



infonova



The R6 REST API Journey

ABOUT US



Herbert Mühlburger

herbert.muehlburger@infonova.com

Senior System Engineer

[@hmuehlburger](#)



Richard Raumberger

richard.raumberger@infonova.com

System Engineer

[@der_raumbaer](#)

AGENDA

- About Infonova and Infonova R6
- RESTful APIs
- Infonova R6 REST API
- New Infonova R6 REST API
- Lessons Learned

ABOUT INFONOVA



- HQ in Graz, Offices Graz & Vienna, >350 employees
- Subsidiary of BearingPoint (European Business & Technology Consulting firm)
- Long established Comms & Media client relationships
- Austrian Leading Companies Award
- "Top 10 to Watch Company" 2015



INFONOVA R6

- Product Management
- Development Center
- Integration, Innovation, Testing Centre

ADVISORY & SOLUTION ENGINEERING

- Digital Strategy & Digital Transformation
- Enterprise Architecture & Solution Design
- Solution Delivery & Operations

OPERATIONS AND SUPPORT SERVICES

- 24x7 global Operations and Maintenance Support Centre
- Certified Database, Storage and Server, Network services
- Network and Security, IT and Servicedesk experts



ABOUT INFONOVA

Scrum

State-of-the-art
Technologies

Continuous Integration

Worldwide Customers Base
in different **Industries**

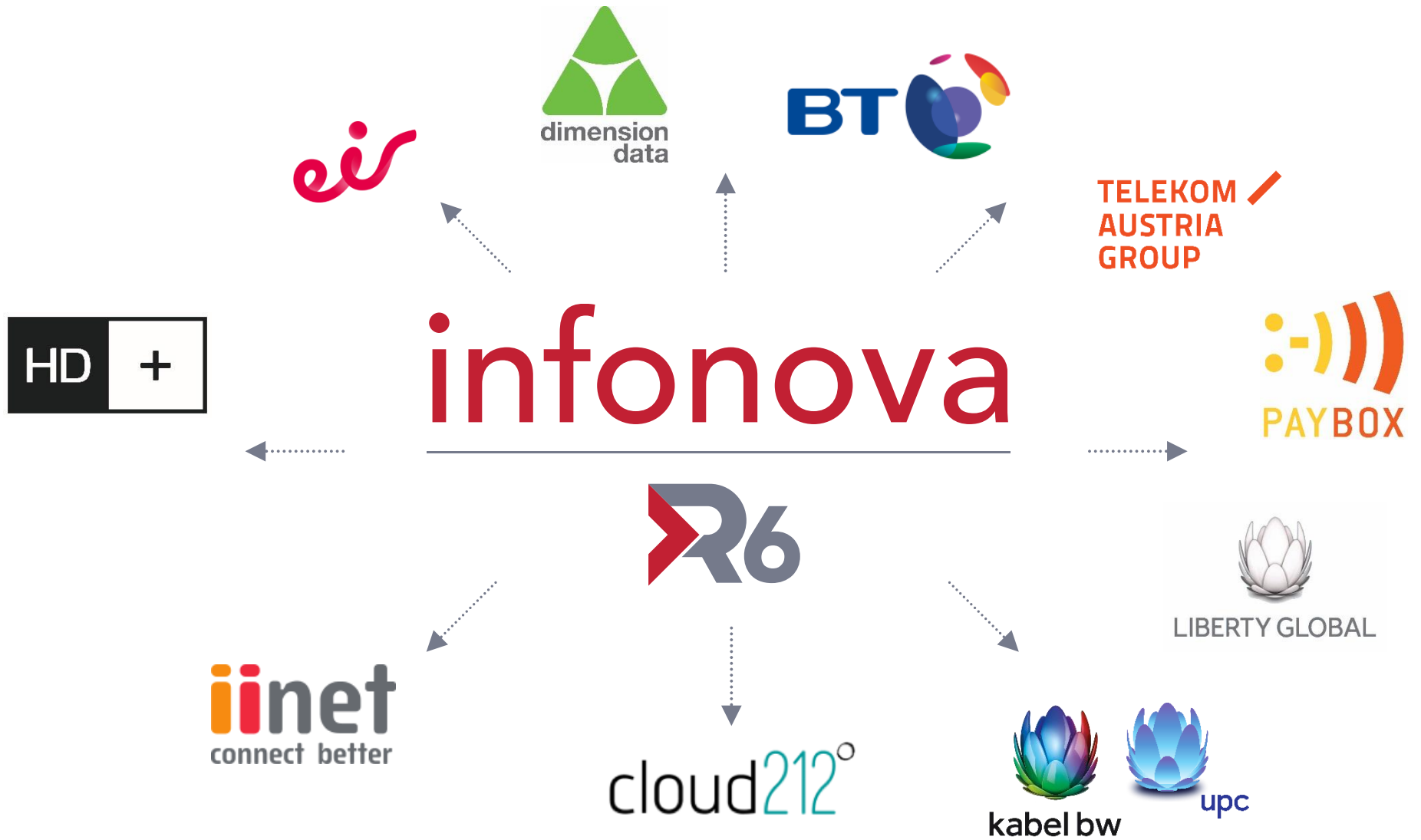
Operations & Support
Services

Advisory & Solution
Engineering

infonova



SELECTION OF INFONOVA R6 DEPLOYMENT AND CATALYST REFERENCES



Our **fun**



Getränke



L-e-g-e-n-d-a-r-y **Weihnachtsfeier**

Sport/Gesundheit



Massage

Team-Events



Essensmarken



Obst



Fortbildung

Parties



Technical Consultant / Architect (m/w)	Graz
Software Tester (m/w)	Graz
Senior Java Software Engineer (m/w)	Graz
Junior Java Software Engineer (m/w)	Graz
Server Spezialist für Microsoft Umgebungen (m/w)	Graz
Application Support Engineer (m/w)	Graz
DevOpS Engineer (m/w)	Graz
Software Architect Software-as-a-Service (m/w)	Graz
Jurist mit Schwerpunkt Vertragsrecht (m/w)	Graz
Agile Coach/Agile Consultant (m/w)	Graz, Wien
Solution Architekt (m/w)	Graz, Wien
Product Owner/Junior Projekt Manager (m/w)	Wien
Business Consultant für Enterprise Solutions (m/w)	Wien
Senior Software Engineer (m/w)	Wien
Junior Software Engineer (m/w)	Wien

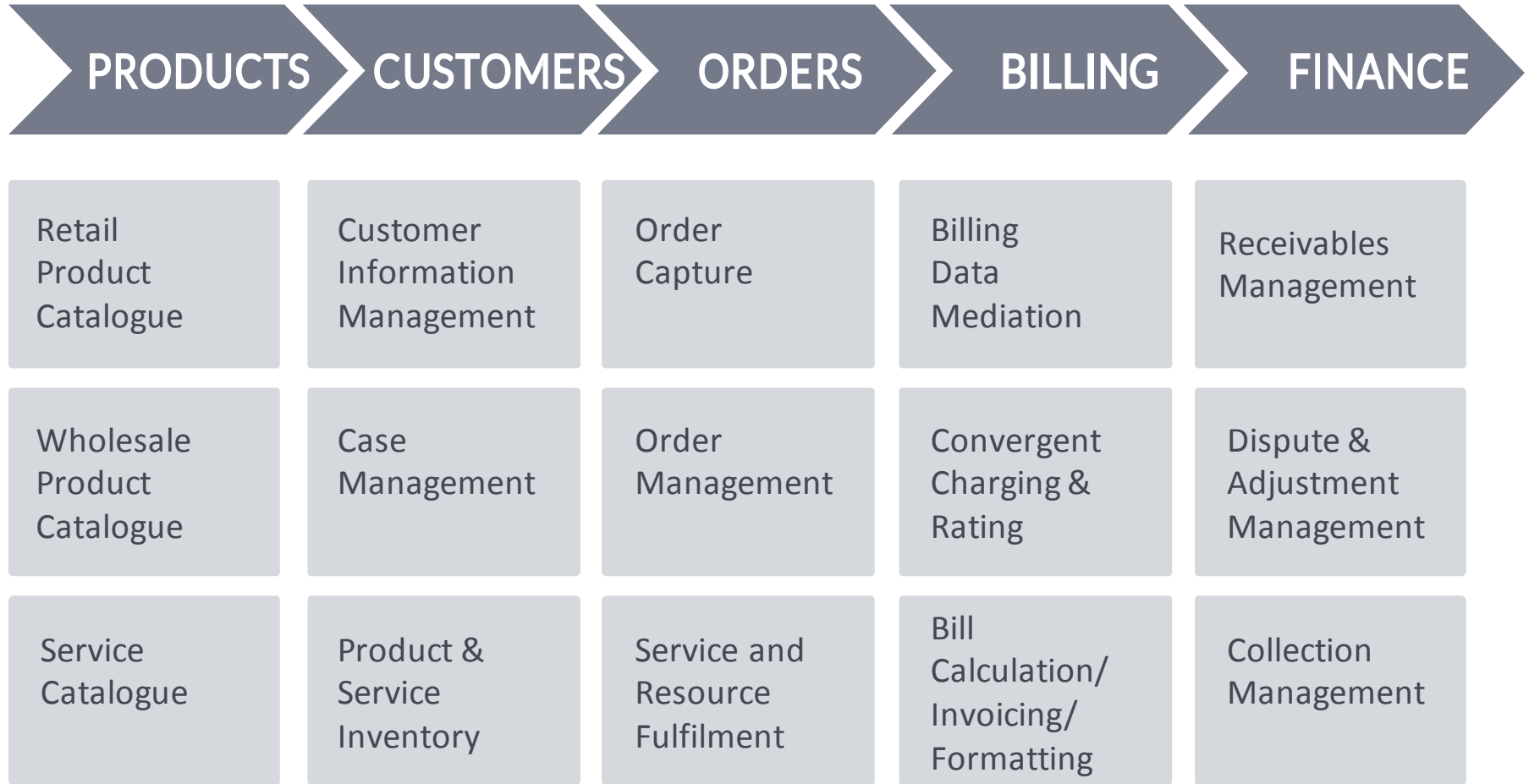
We are hiring!

