTOM Process Type	1.3.1.1	Ensure that all information, materials, systems and resources are available so that the Customer Interface Management processes can operate	Customer Management		

Overview of existing APIs – R17.0





Module 4:
TM Forum Open API Design Patterns

Presented by: Joann O'Brien

Why REST API Design Patterns?

REST is a software architectural style, not a specification, for building scalable web services

RESTful web services do not have any strict specifications for designing and building APIs

Many interpretations exist for how the RESTful Web API should work

Need standards, best practices, conventions, and tips and tricks that you can apply to your RESTful web service applications

What are the TMF REST Design Patterns?

- All TMF APIs are based on a common set of Patterns
- Patterns are documented in the TMF REST Guidelines (also known as TMF 630 and TMF 631)
- Patterns provide answers to recurrent questions in designing REST APIs

Patterns for using Uniform Contract operations for the implementation of Entity CRUD operations and Task operations.

Patterns on filtering and attribute selection.

Patterns for supporting Notification Management in REST based systems.

Versioning Patterns

TMF REST DESIGN PATTERNS

API Resource Archetypes	Resource Collection, Managed Resource, Tasks
Uniform Contract Elements	Resource identifier syntax , Methods, Media types All API operations are based on the REST Uniform Contract operations (GET,POST, DELETE, PUT, PATCH, HEAD, etc.
Query Entities Design Patterns	Patterns used to query resources Uniform Contract GET, Attribute selection, Attribute Filtering
Create Entities Design Patterns	Uniform Contract POST, Mandatory Attributes, Structure of returned Response, Location Header, Exception Code
Update Entities Design Patterns	Uniform Contract PATCH, Mandatory Attributes, Structure of returned Response, Location Header, Exception Code

TMF REST DESIGN PATTERNS

Delete Entities Design Patterns	Uniform Contract DELETE, Mandatory Attributes, Structure of returned Response, Location Header, Exception Code
Tasks Design Patterns	A TASK resource name is a verb representing the task to be executed. TASK creation may result in the creation of a number of related resources or other tasks
Notification Design Patterns	Publish subscribe pattern Hub and event Filtering
Asynchronous Operations Design Patterns	In some use cases the modification of an attribute may result in the execution a long running transaction before the attribute actually is changed. TransactionMonitor resource is returned so that the execution of the operation can be monitored

Operation on Entities	Uniform API Operation	Description
Query Entities	GET Resource	GET must be used to retrieve a representation of a resource
Create Entity	POST Resource	POST must be used to create a new resource
Partial Update of an Entity	PATCH Resource	PATCH must be used to partially update a resource
Complete Update of an Entity	PUT Resource	PUT must be used to completely update a resource identified by its resource URI
Remove an Entity	DELETE Resource	DELETE must be used to remove a resource
Execute an Action on an Entity	POST on TASK Resource	POST must be used to execute TASK resources
Other Request Methods	POST on TASK Resource	GET and POST must not be used to tunnel other request methods

TM Forum REST API Design Pattern Examples

Report Management Example


```
{
   “id”: « 42 »,
   "datetime": "2012-12-20 15:00:00",
   "health_state": "operational",
   "metrics": [
      {
         "code": "123",
         "category_id": "45",
         "date_time": "2012-12-20 14:30:00",
         "reference": "4321",
         "source_id": "8f27ce50-4b00-11e2-bcfd-0800200c9a66",
         "value": 50,
         "metric_id": "b0dbbc50-4b00-11e2-bcfd-0800200c9a66"
      },
      {
         "code": "321",
         "category_id": "48",
         "date_time": "2012-12-20 14:30:00",
         "reference": "4321",
         "source_id": "8f27ce50-4b00-11e2-bcfd-0800200c9a66",
         "value": 50,
         "metric_id": "c50f3e90-4b00-11e2-bcfd-0800200c9a66"
      }
   ],
   "failures": [
      {
         "failure_id": "0bec4420-4b01-11e2-bcfd-0800200c9a66",
         "detail": "Network authentication failure",
         "source_id": "24b2bc00-4b01-11e2-bcfd-0800200c9a66"
      },
      {
         "failure_id": "3b687c50-4b01-11e2-bcfd-0800200c9a66",
         "detail": "Network authentication failure",
         "source_id": "462f1f40-4b01-11e2-bcfd-0800200c9a66"
      }
   ]
     }
```

Query Resource: Attribute Selection Pattern Example

How to select a subset of the attributes of an entity to be present in a returned representation?

- An attribute selector directive called “**fields**” is used to specify the attributes to be returned as part of a partial representation of a resource.

GET {apiRoot} /{resourceName}/{resourceID}/?fields={attributeName*}

- Used to retrieve the partial representation of a resource with the attributes named resourceID.

Example

Retrieve the report with an « id » of 5 and populate the representations with the failures and metrics attributes. Note that the « id » attribute is always present.

REQUEST

```
GET /api/report/5/?fields=failures,metrics
```

RESPONSE

```
200

Content-Type: application/json

{
    “id”: 5,
    "metrics": [
        {
            "code": "123",
            "category_id": "45",
            "date_time": "2012-12-20 14:30:00",
            "reference": "4321",
            "source_id": "8f27ce50-4b00-11e2-bcfd-0800200c9a66",
            "value": 50,
            "metric_id": "b0dbbc50-4b00-11e2-bcfd-0800200c9a66"
        },
        {
            "code": "321",
            "category_id": "48",
            "date_time": "2012-12-20 14:30:00",
            "reference": "4321",
            "source_id": "8f27ce50-4b00-11e2-bcfd-0800200c9a66",
            "value": 50,
            "metric_id": "c50f3e90-4b00-11e2-bcfd-0800200c9a66"
        }
    ],
    "failures": [
        {
            "failure_id": "0bec4420-4b01-11e2-bcfd-0800200c9a66",
            "detail": "Network authentication failure",
            "source_id": "24b2bc00-4b01-11e2-bcfd-0800200c9a66"
        },
        {
            "failure_id": "3b687c50-4b01-11e2-bcfd-0800200c9a66",
            "detail": "Network authentication failure",
            "source_id": "462f1f40-4b01-11e2-bcfd-0800200c9a66"
        }
    ]
}
```

Query Resources: Attribute Filtering Pattern Example

How to retrieve resources using an attribute filtering mechanism?

- The filtering is based on using name value query parameters on entity attributes
- The basic expression is a sequence of attribute assertions being ANDED to formulate a filtering expression:

GET ../{resourceName}/?[{attributeName}={attributeValue}&*]

- Match only for attribute value equality

Returned Resources

When we do a query on a collection there may be a very large number of entities returned, therefore we need to return in chucks and it is the content range header that is used to specify how many entities are returned.

Let’s take a look at another example

Retrieve all Reports with date_time greater than 2013-04-20 and health_state operational.

REQUEST

```
GET /api/report?date_time.gt=2013-04-20&health_state =operational       or
GET /api/report?date_time>2013-04-20&health_state = operational
```

RESPONSE

```
200
Content-Type: application/json
Content-Range: items 1-2/2
[ {
   “id”: « 42 »,
   "datetime": "2012-12-20 15:00:00",
   "health_state": "operational",
   "metrics": [
      {
         "code": "123",
         "category_id": "45",
         ...
         "metric_id": "b0dbbc50-4b00-11e2-bcfd-0800200c9a66"
      },
...
],
   "failures": [
      {
         "failure_id": "0bec4420-4b01-11e2-bcfd-0800200c9a66",
         "detail": "Network authentication failure",
         "source_id": "24b2bc00-4b01-11e2-bcfd-0800200c9a66"
      },
...
]
},
{
   “id”: « 52 « ,
   "datetime": "2012-12-20 15:00:00",
   "health_state": "operational",
   "metrics": [
      {
         "code": "456",
         "category_id": "45",
         ...
         "metric_id": "b0dbbc50-4b00-11e2-bcfd-0800200c9a66"
      },
...
],
   "failures": [
      {
         "failure_id": "0bec4420-4b01-11e2-bcfd-0800200c9a66",
         "detail": "Network authentication failure",
         "source_id": "24b2bc00-4b01-11e2-bcfd-0800200c9a66"
      },
...
   ] }
```

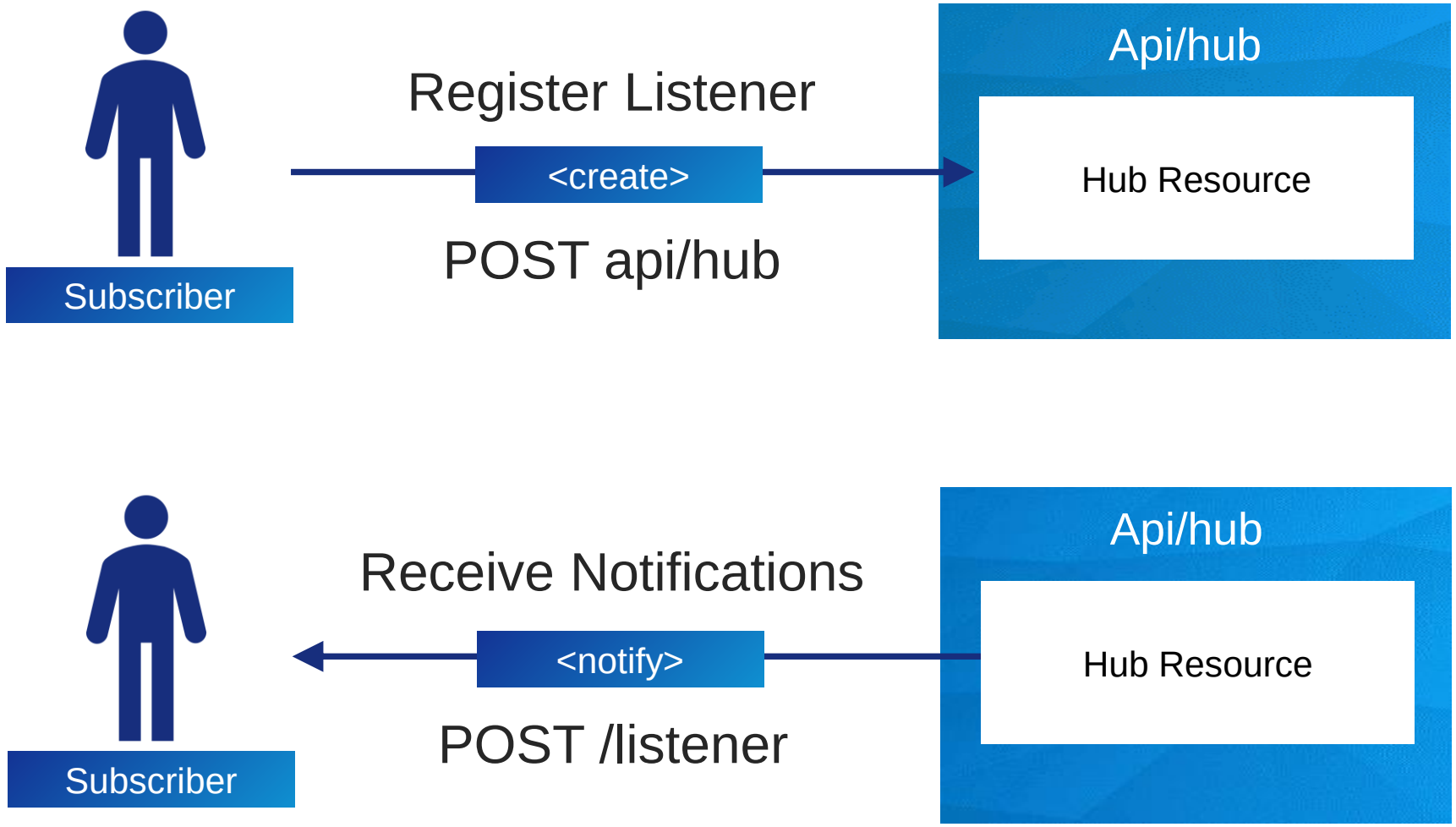
Query Resources with attribute filtering and attribute selection

Attribute filtering and attribute selection **may be combined in a single request** as per the following example.

REQUEST

```
GET /api/report/fields=failures,health_state&report.date_time.gt=2013-04-20&report.state.execution=suspended
```

Notification Pattern



Let’s take a look at an example

Create a Hub to receive events on the “http://in.listener.com"

REQUEST

```
POST /api/hub
Accept: application/json

{"callback": "http://in.listener.com"}
```

RESPONSE

```
201
Content-Type: application/json
Location: /api/hub/42

{"id":"42","callback":"http://in.listener.com","query":null}
```

Publishing Events

- Publishing an event is done by posting the event to the listener address.
- The structure of the event is:

```
{
    "event": {

        EVENT BODY
      },
    "eventType": "eventType"
}
```

- Returns HTTP/1.1 status code 201

Posting a ManagementReport event to the/client/listener endpoint.

