INSURANCE BUSINESS APPLICATIONS

THE PROVEN, NEXT-GENERATION CLOUD INSURANCE PLATFORM

Introduction to IBSuite – Architecture

January 2021

Todays Session

Trainer: Simon Pooley

IBA Solution Architect

Assisted by: Valentin Szel

- In general, we like interactive sessions
- But we are on Teams and quite a few people so:
  - “Raise your hand” if you want to ask a question
  - Or write questions in chat
- We will try to answer all in the session, but some may be leftover and we will answer these in writing
- We may group the questions and answer them at the end of each section
- Please mute your microphone when not speaking

Learning objective of the session

- Understand key IBSuite architecture concepts
  - Three C’s
  - What is an IBSuite instance?
- Integration
  - Understand how IBSuite fits in an Insurer’s landscape
  - Understand preferred integration patterns
- Key domain/data models
  - Product
  - Policy/Policy contract
  - Flexibility
- Infrastructure understanding
  - No need to really understand it as SaaS

Introduction

- IBSuite is SaaS not traditional software
- Overview of architecture

IBSuite Availability and Upgrades

- IBSuite is available **24 / 7**
- IBSuite is true cloud and always "upgraded", i.e. there is no such thing as a cumbersome upgrade project, ever.
- All IBSuite customers are using the same version of IBSuite
- IBSuite is currently upgraded on a monthly basis
- New versions are always backwards compatible

IBSuite is always running and always running on the latest version

Make Your IT simple and your business scalable

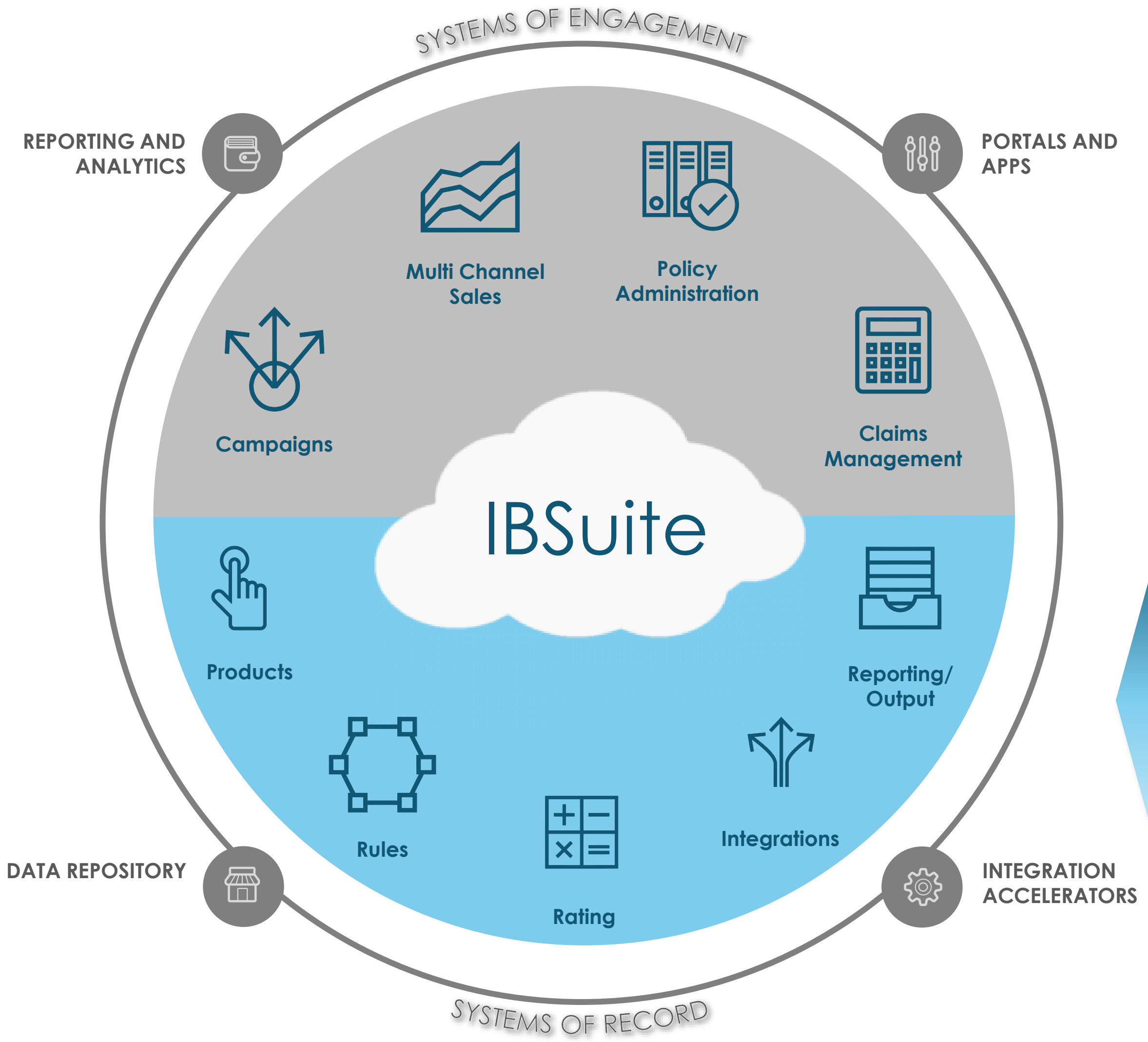
- Achieve scale in transactional volume and performance

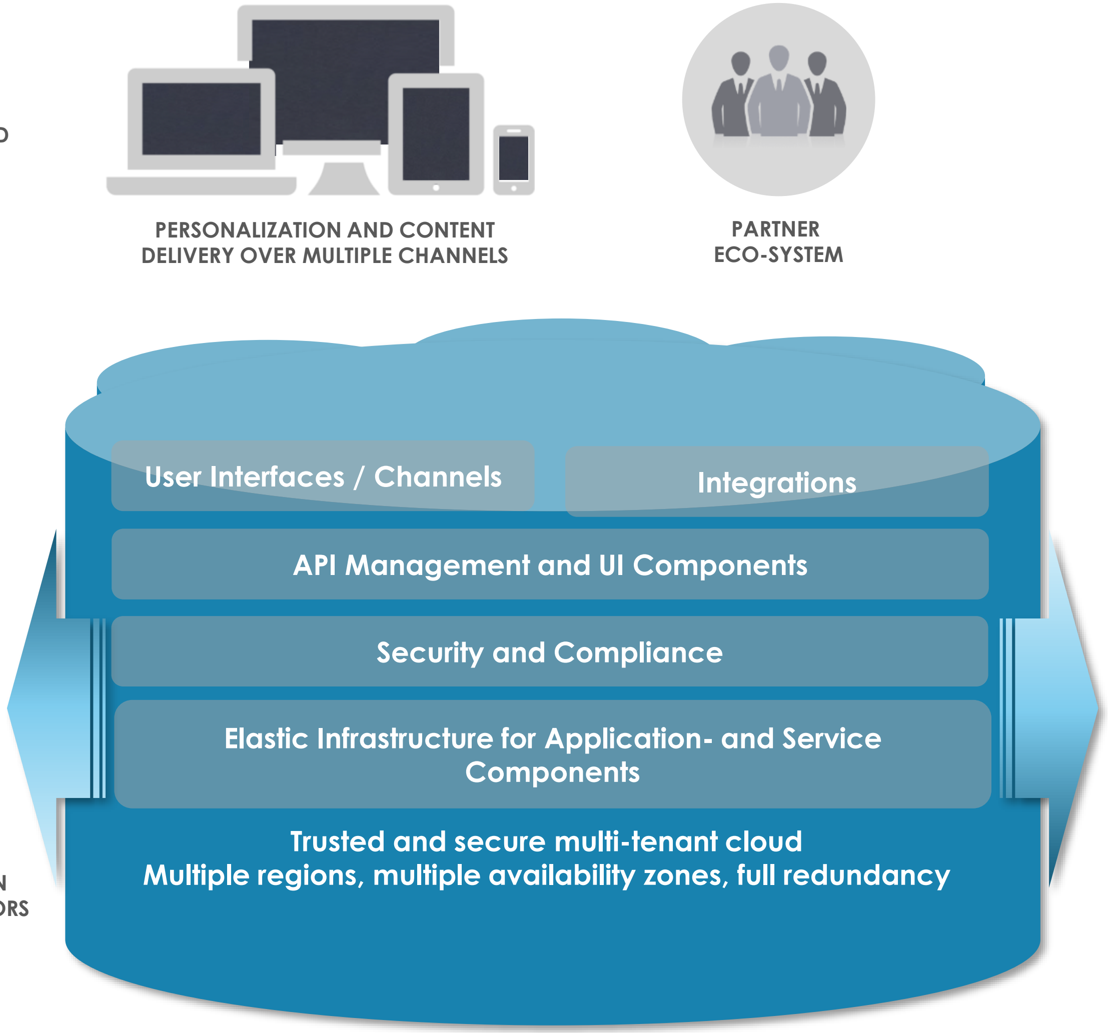
IBA

IBSuite is a SaaS Solution – How does this impact the architecture?

- Delivering a service not just software
  - Impacts EDO methodology, e.g. hypercare, support team
- Access
  - All access through HTTPS
    - Browser – single login page URL per environment for all IBSuite customers
    - API
  - Accessible from any device connected to the internet
  - No access to the Database or File systems
- Standardization
  - All customers share:
    - Infrastructure
    - Same version of the Java code
    - All run is same JVM
  - But there is logically separate database
  - All core changes must be backwards compatible

