TM Forum Input to ONAP Modeling Workshop

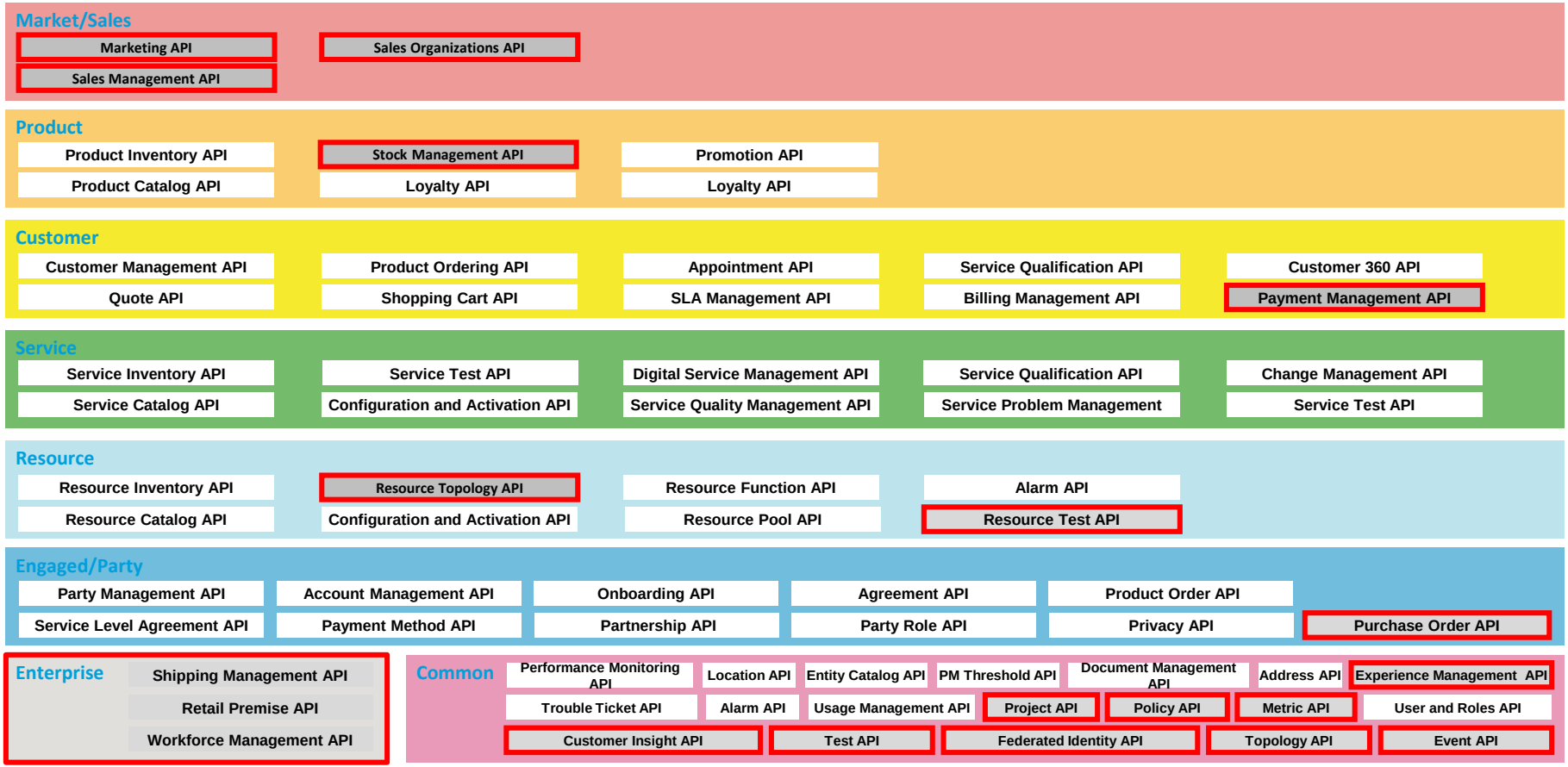
December 14, 2017

John Wilmes
Director of IoT Projects
TM Forum

Model Driven API specifications and the TM Forum Information Framework (SID)