REQUEST

```
POST /client/listener
Accept: application/json

{
   "event": {
      "dateTime": null,
      "id": "42",
      "state": {
         "healthState": "UNKNOWN",
         "executionState": "ACTIVE"
      },
      "metrics":
        [
          {
             "code": "76e27e2b-644e-11e2-8ea1-5c260a86d1e4",
             "categoryID": null,
             "dateTime": 1367266835571,
             "reference": null,
             "sourceID": null,
             "value": "20",
             "metricID": null
                  {
             "code": "7dd589b2-644e-11e2-8ea1-5c260a86d1e4",
             "categoryID": null,
             "dateTime": 1367266835573,
             "reference": null,
             "sourceID": null,
             "value": "89",
             "metricID": null
          }
      ],
      "failures": null
   },
   "eventType": "ManagementReport"
}
```

tmforum
connecting digital ecosystems

Module 5: TM Forum Open APIs and the Information Framework

From Logical Information Model to API Data Models

Presented by: Joann O'Brien

Open APIs and TM Forum Frameworx

- TM Forum's Open APIs are complementary with the Business Process and Information Frameworks.
- The respective data structure provided with every TM Forum Open API is aligned to the Information Framework
- The Information Framework is widely adopted throughout the communications industry
  - helps ensure interoperability – one of the key objectives of the TM Forum Open API program.

Resources and Entities

- Resources are representations of the entities managed by the application.
- Resources are acted upon by the REST API using the Uniform Contract Verbs (POST, GET, PUT, DELETE, PATCH, etc.…).
- Operations on Resources affect the state of the corresponding managed entities.
- There is a direct mapping between the managed entities and the corresponding Resources in the REST model.
- Resource identifiers represent the actual resources that a service exposes.
- A resource can be a trouble Ticket, a Logical Port, an Order, a Task etc...

SID – TMF API Relation

SID is static, it focuses on the data	It describes the business entities It does not describe what you can do with such business entities
SID is an information model not a data model	SID out-of-the-box is typically not implemented as is. Some UML concepts being used cannot be directly mapped to mainstream technologies like XML, Java, etc.
TMF APIs uses SID as the basis for its data model	TMF APIs Entities are REST/JSON realizations of the SID Entities (the Data part of the Shared Information and Data).
TMF APIs adds behaviour to SID	It defines the operations that can be executed on the entities It defines the interaction patterns Request/Response/Exceptions and Notifications

JSON Resource Model and SID Entity Types Patterns

- There is a direct mapping between the SID entity types and the corresponding JSON resources in the TMF REST API resource model and TMF API Data Model.
- The mapping between the SID entity types and the corresponding Resource model is based on a set of patterns called the SID JSON Mapping Patterns

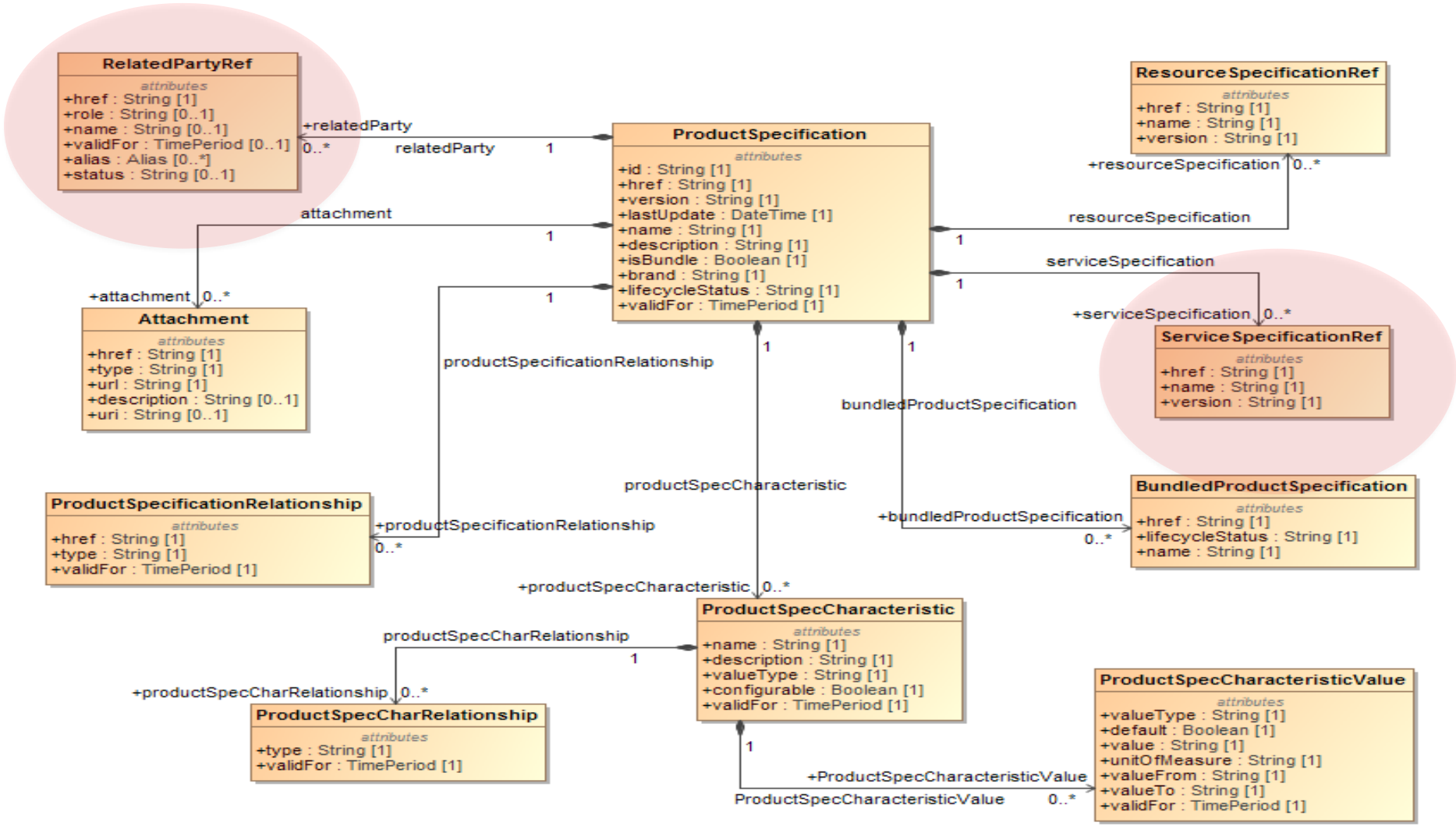
SID Relationships and Links

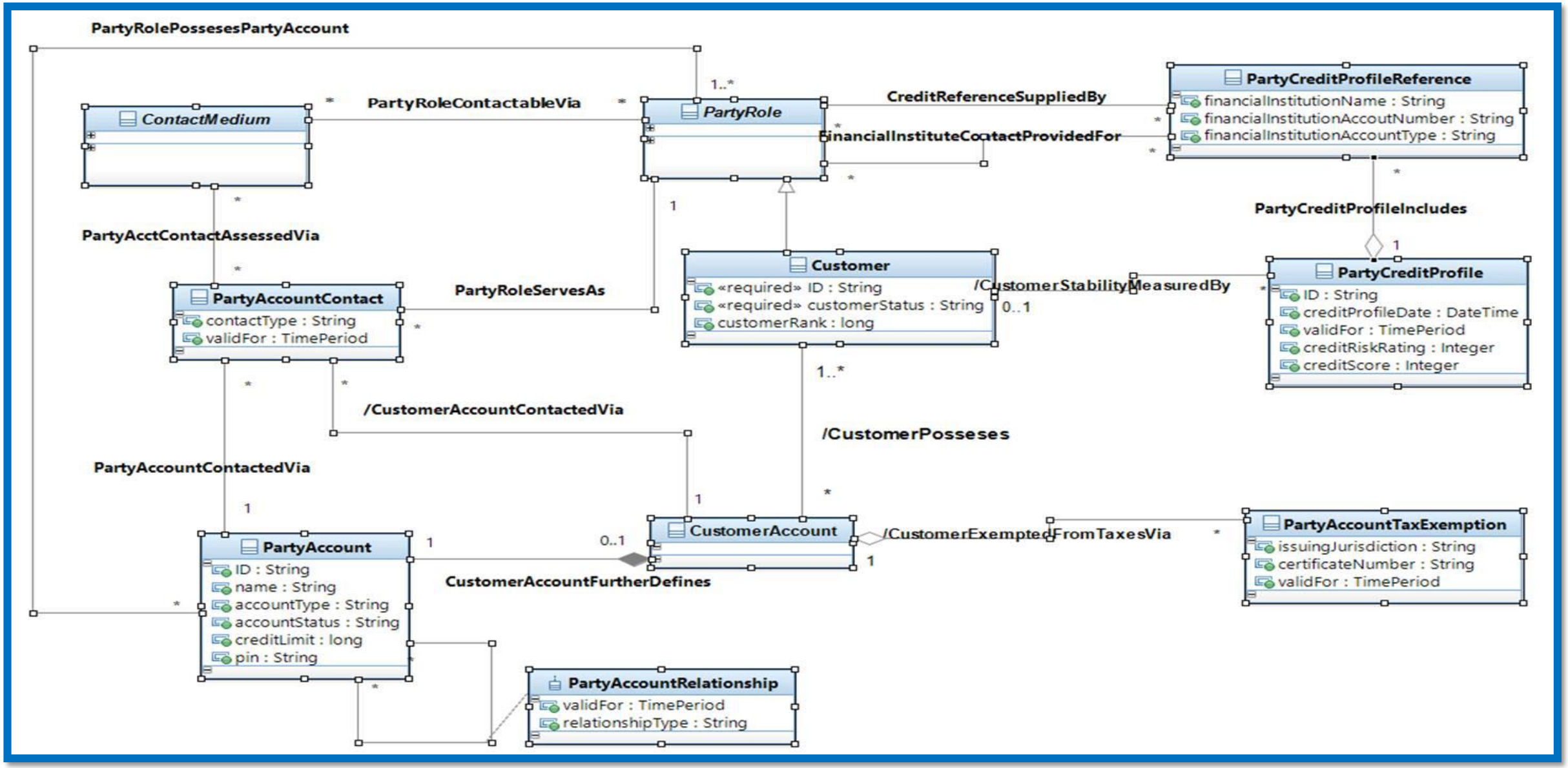
- The SID model define a number of relationships not all of them needs to be implemented for a particular resource within an API
- Choice of link data model must be made when mapping relationships to a REST based representation model

- Hyperlinks
- Link Header
- Dependent Entities

Simple Data Model with Hyperlinks… based on SID

Product Catalog API




PartyRolePossesesPartyAccount
ContactMedium
PartyRoleContactableVia
PartyRole
CreditReferenceSuppliedBy
FinancialInstituteContactProvidedFor
PartyCreditProfileReference
financialInstitutionName : String
financialInstitutionAccoutNumber : String
financialInstitutionAccountType : String
PartyCreditProfileIncludes
PartyAcctContactAssessedVia
PartyAccountContact
contactType : String
validFor : TimePeriod
PartyRoleServesAs
Customer
«required» ID : String
«required» customerStatus : String
customerRank : long
/CustomerStabilityMeasuredBy
PartyCreditProfile
ID : String
creditProfileDate : DateTime
validFor : TimePeriod
creditRiskRating : Integer
creditScore : Integer
/CustomerAccountContactedVia
/CustomerPosseses
PartyAccountContactedVia
PartyAccount
ID : String
name : String
accountType : String
accountStatus : String
creditLimit : long
pin : String
CustomerAccount
/CustomerExemptedFromTaxesVia
PartyAccountTaxExemption
issuingJurisdiction : String
certificateNumber : String
validFor : TimePeriod
CustomerAccountFurtherDefines
PartyAccountRelationship
validFor : TimePeriod
relationshipType : String

Choice of Link models

There are 3 types of data models:

1. Hyperlinks
2. Link Header
3. Dependent Entities

Choice of Link models

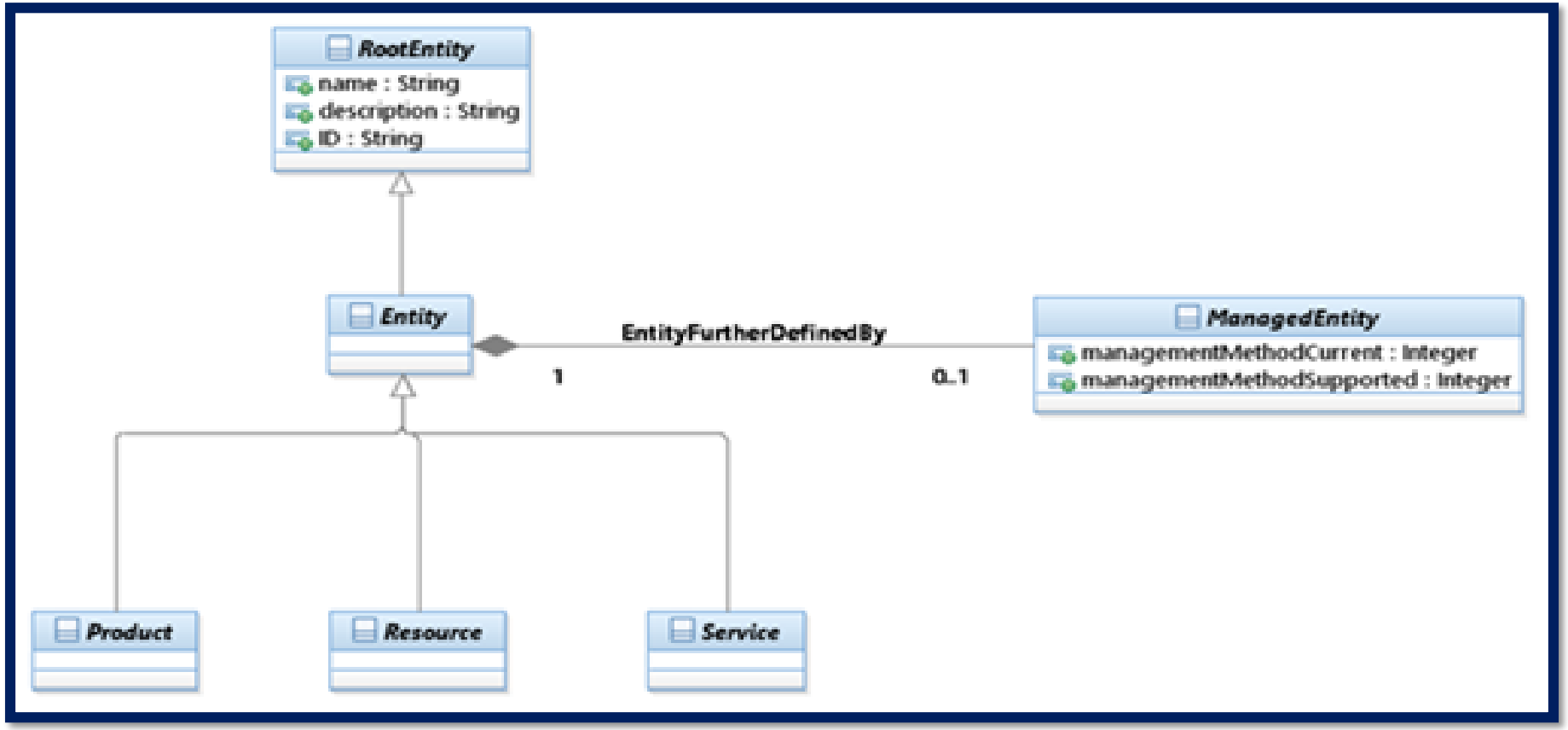
- Embed related object properties within entity representation (transform relationship to dependent entity- data type)
  - This is something used when there is still no resource defined for the associated entity in the API Ecosystem

- Use href or links within the JSON body (hyperlinks) with some useful filtering information
  - This is the preferred approach when the related object is treated as a resource
  - Additional properties are added to the href for filtering and quick retrieval purpose

JSON body for the Customer

```
{
    "id": "c1234",
    "href": "http://serverlocation:port/customerManagement/customer/c1234",
    "name": "DisplayName",
    "status": "Active",
    "description": "Description string",
    ---dependant entity relationship---
    "contactMedium": [
        {
            "type": "Email",
            "validFor": {
                "startDateTime": "2013-04-19T20:42:23.0Z"
            },
            "medium": {
                "emailAddress": "abc@tmforum.com"
            }
        }
    ],
    ----href relationship----
    "customerAccount": [
        {
            "id": "1",
            "href":
    "http://serverlocation:port/customerManagement/customerAccount/1",
            "name": "CustomerAccount1",
            "description": "CustomerAccountDesc1",
            "status": "Active"
        }
    ],
}
```

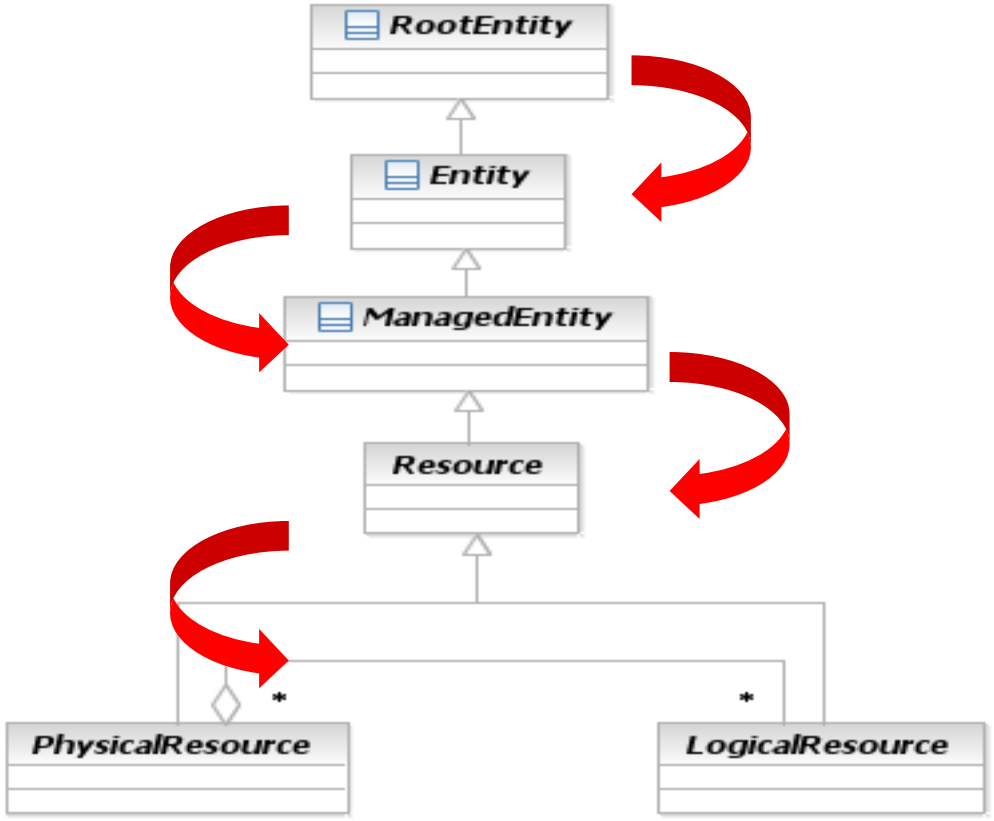
- REST Resources represent SID entities.
- Some SID entities are part of inheritance hierarchies.



Collapsing Classes

- JSON does not support generalization or inheritance (no equivalent of xsi:type at run time)
- In general only the most derived SID classes are used as Resource representations in TMF Open APIs
  - In the SID JSON representation we normally collapse the class into the direct child
  - We then expose the SID Entity as a resource with all the inherited attributes embedded into the JSON representation
- This does not mean that an API SID JSON Resource can’t be extended
  - The REST API Design Patterns support the “@type” property.

Collapsing Classes

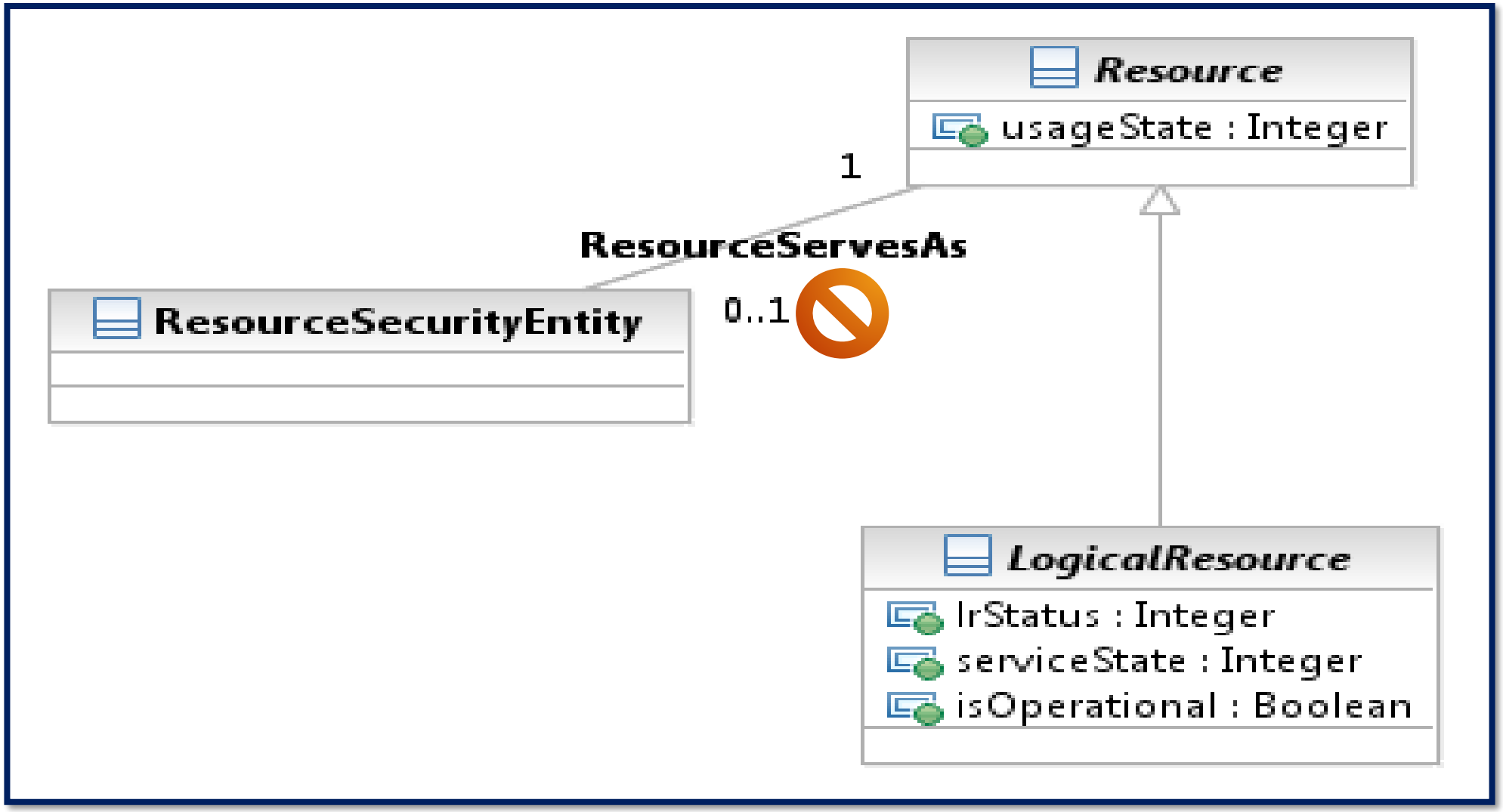


JSON Representation

JSON Representation

- Allow collapsing a class into either its direct parent or its direct child
- Can be recursive as show on figure:
  - In the figure case, LogicalResource inherits from Resource and Resource has no parent
- **If attributes and associations, handle as if present on target class**
- Only valid for direct parent or child today. Extension planned to other associations in future.

In the context of an API, some attributes or associations might not be needed, but might be useful in some other cases.