The IBSuite Platform - High-Level Architecture

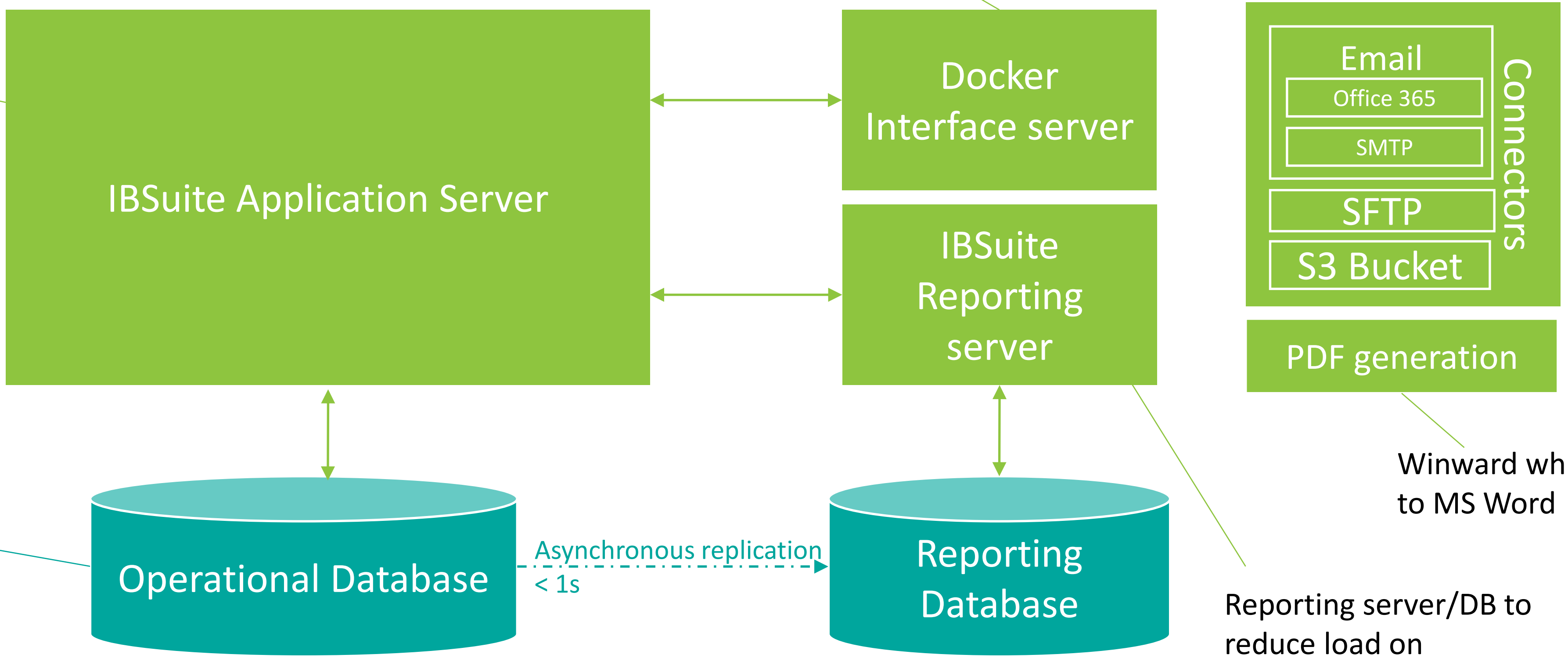



PERSONALIZATION AND CONTENT DELIVERY OVER MULTIPLE CHANNELS
PARTNER ECO-SYSTEM
User Interfaces / Channels
Integrations
API Management and UI Components
Security and Compliance
Elastic Infrastructure for Application- and Service Components
Trusted and secure multi-tenant cloud
Multiple regions, multiple availability zones, full redundancy

Logical IBSuite components

All customers running in same JVM. IBSuite UI makes it appear that each instance is a different application

Where required interfaces are built as separate applications and deployed in a Docker server



Data is logically separated by insurer. Data has two elements:

1. Production data like policies, customers etc.
2. Meta data like product and system config

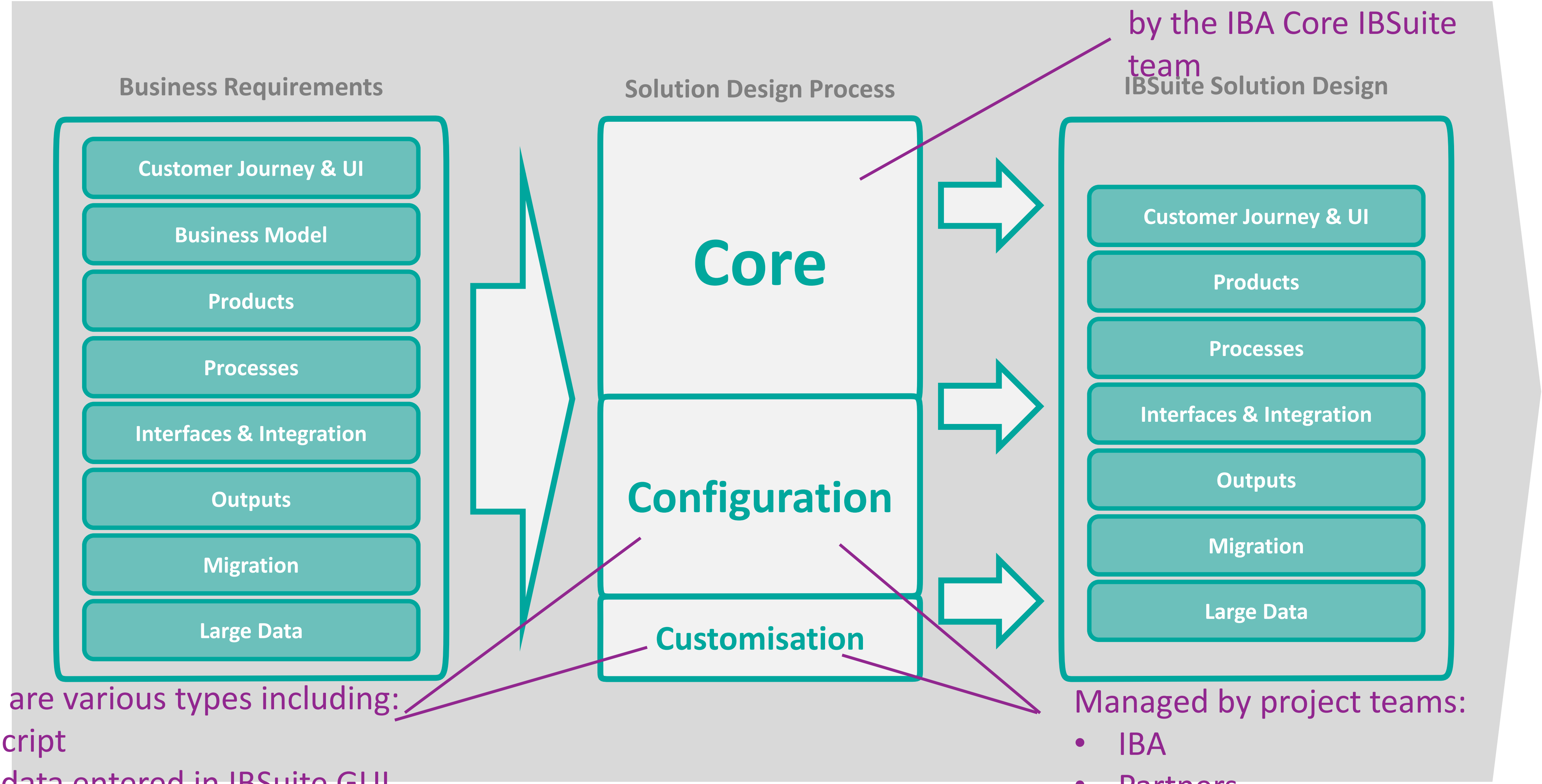
Winward which is a plugin to MS Word

Reporting server/DB to reduce load on operational components

Three C's

- Core
- Configuration
- Customization

IBSuite Solutioning Approach and Out of the Box Functionality



Java – Completely managed by the IBA Core IBSuite team

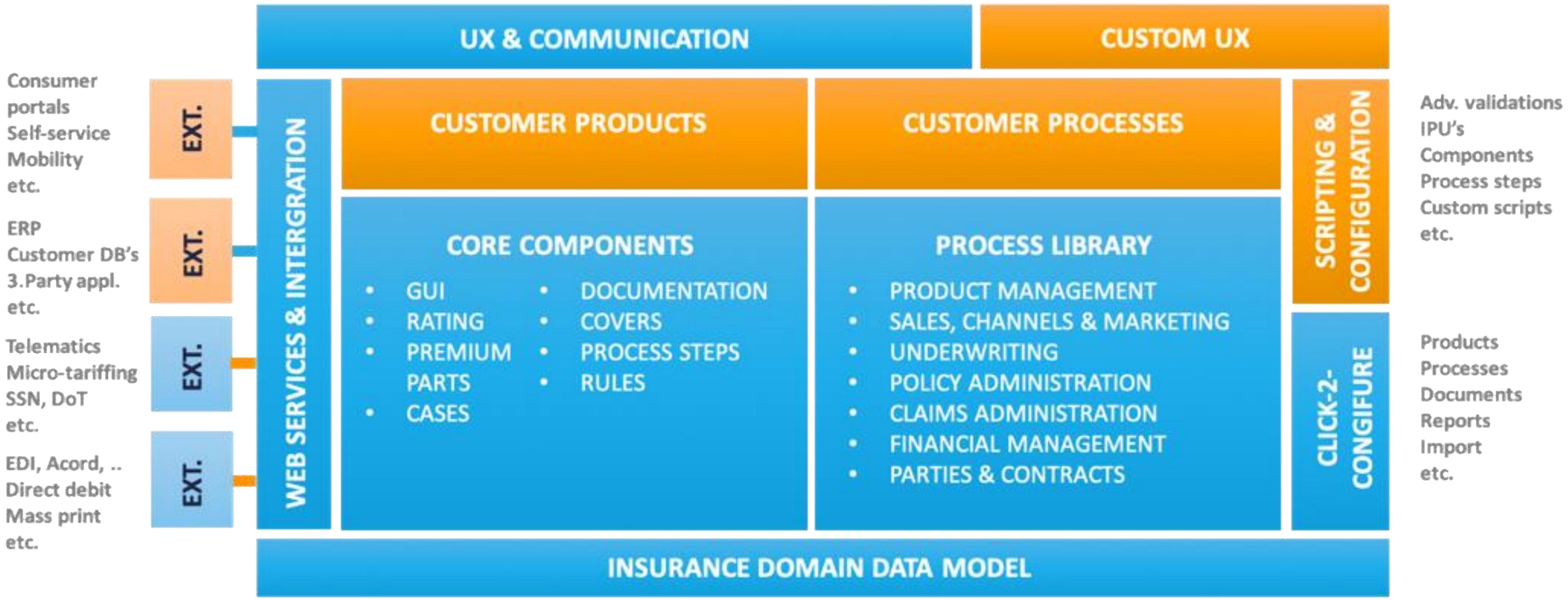
Artifacts are various types including:

- JavaScript
- Metadata entered in IBSuite GUI
- HTML IPUs/Emails

Managed by project teams:

- IBA
- Partners
- Customers

IBSuite Conceptual Model



Consumer portals
Self-service
Mobility
etc.

ERP
Customer DB's
3.Party appl.
etc.

Telematics
Micro-tariffing
SSN, DoT
etc.

EDI, Acord, ..
Direct debit
Mass print
etc.

EXT.

EXT.

EXT.

EXT.

UX & COMMUNICATION

CUSTOM UX

WEB SERVICES & INTERGRATION

CUSTOMER PRODUCTS

CUSTOMER PROCESSES

CORE COMPONENTS

- GUI
- RATING
- PREMIUM PARTS
- CASES
- DOCUMENTATION
- COVERS
- PROCESS STEPS
- RULES

PROCESS LIBRARY

- PRODUCT MANAGEMENT
- SALES, CHANNELS & MARKETING
- UNDERWRITING
- POLICY ADMINISTRATION
- CLAIMS ADMINISTRATION
- FINANCIAL MANAGEMENT
- PARTIES & CONTRACTS

SCRIPTING & CONFIGURATION

CLICK-2-CONGIFURE

Adv. validations
IPU's
Components
Process steps
Custom scripts
etc.