SID is static and focused on data	It describes business entities It does not describe what you can do with those business entities
TMF Open APIs use the SID as the basis for their data model	TMF API Entities are REST/JSON realizations of the SID Entities (the Data part of the Shared Information and Data)
TMF Open APIs add behaviour to the SID	They define the operations that can be executed on the entities They define the interaction patterns Request/Response/Exceptions and Notifications
TMF Open APIs use REST-based design patterns	All TMF Open APIs use a consistent set of API design patterns (TMF 630 & TMF 631) Version 3.0 supports Polymorphism
TMF Open APIs facilitate Dynamic Product & Service Creation	Polymorphism supports the dynamic creation of products and services using a common core API proven in catalyst projects Polymorphic API can carry the service / resource specific payloads defined by MEF, ONF, etc.

Open API Structure – Level 1 mapped to TM Forum Information Framework (SID) level 0



- **The data structure provided with every TM Forum Open API is aligned to the SID**
- **The SID is widely adopted throughout the communications industry**
  - This helps ensure interoperability – one of the key objectives of the TM Forum Open API program
- **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**

- The SID model defines a number of relationships
- Not all of them need 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

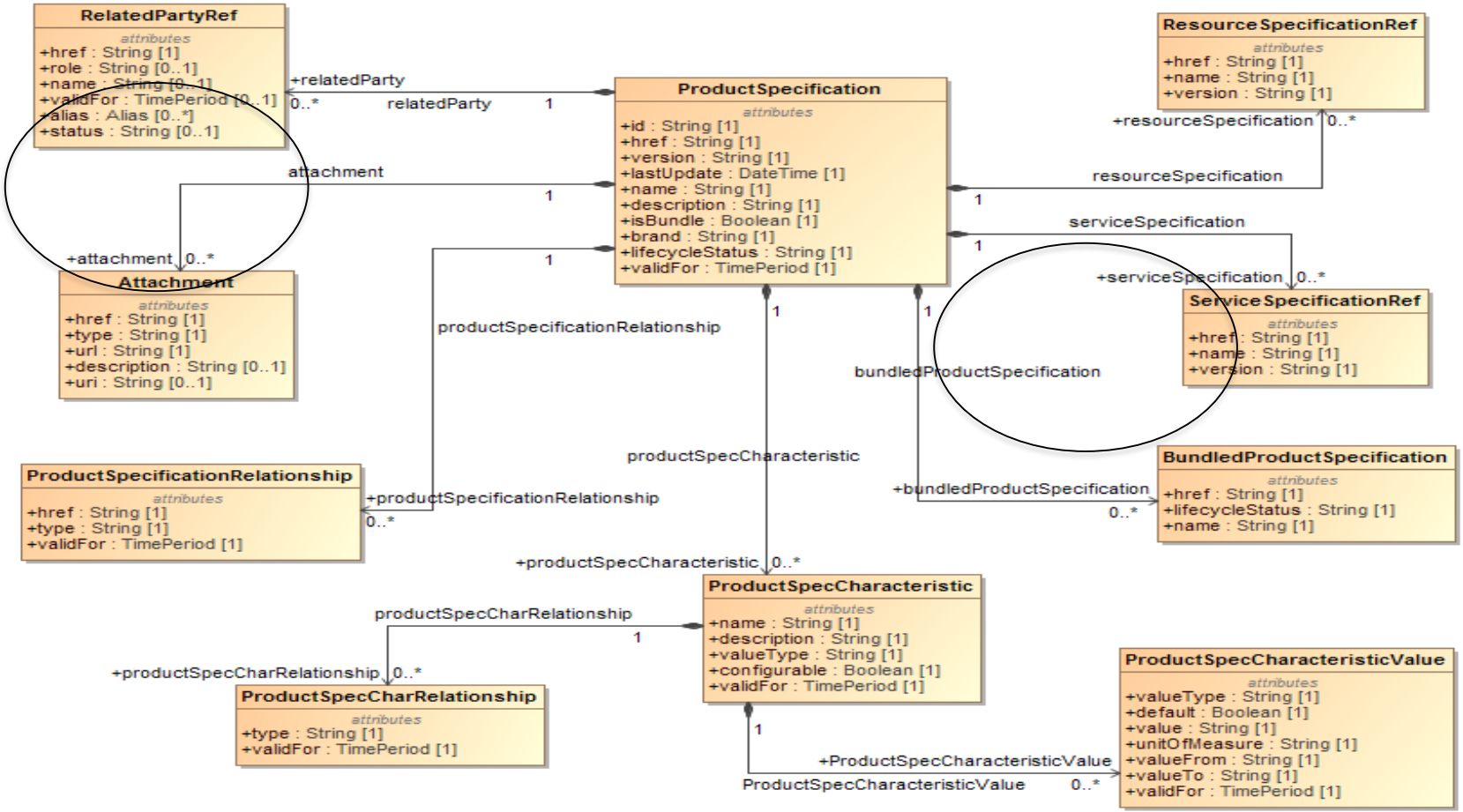
Hyperlinks

Link Header

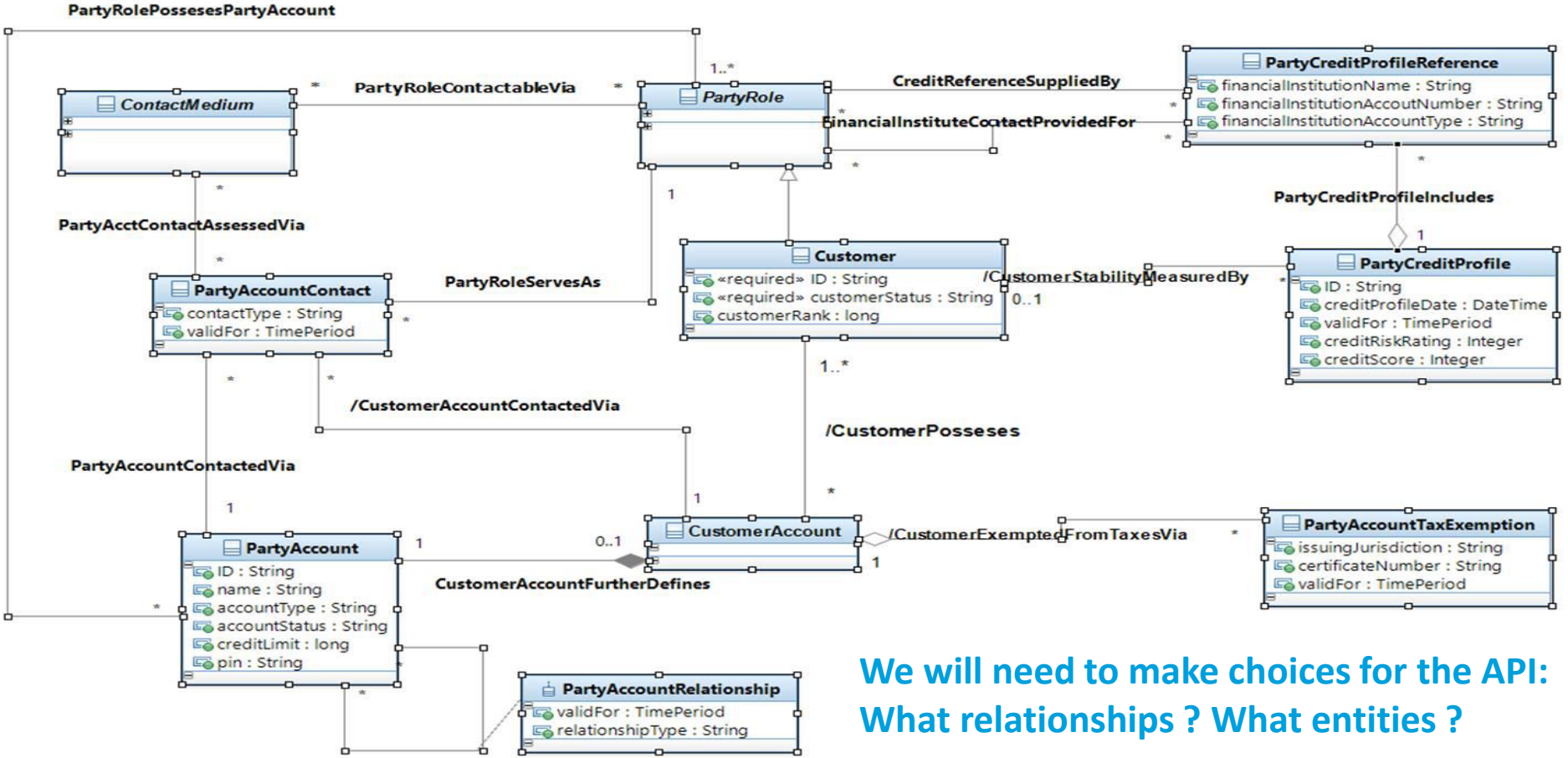
Dependent Entities

Simple Data Model with Hyperlinks… based on TM Forum Information Framework (SID)

Product Catalog API



Customer SID Business Entities and the API Model

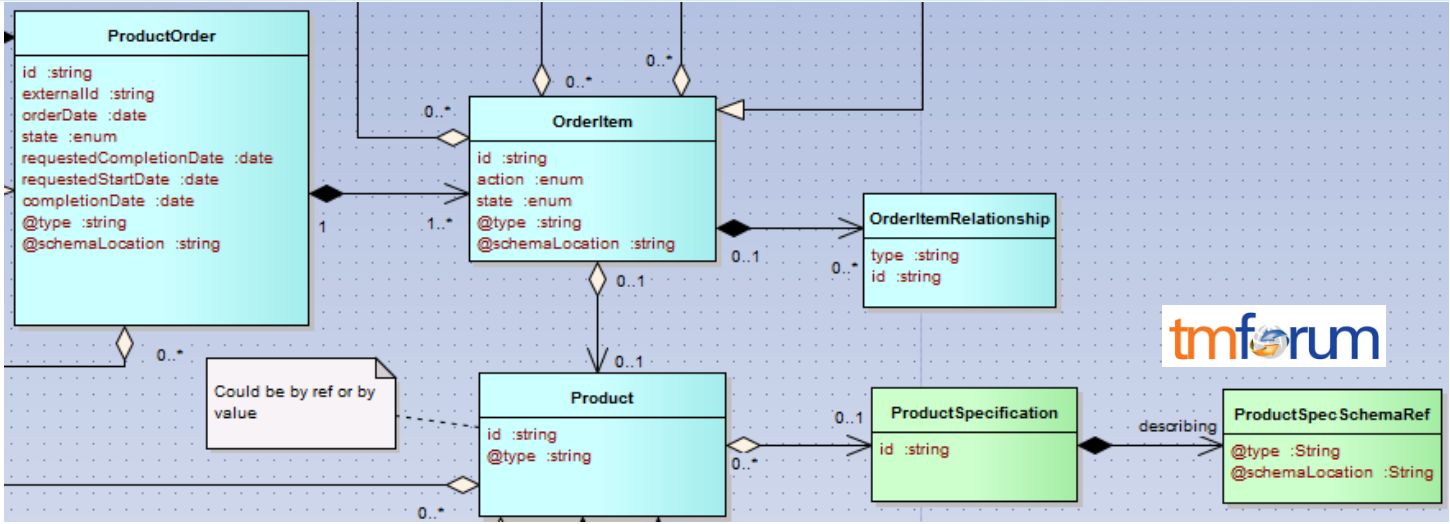


We will need to make choices for the API: What relationships ? What entities ?

