FIWARE
Global
Summit

From Data
to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Marketplace services

Dr. Dennis Wendland, Technical Lead & Architect, FIWARE Foundation

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

Agenda

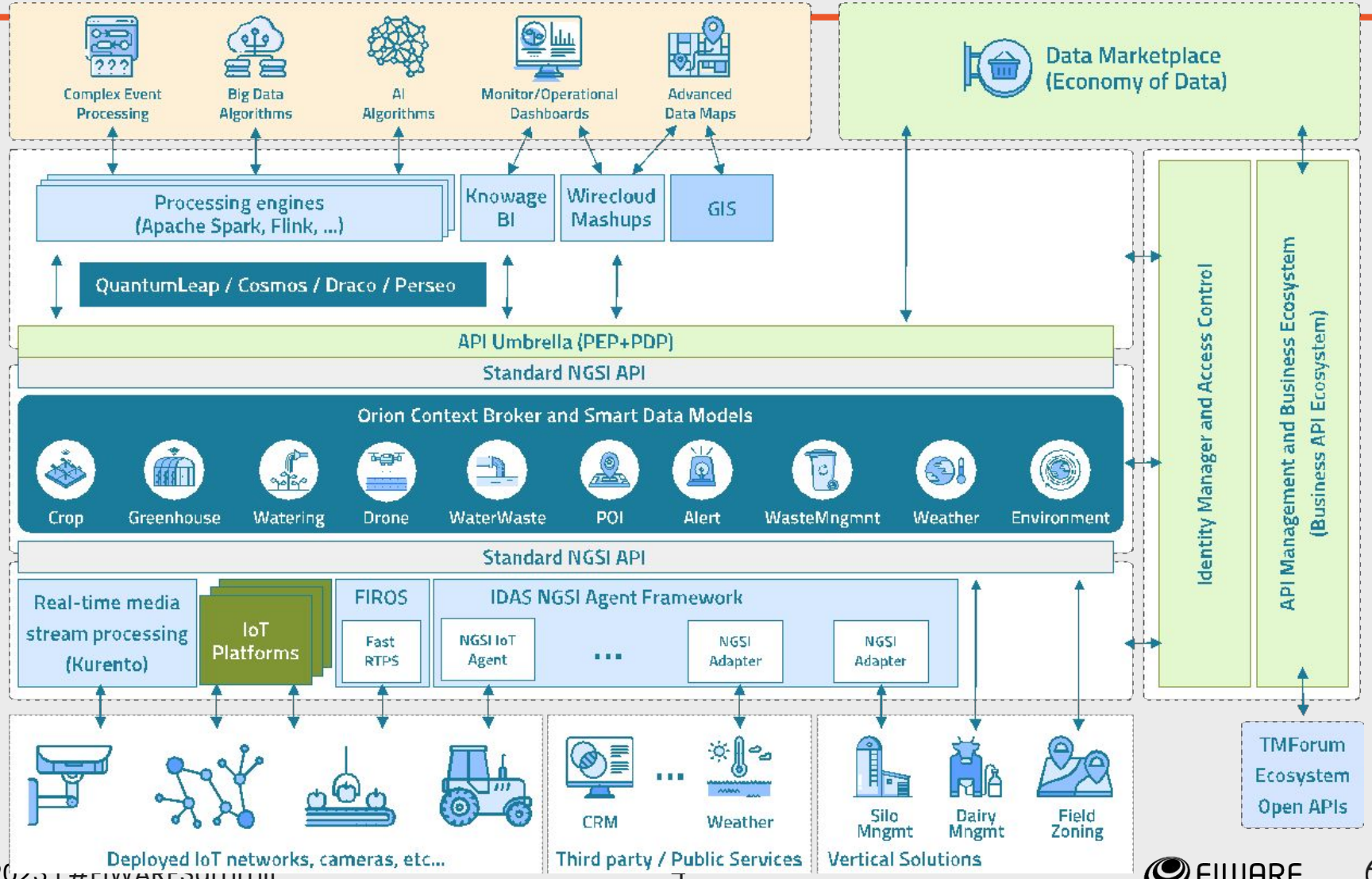
- Introduction to the BAE GE
- Main Concepts of the BAE
- Architecture of the BAE

Introduction to the BAE GE

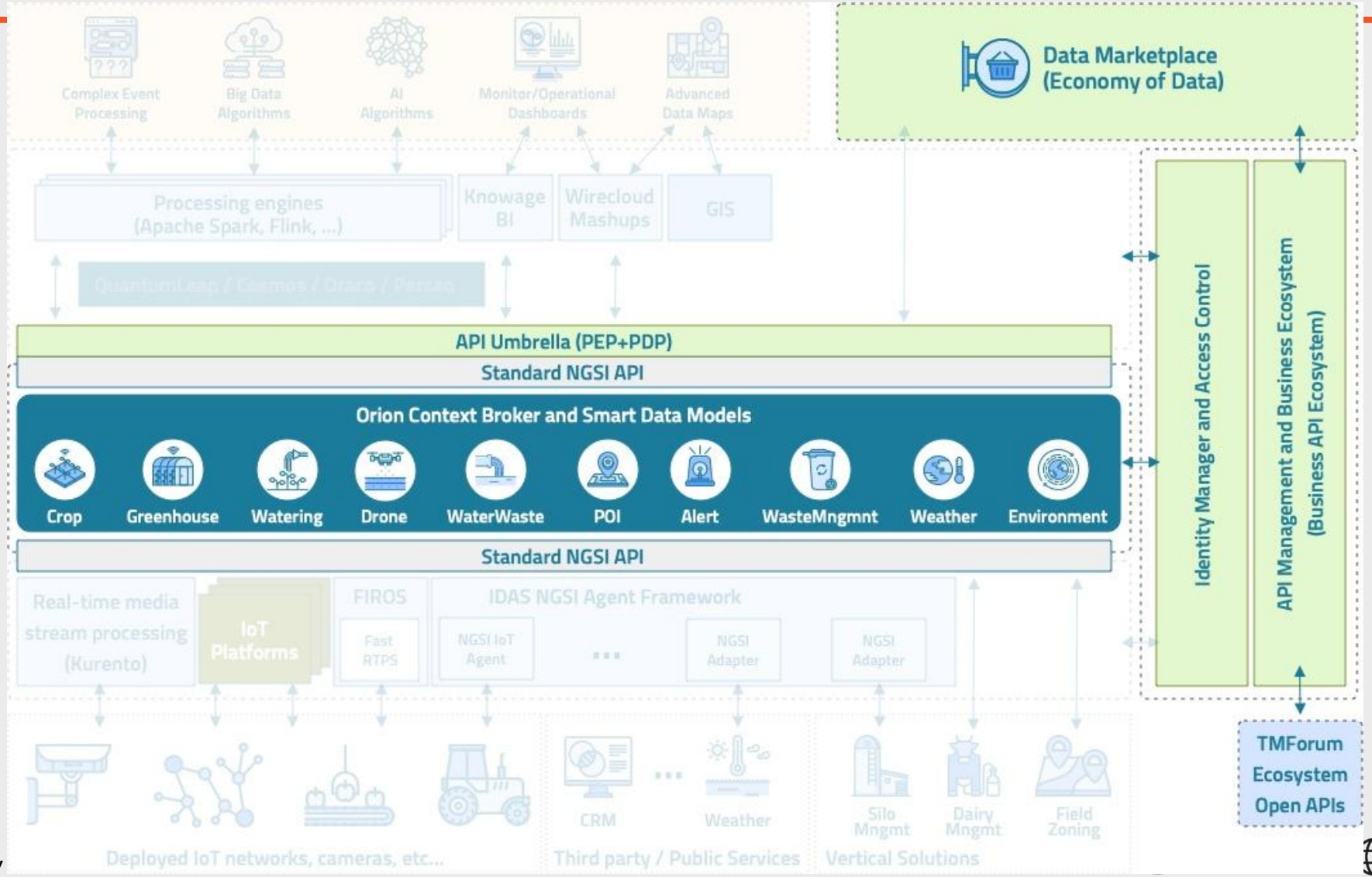
Business API Ecosystem

- Component of the **FIWARE** Platform
- Supports the creation of digital marketplaces for the monetization of digital assets
- Manages the lifecycle of products offers, from product creation to monetisation, billing, payment and revenue sharing
- Relies on **TM Forum** Business Ecosystem Open APIs

Business API Ecosystem

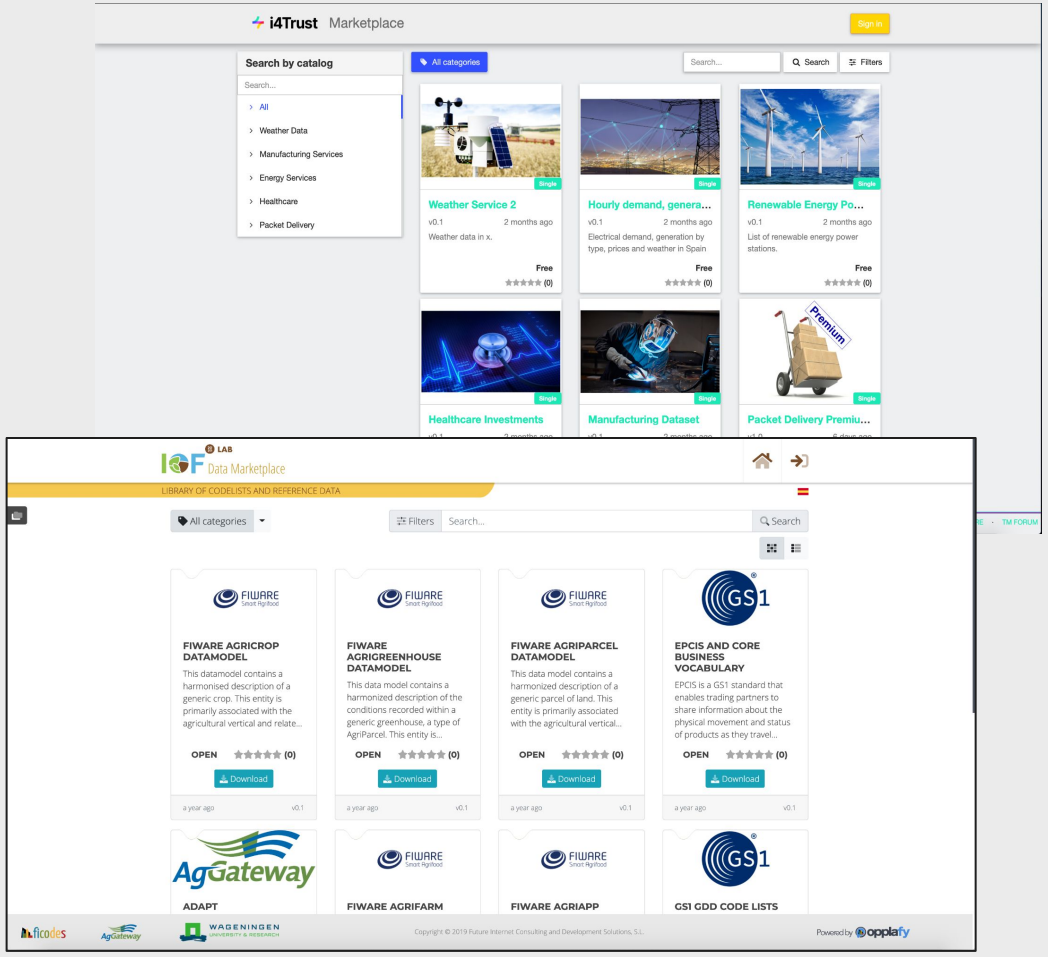


Business API Ecosystem



Business API Ecosystem

- BAE has been designed to be customized for the specific use case
  - Branding and theming support
  - Multiple identity providers supported, including FIWARE Keyrock, Keycloak and GitHub
    - New IDPs and protocols can be integrated in the software
  - Different payment gateways can be integrated