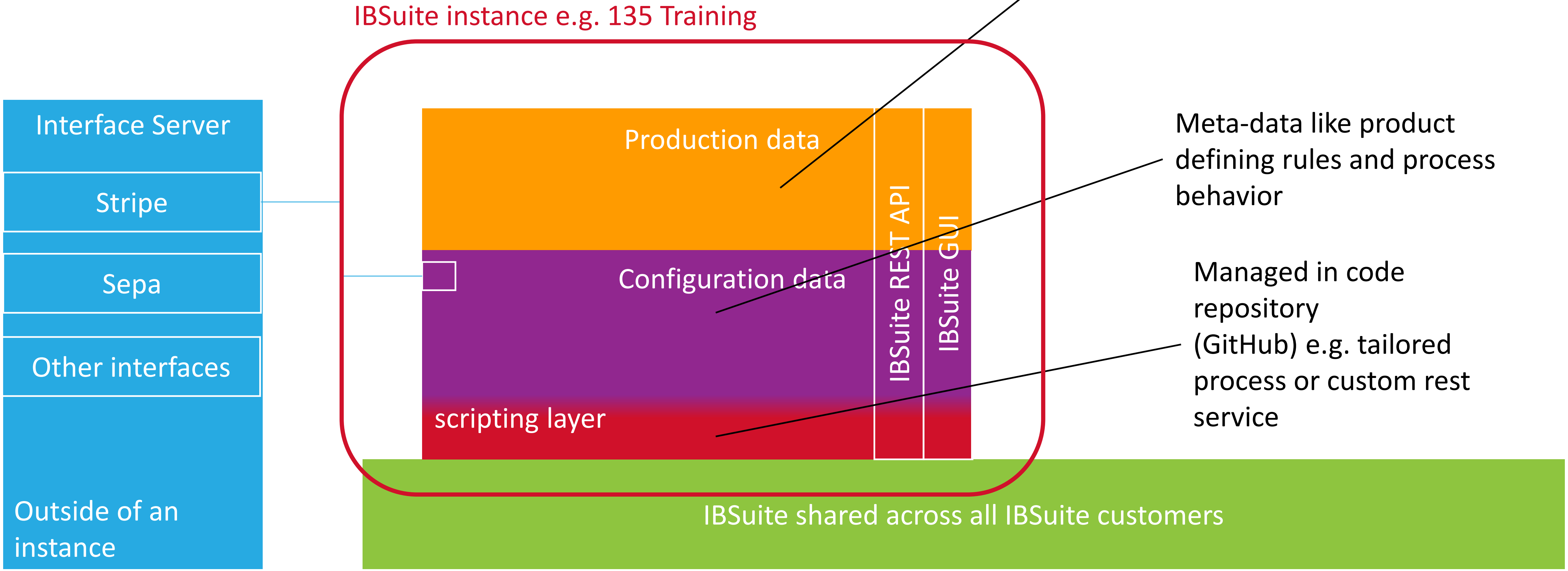
Products
Processes
Documents
Reports
Import
etc.

INSURANCE DOMAIN DATA MODEL

🟦 IBSuite standard components, modules and processes.

🟧 IBSuite products can be configured, processes can be implemented, extensions can be scripted, rules can be defined and integrations/interfaces can be set up.

What is an IBSuite Instance?

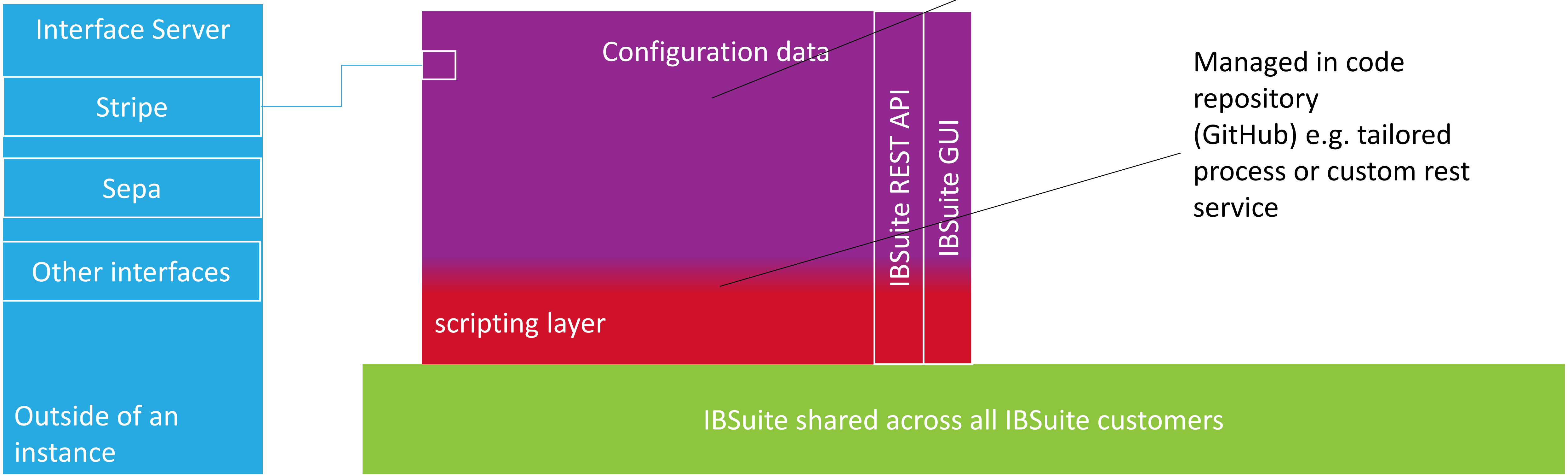


Policies, customers, claims etc.

Meta-data like product defining rules and process behavior

Managed in code repository (GitHub) e.g. tailored process or custom rest service

Code and Configuration Layers in IBSuite



Data managed in the instance e.g. product configuration

Managed in code repository (GitHub) e.g. tailored process or custom rest service

* Diagram is oversimplified as there some other configuration aspects

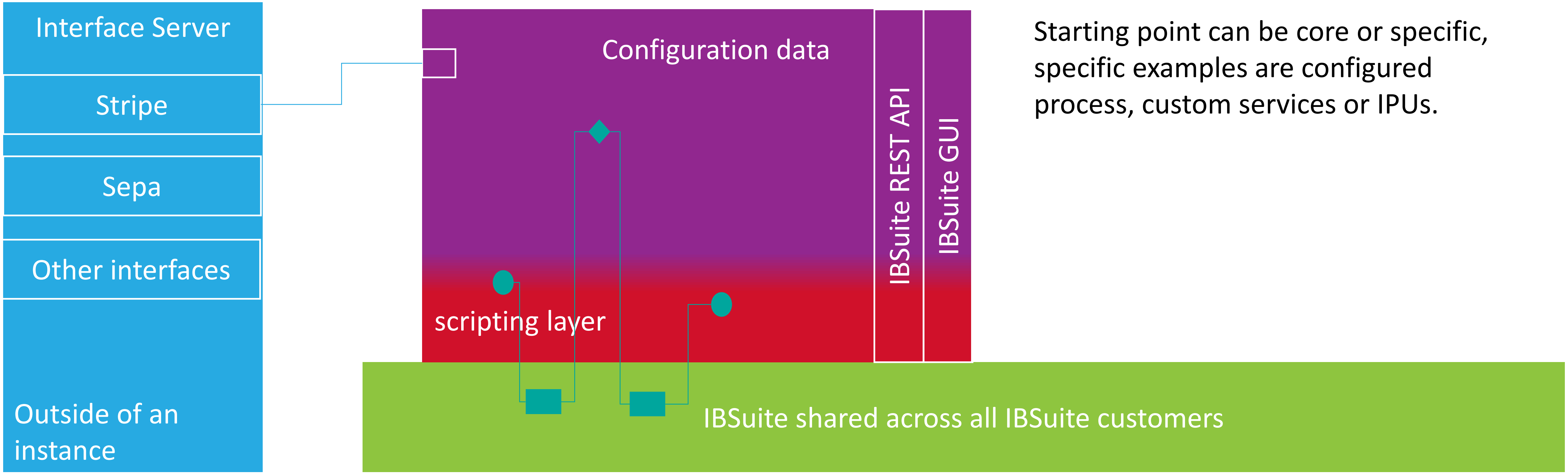
Processes and Functionality Weave Layers Together

Examples:

IBSuite create customer process includes specific extension to send data to Salesforce

IBSuite policy activation process sends data to Llieda.net and stores id and other returned data

Configured process allows Lleida.net to inform about signature data is stored in a mixture of core and configured fields