infonova.com/jobs

infonova



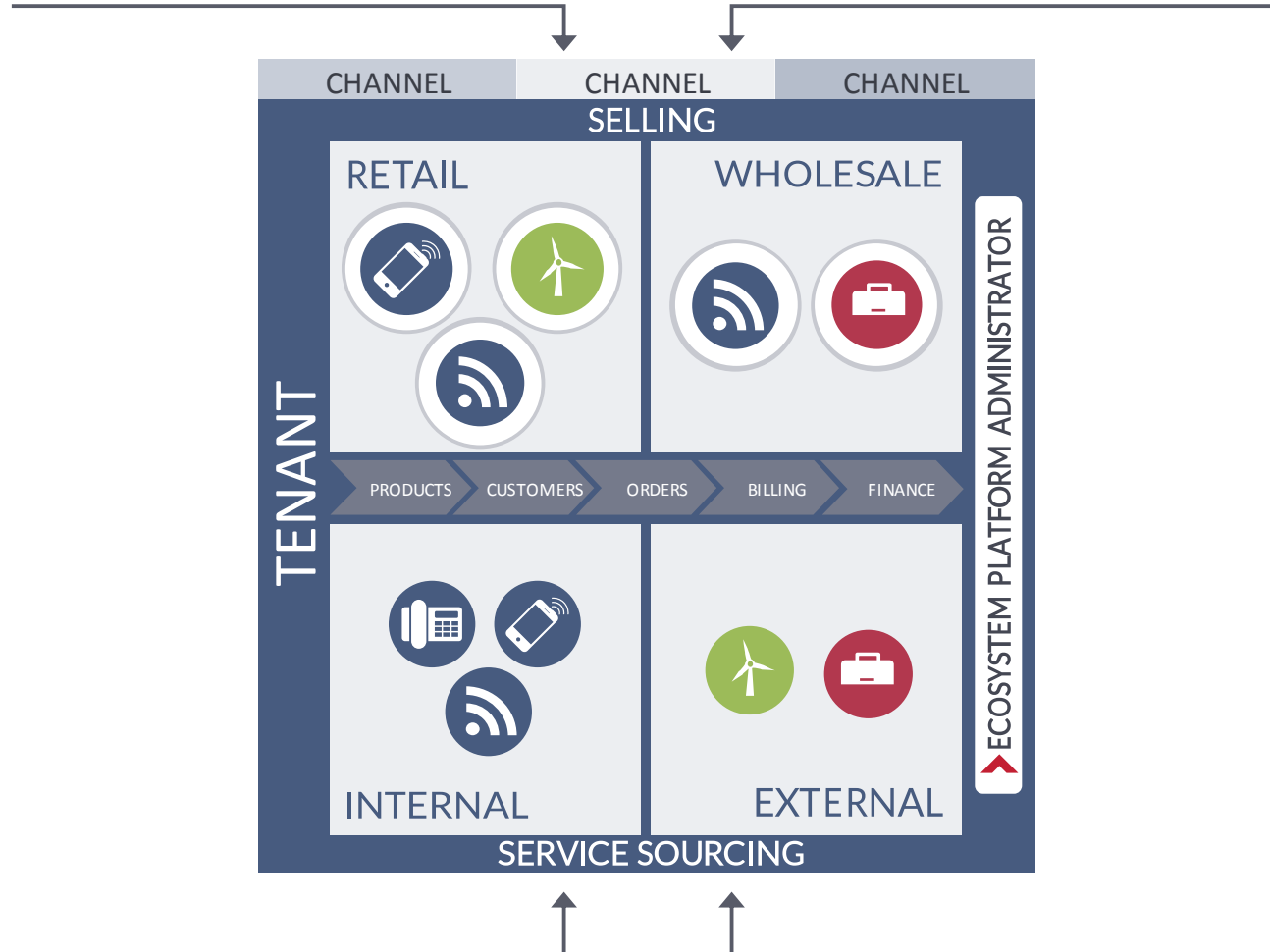
INFONOVA R6 – DIGITAL BUSINESS ENABLEMENT CAPABILITIES