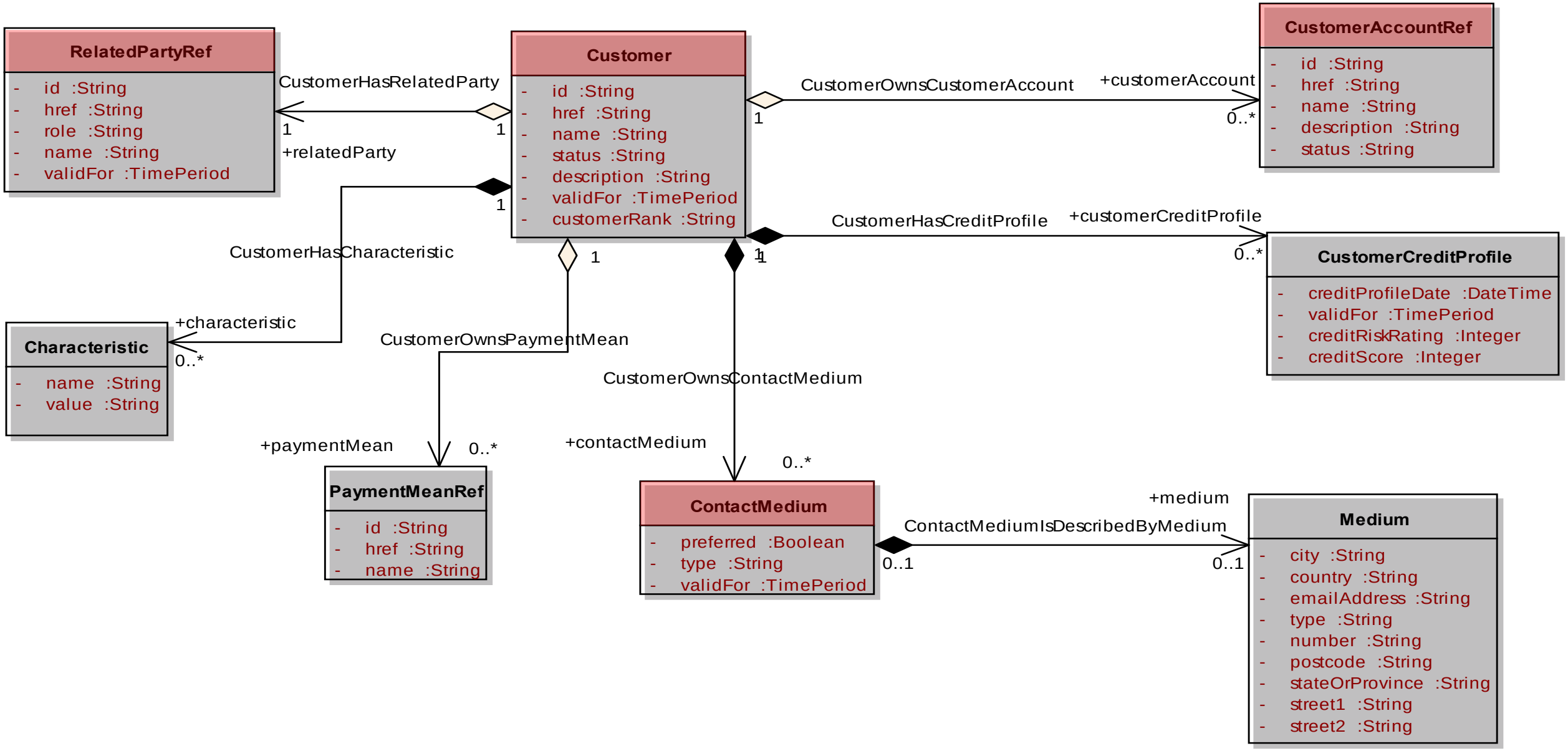
SID/JSON Mapping Methodologies

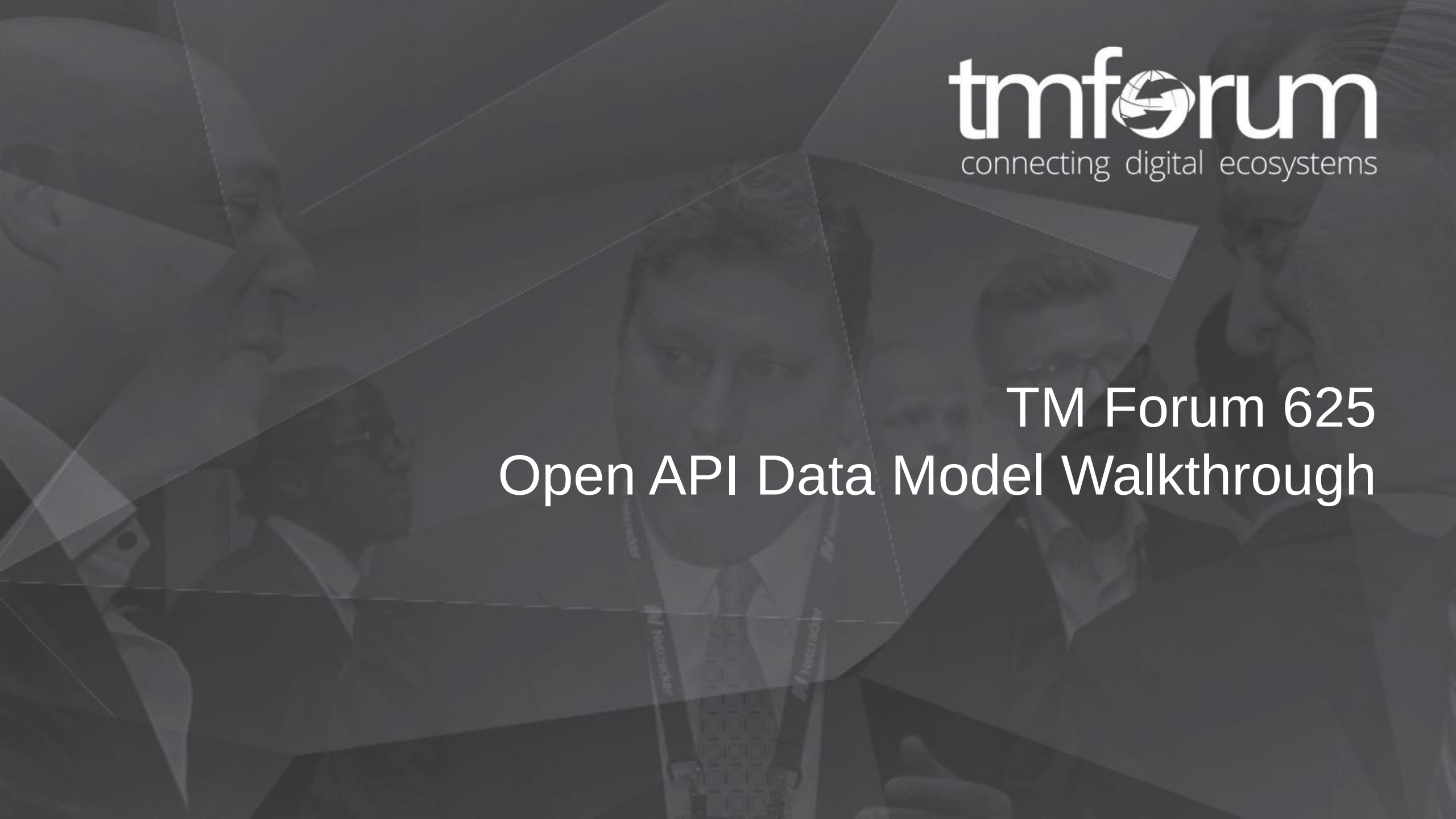
- Choose the Entities to be mapped to API JSON Resources
  - based on management functionality
  - within this or another API Component

- Apply Entity Collapsing Pattern
- Choose related Entities mapped to Dependent Data
  - no need to expose them as resources
  - no API exist to support them as linked entities

Additional SID/JSON Mapping Methodologies

- Transform relationships to other Resource Entities into (array) of href relationships
- Transform relationships to dependent entities into array properties
- Transform Characteristics and Configurable Characteristics into JSON array of Name Value Pairs
- Transform Characteristic Specification into predefined Characteristic Specification JSON construct


RelatedPartyRef
- id :String
- href :String
- role :String
- name :String
- validFor :TimePeriod
CustomerHasRelatedParty
1
1
+relatedParty
Customer
- id :String
- href :String
- name :String
- status :String
- description :String
- validFor :TimePeriod
- customerRank :String
CustomerOwnsCustomerAccount
1
+customerAccount
0..*
CustomerAccountRef
- id :String
- href :String
- name :String
- description :String
- status :String
1
CustomerHasCharacteristic
+characteristic
0..*
Characteristic
- name :String
- value :String
1
CustomerOwnsPaymentMean
+paymentMean
0..*
PaymentMeanRef
- id :String
- href :String
- name :String
1
CustomerOwnsContactMedium
+contactMedium
0..*
ContactMedium
- preferred :Boolean
- type :String
- validFor :TimePeriod
CustomerHasCreditProfile
+customerCreditProfile
0..*
CustomerCreditProfile
- creditProfileDate :DateTime
- validFor :TimePeriod
- creditRiskRating :Integer
- creditScore :Integer
ContactMediumIsDescribedByMedium
0..1
0..1
+medium
Medium
- city :String
- country :String
- emailAddress :String
- type :String
- number :String
- postcode :String
- stateOrProvince :String
- street1 :String
- street2 :String