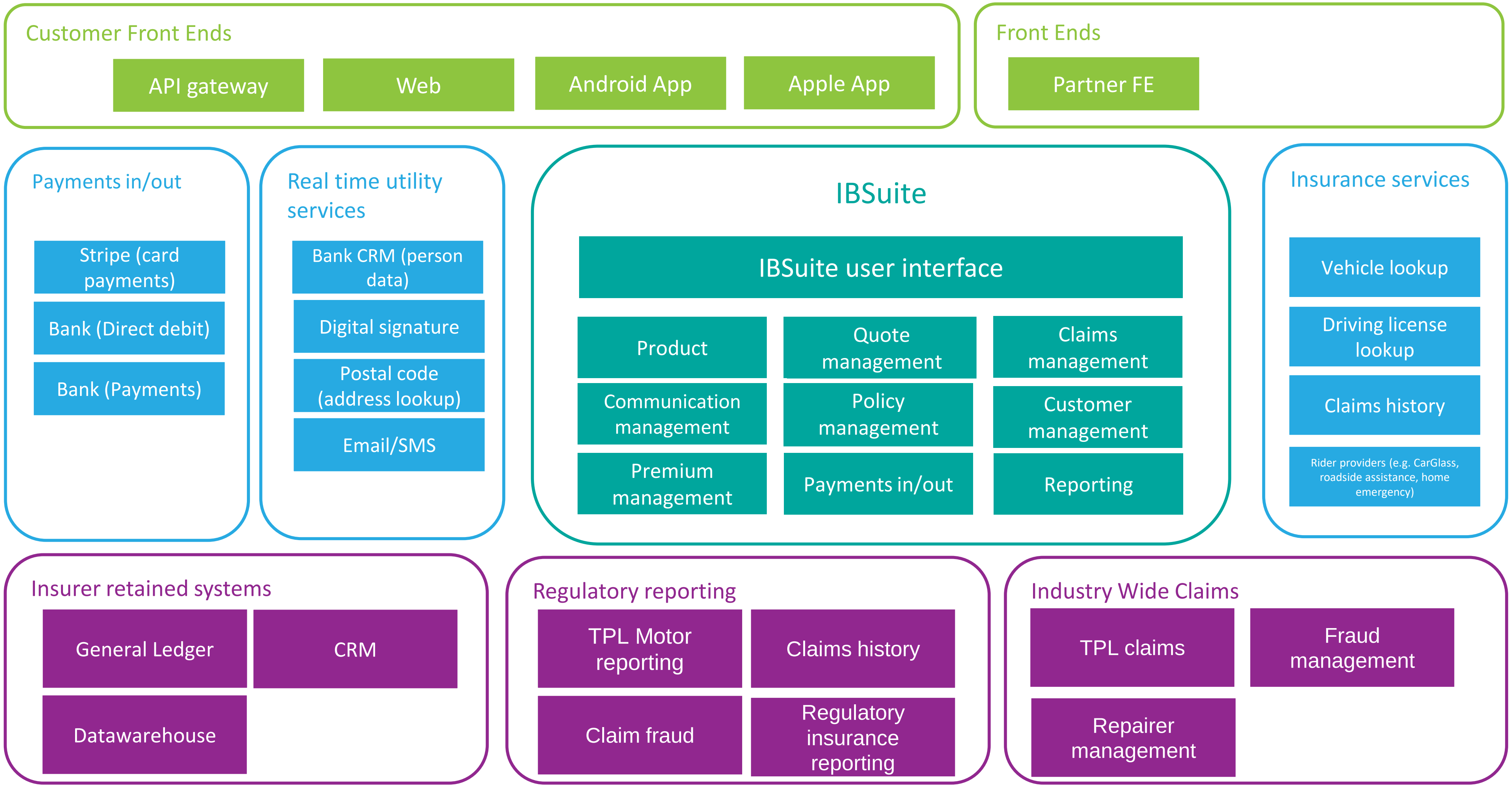


Starting point can be core or specific, specific examples are configured process, custom services or IPUs.

Integration

- IBSuite in an Insurer’s landscape
- Inbound and outbound interfaces
- Real time/ near real time and batch files

Sample Insurance Architecture



Customer Front Ends
- API gateway
- Web
- Android App
- Apple App

Front Ends
- Partner FE

Payments in/out
- Stripe (card payments)
- Bank (Direct debit)
- Bank (Payments)

Real time utility services
- Bank CRM (person data)
- Digital signature
- Postal code (address lookup)
- Email/SMS

IBSuite
- IBSuite user interface
- Product
- Quote management
- Claims management
- Communication management
- Policy management
- Customer management
- Premium management
- Payments in/out
- Reporting

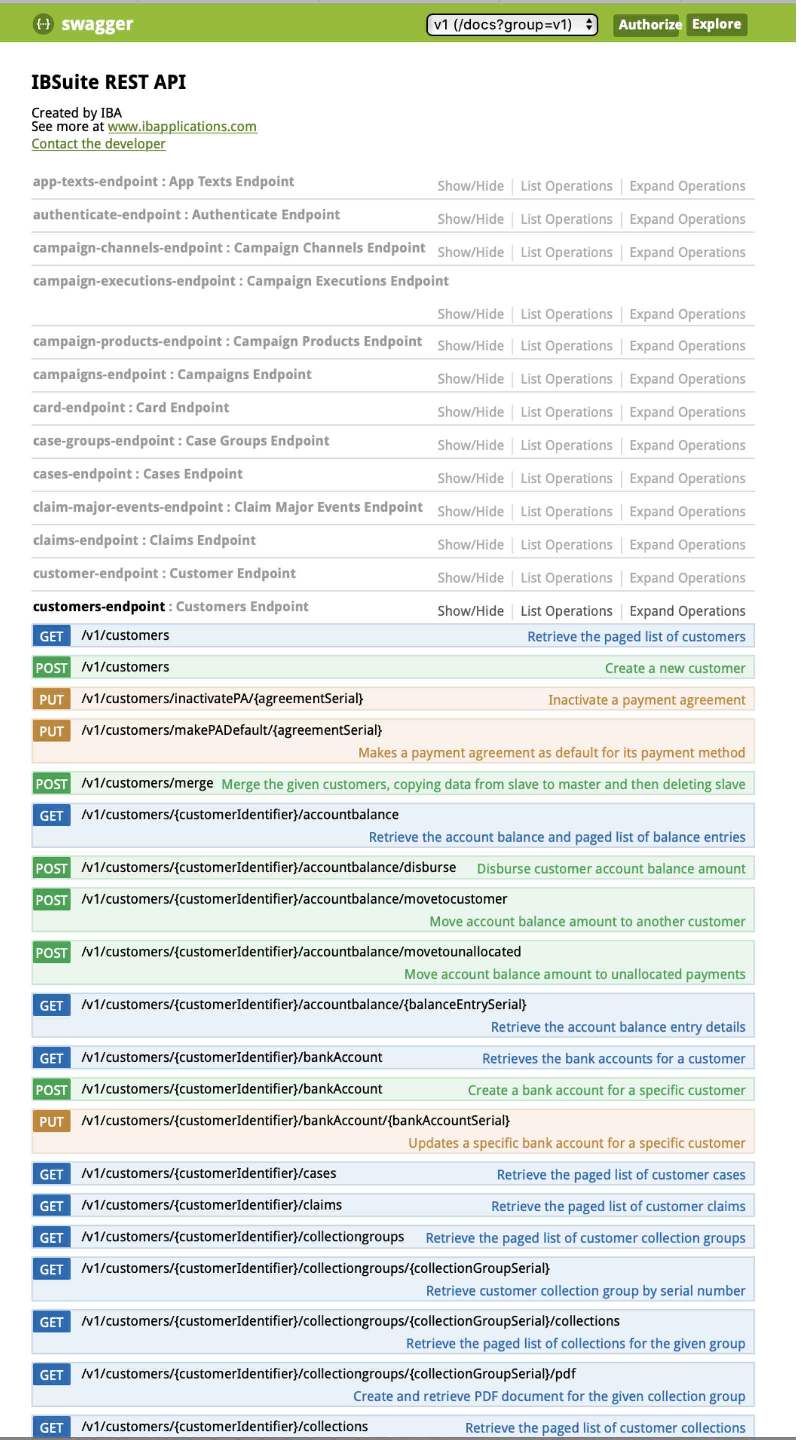
Insurance services
- Vehicle lookup
- Driving license lookup
- Claims history
- Rider providers (e.g. CarGlass, roadside assistance, home emergency)

Insurer retained systems
- General Ledger
- CRM
- Datawarehouse

Regulatory reporting
- TPL Motor reporting
- Claims history
- Claim fraud
- Regulatory insurance reporting

Industry Wide Claims
- TPL claims
- Fraud management
- Repairer management

Interfaces

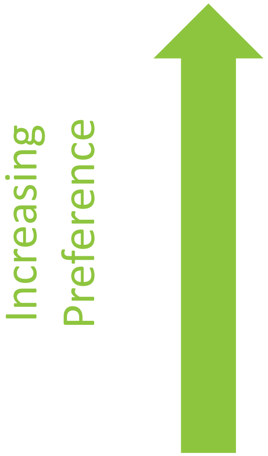


IBSuite REST API

Created by IBA
See more at www.ibapplications.com
Contact the developer

app-texts-endpoint : App Texts Endpoint — Show/Hide | List Operations | Expand Operations
authenticate-endpoint : Authenticate Endpoint — Show/Hide | List Operations | Expand Operations
campaign-channels-endpoint : Campaign Channels Endpoint — Show/Hide | List Operations | Expand Operations
campaign-executions-endpoint : Campaign Executions Endpoint — Show/Hide | List Operations | Expand Operations
campaign-products-endpoint : Campaign Products Endpoint — Show/Hide | List Operations | Expand Operations
campaigns-endpoint : Campaigns Endpoint — Show/Hide | List Operations | Expand Operations
card-endpoint : Card Endpoint — Show/Hide | List Operations | Expand Operations
case-groups-endpoint : Case Groups Endpoint — Show/Hide | List Operations | Expand Operations
cases-endpoint : Cases Endpoint — Show/Hide | List Operations | Expand Operations
claim-major-events-endpoint : Claim Major Events Endpoint — Show/Hide | List Operations | Expand Operations
claims-endpoint : Claims Endpoint — Show/Hide | List Operations | Expand Operations
customer-endpoint : Customer Endpoint — Show/Hide | List Operations | Expand Operations
customers-endpoint : Customers Endpoint — Show/Hide | List Operations | Expand Operations

GET	/v1/customers	Retrieve the paged list of customers
POST	/v1/customers	Create a new customer
PUT	/v1/customers/inactivatePA/{agreementSerial}	Inactivate a payment agreement
PUT	/v1/customers/makePADefault/{agreementSerial}	Makes a payment agreement as default for its payment method
POST	/v1/customers/merge	Merge the given customers, copying data from slave to master and then deleting slave
GET	/v1/customers/{customerIdentifier}/accountbalance	Retrieve the account balance and paged list of balance entries
POST	/v1/customers/{customerIdentifier}/accountbalance/disburse	Disburse customer account balance amount
POST	/v1/customers/{customerIdentifier}/accountbalance/movetocustomer	Move account balance amount to another customer
POST	/v1/customers/{customerIdentifier}/accountbalance/movetounallocated	Move account balance amount to unallocated payments
GET	/v1/customers/{customerIdentifier}/accountbalance/{balanceEntrySerial}	Retrieve the account balance entry details
GET	/v1/customers/{customerIdentifier}/bankAccount	Retrieves the bank accounts for a customer
POST	/v1/customers/{customerIdentifier}/bankAccount	Create a bank account for a specific customer
PUT	/v1/customers/{customerIdentifier}/bankAccount/{bankAccountSerial}	Updates a specific bank account for a specific customer
GET	/v1/customers/{customerIdentifier}/cases	Retrieve the paged list of customer cases
GET	/v1/customers/{customerIdentifier}/claims	Retrieve the paged list of customer claims
GET	/v1/customers/{customerIdentifier}/collectiongroups	Retrieve the paged list of customer collection groups
GET	/v1/customers/{customerIdentifier}/collectiongroups/{collectionGroupSerial}	Retrieve customer collection group by serial number
GET	/v1/customers/{customerIdentifier}/collectiongroups/{collectionGroupSerial}/collections	Retrieve the paged list of collections for the given group
GET	/v1/customers/{customerIdentifier}/collectiongroups/{collectionGroupSerial}/pdf	Create and retrieve PDF document for the given collection group
GET	/v1/customers/{customerIdentifier}/collections	Retrieve the paged list of customer collections



- IBSuite approach to interfaces
  - Industry standard interfaces
  - REST API
    - Inbound - IBSuite has extensive JSON REST API
    - Outbound - REST Services
  - Real time
  - File – CSV, Flat file, Parquet, XML