INFONOVA R6 - DIGITAL BUSINESS ENABLEMENT CAPABILITIES

Each tenant can sell **retail offers** to end customers –
Retail offers can be based on the tenant's own services
or wholesale services from other tenants

Each tenant can sell **wholesale offers** to other tenants –
Wholesale offers can be based on the tenant's own services
or wholesale services from other tenants



Each tenant can directly onboard and
manage its own **internal services**

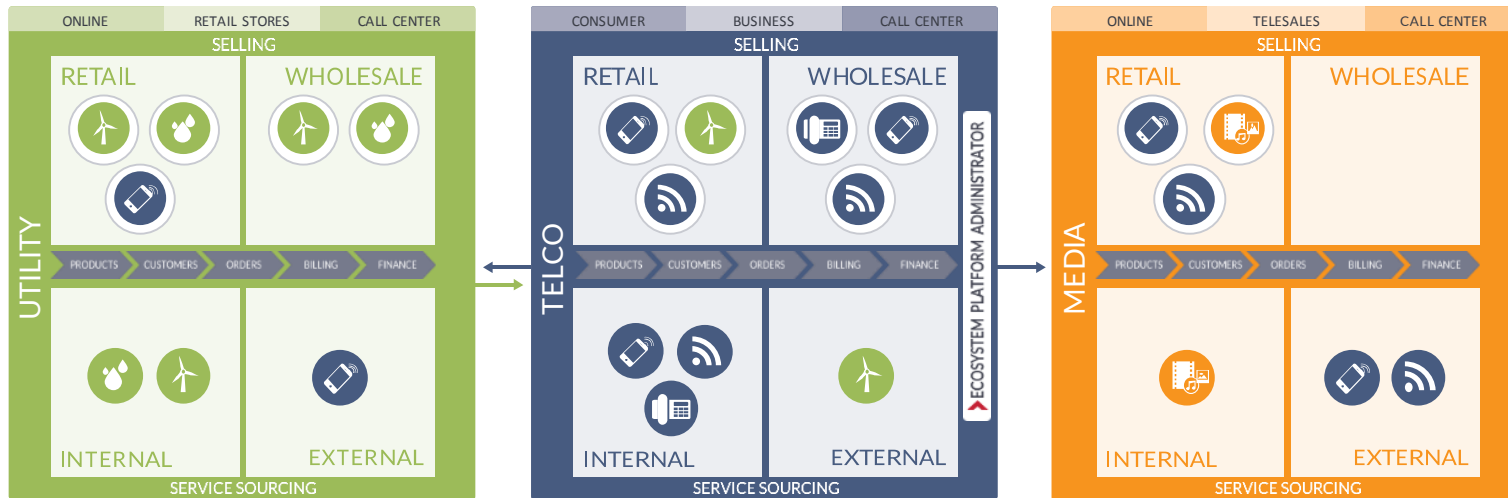
Each tenant can purchase
external services from other tenants