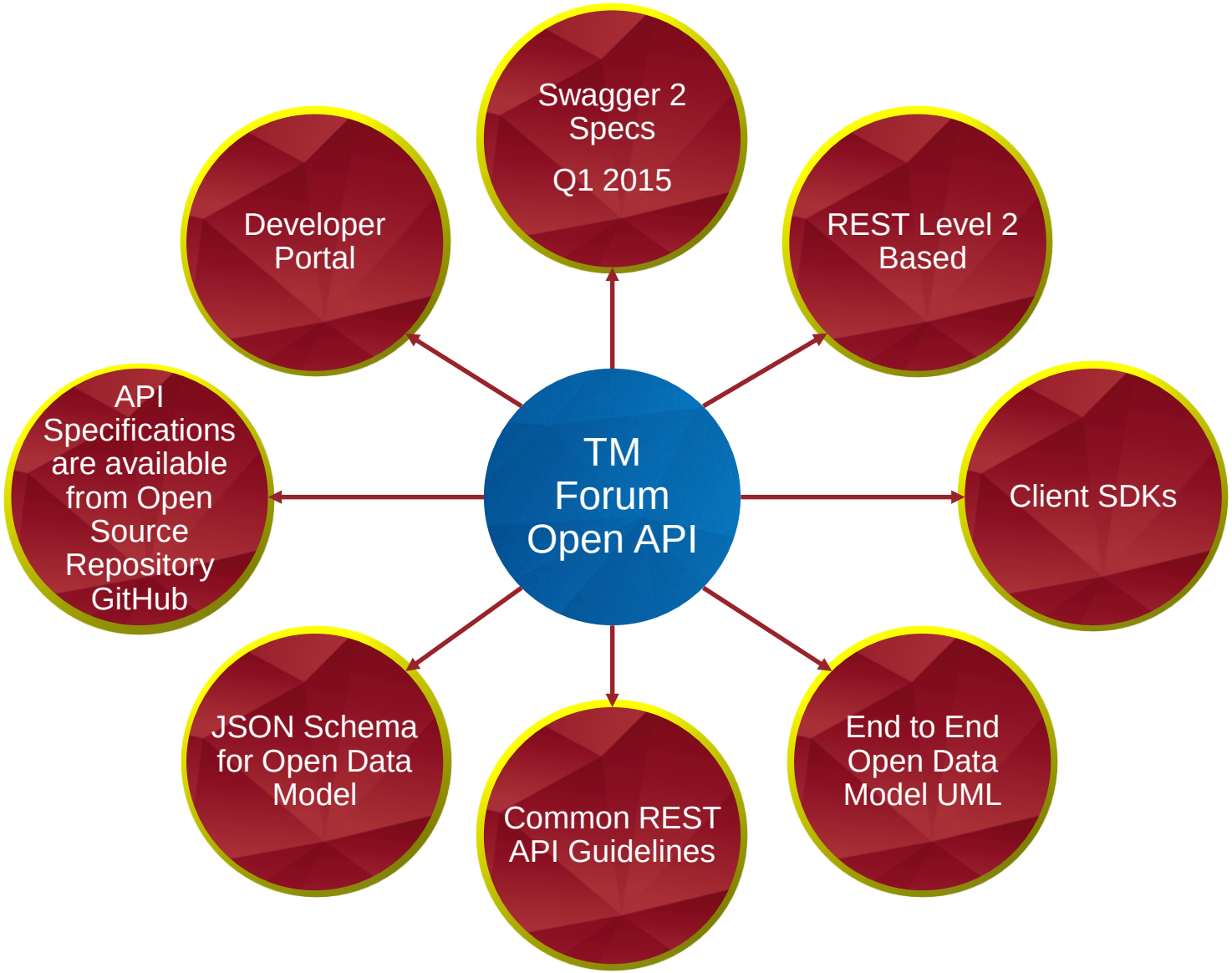


TM Forum 625
Open API Data Model Walkthrough

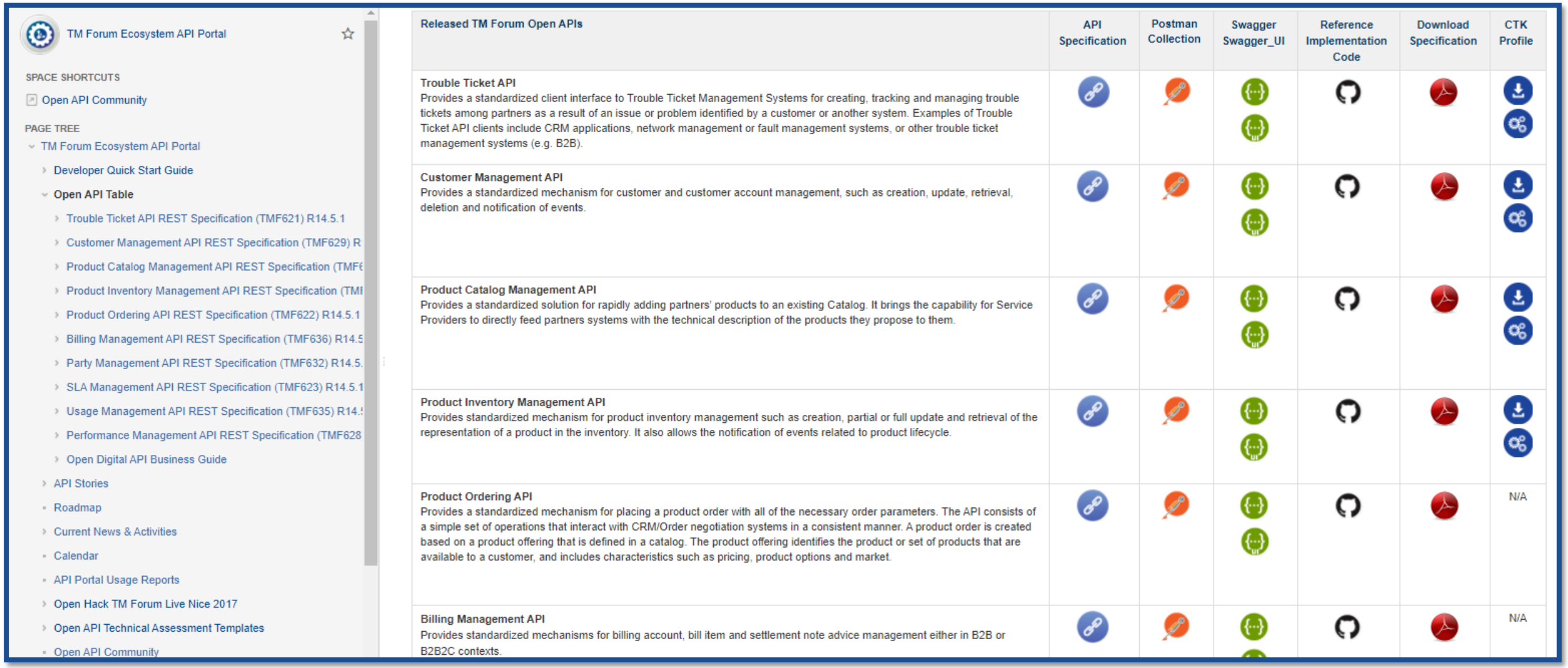
Module 6:
Tooling, Sandbox, Conformance

Presented by: Joann O'Brien

APIs are for Developers…

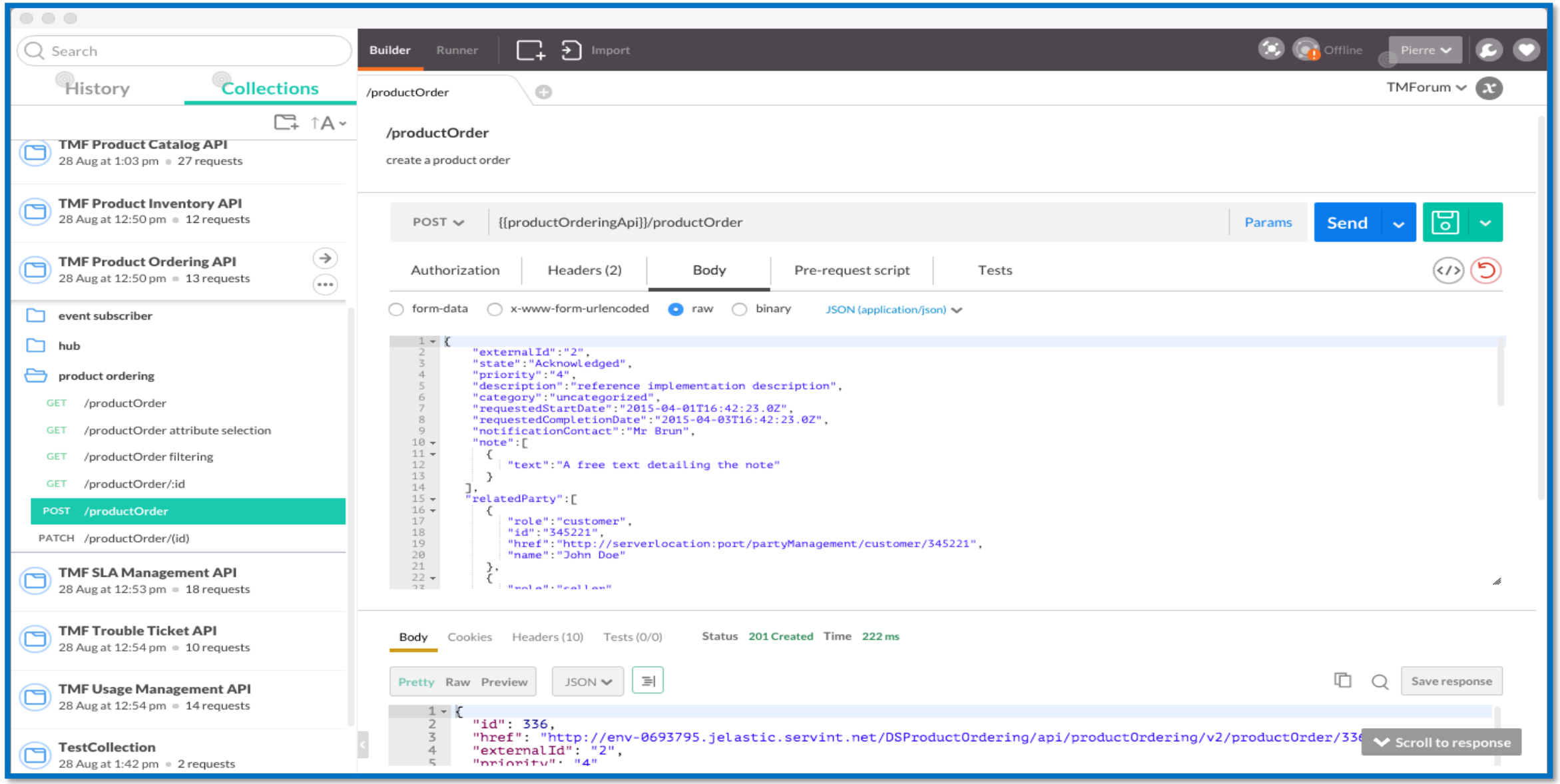


APIs Sandbox and Testing…



http://projects.tmforum.org/wiki/display/API/Open+API+Table

POSTMAN


Search
History
Collections
Builder
Runner
Import
Offline
Pierre
TMForum
/productOrder
TMF Product Catalog API
28 Aug at 1:03 pm • 27 requests
TMF Product Inventory API
28 Aug at 12:50 pm • 12 requests
TMF Product Ordering API
28 Aug at 12:50 pm • 13 requests
event subscriber
hub
product ordering
GET /productOrder
GET /productOrder attribute selection
GET /productOrder filtering
GET /productOrder/:id
POST /productOrder
PATCH /productOrder/(id)
TMF SLA Management API
28 Aug at 12:53 pm • 18 requests
TMF Trouble Ticket API
28 Aug at 12:54 pm • 10 requests
TMF Usage Management API
28 Aug at 12:54 pm • 14 requests
TestCollection
28 Aug at 1:42 pm • 2 requests
/productOrder
create a product order
POST
{{productOrderingApi}}/productOrder
Params
Send
Authorization
Headers (2)
Body
Pre-request script
Tests
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"externalId":"2",
"state":"Acknowledged",
"priority":"4",
"description":"reference implementation description",
"category":"uncategorized",
"requestedStartDate":"2015-04-01T16:42:23.0Z",
"requestedCompletionDate":"2015-04-03T16:42:23.0Z",
"notificationContact":"Mr Brun",
"note":[
{
"text":"A free text detailing the note"
}
],
"relatedParty":[
{
"role":"customer",
"id":"345221",
"href":"http://serverlocation:port/partyManagement/customer/345221",
"name":"John Doe"
},
{
Body
Cookies
Headers (10)
Tests (0/0)
Status 201 Created
Time 222 ms
Pretty
Raw
Preview
JSON
Save response
{
"id": 336,
"href": "http://env-0693795.jelastic.servint.net/DSProductOrdering/api/productOrdering/v2/productOrder/336",
"externalId": "2",
Scroll to response

Swagger

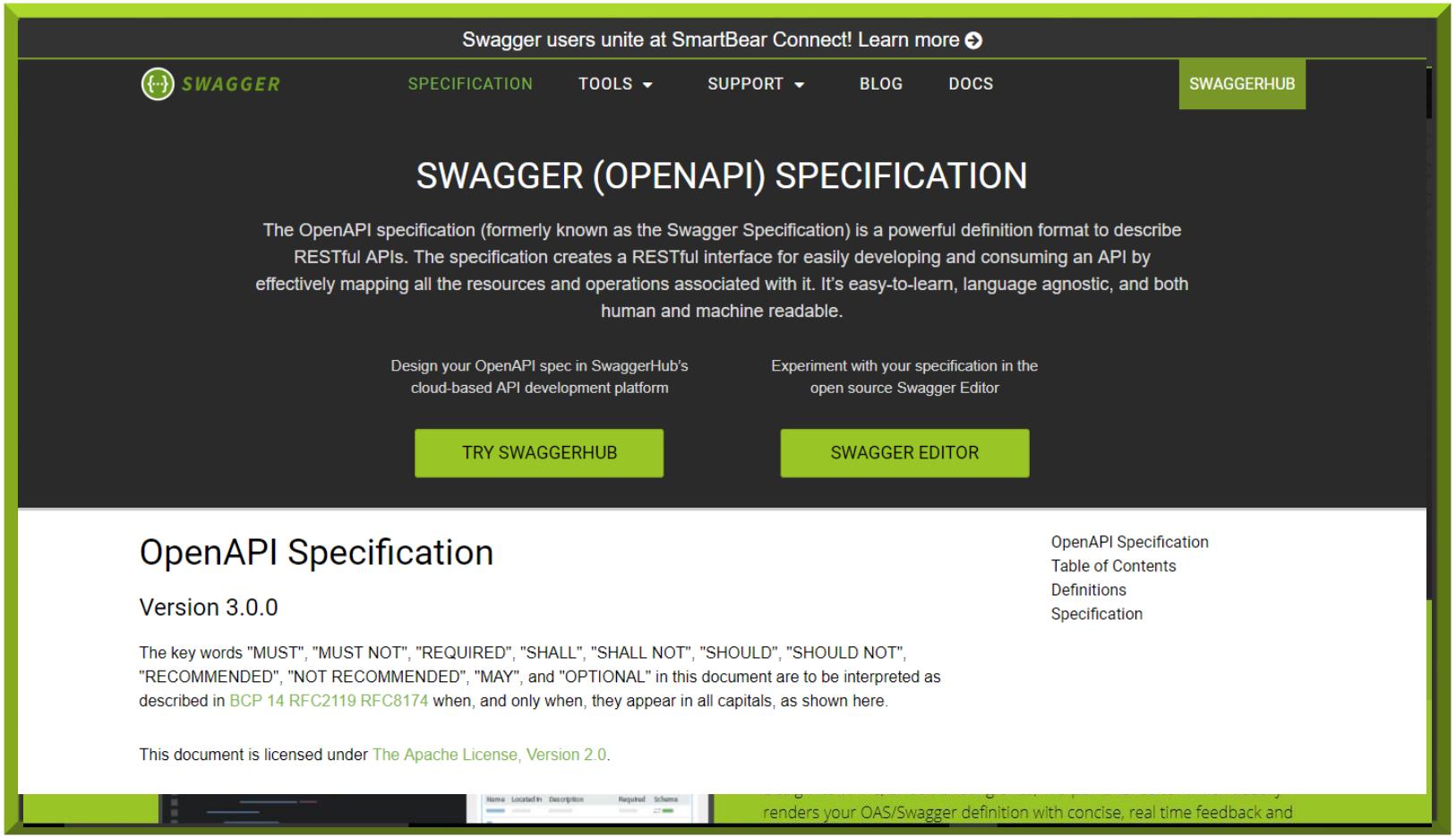
SWAGGER is a powerful open source framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs.

- Swagger Doc
- Swagger Specification
- Swagger Hub

Swagger

SWAGGER is a powerful open source framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs.