Inbound REST Service calls

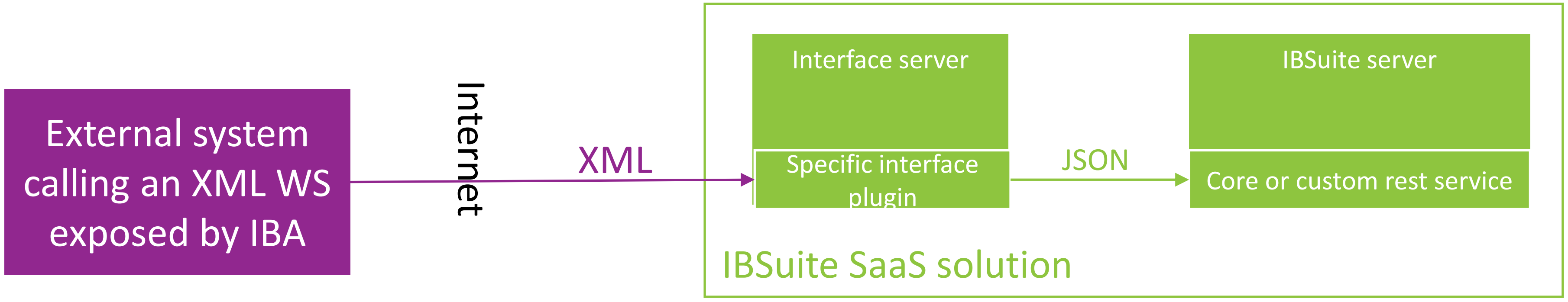
- IBSuite is API first – Core REST Services API
  - The REST API is the primary way to get data in or out of IBSuite and call IBSuite functionality
  - All core IBSuite functionality is exposed through API
  - Extensive list of Core rest service documented in Swagger
  - Logging to IBSuite and then click on the link
    - Training (https://training.dev.ibapplications.com/rest/swagger-ui.html#/)
    - Any dev instance (https://dev.ibapplications.com/rest/swagger-ui.html#/)
    - Any test instance (https://test.ibapplications.com/rest/swagger-ui.html#/)

- Can be extended with Custom REST services – Examples of use:
  - Expose custom functionality to front ends or other systems
  - Where the core services do not expose functionality in a way that is convenient/usable to the FE or other system
  - To wrap external services so all business connectivity is managed in IBSuite and a FE can reuse this connection
- Postman typically used as by product specialists to dev/test IBSuite services

Other Inbound service calls

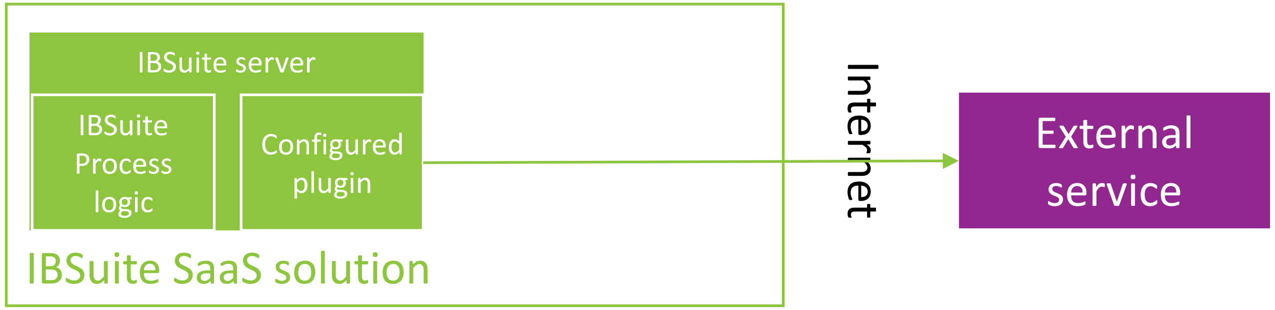
- As first preference is “Industry standard interfaces” need a mechanism to support other protocols
- “Plugin” must be created and deployed to the Docker interface server
  - Primarily for protocol transformation
  - Could also be used for:
    - Transformation
    - Orchestration

  But best practice is that the plugin is purely the technical aspects of the interface
- Plugins are built and managed by the core IBSuite team not project teams

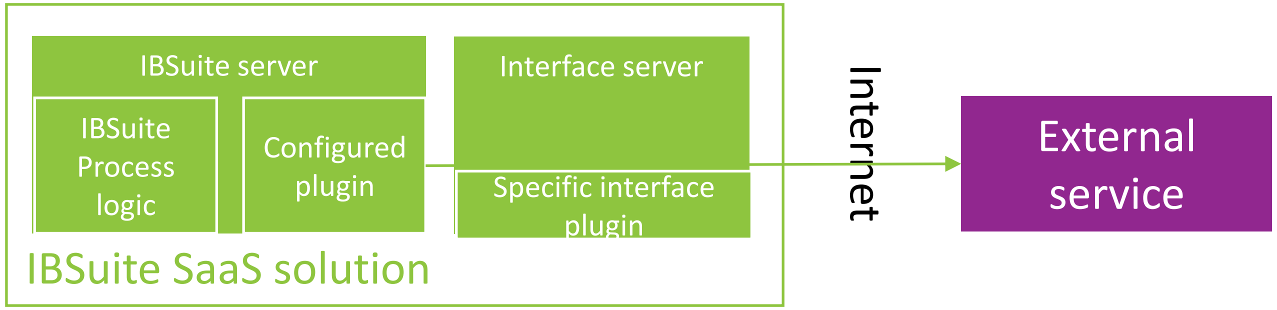

External system calling an XML WS exposed by IBA
Internet
XML
Interface server
Specific interface plugin
JSON
IBSuite server
Core or custom rest service
IBSuite SaaS solution

Implementation of outbound real time interfaces

- Clear separation between:
  - Technical details of the interface
  - The mapping of the data to and from and the uses of the interface in business process
- Simplest example out bound JSON REST service:
  - Rest service plugin holds technical configuration
    - End point URL
    - HTTP header parameters
  - The process logic and the mapping is defined in in the process



- Other real time (e.g. Soap web service)
  - IBA create separate plugin application
    - Does protocol transformation
    - Deploy to Docker interface server



File/Batch interfaces

- Pull/push files to SFTP/S3 Bucket server on client side

- Two approaches
  - Using a “plugin” **– Best Practice**
    - Real time interface between IBSuite and Plugin to trigger the data needed when appropriate
      - Can happen throughout the day even if the file is nightly/weekly/etc.
        - Data stored in plugin
    - File creation process triggered as a rest service
    - Plugin manages all technical aspects of the file e.g. Headers/ footer, row counts, data formatting

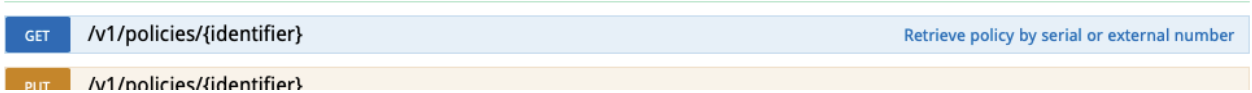
  - Configure file creation in custom IBSuite job
    - Push/pull file to SFTP/S3 bucket using connector

Domain/data models

- Different data models
- Key models
  - Product
  - Policy/policy contract
  - Quote/Quote entry
- Managing dynamic data models in a fixed physical model

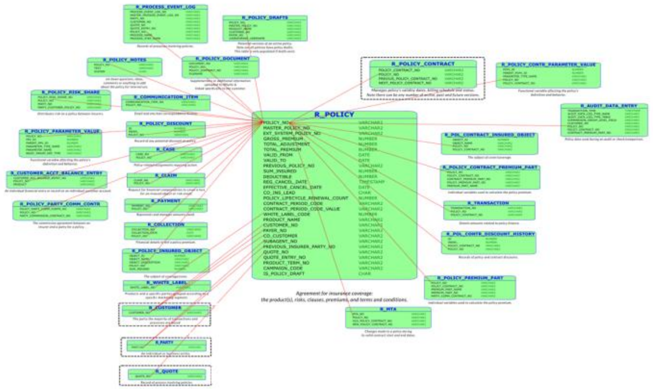
Domain or data models in IBSuite

- The two important data models in IBSuite are:
  - The data exposed through GET REST service e.g.
  - Reporting or IQL data model