- Our proposal is to do the same for the APIs used to automate interactions between BSS and ONAP
- Introducing polymorphism pattern in Service Order API (in progress), Service Catalog API (already done in TMF release 17.5) and Service Inventory (in progress) will allow the creation of flexible ‘service-agnostic’ API

- MEF SONATA LSO-SDK-R1 leveraged TMF Open API features introduced with TMF API guidelines 3.0
- In particular, polymorphism pattern was used in productOrder API and productOfferingQualification API to describe Product Specification and use this to dynamically extend product ordering configuration
- Benefit of this pattern is to decouple API model (which are fully product-agnostic) from Product Description that could be model driven and be described in JSON / YML file
- These APIs were used in a PoC between AT&T, Colt and Orange

API ProductOrder resource model extraction



- A ProductOrder is made of OrderItem(s)
- An OrderItem describes an operation on a product (or a future product)
- A product is defined by a ProductSpecification

The "describing" attribute allows us to describe

- Type of the productSpec to be described…. An UNI, an E-Line, etc…
- Schema location where product specification is described

Product Specifications are defined in JSON file in MEF-GIT

MEF-GIT / **MEF-LSO-Sonata-SDK** Private

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

Branch: master

MEF-LSO-Sonata-SDK / experimental / api / ProductSpecDescription / **Ordering** /

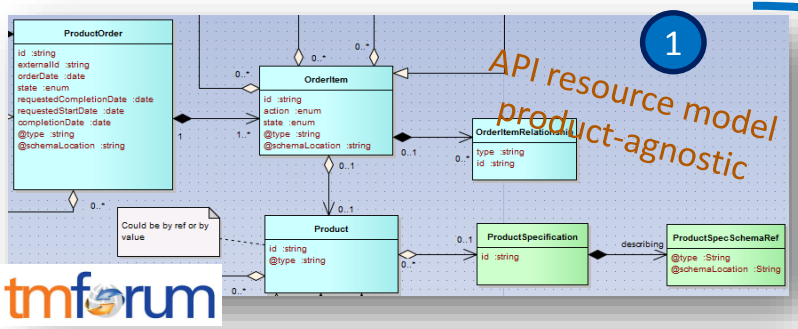
bigludo7 Add files via upload

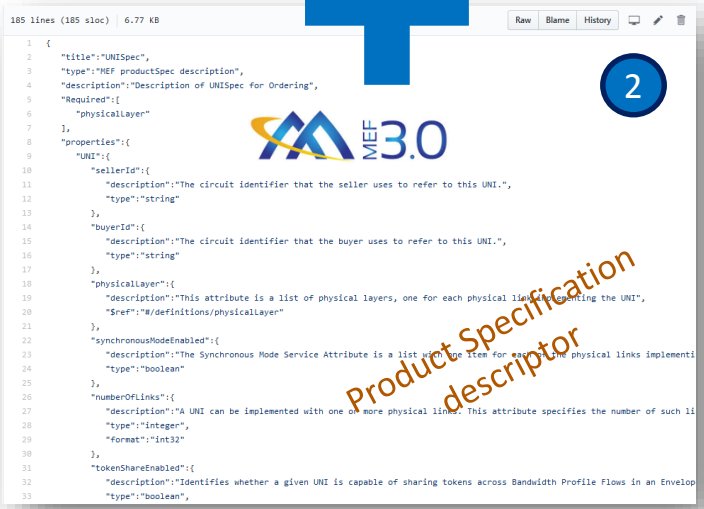
.gitignore	Create .gitignore
ELineSpec.json	Add files via upload
ENNICEEndPointSpec.json	Add files via upload
UNICEEndPointSpec.json	Add files via upload
UNISpec.json	Add files via upload