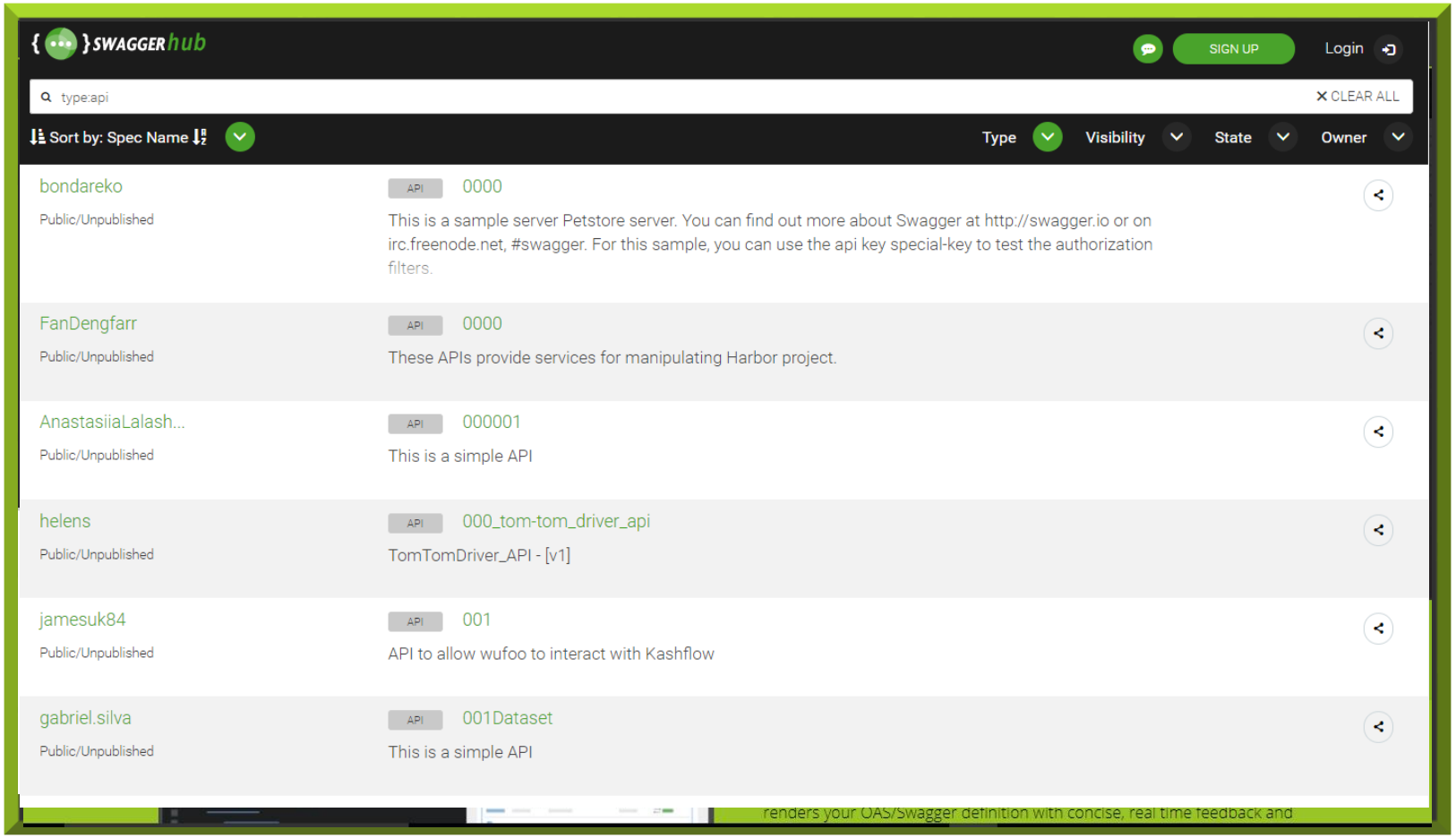
- Swagger Doc
- Swagger Specification
- Swagger Hub



Swagger

SWAGGER is a powerful open source framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs.

- Swagger Doc
- Swagger Specification
- Swagger Hub



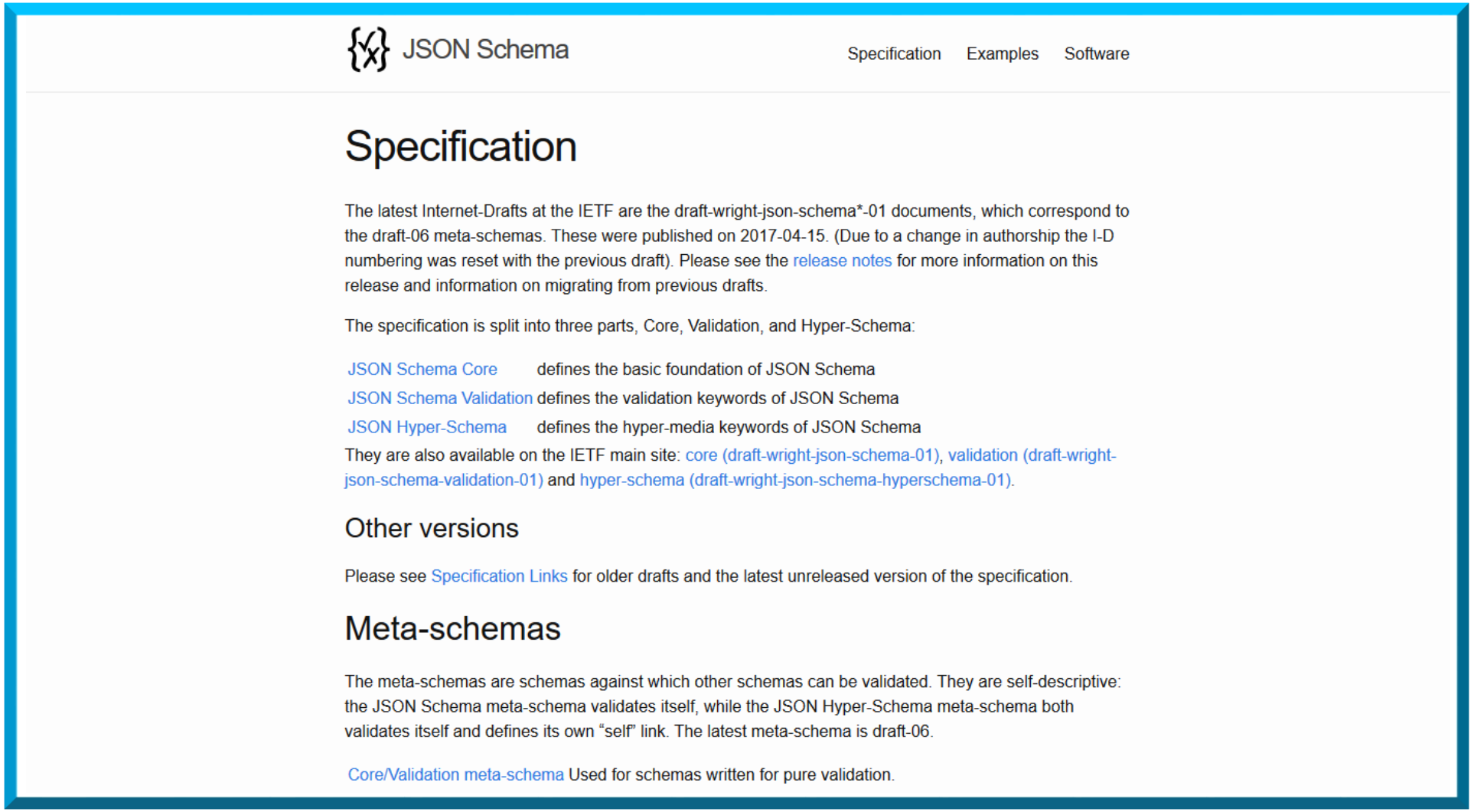
JSON Schema

The equivalent of XML Schema for JSON

JSON Schema Documentation
http://json-schema.org/documentation.html

JSON Schema

The equivalent of XML Schema for JSON



JSON Schema

Specification Examples Software

Specification

The latest Internet-Drafts at the IETF are the draft-wright-json-schema*-01 documents, which correspond to the draft-06 meta-schemas. These were published on 2017-04-15. (Due to a change in authorship the I-D numbering was reset with the previous draft). Please see the release notes for more information on this release and information on migrating from previous drafts.

The specification is split into three parts, Core, Validation, and Hyper-Schema:

JSON Schema Core defines the basic foundation of JSON Schema
JSON Schema Validation defines the validation keywords of JSON Schema
JSON Hyper-Schema defines the hyper-media keywords of JSON Schema

They are also available on the IETF main site: core (draft-wright-json-schema-01), validation (draft-wright-json-schema-validation-01) and hyper-schema (draft-wright-json-schema-hyperschema-01).

Other versions

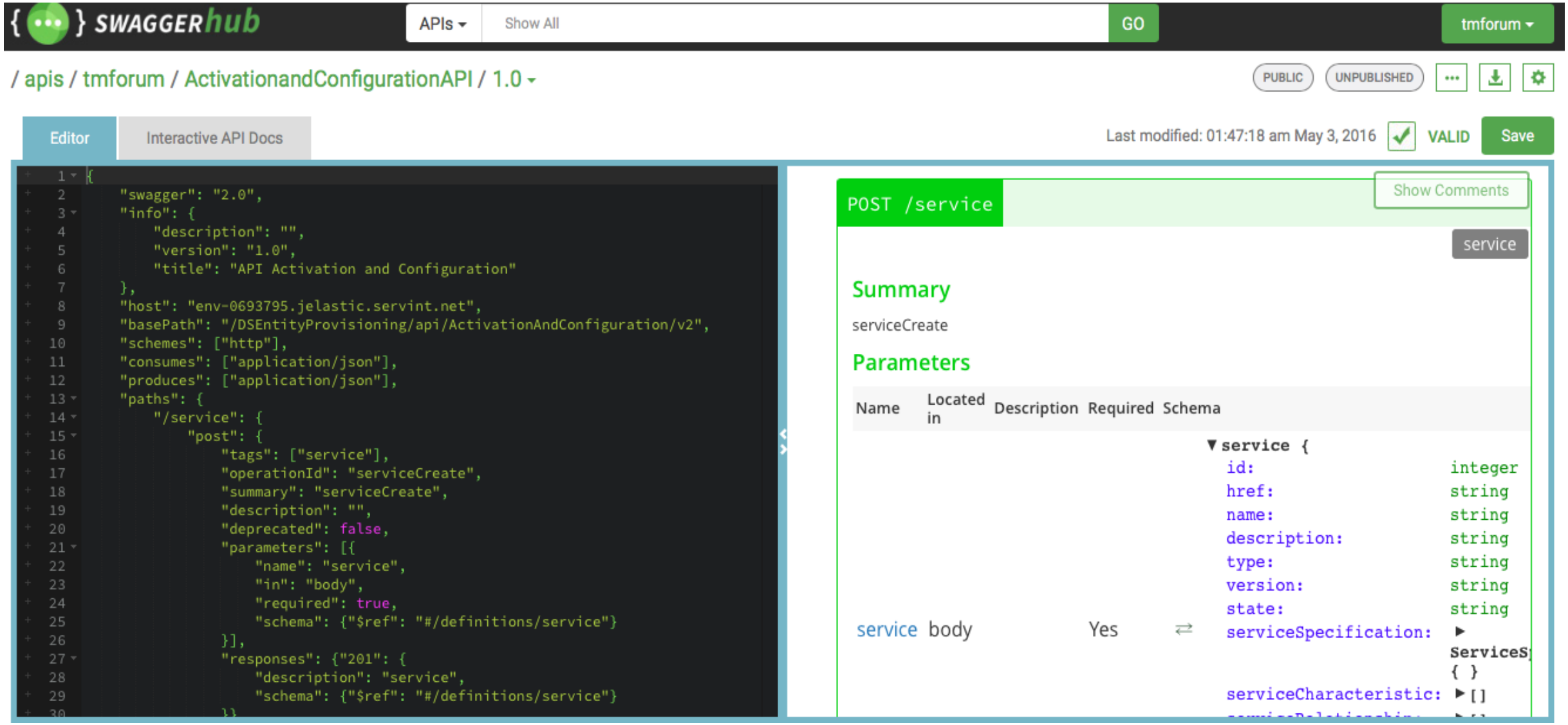
Please see Specification Links for older drafts and the latest unreleased version of the specification.

Meta-schemas

The meta-schemas are schemas against which other schemas can be validated. They are self-descriptive: the JSON Schema meta-schema validates itself, while the JSON Hyper-Schema meta-schema both validates itself and defines its own "self" link. The latest meta-schema is draft-06.

Core/Validation meta-schema Used for schemas written for pure validation.