GET /v1/policies/{identifier} Retrieve policy by serial or external number

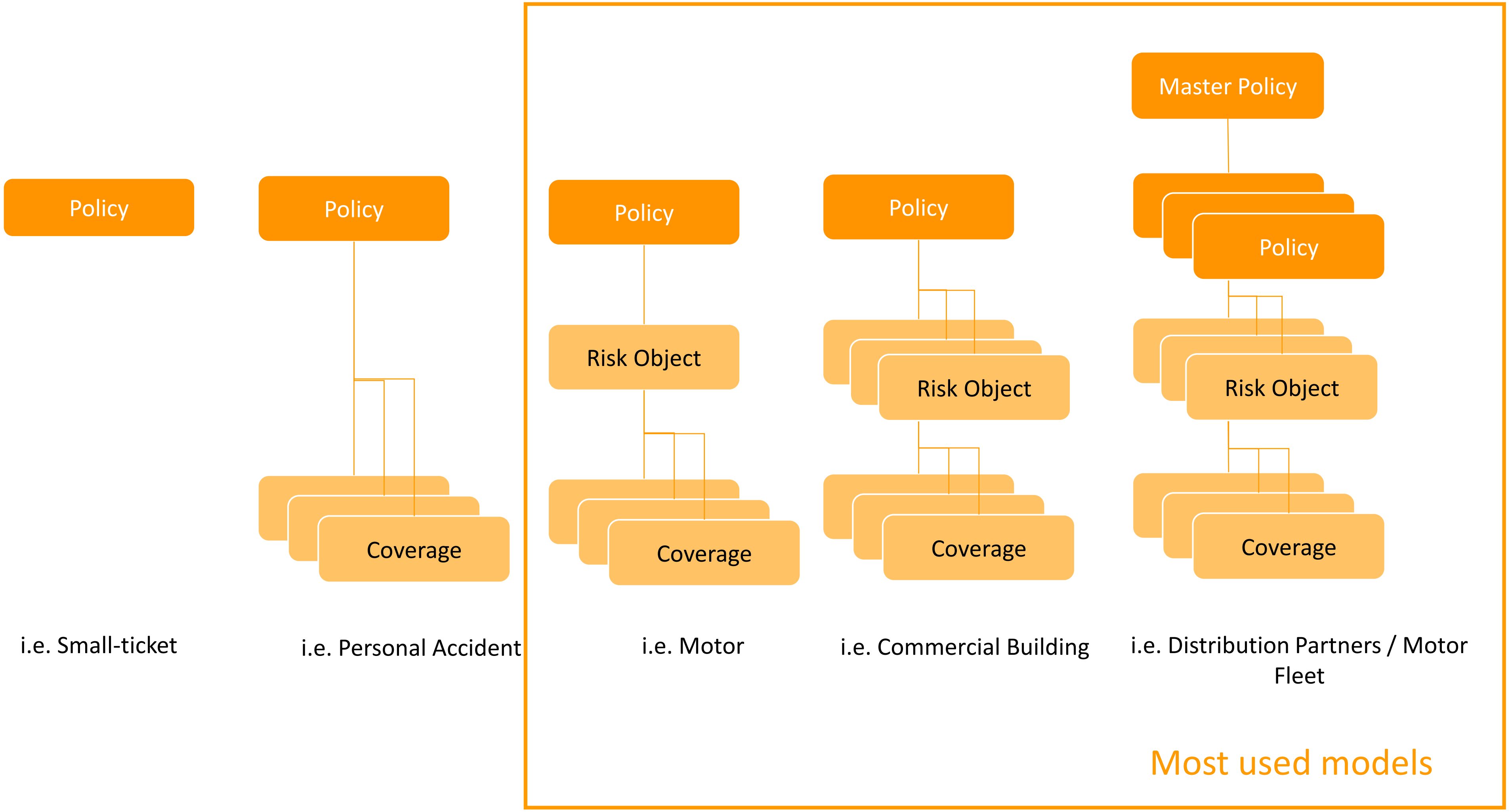
PUT /v1/policies/{identifier}



- Both of these models are part of the IBSuite backward compatibility guarantee

- IBSuite's real data base model is not shared by IBA and should not be used in any implementation
  - IBA proprietary data
  - **NOT backwards compatible** – Using it risks introducing causing defects with future software releases

Standard IBSuite Product Models



i.e. Small-ticket

i.e. Personal Accident

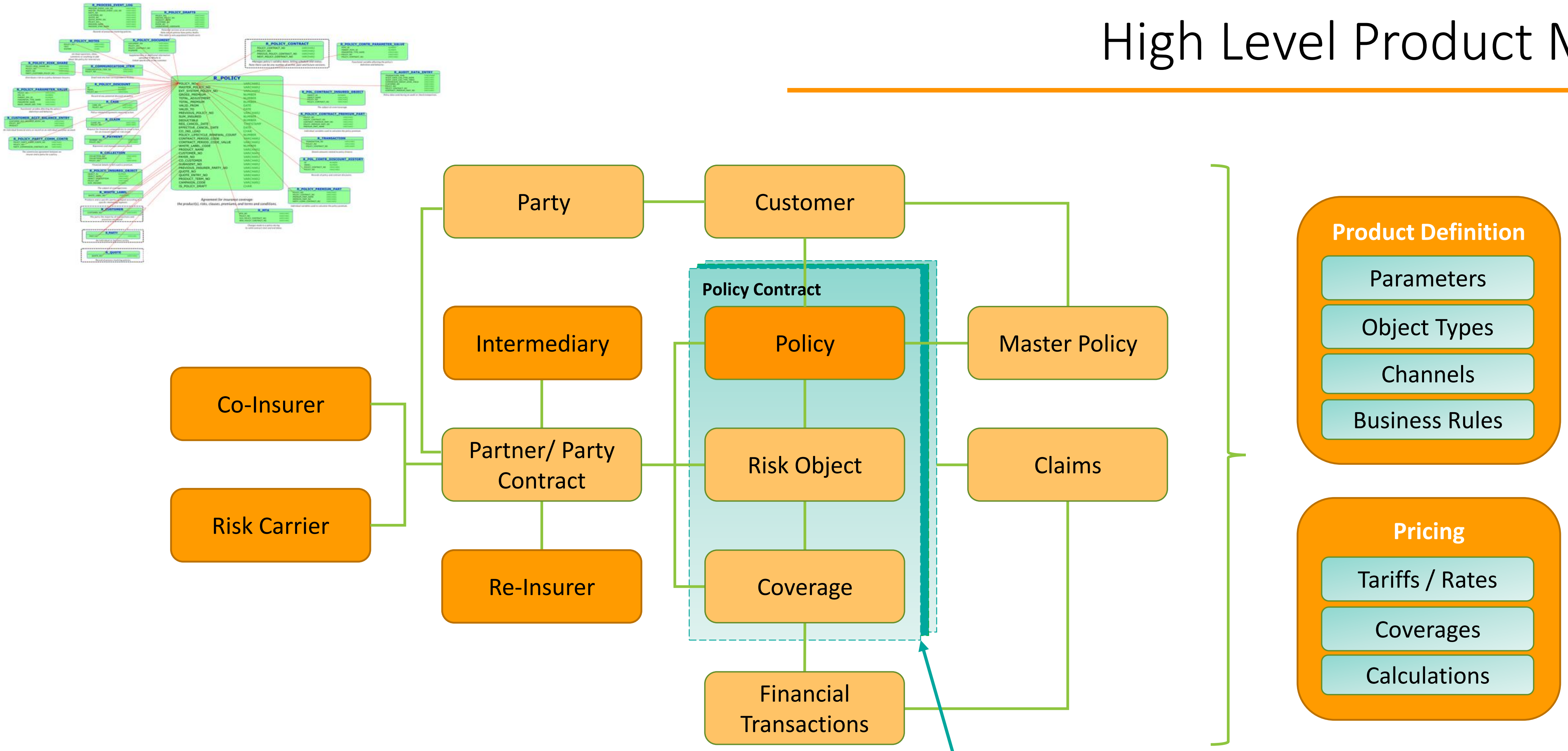
i.e. Motor

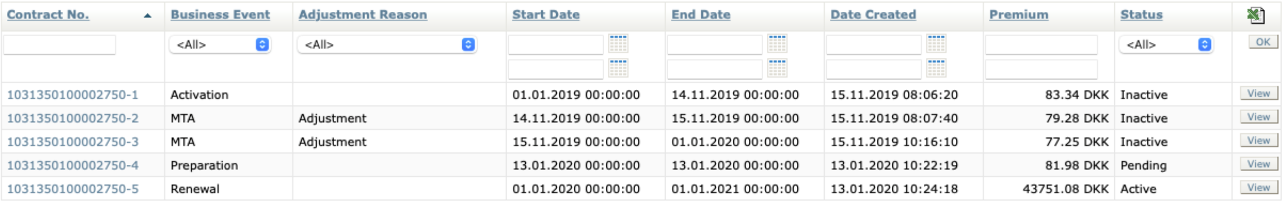
i.e. Commercial Building

i.e. Distribution Partners / Motor Fleet

Most used models

High Level Product Model





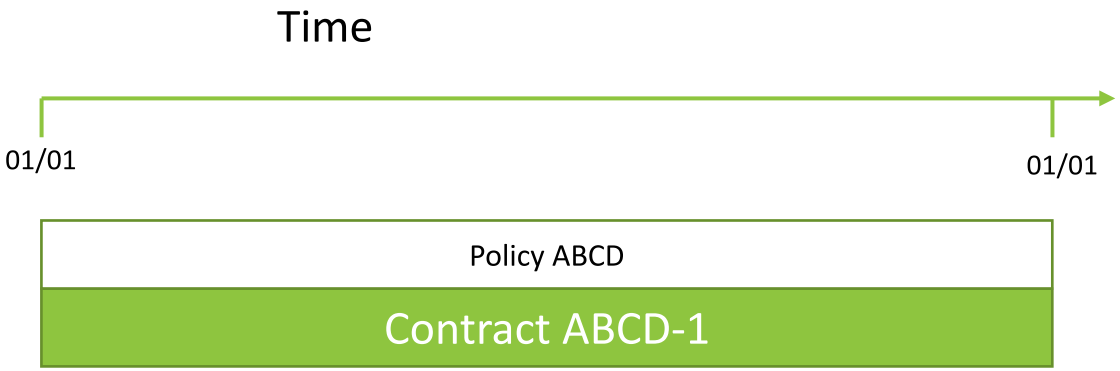
Contract No.	Business Event	Adjustment Reason	Start Date	End Date	Date Created	Premium	Status	
	<All>	<All>					<All>	OK
1031350100002750-1	Activation		01.01.2019 00:00:00	14.11.2019 00:00:00	15.11.2019 08:06:20	83.34 DKK	Inactive	View
1031350100002750-2	MTA	Adjustment	14.11.2019 00:00:00	15.11.2019 00:00:00	15.11.2019 08:07:40	79.28 DKK	Inactive	View
1031350100002750-3	MTA	Adjustment	15.11.2019 00:00:00	01.01.2020 00:00:00	15.11.2019 10:16:10	77.25 DKK	Inactive	View
1031350100002750-4	Preparation		13.01.2020 00:00:00	13.01.2020 00:00:00	13.01.2020 10:22:19	81.98 DKK	Pending	View
1031350100002750-5	Renewal		01.01.2020 00:00:00	01.01.2021 00:00:00	13.01.2020 10:24:18	43751.08 DKK	Active	View

Policy contract is a full version of the policy and details exactly what is covered and when.