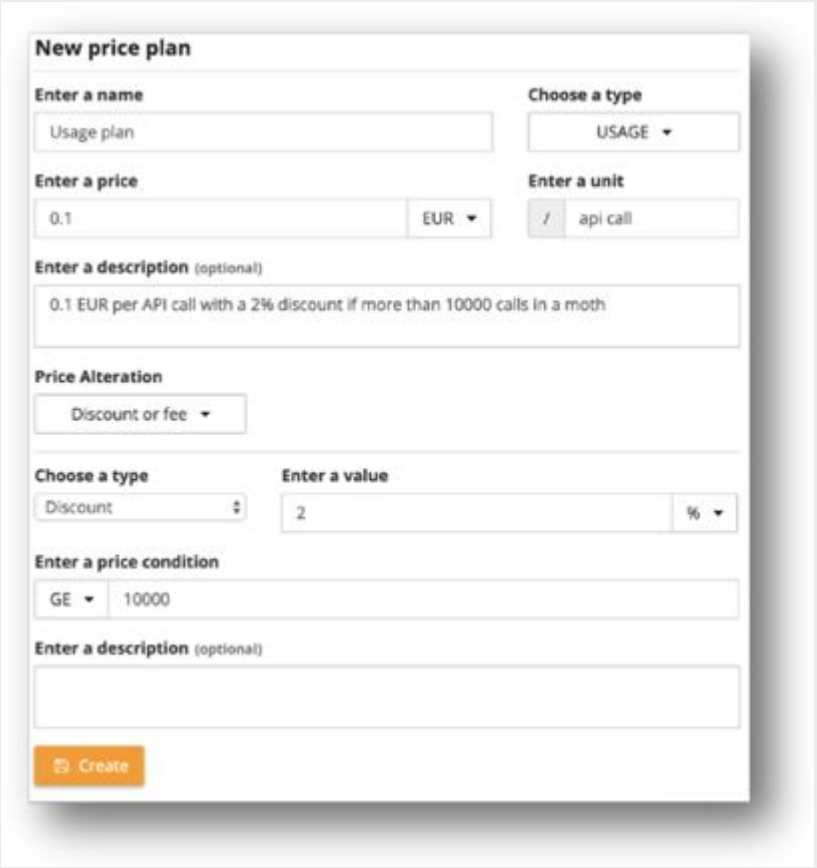
Business API Ecosystem

- Support for different kind of digital asset monetization via plugins
  - Define the asset type
  - Define the metadata to be retrieved and stored
  - Implements handlers to validate and process asset data during the product lifecycle
  - Performs service activation during acquisition
- Plugin are managed as zip files, including:
  - JSON file with plugin configuration
  - Python class with handler implementation
  - Any needed python code

```
{
    "name": "NGSI-LD Query",
    "author": "fdelavega",
    "version": "0.1",
    "module": "ngsi_query.NGSIQuery",
    "media_types": ["application/ld+json"],
    "formats": ["URL"],
    "overrides": [],
    "pull_accounting": true,
    "form_order": [
        "app_id", "role", "entities", "service"
    ],
    "form": {
        "app_id": {
            "type": "text",
            "label": "Application ID",
            "placeholder": "app_id",
            "mandatory": true
        },
        "role": {
            "type": "text",
            "label": "Acquisition Role",
            "placeholder": "customer",
            "mandatory": true
        },
```

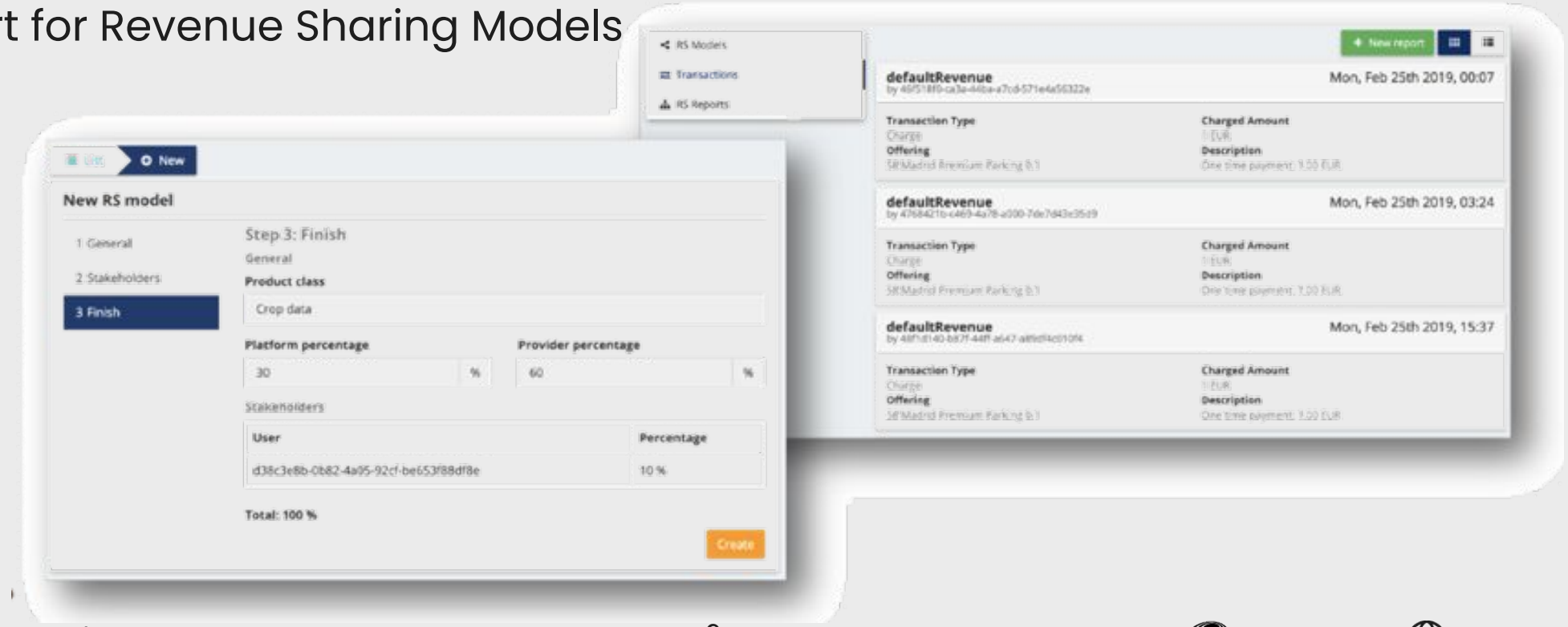
Business API Ecosystem

- Supports multiple price plans
  - **Open**: Published digital services can be accessed or downloaded without buying
  - **Free**: Require offer to be acquired and terms of service accepted
  - **One time payments**: Single payment for the service
  - **Recurring payments**: Weekly, monthly, etc
  - **Usage payments:** Actual payment calculated based on usage
    - Require accounting information that can be submitted via TMForum Usage API or retrieved by the digital asset plugin
  - **Advanced models**
    - Fees, discounts, dynamic pricing



Business API Ecosystem

- Support for usage terms and conditions
- Support for specifying SLAs
- Support for Revenue Sharing Models



Main Concepts of the BAE

TM Forum Open APIs

- Enable seamless connectivity, interoperability and portability across complex ecosystem services
- TM Forum Open APIs are based on REST. They are technology agnostic and can be used in any digital service scenario
- TM Forum Open APIs aim at:
  - Enabling services to be managed end-to-end throughout their lifecycle
  - Working in an environment where multiple partners are involved in service delivery
- There are around 60 APIs covering different aspects of digital service management

Catalog Elements

- Catalog Models taken from TMForum Catalog Management API
- Product Offering classification
  - **Product Category**: Created by a system administrator, can be used by providers to categorize offers. Product Categories can be nested to create a tree
  - **Product Catalog**: Created by providers to group their own offers. All offers of the system need to be part of a Catalog
- Product Offering creation
  - **Asset**: Real digital product registered in the system (the data, file, service, etc).
  - **Product Specification**: Definition of product-related information linked to the asset in the system. It incorporates asset link, product characteristics, and attachments.
  - **Product Offering**: Definition of the business-related information linked to the product. It includes link to the catalog, product specification, license, SLAs, pricing models and revenue sharing models.

Usage Elements

- Usage Models taken from TMForum Usage Management API
- Models:
  - **Usage Specification**: Defines the kind of usage information expected for a given product offering, including the units and metrics.
  - **Usage Document**: Includes an actual usage made of a product specification asset during a period of time. It includes also its associated price and whether it has been charged or not.

Party and Billing Elements

- Party Models are taken from TMForum Party Management API
- Party models: Include information about the different participants of the system. All the models managed in the BAE point to one or multiple parties
  - **Individual**: Provides information about the different users registered in the system, including personal and contact information
  - **Organization**: Provides information about the different organizations registered in the system, including trading, legal and contact information. Multiple individuals can be part of an Organization

- **Billing Account**: Provides the billing information of a Party including billing addresses. Billing account mode is taken from TMForum Billing Management API

Ordering Elements

- **Product Order**: Includes a request from a customer to manage one or multiple agreements. It may be used to acquire Product Offerings but also to modify or cancel existing product offering acquisitions. Product Order model is taken from Product Ordering management API
- **Product**: Is the result of acquiring a Product Offering, and includes all the relevant information of such an acquisition, including selected characteristics, selected pricing model and charges. Product model is taken from Product Inventory API

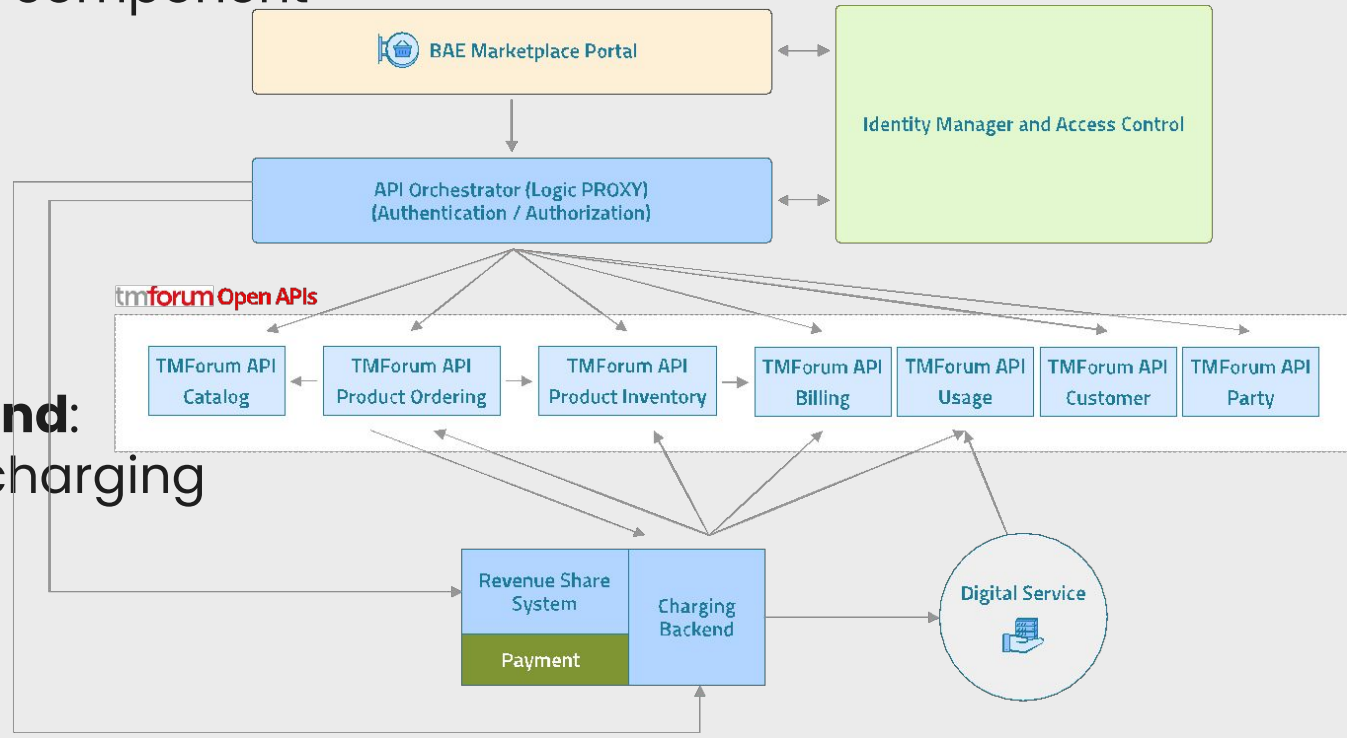
Revenue Sharing

- Models:
  - **Revenue Sharing Model**: It establishes how the incomes generated by a set of offers has to be distributed among the different stakeholders involved. It includes, the platform percentage, the provider percentage and the stakeholders percentage.
  - **Transaction**: A payment made by a customers linked to a particular revenue sharing model.
  - **Revenue Sharing Report**: Result of applying the a revenue sharing model to its linked transactions. It include the amount to be paid to the different stakeholders.