185 lines (185 sloc) | 6.77 KB — Raw | Blame | History

```
{
   "title":"UNISpec",
   "type":"MEF productSpec description",
   "description":"Description of UNISpec for Ordering",
   "Required":[
      "physicalLayer"
   ],
   "properties":{
      "UNI":{
         "sellerId":{
            "description":"The circuit identifier that the seller uses to refer to this UNI.",
            "type":"string"
         },
         "buyerId":{
            "description":"The circuit identifier that the buyer uses to refer to this UNI.",
            "type":"string"
         },
         "physicalLayer":{
            "description":"This attribute is a list of physical layers, one for each physical link implementing the UNI",
            "$ref":"#/definitions/physicalLayer"
         },
         "synchronousModeEnabled":{
            "description":"The Synchronous Mode Service Attribute is a list with one item for each of the physical links implementi
            "type":"boolean"
         },
         "numberOfLinks":{
            "description":"A UNI can be implemented with one or more physical links. This attribute specifies the number of such li
            "type":"integer",
            "format":"int32"
         },
         "tokenShareEnabled":{
            "description":"Identifies whether a given UNI is capable of sharing tokens across Bandwidth Profile Flows in an Envelop
            "type":"boolean",
```

MEF API leveraging TMF polymorphism





POST productOrder/


```
        "id":"3",
        "action":"add",
        "productOffering":{
          "id":"ELINE_EP_UNI_Offering"
        },
        "product":{
          "productSpecification":{
            "id":"UNICEEndPointSpec",
            "describing":{
              "@type":"UNICEEndPointSpec",
              "@schemaLocation":" https://github.com/MEF-GIT/MEF-LSO-Sonata-SDK/blob/master/experimental/api/ProductSpecDescription/Ordering/UNICEEndPointSpec.json"
            }
          },
          "cvlanId": [
              20
          ],
          "ingressBWProfile":[
            {
              "cosId":"Gold",
              "cir":{
                "amount":10,
                "unit":"Mbps"
              }
            }
          ]
        },
```