Policy contracts and MTA's

New Business

Original Policy


Time

01/01

01/01

Policy ABCD

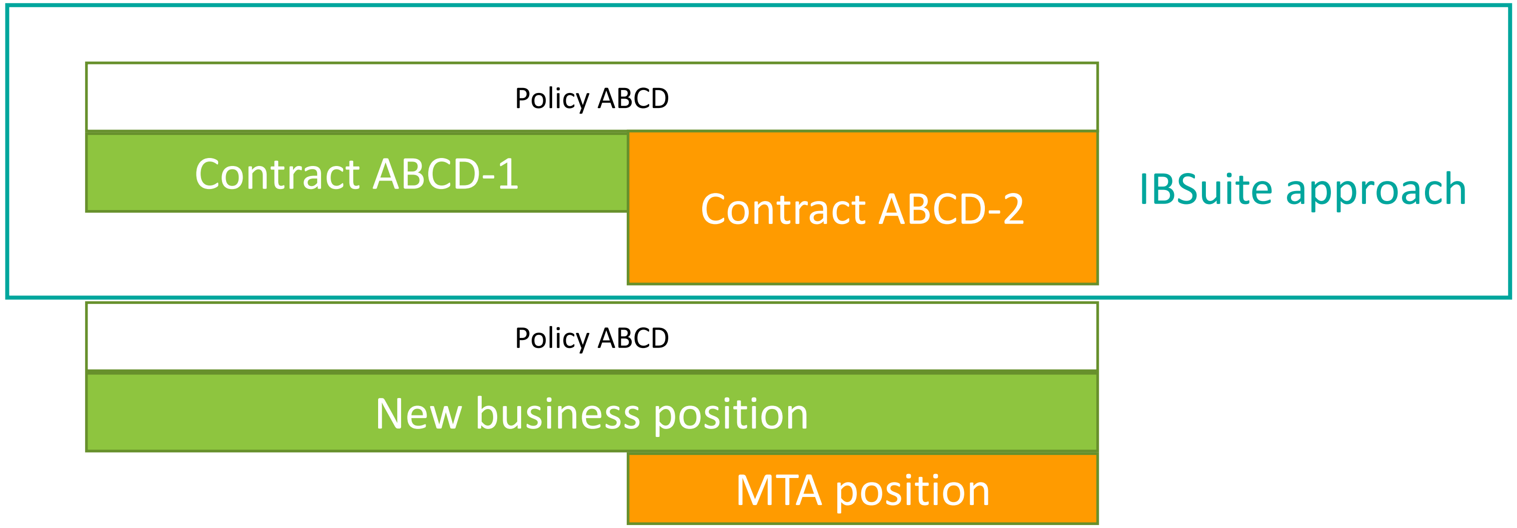
Contract ABCD-1

Financial view

Contract 1
+120

MTA

Two ways to think of an MTA


Policy ABCD

Contract ABCD-1

Contract ABCD-2

IBSuite approach

Policy ABCD

New business position

MTA position

Contract 1
+120
-60

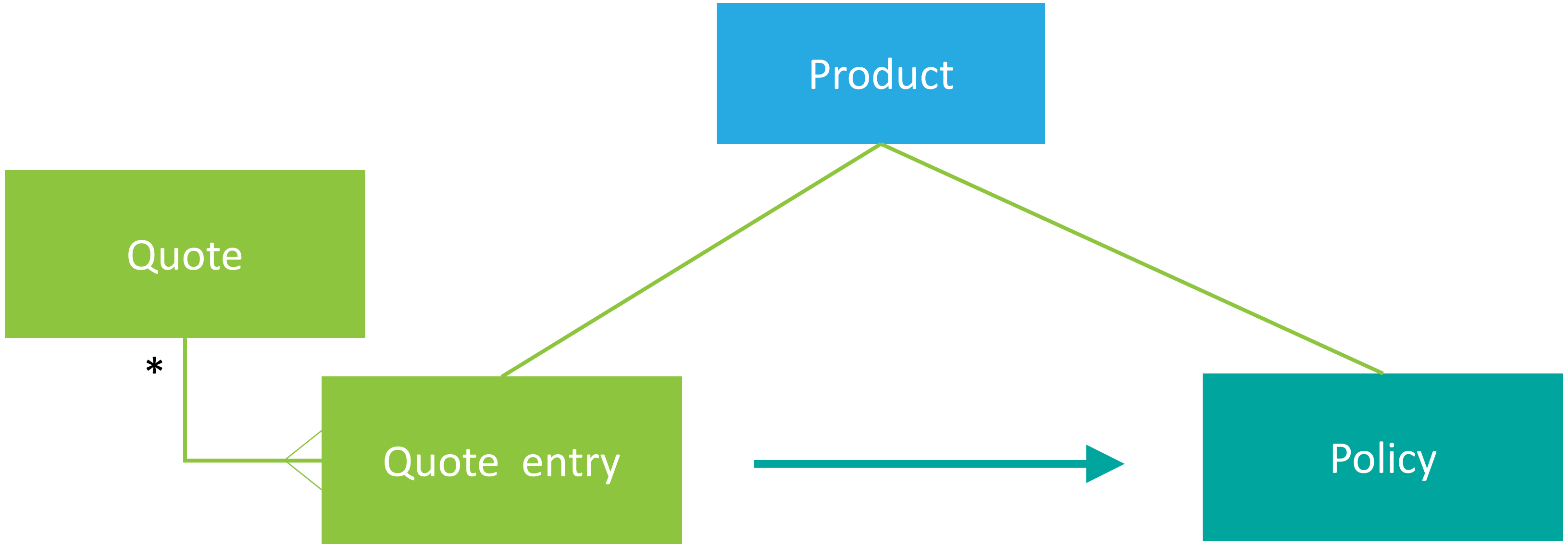
Contract 2
+72

Total
+132

Quote – Quote Entry

Can have multiple quote entries within a quote. For example:

- Different options
- Package offering covering multiple products



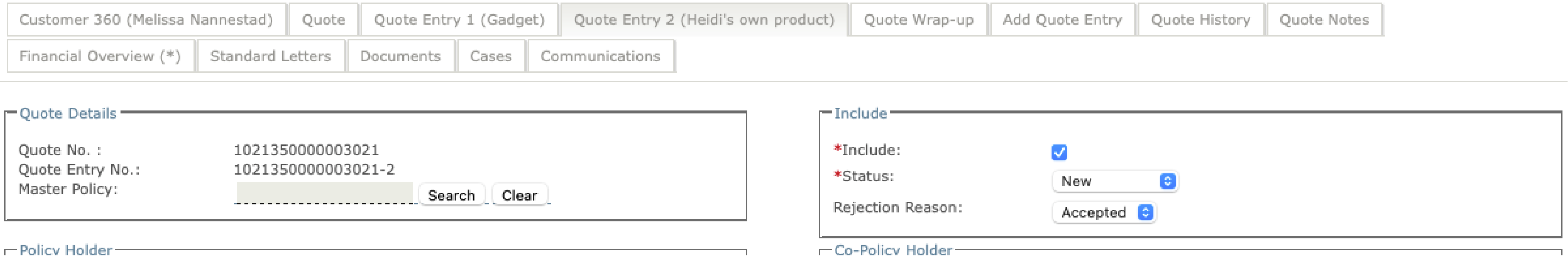
Product

Quote

*

Quote entry

Policy



Quote Details

Quote No. : 1021350000003021
Quote Entry No.: 1021350000003021-2
Master Policy:

Include

*Include: ☑
*Status: New
Rejection Reason: Accepted

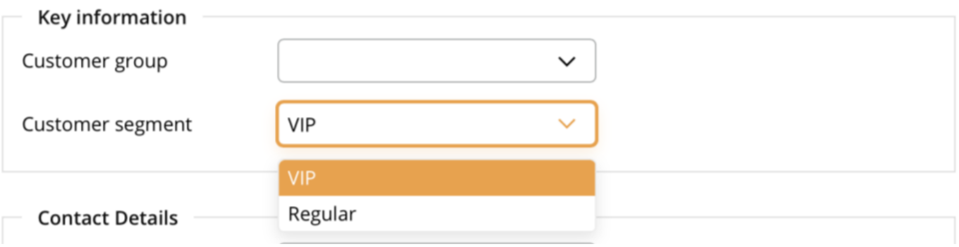
Policy Holder

Co-Policy Holder

Solution to dynamic data models in IBSuite

IBSuite has a fixed physical data model used by all customers so how can we have customer specific data structures?

- Flex fields
  - Many entities in IBSuite have flex fields
  - These are spare fields that serve no purpose in core IBSuite
  - Used by customers to capture data that is relevant to them
  - Label on screen is configurable



Key information

Customer group

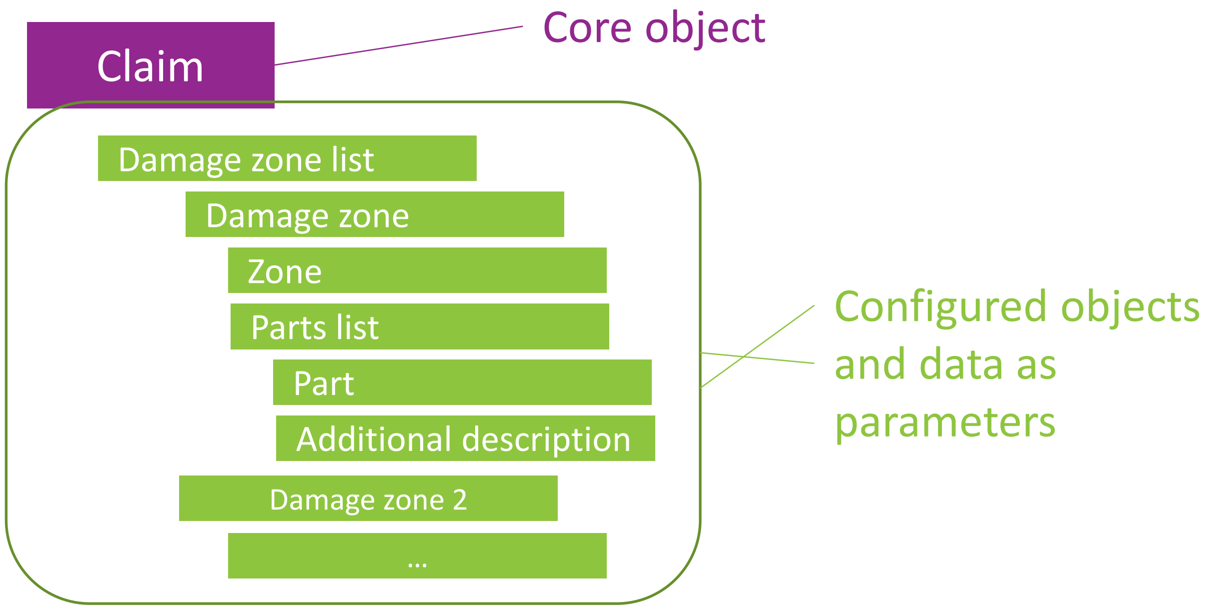
Customer segment: VIP

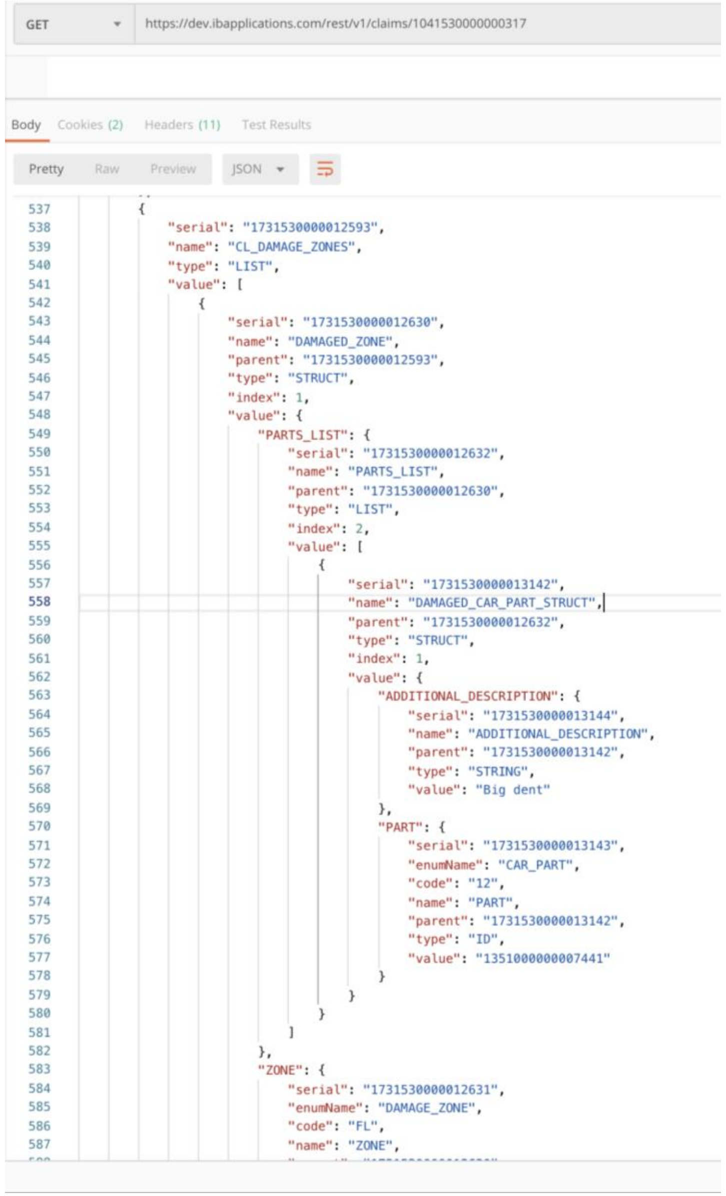
Contact Details

- Custom tables
  - Similarly, the IBSuite data model includes whole tables that serve no purpose in core IBSuite
  - These have some standard fields like custom_type and foreign keys to customer/policy/claim, but the rest of the columns have generic names like String1
  - Using the custom_type allows many logical entities to be stored in the same “physical” table