Architecture of the BAE

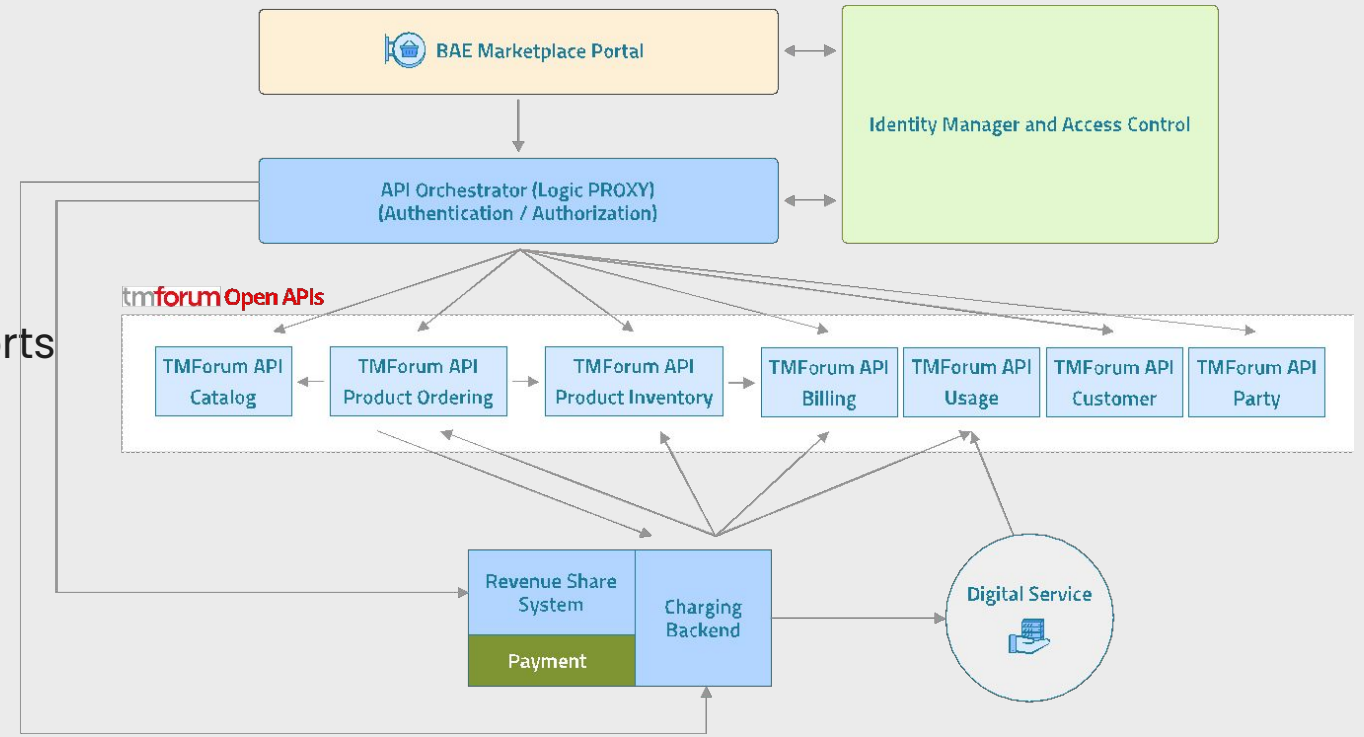
Architecture of the BAE

- The Business API Ecosystem software is made up of a couple or orchestrated microservices
  - **Business Ecosystem Logic Proxy**: This component provides the entry point to the system
    - Gives API Access
    - Serves the GUI
    - Integrates with the IDP
    - Validates the user permissions
    - Orchestrates the APIs
  - **Business Ecosystem Charging backend**: This component provides billing and charging processing
    - Manages digital assets
    - Calculates and executes payments
    - Activates and deactivates services



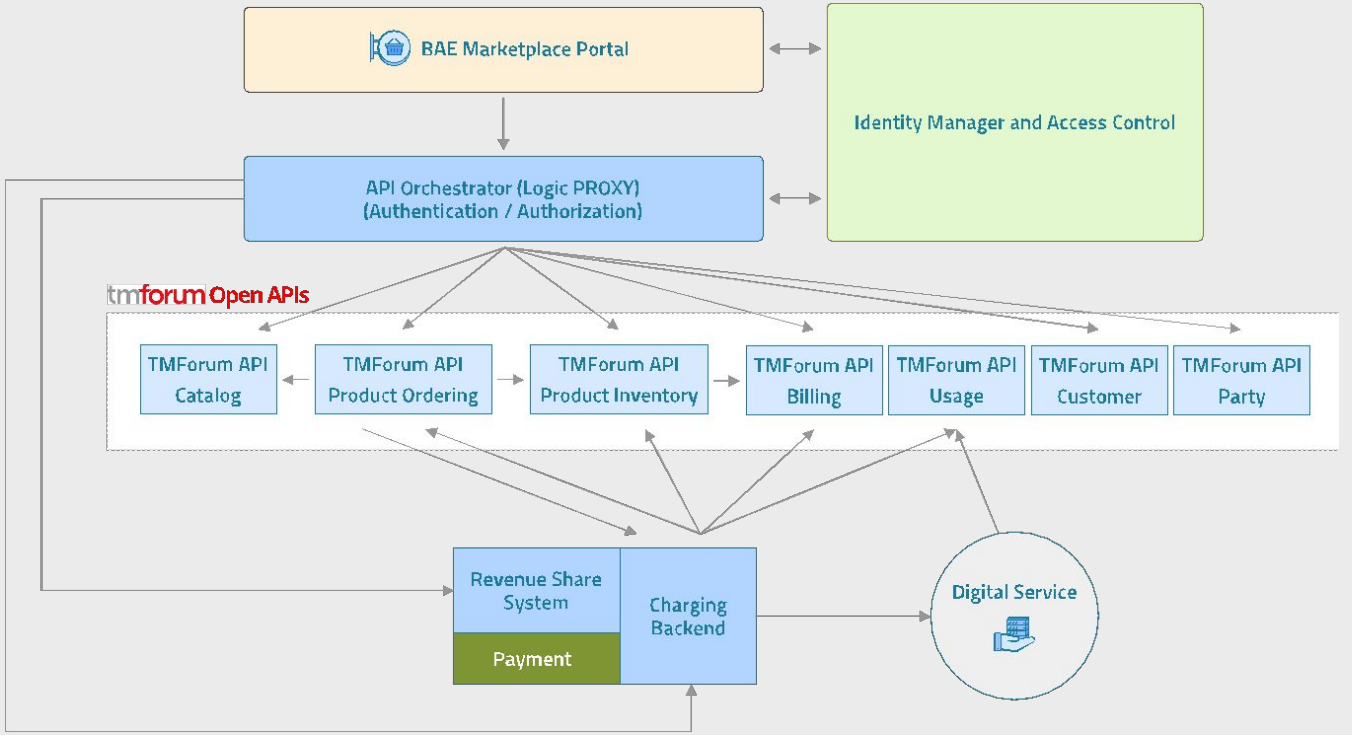
Architecture of the BAE

- The Business API Ecosystem software is made up of a couple or orchestrated microservices
  - **Revenue Sharing System:** This component manages the revenue sharing
    - Stores the revenue sharing models
    - Stores the different transactions
    - Generates the revenue sharing reports

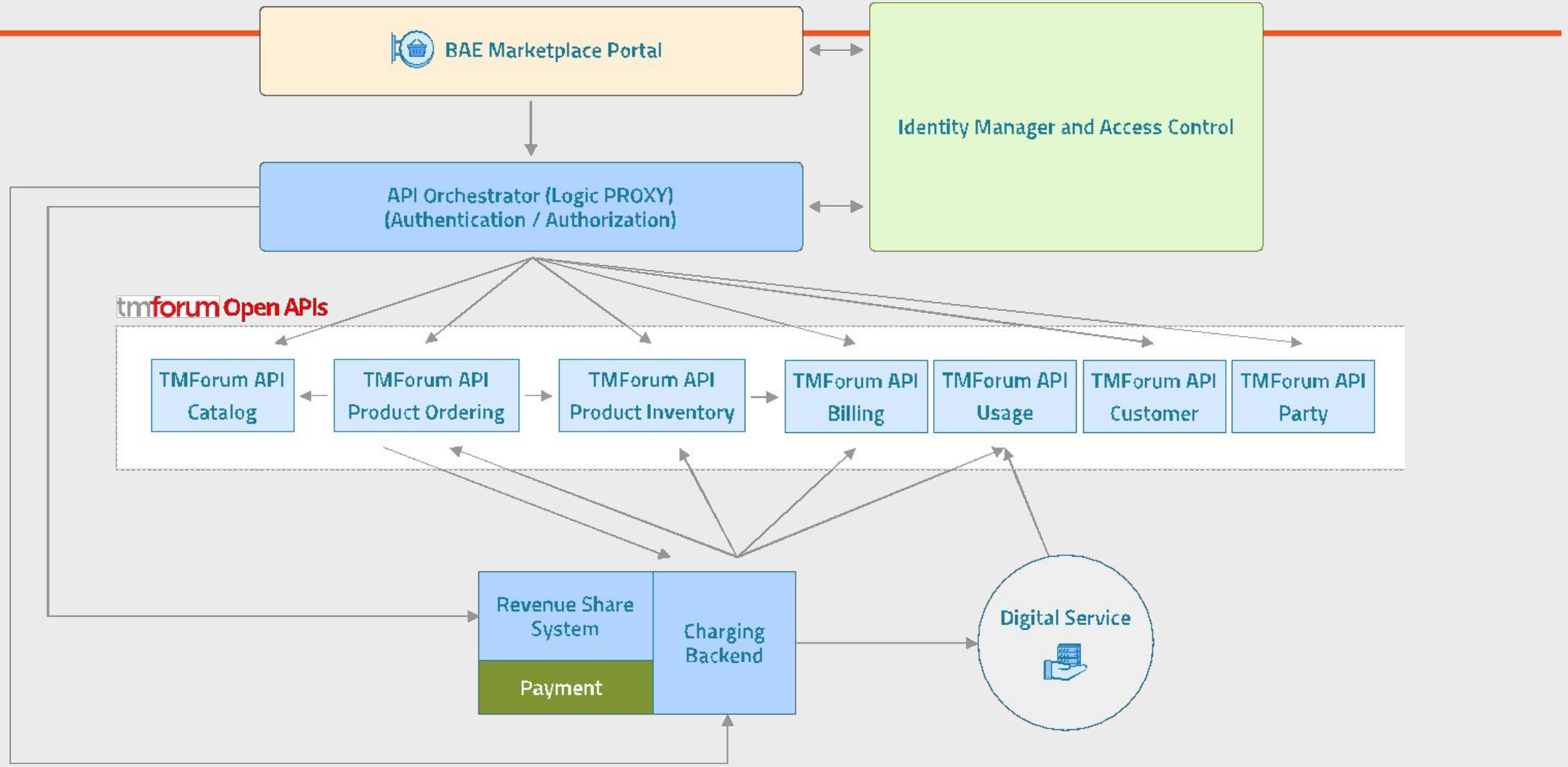


Architecture of the BAE

- The Business API Ecosystem implements a few set but core TM Forum Open APIs for digital service monetisation features
- Product Lifecycle
  - Product Catalog Management API
  - Product Ordering Management API
  - Product Inventory Management API
- User and customer management
  - Party Management API
  - Customer Management API
  - Billing Management API
- Usage Management API