INFONOVA R6 - DIGITAL BUSINESS ENABLEMENT CAPABILITIES

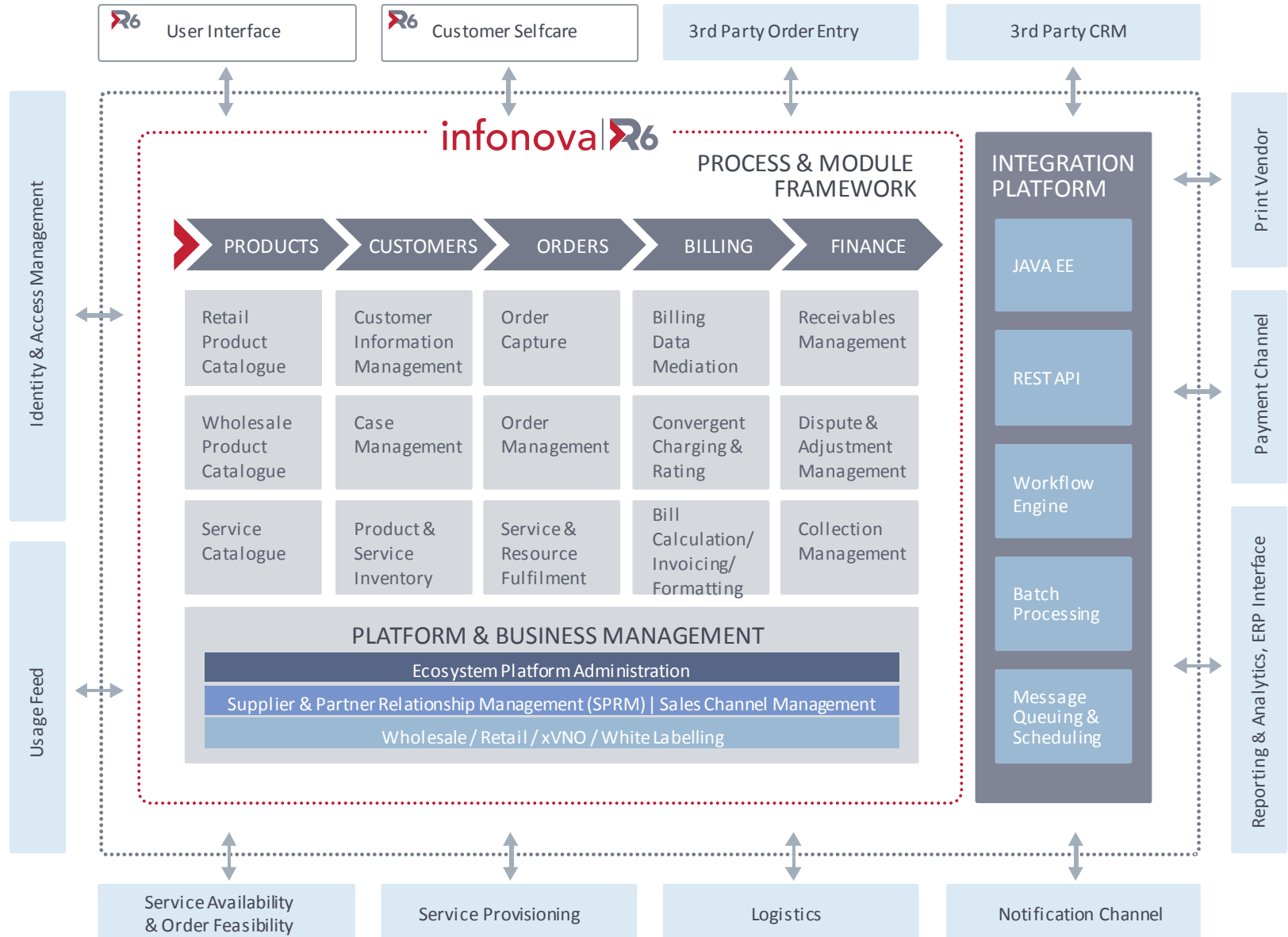


infonova | R6

MULTI-TENANT BUSINESS ARCHITECTURE



INFONOVA R6 INTEGRATION FRAMEWORK



RESTful APIs

REST

- Representational State Transfer (REST)
- Architectural pattern for distributed systems
- PhD Thesis of Roy Fielding "Architectural Styles and the Design of Network-based Software Architectures"
(2000)

Formal Constraints of a RESTful Architecture

- Client-Server architectural pattern
- Stateless
- Cacheable
- Layered
- Code-on-Demand
- Uniform Interface



Richardson Maturity Model

- Level 0 – RESTless
 - ... one endpoint to fit them all

- Level 1 - Resources
 - ... using divide and conquer, breaking a large service endpoint down into multiple resources