Swagger and JSON Schema V4



https://app.swaggerhub.com/apis/tmforum/api-trouble_ticket/2.0

Read Only

```yaml
swagger: '2.0'
info:
  description: ''
  version: '1.0'
  title: API Appointment
host: api.orange.com
basePath: /
schemes:
  - http
consumes:
  - application/json
produces:
  - application/json
paths:
  /appointment:
    post:
      tags:
        - appointment
      operationId: appointmentCreate
      summary: create
      description: 'After checking free slots, this operation is used to create an
        appointment with all its characteristics.'
      deprecated: false
      parameters:
        - name: appointment
          in: body
          required: true
          schema:
            $ref: '#/definitions/Appointment'
      responses:
        '201':
          description: Success
          schema:
            $ref: '#/definitions/Appointment'
        '400':
          description: bad_request
          schema:
            $ref: '#/definitions/Error'
    get:
      tags:
        - appointment
      operationId: appointmentList
      summary: find
      description: "This operation is used to retrieve appointment information using
```

API Appointment

Version 1.0

Paths

/appointment

POST /appointment

appointment

Summary

create

Description

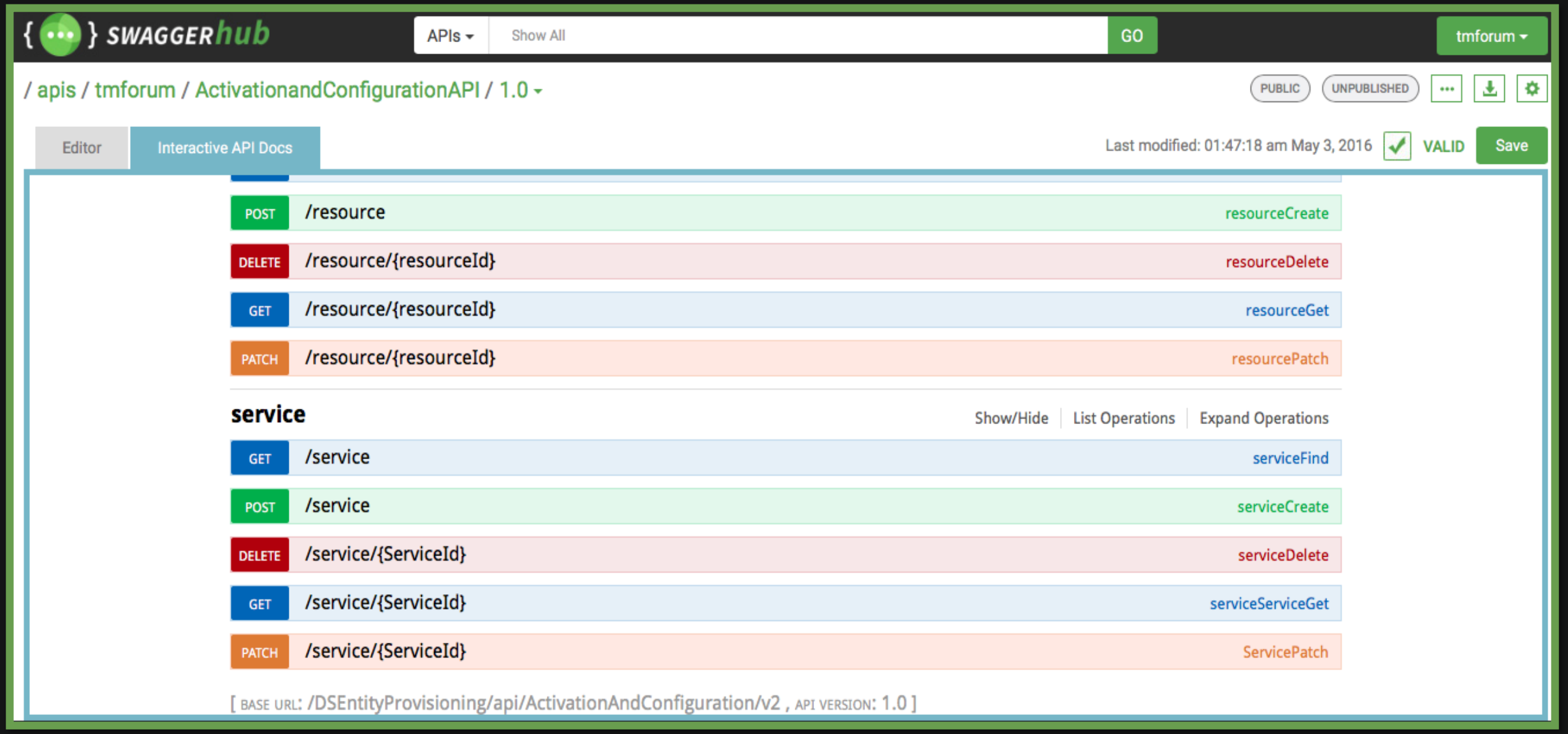
After checking free slots, this operation is used to create an appointment with all its characteristics.

Parameters

Name	Located in	Description	Required	Schema

▼ **Appointment** {
alarm: boolean
attachment: ▶ []
category: string
endDate: string

SWAGGERhub User Interface



What is API Certification?

- Certification is the process by which an implementation is confirmed to support the standard definition of an API
- TM Forum provides the tools required to certify implementations against API definitions
- Certification is only against Mandatory features of an API

Main Artifacts

API Conformance Profile document	Describe what is tested Mandatory attributes and Operations
CTK script	Provided by TMF to be executed against certified implementation of the API
Cloud based Reference Implementation of the API	Provided by TMF as implementation test bed Is an example of a certified API

What is tested by a CTK?

1. Implementation of all the mandatory operations
2. Support of the core data model of the API presence and proper constructs for entities in the data model
3. Support of mandatory filters and attribute selection constructs as defined in the API specification
4. Support of all mandatory notifications
5. Support of the core event data model of the API
6. Mandatory relationships between entities (resources) **defined** and **managed** by the API are present and operation(href leads to real resources within the API)
7. Mandatory relationships between entities (resources) not **defined** and **managed** by the API are present

What is required to be conformant?

An API must pass all the CTK tests in order to be approved as a valid implementation of the TMF specification

An API must have implemented all the CORE or MANDATORY features to be conformant.

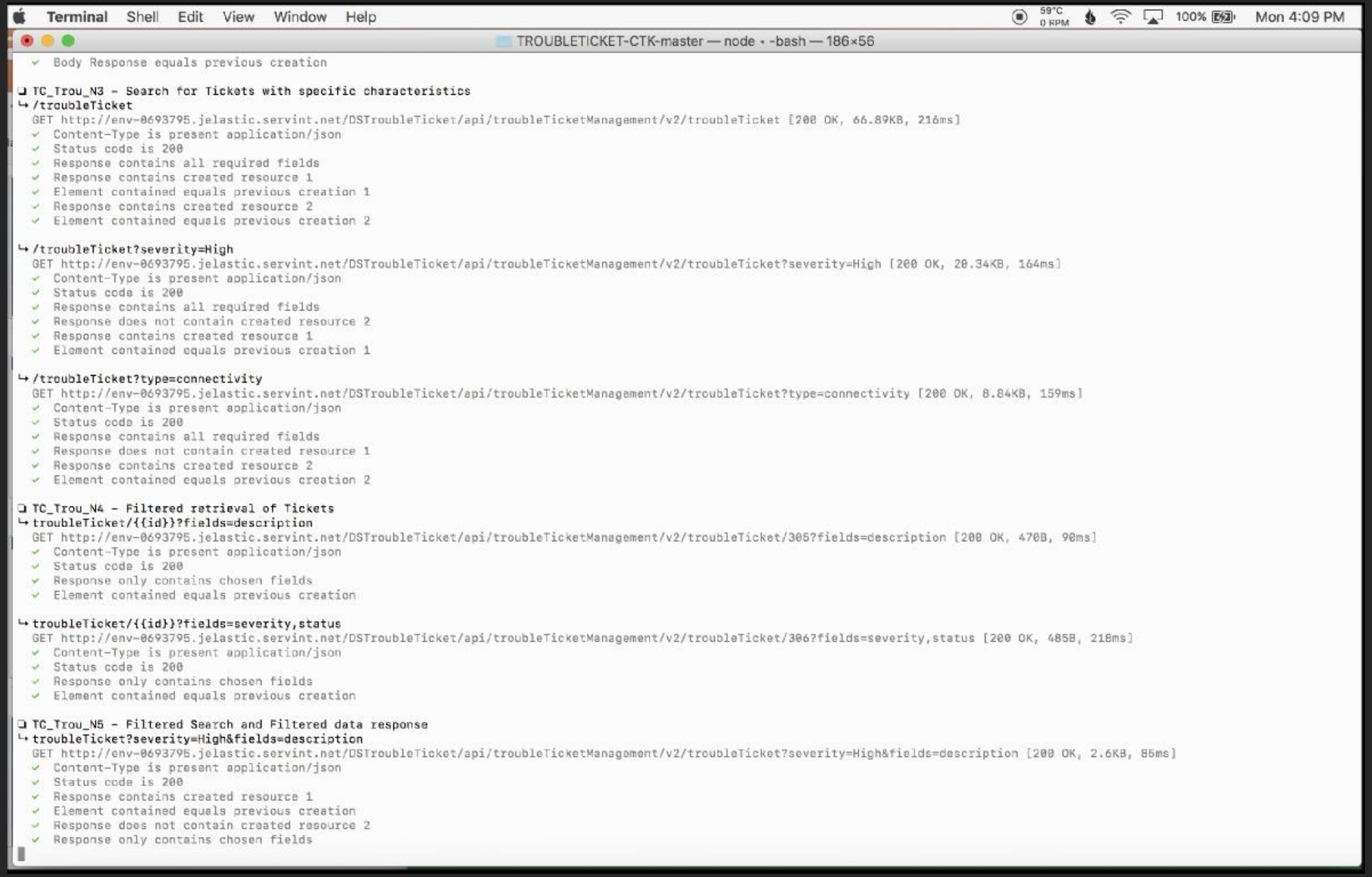
If a mandatory feature is missing then the API is eliminated and out of consideration for conformance

Trouble Ticket API Conformance Profile Walkthrough

Attribute Name	Mandatory or Optional	Comments
id	M (in response messages) O (otherwise)	Generated by the server and provided in the response upon resource creation. Accepted in entity-creation requests if the server supports the incoming identifier as the reference to create new resources
href	M (in response messages) O (otherwise)	Value in response must be the same as the one set in Location header provided upon entity creation
correlationId	O	
description	M	
severity	M	This is the severity as identified by the requestor (perceived severity). The server may change its

http://projects.tmforum.org/wiki/display/API/Open+API+Table

Trouble Ticket API CTK Example


Terminal Shell Edit View Window Help
59°C 0 RPM
100%
Mon 4:09 PM
TROUBLETICKET-CTK-master — node • -bash — 186×56
✓ Body Response equals previous creation
❑ TC_Trou_N3 - Search for Tickets with specific characteristics
↳ /troubleTicket
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket [200 OK, 66.89KB, 216ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response contains all required fields
✓ Response contains created resource 1
✓ Element contained equals previous creation 1
✓ Response contains created resource 2
✓ Element contained equals previous creation 2
↳ /troubleTicket?severity=High
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket?severity=High [200 OK, 20.34KB, 164ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response contains all required fields
✓ Response does not contain created resource 2
✓ Response contains created resource 1
✓ Element contained equals previous creation 1
↳ /troubleTicket?type=connectivity
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket?type=connectivity [200 OK, 8.84KB, 159ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response contains all required fields
✓ Response does not contain created resource 1
✓ Response contains created resource 2
✓ Element contained equals previous creation 2
❑ TC_Trou_N4 - Filtered retrieval of Tickets
↳ troubleTicket/{{id}}?fields=description
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket/305?fields=description [200 OK, 470B, 90ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response only contains chosen fields
✓ Element contained equals previous creation
↳ troubleTicket/{{id}}?fields=severity,status
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket/306?fields=severity,status [200 OK, 485B, 218ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response only contains chosen fields
✓ Element contained equals previous creation
❑ TC_Trou_N5 - Filtered Search and Filtered data response
↳ troubleTicket?severity=High&fields=description
GET http://env-0693795.jelastic.servint.net/DSTroubleTicket/api/troubleTicketManagement/v2/troubleTicket?severity=High&fields=description [200 OK, 2.6KB, 85ms]
✓ Content-Type is present application/json
✓ Status code is 200
✓ Response contains created resource 1
✓ Element contained equals previous creation
✓ Response does not contain created resource 2
✓ Response only contains chosen fields

Module 7:
Introduction to TM Forum MicroServices

Presented by: Joann O'Brien

MicroService Architecture

Microservice architecture is a design pattern to build software system composed of multiple reusable services that interacts with each other to provide a business capability.