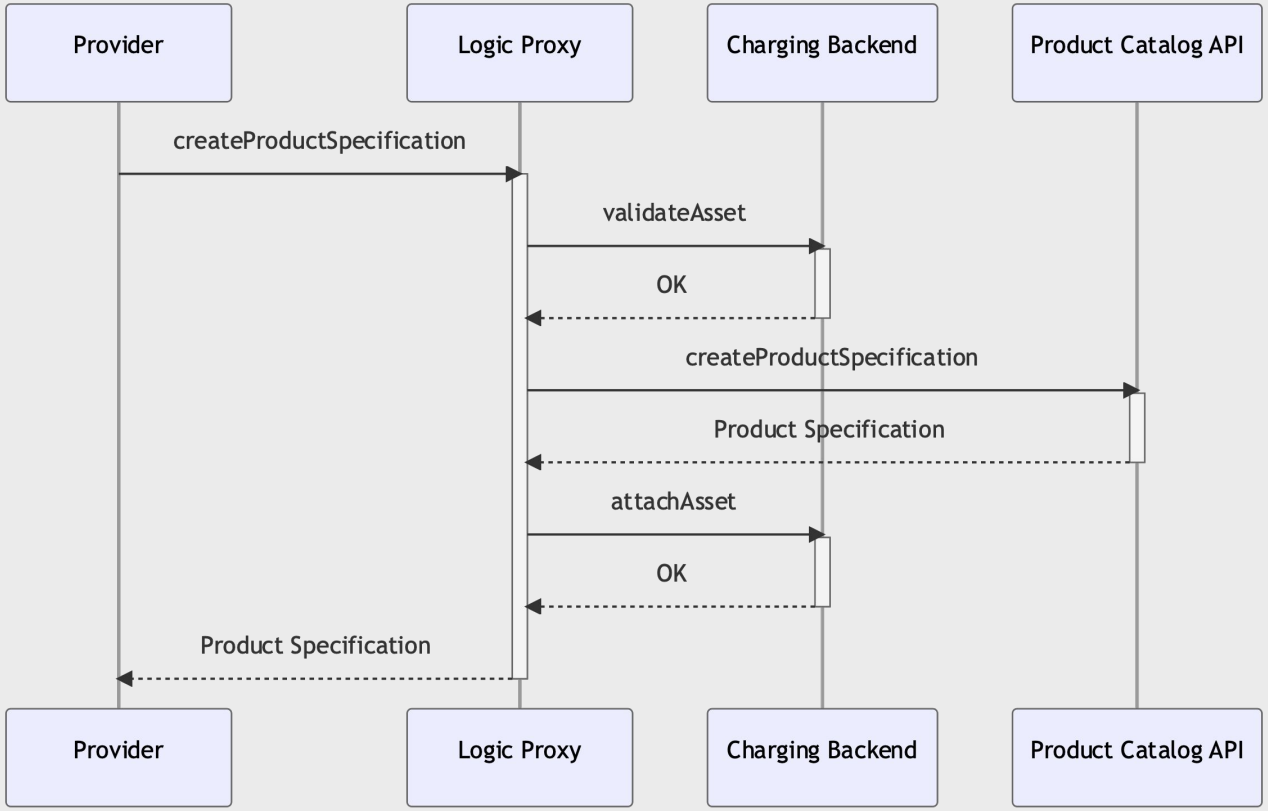


Architecture of the BAE



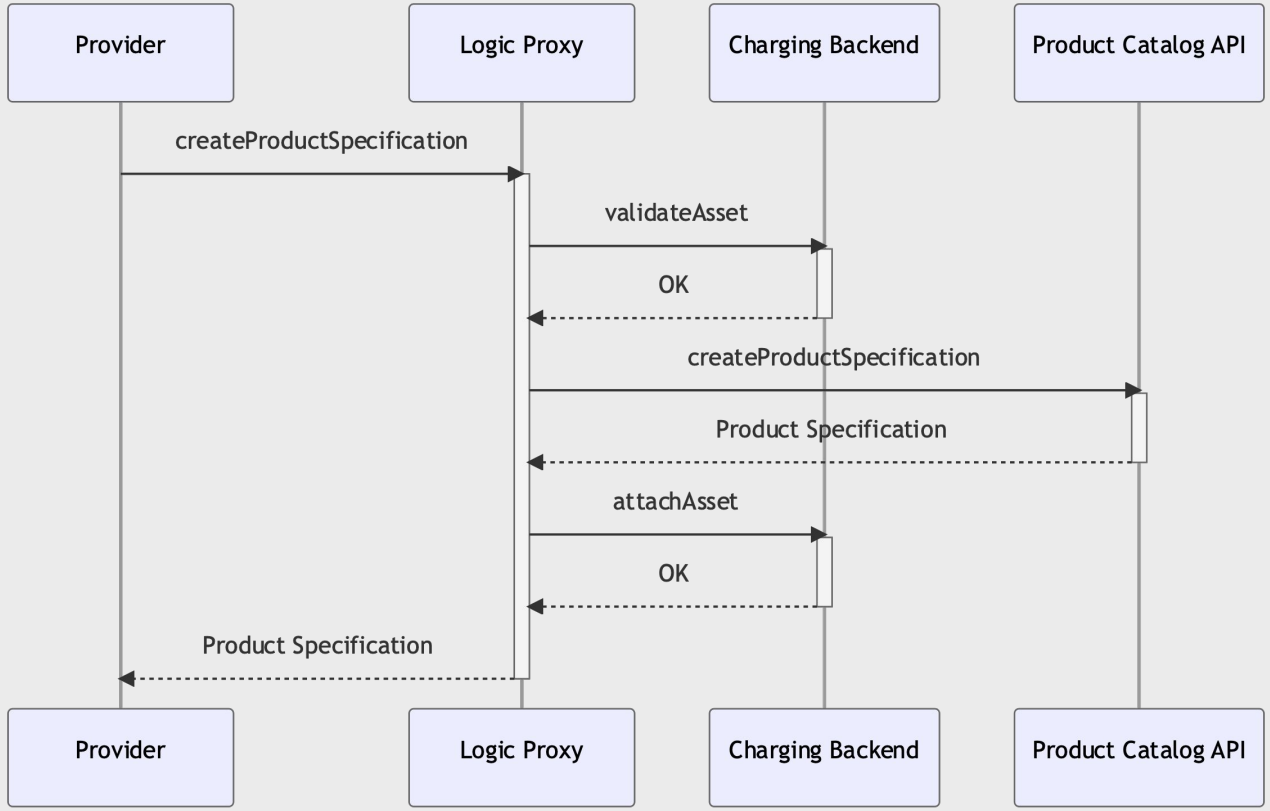
Registering a Product Specification

1. The provider creates a Product Specification (Using the GUI or the API)
2. The Proxy validates user permissions for creating a Product Specification
3. The proxy takes the asset related characteristics and send them to the charging backend for validation.
4. The charging backend validates that the asset is registered and owned by the provider and triggers “on product validation” plugin handlers
5. The Proxy creates the Product Specification in the Product Catalog API and retrieves the new created Product Specification

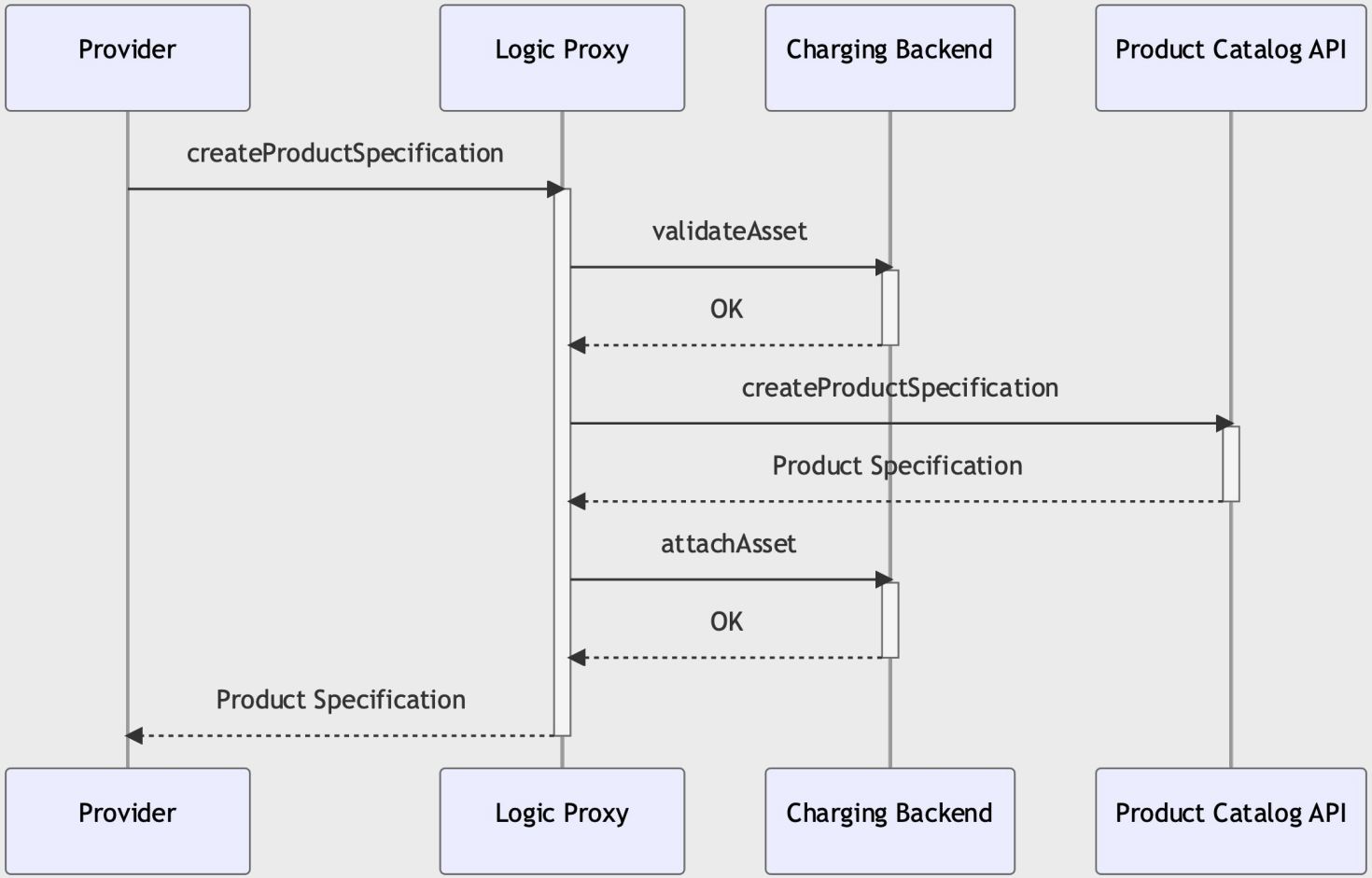


Registering a Product Specification

6. The Proxy sends the information of the new Product Specification to the Charging Backend
7. The Charging Backend attaches the ID of the Product Specification to the asset for future use and triggers “on product attached” plugin handler
8. The provider receives the new Product Specification