- Level 2 – HTTP Verbs
 - ... introduces a standard set of verbs so that we handle similar situations in the same way, removing unnecessary variation

- Level 3 – Hypermedia Controls (HATEOAS)
 - ... introduces discoverability, providing a way of making a protocol more self-documenting

(via martinfowler.com)

Infonova R6 REST API

Infonova R6 REST API - Architecture

➤ R6 REST API

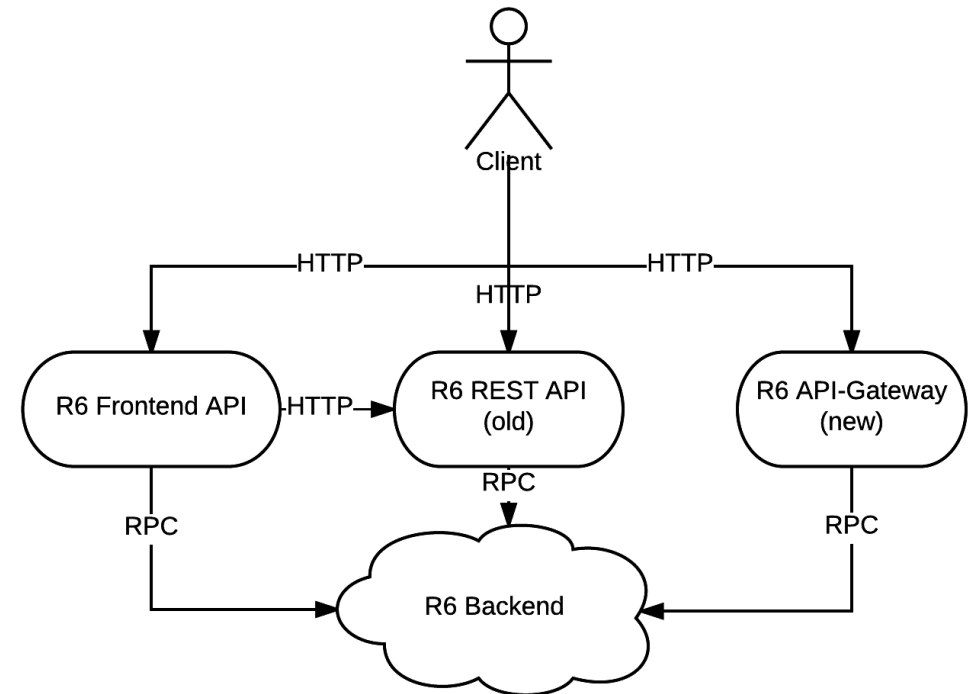
- Existing legacy API
- Number of endpoints: 85
 - Not all areas covered

➤ R6 API-Gateway

- Replacement for legacy API

➤ R6 Frontend API

- „Backend for Frontend“



Infonova R6 REST API – Current Status

- Organically grown out of a monolith
 - Resources extracted in their own artifact

- No Versioning
 - New and updated endpoints may change behavior

- Maintainability
 - Breaking changes difficult due to dependent projects

Infonova R6 REST API – Current Status

➤ Consistency

- Endpoints do not use the HTTP semantics consistently; e.g. POST vs. PUT
- Mix-up of HTTP status codes; e.g. 201 (Created) vs. 202 (Accepted) vs. 204 (No Content)
- Error handling
 - Not as transparent as we want it to be

➤ Documentation maintained separately

- Might be out-of-sync

➤ Large Payloads

- Backed by XSD
- Shared models

Goals for the new API

- Consistent APIs
 - Endpoints
 - Error messages and error handling
 - Documentation

- Versioning from the very beginning

- Loosely coupled API modules
 - Hypermedia links between resources

Goals for the new API

- Expose all business relevant information via REST

- Support different users
 - developers (internal view)
 - Hide HTTP semantics from developers
 - customers (external view)
 - Support different business cases from a customer perspective