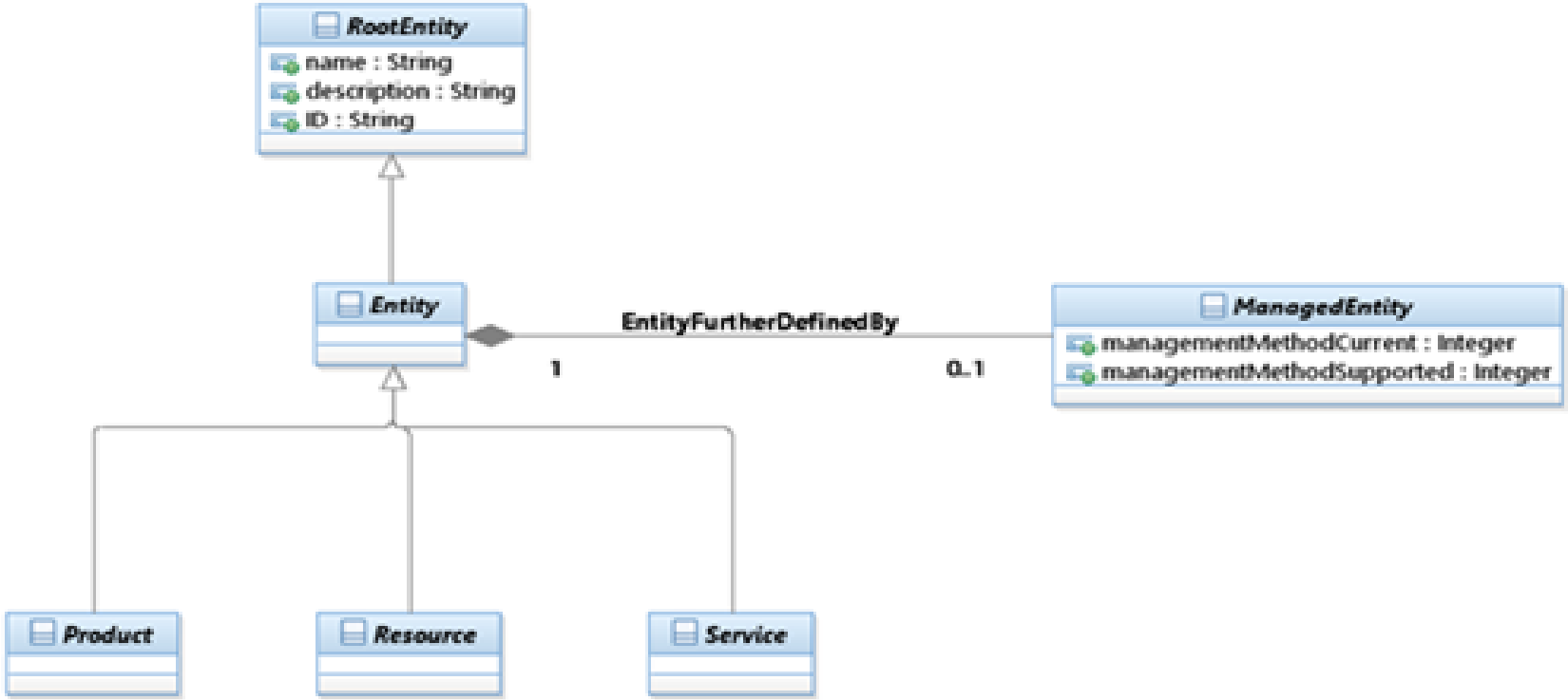

3

Additional Material

- 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

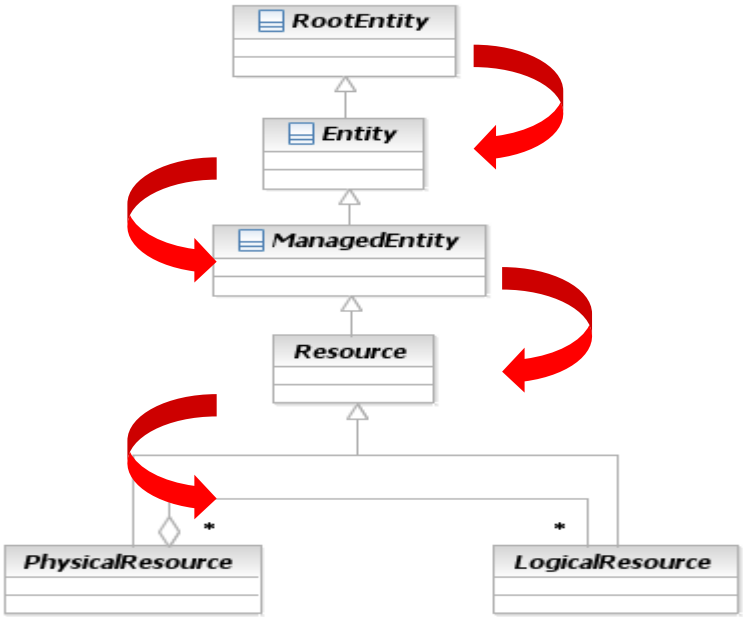
```
{
    "id": "c1234",
    "href": "http://serverlocation:port/customerManagement/customer/c1234",
    "name": "DisplayName",
    "status": "Active",
    "description": "Description string",
    ---dependant entity relationship---
    "contactMedium": [
        {
            "type": "Email",
            "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



- 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 shown 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

- 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

- 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

Customer API Data Model