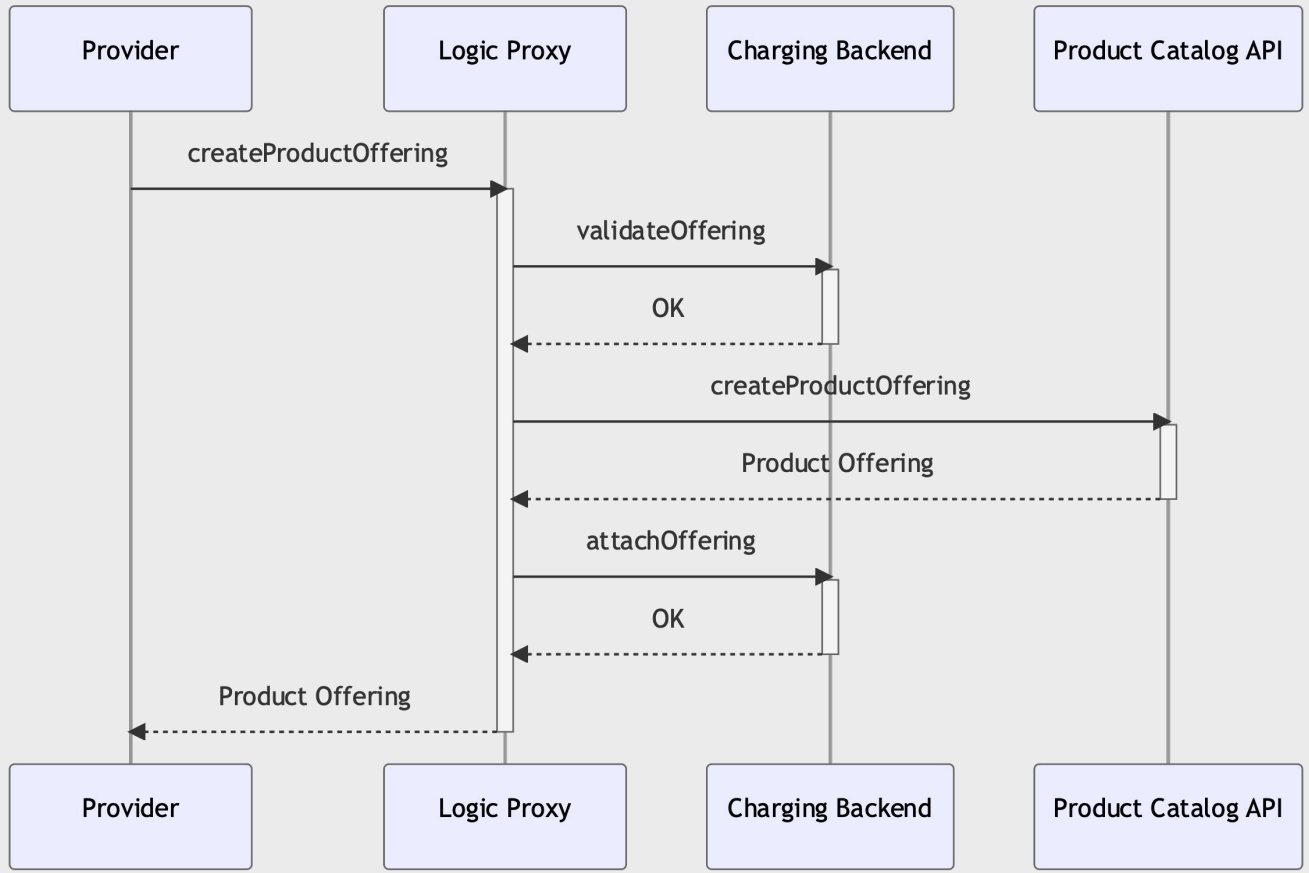


Registering a Product Specification



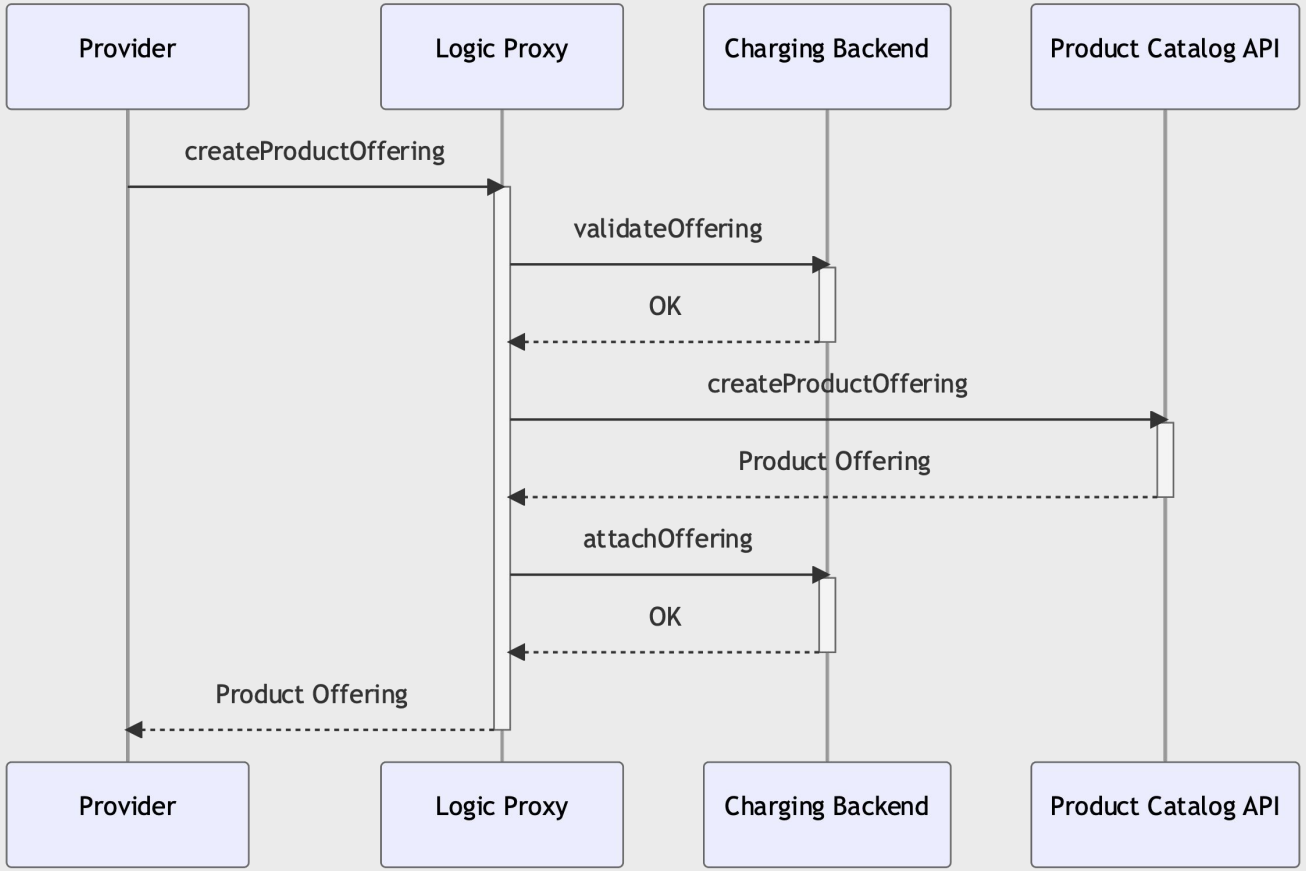
Create a Product Offering

1. The provider creates a Product Offering (Using the GUI or the API)
2. The Logic Proxy Validates the permissions of the provider to create the Product Offering and to include the linked resources (Product Specification, Categories, etc)
3. The Logic Proxy sends the Product Offering to the Charging Backend for validation
4. The Charging Backend validates the pricing model and triggers the “on product offering validation” plugin handlers

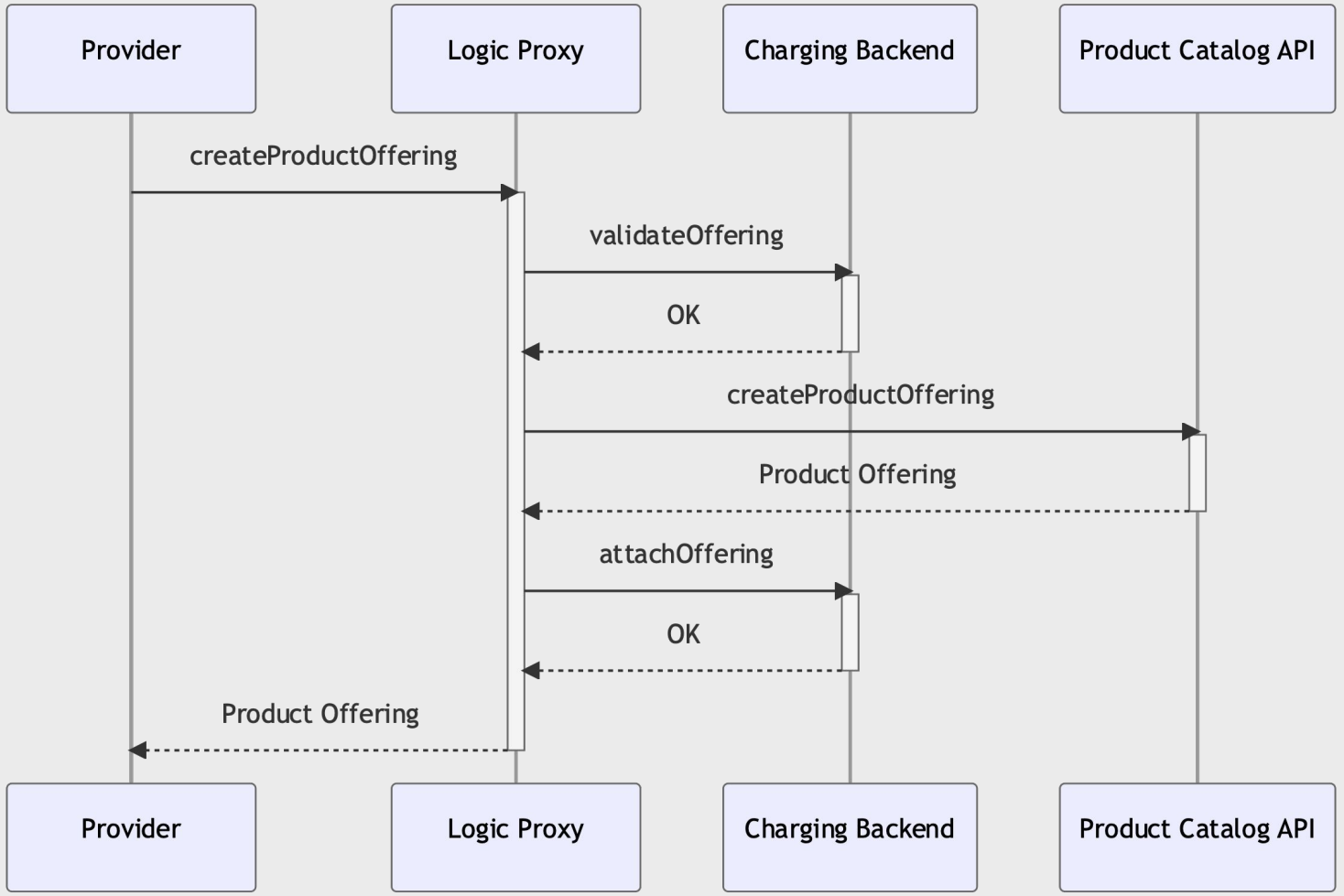


Create a Product Offering

5. The Logic Proxy Creates the Product Offering in the Product Catalog API that returns the new Created Product Offering
6. The Logic Proxy sends the information of the new Product Offering to the Charging Backend that attaches the ID of the Product Offering to the asset for future use
7. The proxy returns the new Product Specification to the provider

