Confidential

BIAN Webinar

BIAN as a functional language - the journey (so far!) toward fully compliant Domain Driven landscape.

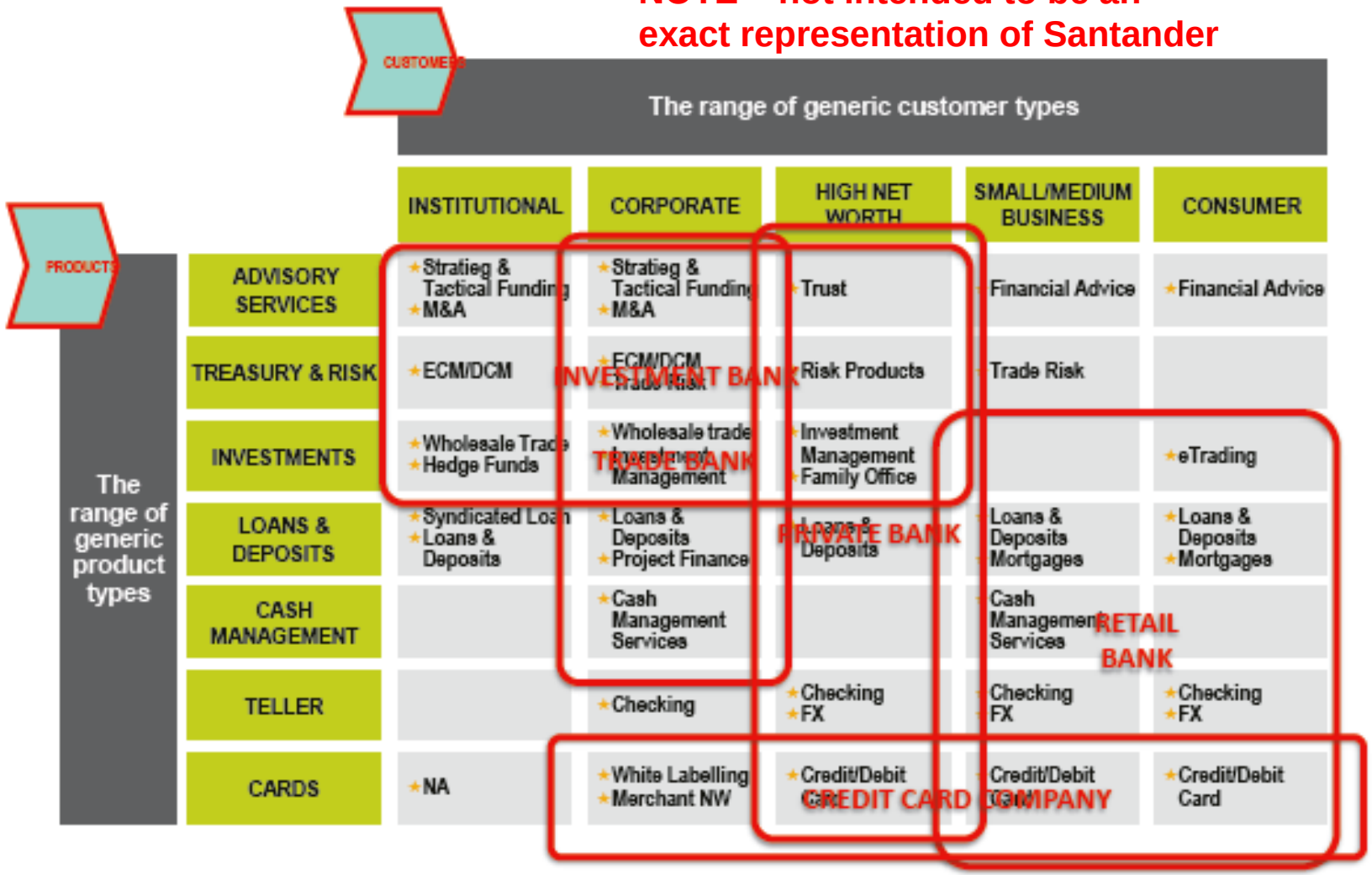
21st October 2020

View of Generic Banking Institution

BIAN Banking Industry Architecture Network

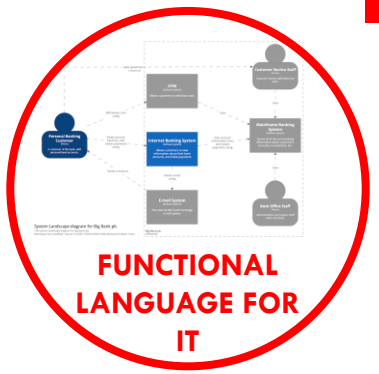
NOTE – not intended to be an exact representation of Santander

Santander is a complex global, multi-entity organization.
For purpose of demonstrating, this is a an example of the generic global bank view across the customer and product dimensions