Each MicroService is

- Isolated
- Autonomous
- Context-bound

Isolation

> Isolation is at the heart of microservices. A microservice is meant to be architected, created, deployed, maintained and retired without affecting any other microservice. You can't do any, let alone all, of that without good isolation
>
> *Donald Belcham*

Isolation helps in building and maintaining a single capability without impacting other functions. Isolation principle enables:

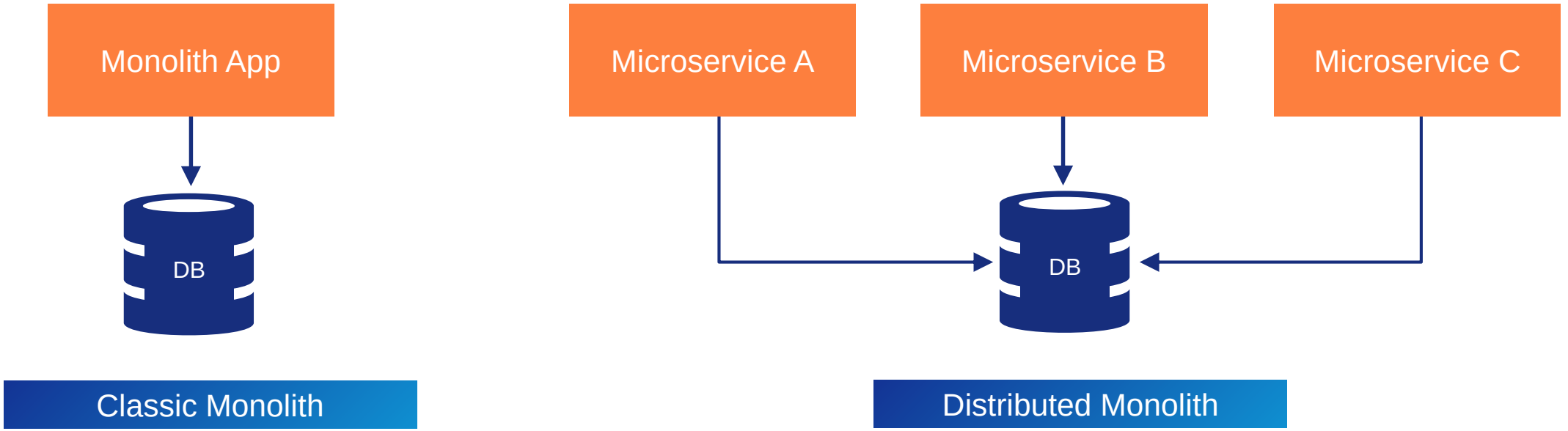
- Design, Build, Test and Release isolation - Enables continuous delivery, reduced time to market and time to value as change is isolated to a specific business function
- Database isolation - Eliminates integration at database level. Any changes in database affects only one microservice and thus reduces surface area for testing, validation and deployment
- Failure isolation - Contain propagation of failure and prevent cascading impact to whole system

Autonomous

- Microservices are autonomous, self-governing and controls their own destiny.
- Consumers cannot impose obligation (Command & Control) on provider.
- An autonomous provider service can only promise its own behavior by publishing its interface (API)
- Consumer can choose to accept or reject the promise offered by provider but cannot direct it to behave in a particular way.

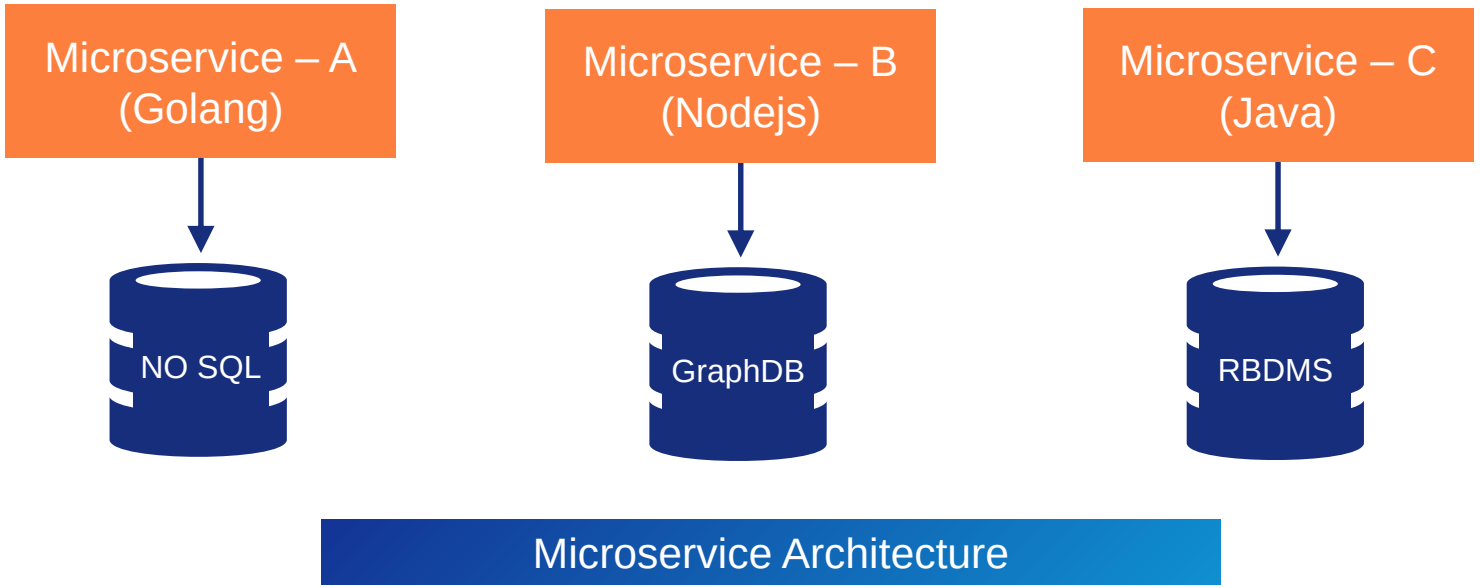
Data Encapsulation

Distributed Monolith Anti Pattern

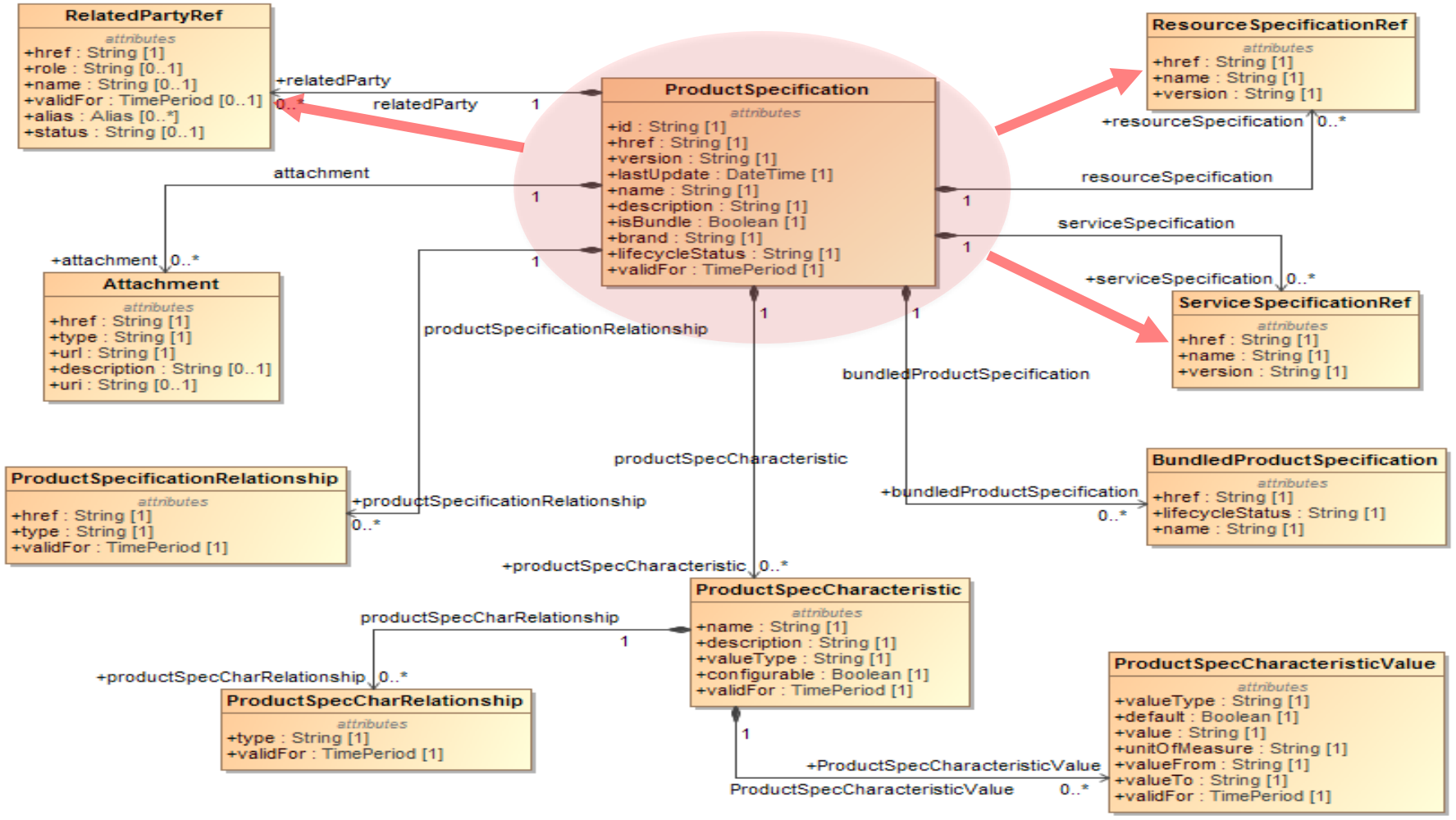


Distributed Monolith is a major anti-pattern where bunch of so called independent services shares a common database for persistent state.

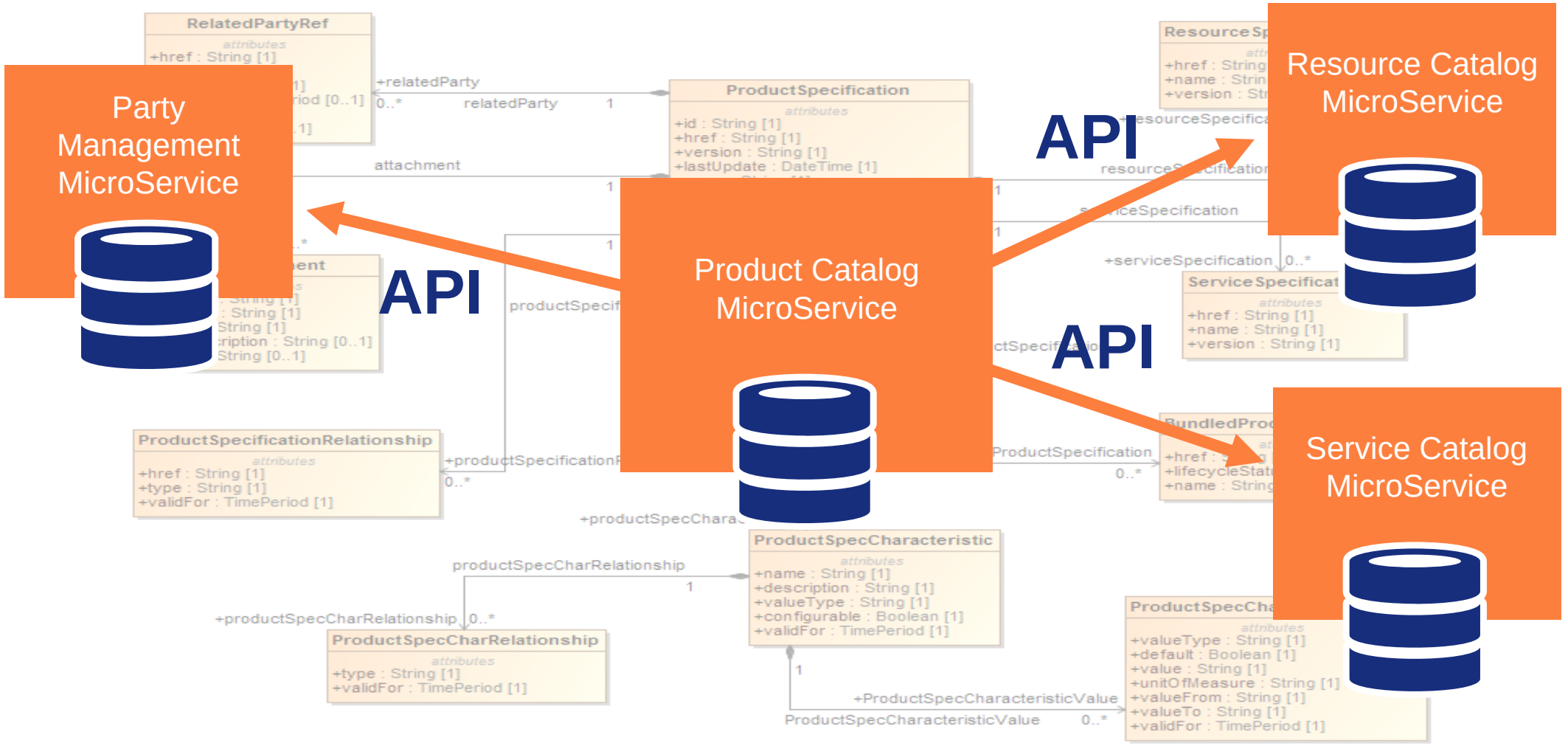
Persistency Isolation



Domain Driven Open API Micro Services



Domain Driven Open API Micro Services



Expose one MicroService per API managed Resource.