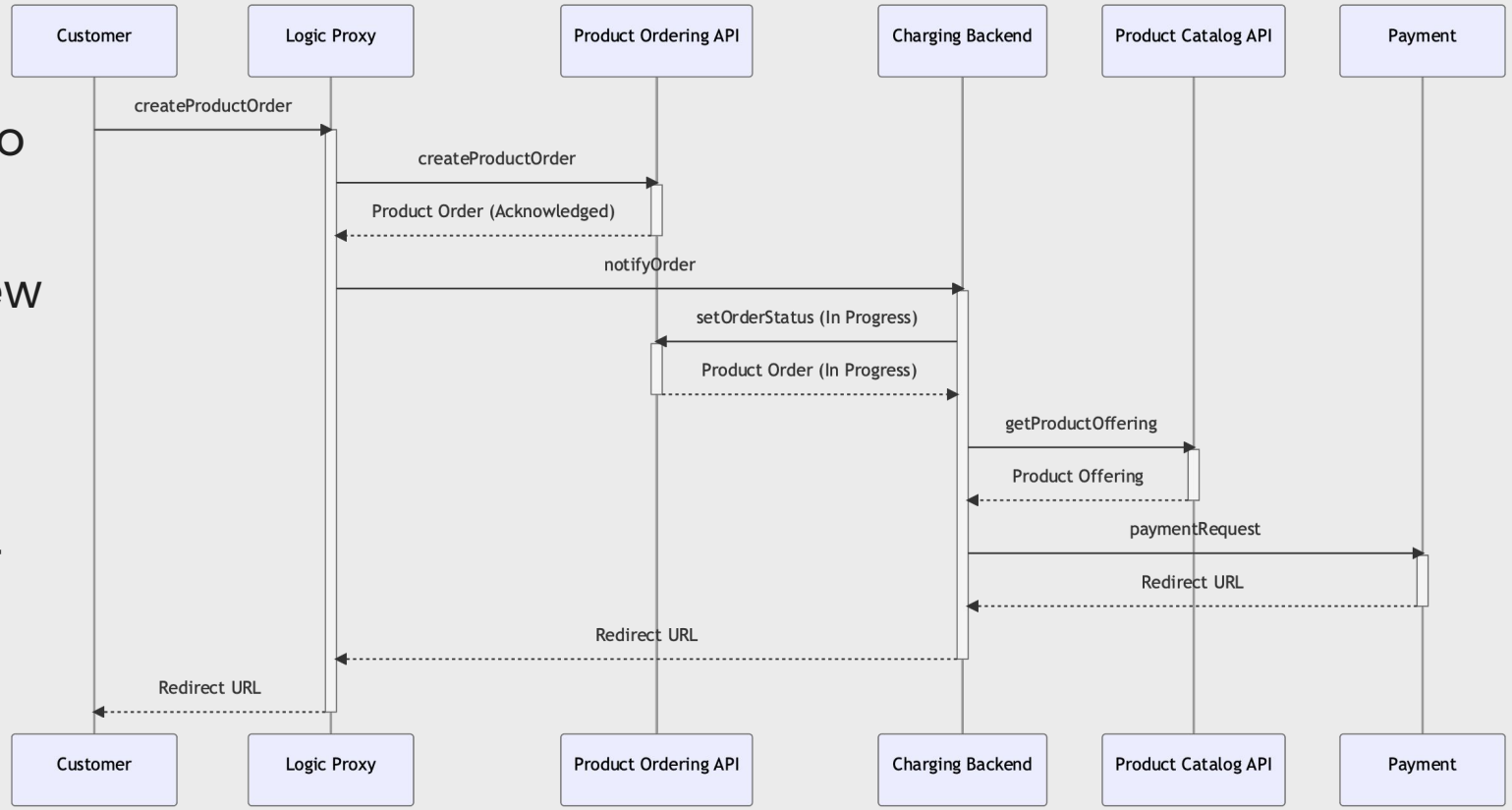


Creating a Product Offering



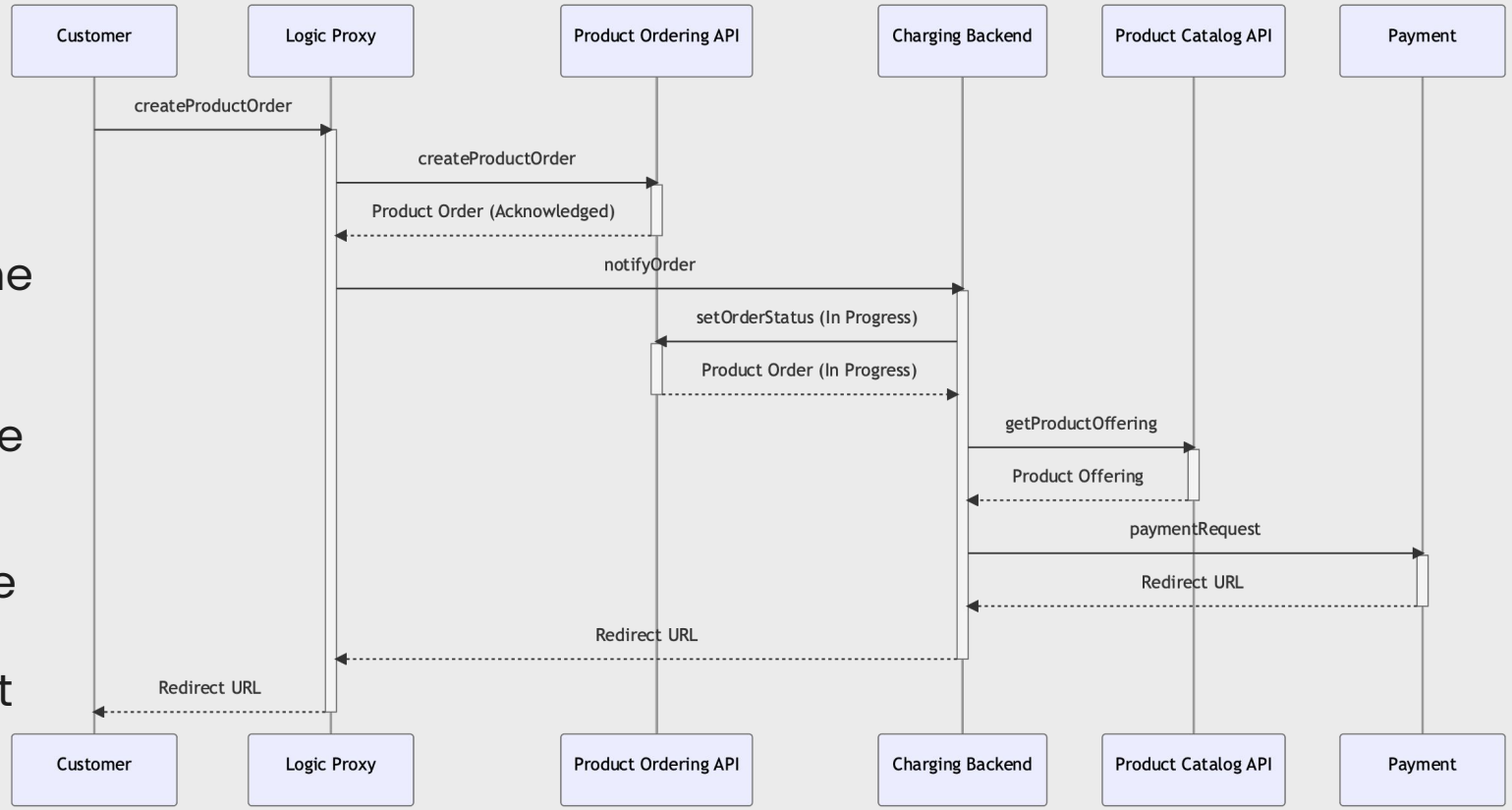
Acquiring a Product Offering – Create Product Order

1. The customer creates a new Product Order
2. The Logic Proxy validates the permissions of the customer to create a new Product Order
3. The Logic Proxy creates the new Product Order in the Product Ordering API
4. The Product Ordering API returns the new Product Order in Acknowledged status

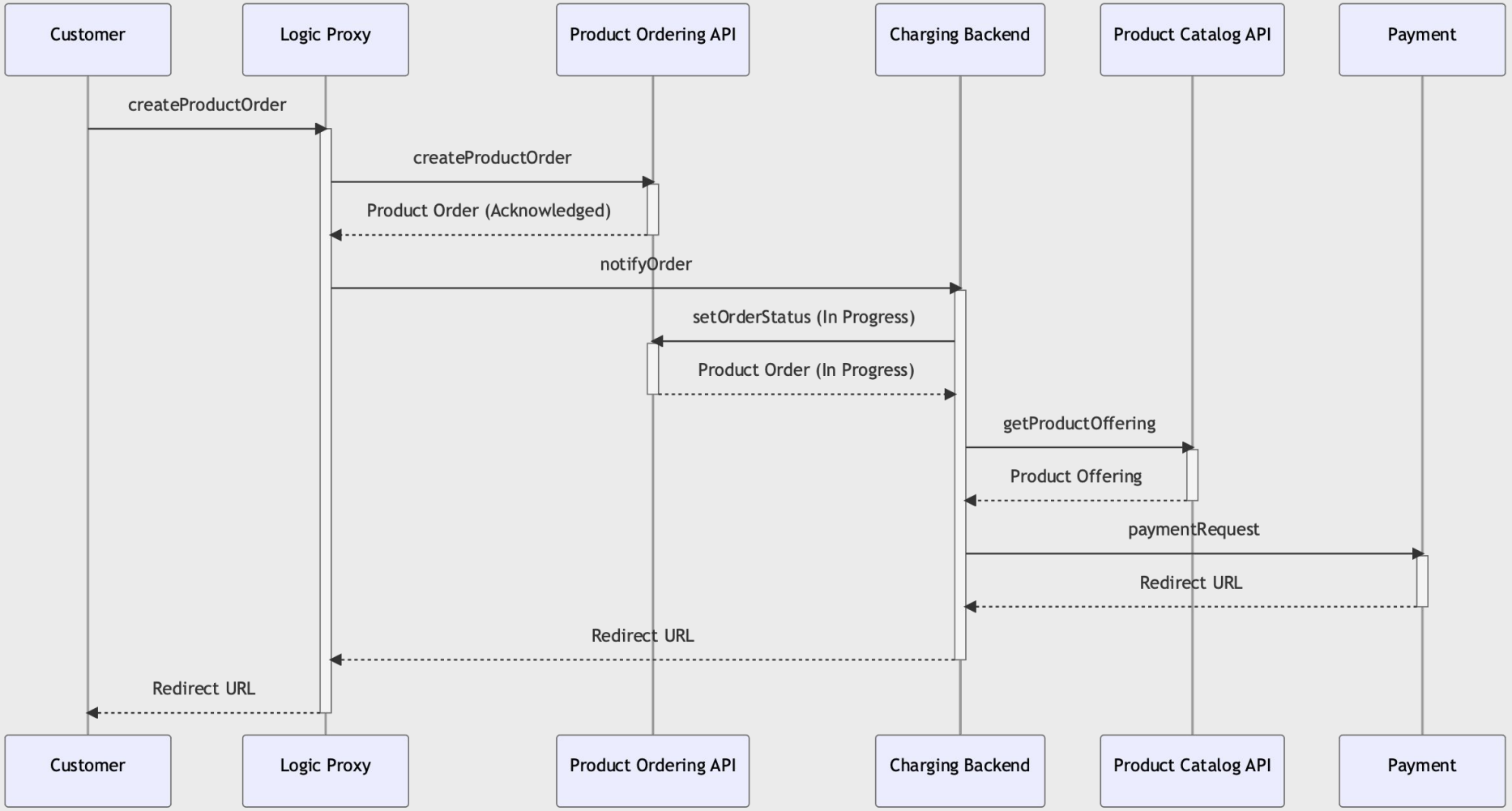


Acquiring a Product Offering – Create Product Order

5. The Logic Proxy sends the Product Order to the Charging Backend
6. The Charging Backend changes the status of the Product Order to “In Progress”
7. The Charging Backend retrieves the Product Offering being acquired from the Product Catalog API and uses the pricing model to calculate the amount to be paid
8. The Charging Backend initiates the payment process calling the payment gateway (i.e PayPal) that returns a redirect URL for the customer