Examples of product segmentation and "bank type" coverage

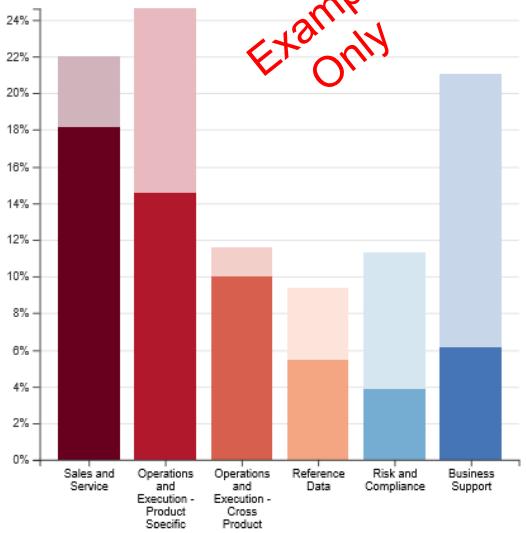
Example BIAN Use Case 1

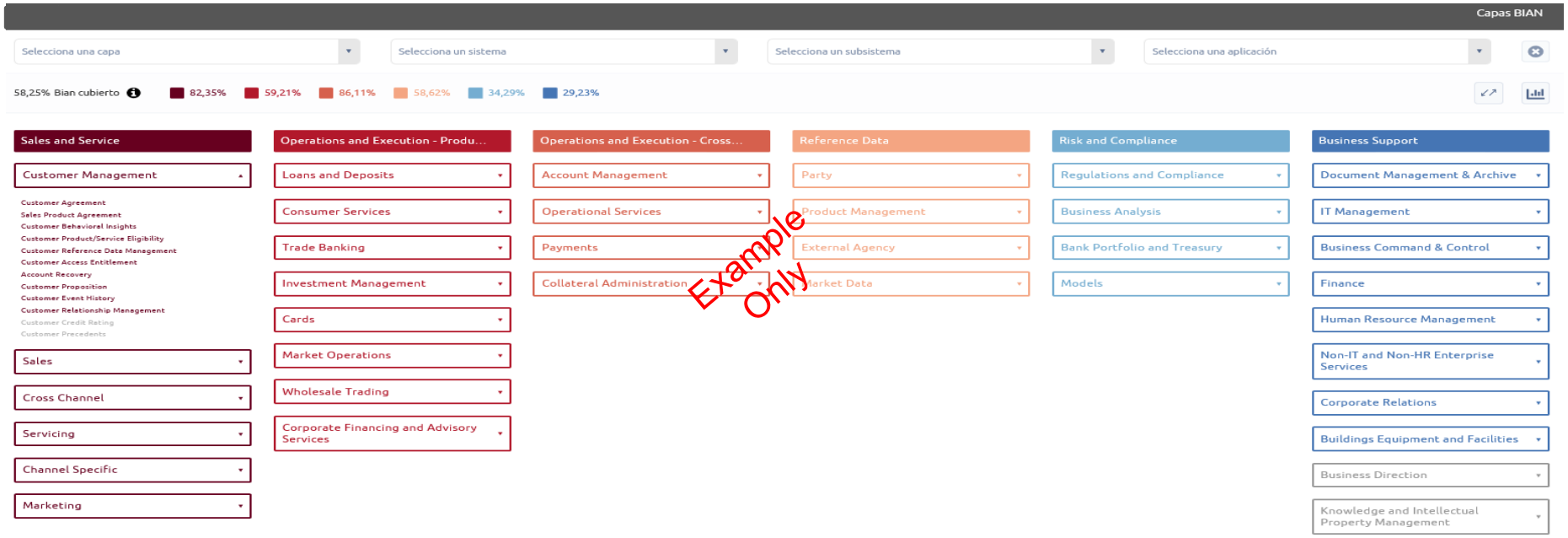


BIAN AS A FUNCTIONAL LANGUAGE TO ORGANIZE IT LANDSCAPE

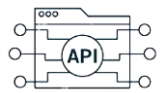
- In a complex global, multi-entity organization such as Santander Group, BIAN provides value as a common functional language to organize and manage the Applications Portfolio.
- Key IT portfolio tools are being updated to include BIAN Service Landscape as a new dimension to manage our assets.
- This also provides a view into opportunities such as simplification of the application landscape, reducing costs, decommissioning.

BIAN Coverage





Example BIAN Use Case 2



API DESIGN

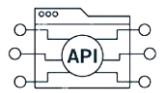
Domain Driven Design

BaaS - API FUNCTIONAL REFERENCE FRAMEWORK

- A reference architecture framework has been defined in order to provide common standards across the group.
- API Functions are published in the Intranet API Portal
- Enables more efficient identification of required APIs, also preventing inadvertent duplication.
- Each API is classified under its corresponding BIAN Service Domain.
- API Design: Each API must expose functions from only one BIAN Service Domain.
  - Also recognises, in some cases, a service domain may have more than one API
- An API can delegate responsibility to another API, matching the Service Domain model of delegating responsibilities to other service domains. This is hidden from the consumer.