Solution to dynamic data models in IBSuite

- Parameters
    - Most flexible solution to allow multi-level data structures to be configure
    - Most common use in configuring the product specific data for quotes, policies and claim
    - Automatically available on the relevant REST API
    - Example from PoC





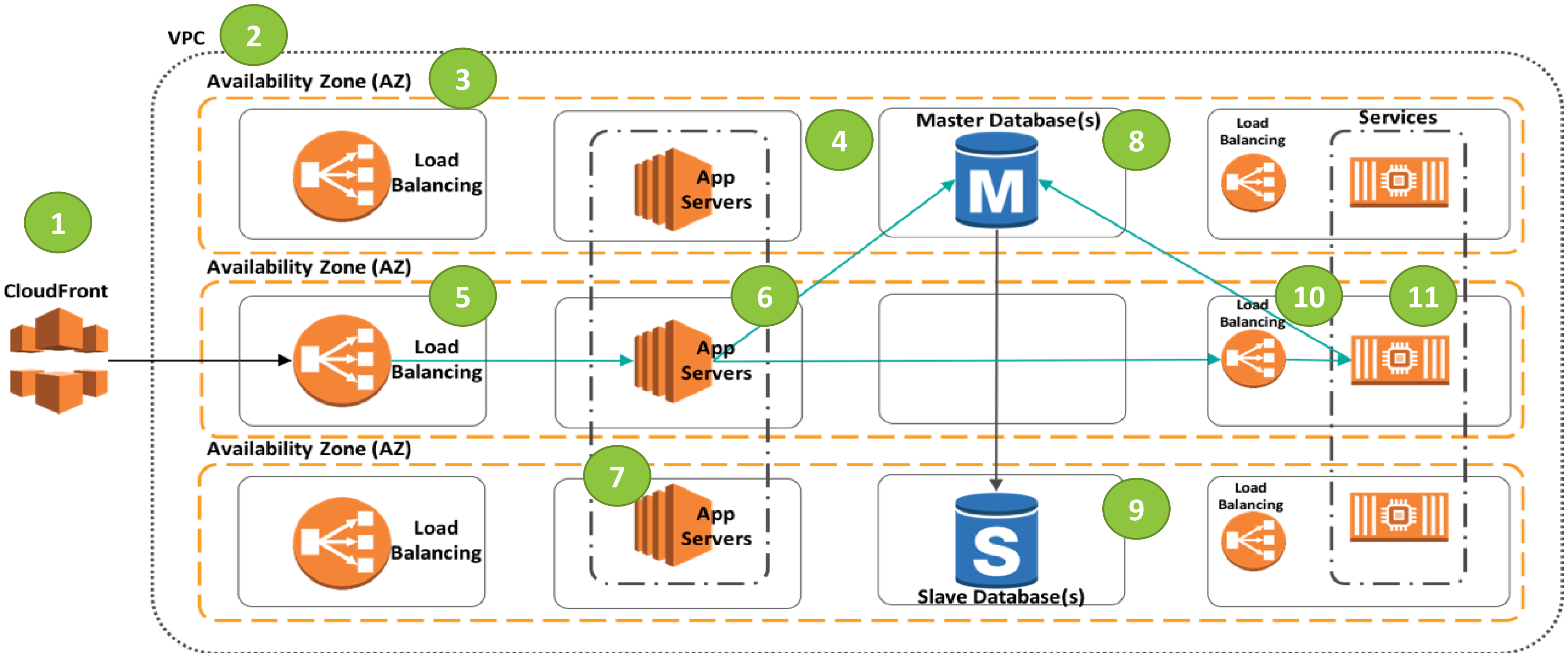
https://dev.ibapplications.com/rest/v1/claims/1041530000000317

```
        {
            "serial": "1731530000012593",
            "name": "CL_DAMAGE_ZONES",
            "type": "LIST",
            "value": [
                {
                    "serial": "1731530000012630",
                    "name": "DAMAGED_ZONE",
                    "parent": "1731530000012593",
                    "type": "STRUCT",
                    "index": 1,
                    "value": {
                        "PARTS_LIST": {
                            "serial": "1731530000012632",
                            "name": "PARTS_LIST",
                            "parent": "1731530000012630",
                            "type": "LIST",
                            "index": 2,
                            "value": [
                                {
                                    "serial": "1731530000013142",
                                    "name": "DAMAGED_CAR_PART_STRUCT",
                                    "parent": "1731530000012632",
                                    "type": "STRUCT",
                                    "index": 1,
                                    "value": {
                                        "ADDITIONAL_DESCRIPTION": {
                                            "serial": "1731530000013144",
                                            "name": "ADDITIONAL_DESCRIPTION",
                                            "parent": "1731530000013142",
                                            "type": "STRING",
                                            "value": "Big dent"
                                        },
                                        "PART": {
                                            "serial": "1731530000013143",
                                            "enumName": "CAR_PART",
                                            "code": "12",
                                            "name": "PART",
                                            "parent": "1731530000013142",
                                            "type": "ID",
                                            "value": "1351000000007441"
                                        }
                                    }
                                }
                            ]
                        },
                        "ZONE": {
                            "serial": "1731530000012631",
                            "enumName": "DAMAGE_ZONE",
                            "code": "FL",
                            "name": "ZONE",
```

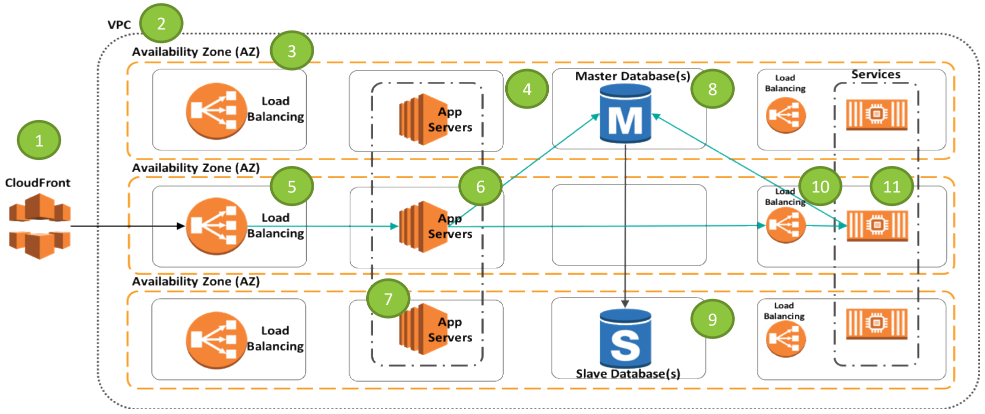
Infrastructure

- Key point it is a SaaS solution so infrastructure is purely an IBA problem
- Infrastructure explained here but for interest only

Detailed Data Flow



Detailed Data Flow



IBSuite works in a global infrastructure model, supporting multiple regions, however, for the readability of this document the eu-west-1 region is in focus.

Placed in eu-west-1 region the solution is deployed in 3 availability zones (data-centres) running in an active-active setup. Each AZ is in separate locations, separate internet uplinks, different flood-planes and different parts of power-grid. Test- and production instances are deployed in a similar way and undergoes same security levels. However, test- and production is deployed in separate AWS accounts and each environment in an isolated VPC (separated network) and no environment has any access to other environments.

1. CloudFront + WAF (Web Application Firewall). Global infrastructure with DDoS protection, advanced caching and Web Application Firewall.

2. VPC in AWS Region – eu-west-1 – Dublin, Ireland. A VPC is private, non-public IP space (VLAN). Each environment is in its own VPC with an allocated /16 network.

3. Availability Zones – we are using all three available in eu-west-1 for maximum availability in the setup

4. Separate subnets and route tables for each infrastructure layer. Only the DMZ subnet can route traffic to and from the internet. Application subnets can reach internet via NAT, database subnets cannot reach internet in any way.

5. Application Load Balancer from AWS. Capacity scales automatically with load and health-checks ensure traffic is only routed to active servers. SSL is terminated here, before traffic is passed on in to the VPC. Firewalled using Security Groups to only allow traffic from AWS CloudFront IPs.

6. Autoscaling group for Application servers. Based on load, the number of servers will in- or decrease to ensure the solutions scales to any load.

7. Application servers – Tomcat 8 on AWS Linux 2 LTS (CentOS variant). Servers are fully scripted and does not allow even SSH access. Security Group ensures that traffic is only accepted from Application Load Balancers.

8. Database is Oracle 12c on AWS RDS. RDS is a fully managed database service. Data is encrypted at REST using the PCI compliant AWS KMS service. Encrypted backups are moved to S3 which in a single region offers 99.999999999% durability – and automatically replicated to a secondary AWS region (eu-central-1) in Frankfurt.

9. Database Slave – operated by RDS. Hot-standby in a separate AZ. In case of Master failure DNS will switch traffic to Slave instance and automatically create a new master. Replication lack is monitored and normally below 50 ms.

10. Internal Load Balancer only available within the VPC – allows for scalable and available plugins.

11. IBSuite plugins running in Docker on AWS Fargate – a scalable service for Docker. Each service is allowed to scale based on load and container instances are spread out over the available AZ's automatically. Where needed containers can access Oracle.

INSURANCE BUSINESS APPLICATIONS

THE PROVEN, NEXT-GENERATION CLOUD INSURANCE PLATFORM

https://ibapplications.com - info@ibapplications.com
https://www.linkedin.com/company/insurance-business-applications