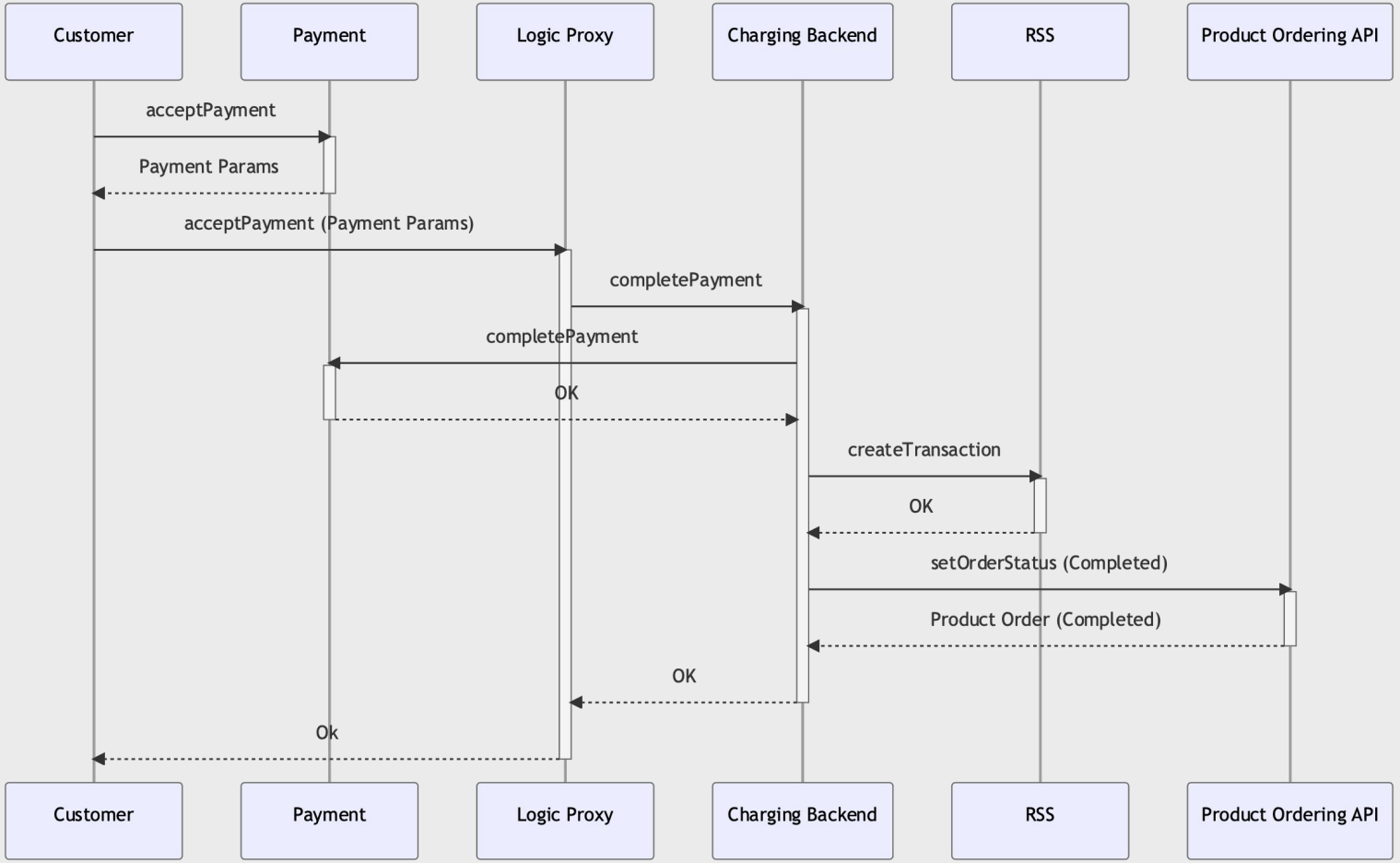


Acquiring a Product Offering – Create Product Order



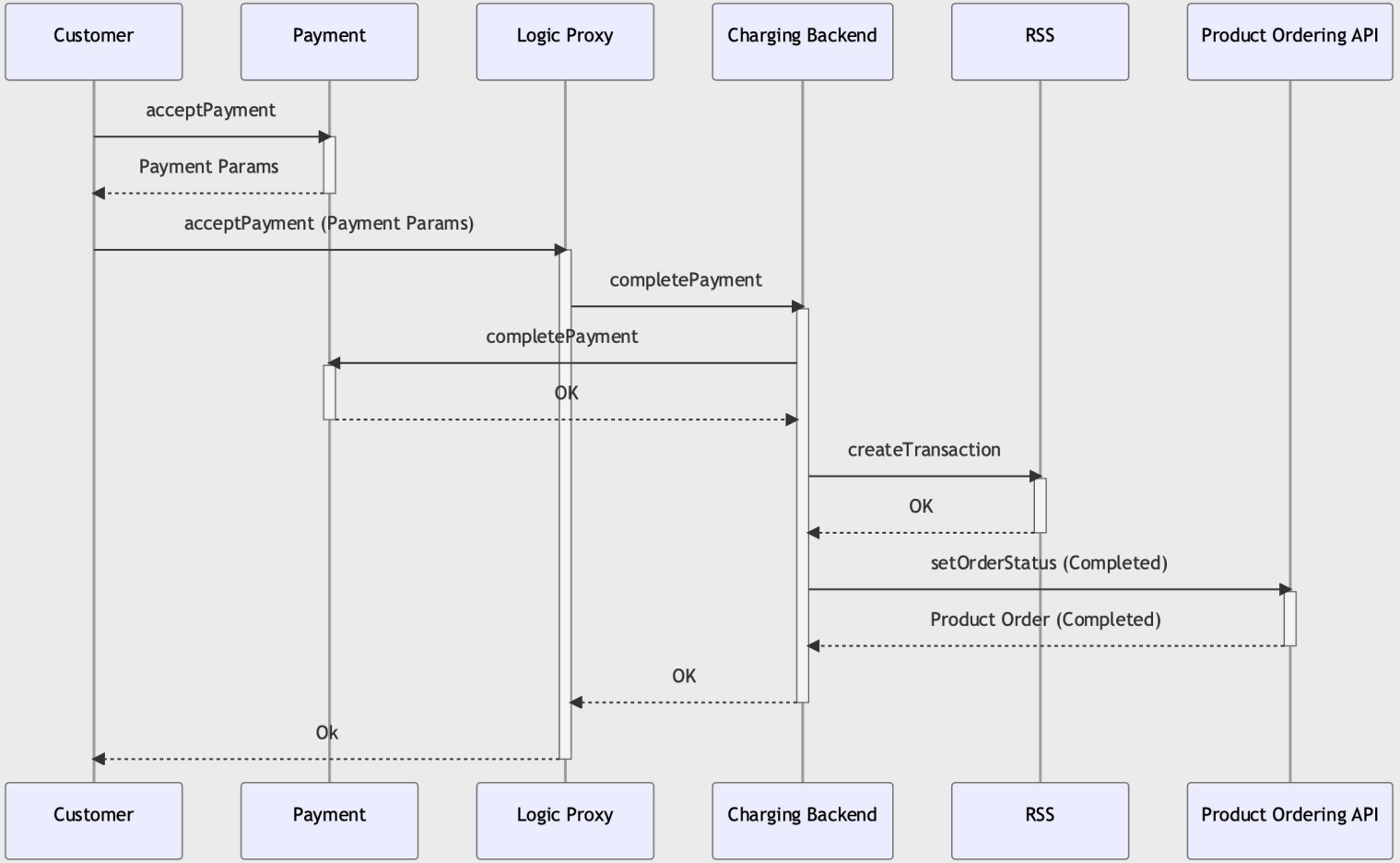
Acquiring a Product Offering – Accept Payment

1. After receiving the redirect URL for the customer, the Customer is redirected to the Payment Gateway site to accept the payment (the GUI does it automatically)
2. The customer sends the payment params to the Logic Proxy that sends the information back to the Charging Backend
3. The Charging Backend uses the payment information to complete the payment in the Payment Gateway

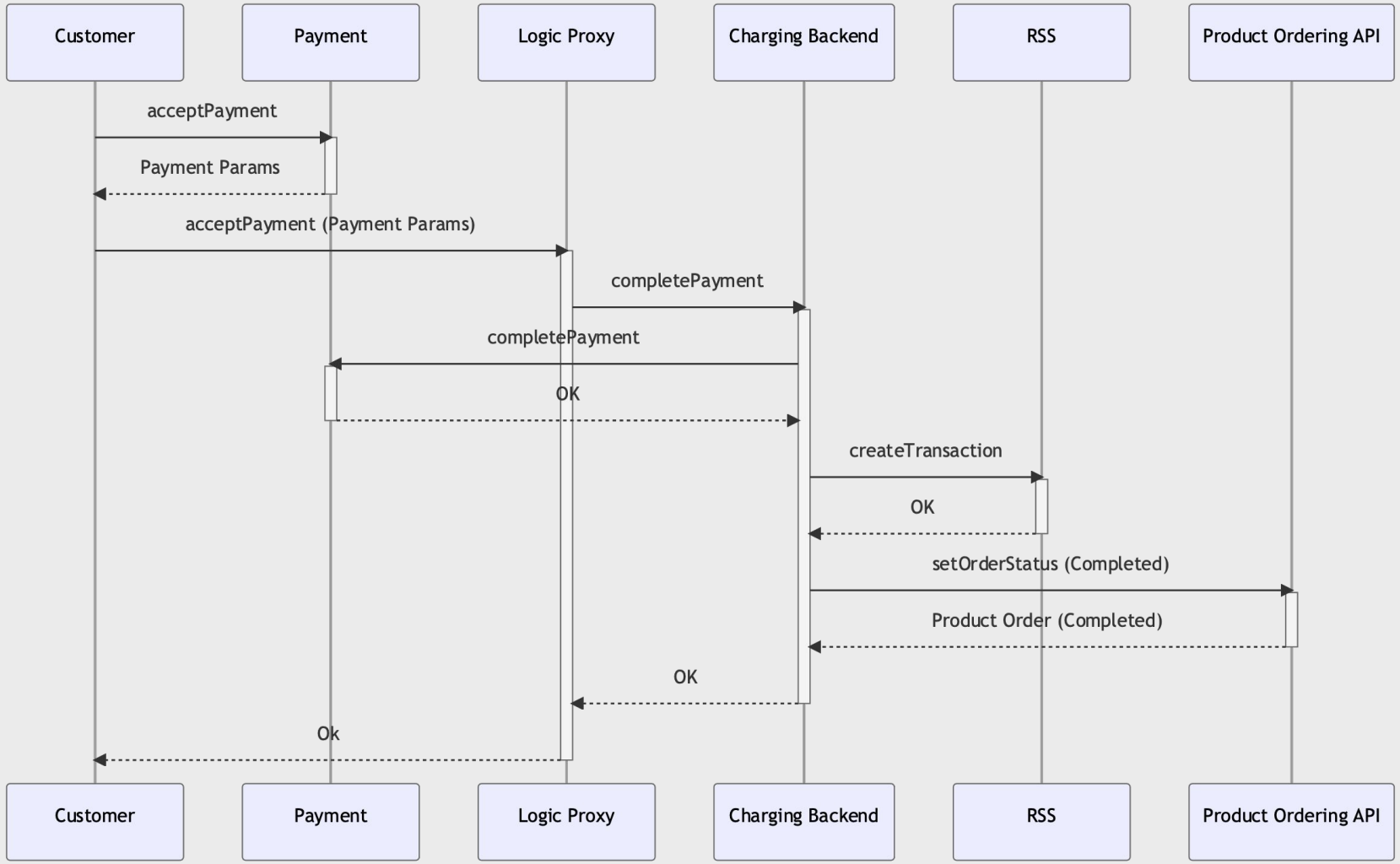


Acquiring a Product Offering – Accept Payment

4. The charging backend creates a transaction in the Revenue Sharing System for the amount charged to the customer
5. The charging backend changes the status of the Product Order to “Completed” in the Product Ordering API