Example BIAN Use Case 2

BIAN Banking Industry Architecture Network

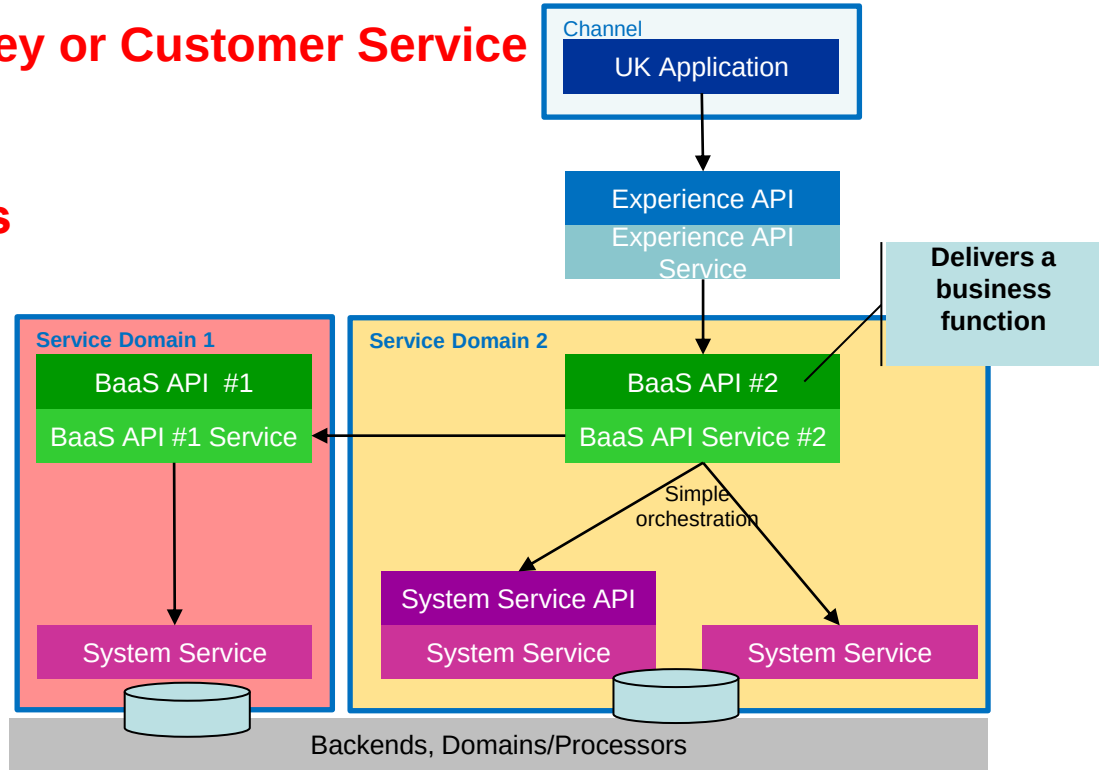


API DESIGN

Domain Driven Design

BaaS - API FUNCTIONAL REFERENCE FRAMEWORK

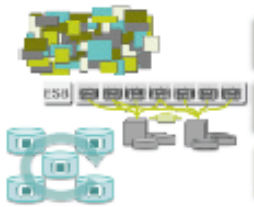
- Banking as a Service (BaaS) APIs can be reused and have business logic that is common for many applications according to its BIAN Service Domain (Capability) which sets the logical boundaries and scope of functionality and data (I.e. Domain Driven Design)
- BaaS APIs are Touchpoint, Customer Journey or Customer Service capabilities
- Each BaaS API must exclusively manage it's own data, for the whole lifecycle of that data.



Direction of travel towards service orientated fully compliant landscape

BIAN
Banking Industry Architecture Network

	Type 1. Direct to Core	Type 2. Wrapped Host	Type 3. Component Architecture
Definition	The API routes direct to the core system providing the service. Intermediate channel based access control and 'buffering' is required	Integrating service middleware – a service bus – 'wraps' the host systems. The service bus can offer various host access mitigation capabilities/enhancements	The host services are implemented as loose coupled microservices with complex interactions supported by sophisticated connective middleware
API Service Description	Read only or simple 'atomic' update transactions supported by a single host system. The solution is likely to be host application specific	Enhanced 'simple access' services aligned to established standards. Wrapping may enhance service capabilities and some hosts may support more complex exchanges	Support for flexible and complex interactions involving multiple business activities and processing/decision chains
Examples	◆ Retrieve a balance/account statement ◆ Reference a product/service directory ◆ Initiate a payment	Message conforms to industry standards (e.g. ISO20022) ◆ Retrieve a balance/account statement ◆ Reference a product/service directory ◆ initiate a payment ◆ Customer on-boarding/offers	◆ Prospect on-boarding and origination ◆ Customer dispute/case resolution ◆ Customer relationship development/up-sell/cross-sell campaigns ◆ Third party service integration
Business Drivers	Provide application based access to an established/existing type of customer exchange	Provide application based access with a high degree of standards alignment. Mask/augment host/legacy system limitations.	◆ Support sophisticated interactions ◆ Support new business models ◆ Support for 3rd party integration ◆ Leverage advanced technolgies/architectures



Direct to Core

Wrapped Host

Microservice Architecture

BIAN Aligned