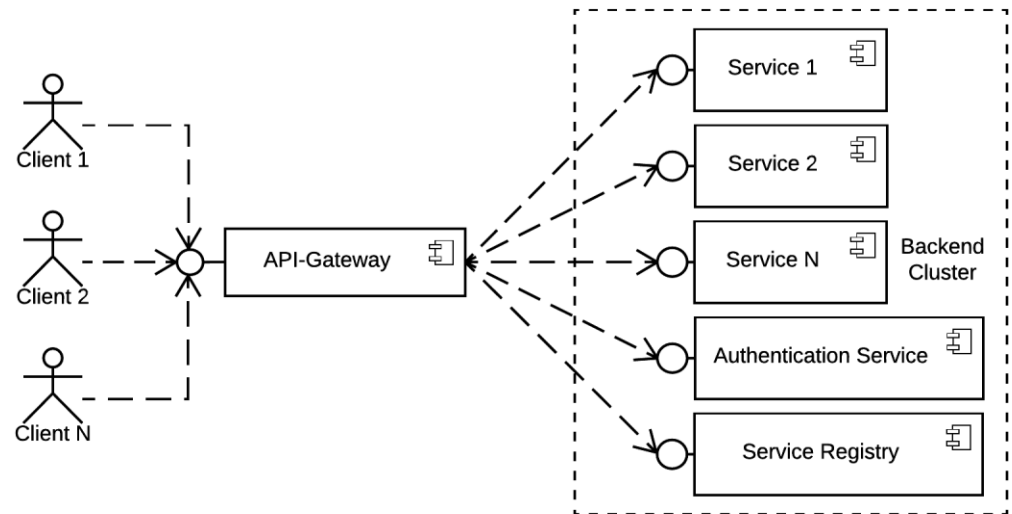
- Prepare for future technology changes

Disclaimer



Infrastructure

- Realized using an API-Gateway
- Based on
 - Spring Boot
 - Embedded Servlet Container
- Resources are packaged, developed and deployed in their respective services
 - e.g. all customer related resources in the customer related service
- API-Gateway delegates calls to the services



Documentation

- Markdown? Latex? AsciiDoc? Swagger? Other?
- Result:
 - AsciiDoc
 - Spring REST Docs
- Examples are generated via unit-tests
 - No more out of sync examples
- Why not Swagger?
 - There's more to it, than "just" resources
 - No mutual exclusion, but a question of prioritization



Versioning

- What should be versioned?
 - Whole API?
 - Do I want to release the whole API for every change?
 - Single Resource?
 - Do I want to manage this granularity?

- Solution: logical modules (e.g. customerAccounts, orders, etc.)

- Releasing a new stable version of an API leads to deprecation of previous versions

- Support for deprecated APIs will be provided for an additional major release
 - Support will be discontinued afterwards

Versioning

➤ Breaking Changes

- May alter the overall behavior and functionality of the API
- May be applied to released beta APIs
- May only be applied to beta APIs or to a new version of stable APIs
- Do not affect released stable APIs

➤ Non-breaking Changes

- Do not alter the overall behavior of the APIs
- Do not require client adaptations. Existing requests work as before.
- May be added for all released stable APIs

Versioning

➤ Methods

- URL? (<http://host/api/v1/resource>)
- Accept-Header? (<http://host/api/resource> Accept: vnd.infonova.r6.v1+json)
- Custom-Header? (<http://host/api/resource> Api-Version: 1)
- None

➤ Each method kinda sucks...

➤ Results

- Method: URL
- Target: Module
- e.g. GET <http://host/api/customerAccounts/v1/history>

API - Rollout Process

- Internal Review → Feedback!
- Public Review → Feedback!
- Public Beta
- Stable Release
- Maintenance

General Improvements

➤ Complete API Guidelines

- including a framework supporting devs to produce compliant APIs

➤ Reviews

➤ (Almost) no shared models between APIs

➤ Hypermedia

- Linking between resources; e.g.

```
"_links": {  
  "reference:HistoryType": {  
    "templated": true,  
    "href": "/r6-api/tenant/customers/v1/historyTypes/{type}"  
  }  
}
```

Lessons Learned

- Collect feedback as early as possible

- Don't make it perfect, but perfectly useable
 - Purism vs. Pragmatism
 - Build APIs for the clients, not the backend

- Software evolves, so does your API → Plan for it!
 - Look in the magic software crystal ball!

- Keep payloads small
 - Don't cover every possibility in one response
 - Instead, enable resource aggregation

- Be careful with common schemas
 - Adding elements to one resource might not be suitable for another resource

References

- Richardson Maturity Model
<https://martinfowler.com/articles/richardsonMaturityModel.html>
- Architectural Styles and the Design of Network-based Software Architectures
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

infonova

GLT17 Besucherumfrage

<https://survey.linuxtage.at>