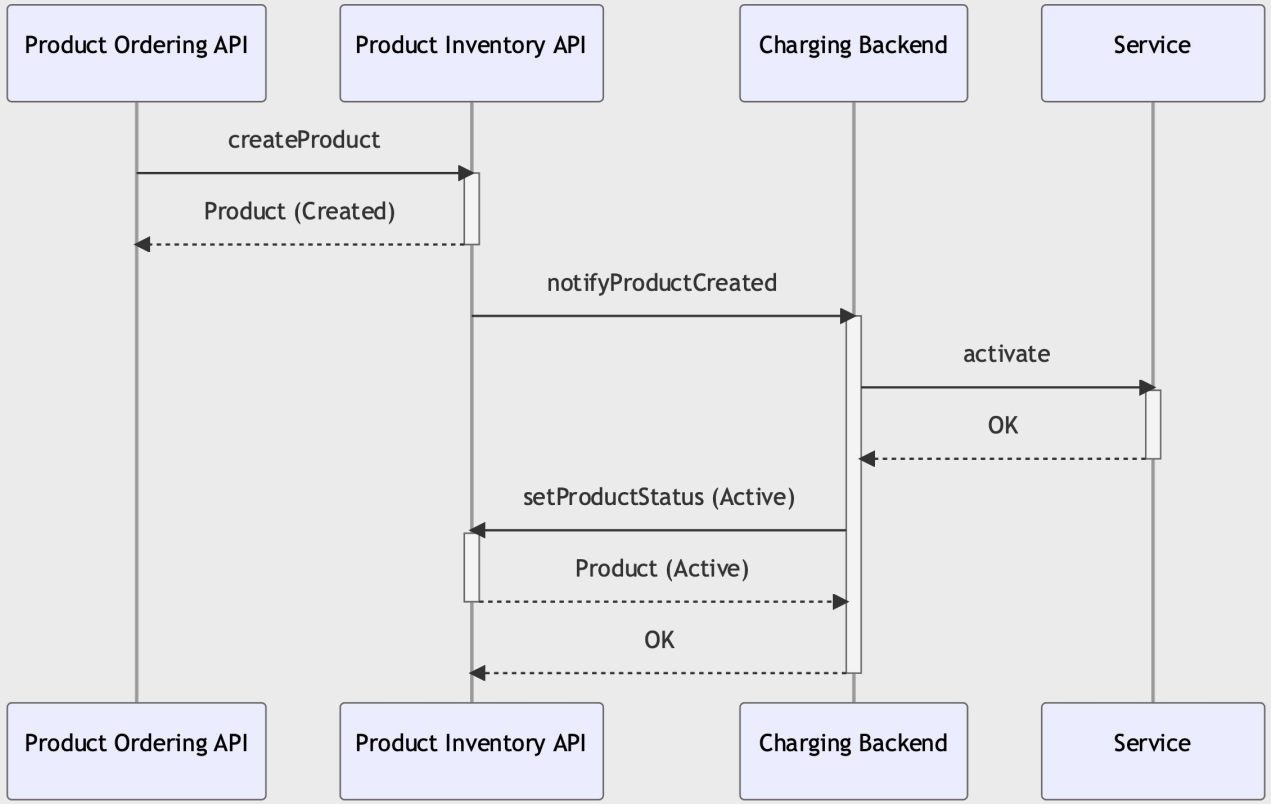


Acquiring a Product Offering – Accept Payment

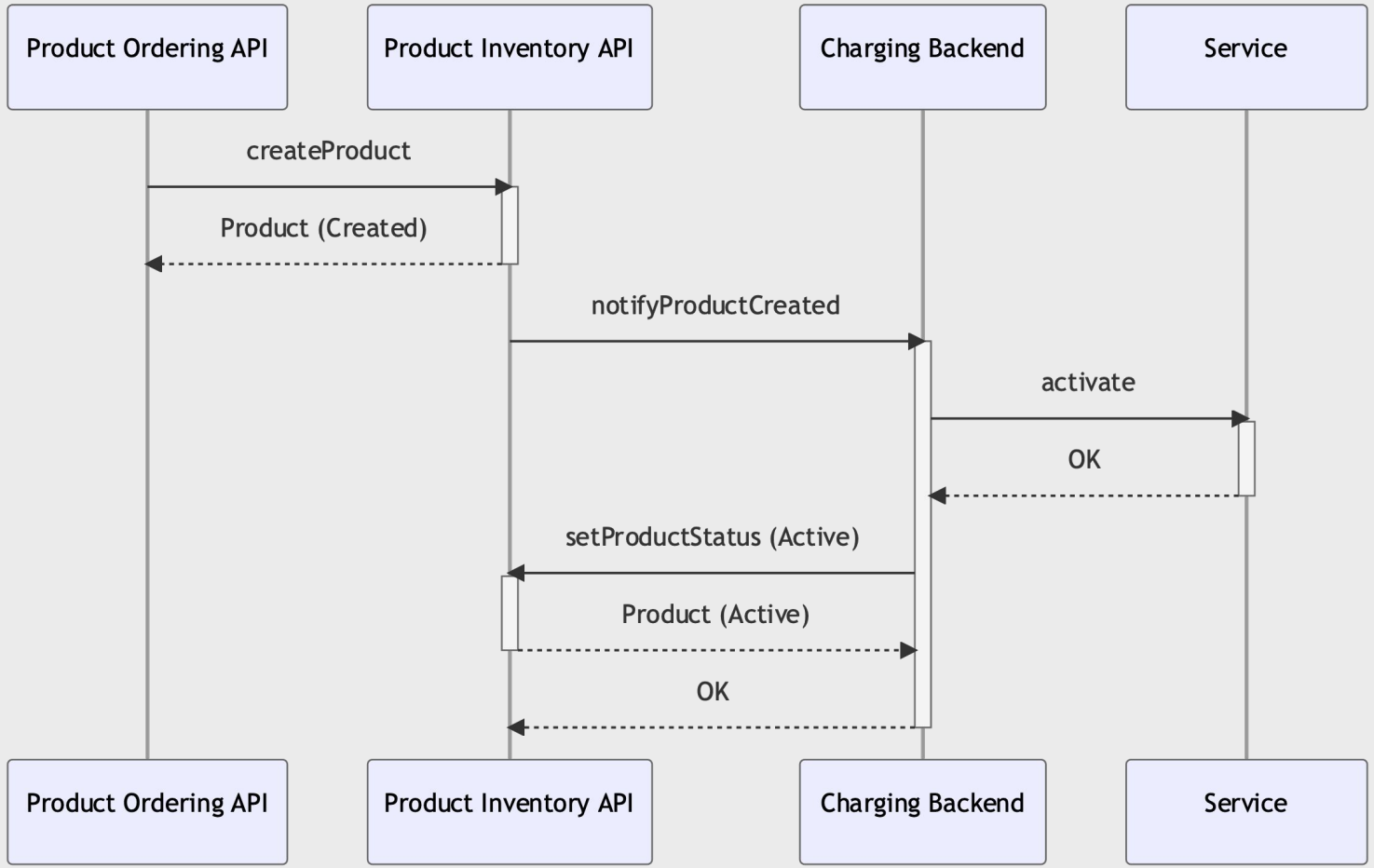


Acquiring a Product Offering – Service Activation

1. When a Product Order status is “Completed”, the Product Ordering API automatically created a Product in the Product Inventory for each of the items
2. When the new Product is Created, the Product Inventory API notifies the Charging Backend with the new Product in “Created” state.
3. The Charging Backend calls the on service acquired plugin handler for service activation
4. The Charging Backend changes the status of the Product to “Active” in the Product Inventory API

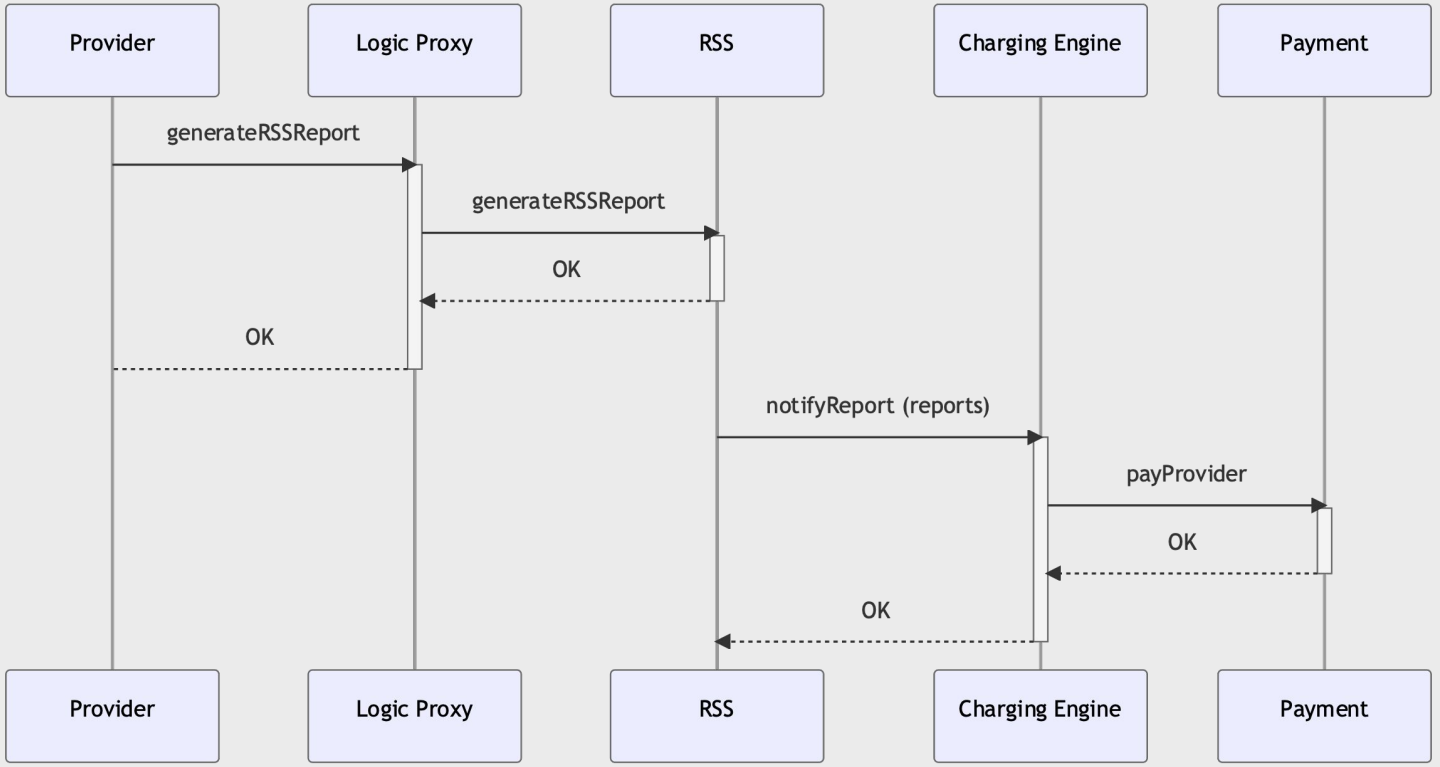


Acquiring a Product Offering – Service Activation



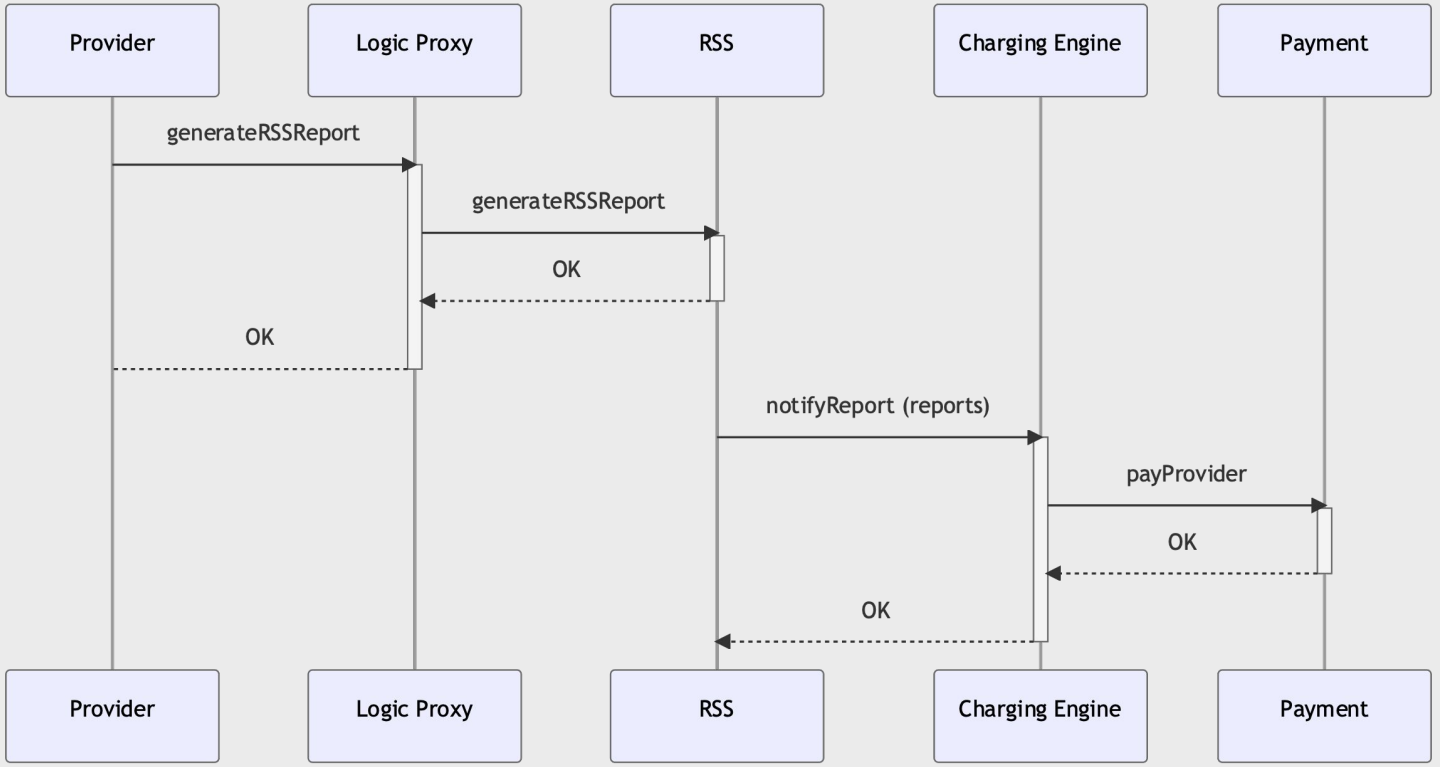
Generate RSS Report

1. The provider ask for generating a Revenue Sharing Report for a specific Product Class
2. The Logic Proxy validates the permissions of the Provider and sends the request to the Revenue Sharing System
3. The Revenue Sharing System starts the report generation in background and sends an Accepted response to the Logic Proxy



Generate RSS Report

1. When the report are generated the Revenue Sharing System notifies the Charging Backend providing the report information
2. The Charging Backend uses the Report information to pay the different stakeholders involved using the Payment Gateway



TM Forum – Target Architecture

- Update of current APIs
  - Usage of JSON-LD provided with new versions of the API definitions
  - New implementation using the **FIWARE Context Broker** for storing of relevant information, enabling the usage of FIWARE components for managing the business data
- New APIs are being integrated
  - Support for Service and Resource lifecycles integrating Catalog, Ordering and Inventory APIs
  - Support for Agreement API
  - Standardize service activation by integration Service Activation and Configuration API
  - Extend Party model including the Party Role Management API

FIWARE Global Summit

Find Us On

Stay up to date

JOIN OUR NEWSLETTER

Be certified and featured

FIWAREMarketplace

Hosting Partner

City of Vienna

vienna business agency

Keystone Sponsors

aws

ms.gis

Raseel

Red Hat

Media Partners

bee smart city

CITIES FORUM

CitiesToday

Our Future Water

RM RENEWABLE MATTER

Smart Cities World

the smartcity journal

FIWARE
Global
Summit

Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds