Table of Contents

new-page

Here's the updated list of figures without the page numbers:

Table of Diagrams

\nnew-page\nHere's the structured content from the image, organized by main topics and subtopics:

1. Conventional Process Oriented Models vs. Component Business Models

- Process oriented models represent business activity as a linked sequence of work tasks
- Suitable for systems that automate actions as a repeating workflow (e.g., factory production line)
- Well-suited for transaction processing in banking, especially for automation and straight through processing (STP)
- Component business models are more representative for interactive/collaborative workflows and systems integrating advanced analytics for decision making

2. BIAN's Challenge in Selecting a Banking Component Model

- Need for a model suited to service oriented design
- Goal: Isolate comprehensive collection of discrete (non-overlapping) business functional components
- Define service exchanges between operational 'building blocks'
- Ensure scope/role of each conceptual building block and associated service exchanges are 'canonical' (consistently interpretable by all industry participants)

3. BIAN's Component Model: "Asset Leverage" Model

- Empirical rather than theoretical technique
- Process:
  1. Define candidate component
  2. Test in practice
  3. Model behavior using real world scenarios
  4. Confirm and refine definition
- BIAN members spent years considering business events to define "Service Domains" covering most banking activities

4. Two Aspects of the Asset Leverage Model

a. First Aspect: Business Assets

- Things of value or 'assets' possessed/accessed by the business

- Can be tangible (buildings, computers) or intangible (relationships, knowledge/knowhow, goodwill)

b. Second Aspect: Asset Control and Manipulation

- Day-to-day execution involves controlling/manipulating assets

- Uses IT control systems to enhance/extract commercial value

- Control types categorized by "functional patterns"

Examples:

- Computer is operated

- Customer relationship is managed

- Market knowledge is analyzed to extract insights

This structure provides an overview of BIAN's approach to modeling banking components, emphasizing the shift from process-oriented to component-based models, and detailing the "Asset Leverage" model used by BIAN.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents a concept diagram and explanation of a Service Domain in the context of banking operations. Here's a structured breakdown of the content:

A Service Domain is responsible for "doing something to something"

Figure 3 - A Service Domain Does Something to Something

Components of the Service Domain:

1. **Bank Resources**:

- The Bank has resources, tangible & intangible...

- Includes: Reputation & Capital, Servicing Capacity, Production Capacity, Employee Commitments

2. **Service Domain Actions**:

- Manage
- Administer
- Operate
- Design
- Fulfill

3. **Outcomes**:

- ...that it controls or leverages in various ways...
- Track
- Analyze
- Process

4. **Purpose**:

- ... in order to create commercial value.

Key Points:

- Components/Service Domains are generic and stable over time
- The Service Domain is represented by a hexagonal shape with "Life Cycle" and "Local State" components

Explanation:

A BIAN Service Domain is defined to be responsible for implementing one pattern of control to instances of just one type of asset. The Service Domain does this from start to finish for the complete life cycle, as often as required by the business.

Unlike the production-line process implementation where the steps in the processing logic are tightly linked together from end-to-end for one specific purpose, the BIAN Service Domain functional components can be implemented in a manner that allows them to be more loosely coupled together. They can be engaged in any suitable combination and sequence as necessary to address many different business events, in parallel if needed. These design properties are revisited in more detail with examples in the next section.

This approach allows for flexibility and adaptability in banking operations, enabling the service domains to handle various business scenarios efficiently.\n\nend-of-page\n\n

---

\nnew-page\nThe BIAN approach applies a different model view of business activity in order to define canonical (standard) business functional partitions – the BIAN "Service Domains"

Figure 4 - Process Vs Component Model View

This figure compares two approaches:

1. Traditional business models:

- Process oriented

- Business activity viewed as a series of linked decisions and actions

- Design usually assumes access to a common 'shared' view of all processing data

- Illustrated with a flowchart-like diagram connected to a shared database

2. BIAN specific type of service:

- Based design to isolate discrete and autonomous business functions

- Connections between components use a common business vocabulary, but each 'encapsulates' its own processing data

- Illustrated with interconnected components sharing a central "Shared Message Vocabulary"

The Service Domains can represent concentrated centers of specific business expertise or capabilities that can be assigned. When appropriate they can be directly associated with responsible organizational units of the enterprise

A brief aside: BIAN's original intent in selecting an approach was to help eliminate the complexity in the bank's application portfolio to improve interoperability within banks by defining a service oriented architecture (SOA). It is a convenient coincidence that the same BIAN Service Domain functional partitions define highly encapsulated and autonomous components that are well suited to implementation in the highly distributed cloud/container technical environments that are being increasingly adopted with the support of APIs today. Indeed, without a robust business partitioning approach such as that used by BIAN, it is extremely difficult to develop highly distributed systems of any scale mostly because handling shared business information quickly becomes excessively complicated.

2.3 Component Vs Process Business Designs – an example for comparison

The difference between a more conventional process model view and the component type of model can be demonstrated using the simple example shown below for processing a mortgage offer made to a customer. In the process model on the left the workflow is broken down into ever finer grained actions that can then be programmed as an (partially) automated workflow. The component view on the right however simply defines a linked collection of specialized business functions (components).\n\nend-of-page\n\n

---

\nnew-page\nThe image presents a comparison between two different model views for a mortgage application process: a conventional process model and a service-centered design using BIAN Service Domains. Here's a breakdown of the content:

Figure 5 - Process Vs Component Model of a Mortgage Application

1. Conventional Process Model:
   - Shown on the left side of the image
   - Represents a linear, step-by-step process
   - Includes steps like:
     - Gather customer & loan details
     - Negotiate and agree terms

- Check ability to fund the loan
- Check credit worthiness
- Value the property
- Get underwriting decision
- Uses a shared database for storing and accessing information

2. Service-Centered Design (BIAN Service Domains):
- Shown on the right side of the image
- Represents interconnected components or services
- Includes components like:
- Product Directory
- Customer Data Management
- Underwriting Decision
- Customer Offer Processing
- Credit Administration
- Collateral Asset
- Document Services
- Mortgage Fulfillment

The text below the image explains the differences and implications of these two approaches:

1. Conventional Process Model:
- Streamlined and automated end-to-end sequence of actions
- May be difficult to amend for different business situations
- Example: pre-approval and checks before property is found

2. Service-Centered Design:
- Specialized business elements interact as needed
- No specific sequence inferred

- Flexible to support different processing sequences/variations

- Can support many different business events with little change

3. Additional Benefits of Service-Centered Design:

- Flexibility to support various collaboration patterns

- Potential for re-use of operational capabilities

- Service Domains can be designed to operate autonomously

- Can be engaged in many different business events

The key takeaway is that while the conventional process model is streamlined, it may lack flexibility. In contrast, the service-centered design offers more adaptability and potential for re-use across different business scenarios.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Componentization vs. Service Enablement

Componentization

- **Definition**: Componentization defines discrete business capabilities. It involves identifying specialized business functional partitions that can be defined in isolation and reused in various combinations to support different behaviors.

- **Benefits**:

- Eliminates operational redundancy.

- Enhances business operations.

- Supports highly distributed/decoupled operating models and their supporting systems when appropriate.

Service Enablement

- **Definition**: Service enablement is an implementation option that can be applied to the components to support a more flexible and dynamic operating model.

- **Considerations**:

- Introduces significant overheads in terms of service latency and performance orchestration.

- Particularly suitable for front office applications where collaboration and flexibility to mix and match capabilities are needed.

- For back office transaction processing, where activities are more structured and fixed-sequence, component connections tend to be more rigid. Here, throughput and performance are prioritized over flexibility.

Transactional Systems

- The component view remains useful in transactional systems to better decouple activities and streamline, optimize, or batch processes.

- Standard ‘service aligned’ interfaces can be used for importing/exporting data.

- However, transaction processing exchanges within and between systems are typically better implemented as ‘hard-wired’ point-to-point connections rather than service-enabled connections.

The BIAN Service Domain – Some Example Definitions

- BIAN membership has defined a collection of over 320 Service Domains using real-world banking examples.

- The designs are based on a mutually exclusive and collectively exhaustive (MECE) hierarchical classification of over 250 banking asset types combined with nineteen distinct patterns of behavior (Functional Patterns), which include:

- Operate

- Manage

- Analyze

- These design principles are fully defined in the next section of this guide, with examples used to describe the core architectural properties of the Service Domains in a more general context.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Service Domains

A “Service Domain” represents the combination of one pattern of behavior being applied to instances of one type of asset. The Service is defined to be responsible for fulfilling this business function for the lifecycle and for as many times as might be necessary.

Examples of Service Domains

1. **Systems Operations**

- Responsible for operating a computer facility (such as a production application platform) from the time it is turned on to the time it is switched off. This occurs for as many operating sessions as might be required over the production life of the technology.

2. **Customer Relationship Management**

- Responsible for the set-up, maintenance, and execution of a customer relationship plan from the time the relationship is first established through to its termination. This applies to every active customer relationship that it manages.

3. **Market Analysis**

- Responsible for consolidating market information/research and applying whatever type of analysis might be required at any time to develop particular market insights when requested.

Operational Behaviors of Service Domains

Every Service Domain applies a single pattern of behavior to instances of a type of asset for the complete lifecycle. The descriptions above show that Service Domains may manifest a wide range of operational behaviors depending on their business role.

- Some Service Domains may act on only one or a very few concurrent instances, whereas others may handle many millions of instances at the same time.

- For example, Systems Operations handles a single production system operating at a time, while Customer Relationship Management may manage many active relationships concurrently.

- The lifecycle for some Service Domains may be extremely long, while others may ‘churn’ quickly.

- For instance, the Service Domain “Product Design” may deal with product design specifications that can persist for many years, whereas the Service Domain “Customer Contact” handles call center conversations with customers that may often last just a few seconds.\n\nend-of-page\n\n

---

\nnew-page\nHere's the structured content from the image, organized according to the titles and subtitles:

A Service Domain represents the responsibility to implement a specific pattern of commercial behavior to instances of a specific type of asset

Figure 6 - The Functional Patterns and Asset Decomposition Hierarchy

[The image shows a diagram illustrating the concept of Service Domains, Asset Types, and Functional Patterns]

here is a transcription:

Transcription of the Image

A Service Domain represents the responsibility to implement a specific pattern of commercial behavior to instances of a specific type of asset

---

Assets are identified using a "MECE" decomposition hierarchy

The Bank has resources, tangible & intangible...

...that it controls or leverages in various ways...

...in order to create commercial value

---

19 standard patterns of commercial behavior have been identified and refined in use

1. Production Capacity
2. Reputation & Capital
3. Servicing Capacity
4. Customer Relationships
5. Knowledge & Knowhow
6. Employee Commitments

One Asset Type

One Pattern

The Service Domain uses a 'Control Record' to keep track of each time/instance it executes its responsibilities for a complete lifecycle

1. Manage
2. Operate
3. Administer
4. Design
5. Fulfill
6. Analyze
7. Track
8. Process

This transcription aims to keep the same structure and flow as the image for clarity.

Asset Decomposition

- Assets are identified using a "MECE" decomposition hierarchy

- The diagram shows different asset types: Production Capacity, Regulation & Compliance, Customer Relationship, Knowledge & Knowhow, Employee/Commitments

Functional Patterns

- 19 standard patterns of commercial behavior have been identified and refined in use

- The diagram shows a table with various functional patterns such as Manage, Fulfill, Process, Track, etc.

The Service Domain uses a 'Control Record' to keep track of each time/instance it executes its responsibilities for a complete lifecycle

The top level of the asset decomposition and nineteen general functional patterns

BIAN has used these to develop Service Domains as shown schematically above. Note that it is not necessary for implementers to be overly familiar with these terms. But when considering the role/purpose of a Service Domain it can help to focus on its functionality by considering the main behavior it implements (its associated functional pattern) and the subject it acts upon (the asset type) as represented by the Service Domain's 'control record'.

The Service Domain's Control Record

Every BIAN Service Domain specification defines a single associated operational artifact called its "control record". This is simply the mechanism it uses to control or trace the execution of one occurrence of it performing its business role for a complete lifecycle. The control record is an important feature of a Service Domain for development because it contains most of the key information that is likely to be referenced and exchanged in service operations between the Service Domains. As already described, the control record reflects the combination of the type of asset acted upon and the functional pattern being applied. Expanding on the three Service Domain examples already mentioned:

Systems Operations

- Its control record is the system operating session that defines the schedule of actions taken and captures any details of operational\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Overview of Service Domains

Events and Tasks

- Events that occur or tasks that are processed throughout a complete operating session cycle for the production system.

Customer Relationship Management

- **Control Record**: The customer relationship plan/charter.

- **Purpose**: Sets out the goals, relationship development actions, and records key events and performance over the complete duration of an individual customer relationship, which often spans many years.

Market Analysis

- **Control Record**: The market analysis perspective developed.

- **Details**: Includes the information referenced and the analysis algorithms applied to develop specific insights for one analysis request.

Operating Control and Reporting

- General operating control and reporting information maintained by every Service when implemented as a service center.

- The significant majority of the accessed information by other Service Domains through service exchanges is extracted from one (or more) of its active control record instances.

Service Domains

- Each Service Domain supports a discrete and non-overlapping business partition that collectively covers all banking activities.

- The discrete business information governed and exchanged through service operations (as defined by their control records) together defines a type of high-level information architecture.

Domain Service Operation Action Terms

- The last key design feature of the Service Domain relates to the Service Domain's service interface.

- BIAN has defined a standard set of "action terms" that characterize the type of service operation exchanges that a Service Domain can perform.

Standard Action Terms

- The complete collection of action terms is described later, but for a quick preview, some BIAN actions include:

- **Initiate**

- **Retrieve**

- **Evaluate**

Examples of Service Operations

- **Systems Operations**:

- **Control Record**: The system operating session.

- **Example Service Operation**: A call to “initiate” an operational service or feature, such as initiating an ATM withdrawal transaction from a teller device on the ATM network.

- **Customer Relationship Management**:

- **Control Record**: The customer relationship plan.

- **Example Service Operation**: A call to “retrieve” an assessment of performance to plan for a particular customer relationship, maintained in their associated control record instance.\n\nend-of-page\n\n

---

\nnew-page\n# Market Analysis

Overview

Market Analysis is controlled by its perspective, with an example service operation being a call to "evaluate" a specific aspect of the market. This service call results in the creation of a new control record instance that includes:

- Input market reference information

- Applied analysis algorithms

- Findings of the analysis

These findings are then returned to the requester.

Service Domain Encapsulation

Definition

The Service Domain manages all activities throughout the complete life cycle. It encapsulates complex processing logic and business information, allowing other Service Domains to access only the externally visible information contained in its offerings.

Example: Customer Relationship Management Service Domain

The Customer Relationship Management (CRM) Service Domain contains:

- Detailed analysis of product and service utilization
- Performance metrics
- Records of meetings
- Product utilization
- Relationship development ideas

However, only a higher-level extract of this information is provided through its external services. For instance, a request for details on a customer's profitability does not require an understanding of the intricate aspects of relationship management; it only needs a simple performance report derived from the comprehensive data.

Information Structure

The encapsulation of processing logic and information results in two distinct types of business information within the Service Domain:

1. **Detailed Internal Logic and Data**:
   - This is specific to the Service Domain and supports its processing.
   - It can adopt any suitable format/schema and is typically maintained in an internal database.

2. **Shared/External Business Information**:
   - This is a common business vocabulary that represents a limited subset of the managed business information.
   - It is essential for the services offered to calling Service Domains.

BIAN Business Object Model (BOM)

The BIAN Business Object Model (BOM) is a continually developed data model that captures the shared business vocabulary. It provides:

- Definitions of control record content

- Specifications for the exchanged business information passed by services between Service Domains

This model ensures consistency and clarity in the communication of business information across different Service Domains.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Precision in Business Vocabulary

Definition and Importance

- Shared business vocabulary encompasses concepts that span the banking business.

- It typically includes general and widely recognized banking terms.

- This vocabulary must conform to a common definition/specification.

Variability in Precision

- The precision of definitions varies based on the nature of the information and its handling.

- Shared business information requires precise definitions, especially in financial contexts.

- Financial data consists of elements/attributes with exact specifications, such as:

- Amount

- Currency

- Processing dates

- Involved accounts/counterparties

- Other types of information may have less inherent precision and can be represented in unstructured formats, such as:

- Customer credit evaluations

- Product preferences

- Market performance analyses

Service Exchange and Precision Requirements

Machine vs. Human Readers

- The required precision for service exchange differs based on the end user:

- **Machine**: Requires precise, machine-readable data elements that are fully specified in advance.

- **Cognitive/Human Reader**: Capable of handling complex and variable data, interpreting and extracting relevant information from unstructured presentations.

APIs in Banking

Definition and Evolution

- The term API (Application Programming Interface) traditionally referred to structured machine-to-machine connections.

- APIs are now increasingly defined to support screen-based dialogues where:

- The provider or consumer, or both, are cognitive/human.

Importance of Distinction

- This distinction is crucial when relating BIAN service operations to underlying system APIs, as will be explained with examples later in this guide.\n\nend-of-page\n\n

---

\nnew-page\n# Summary

Objective

The objective of this section is to clarify that the BIAN representation of business as functional components that can be service-enabled is fairly novel and distinct from conventional process-based designs. It explains that aligning development to the component boundaries can have significant beneficial architectural implications, but that aligning to the specifications requires an investment of effort in interpreting the BIAN specifications.

Service Domain

This section describes some of the main properties of the foundational building block of the BIAN model – the Service Domain. It explains that each Service Domain's role is to be the application of a

pattern of behavior (functional patterns) to instances of a type of business asset. It also outlines that BIAN defines standard types of service (action terms) offered by a Service Domain to others requiring access to its services.

Next Steps

In the next section, the different BIAN design artifacts are described in far greater detail.\n\nend-of-page\n\n

---

\nnew-page\n# The BIAN Design Artifacts

Overview

The BIAN Design Artifacts section describes the available BIAN design artifacts. These include:

1. **The BIAN Service Landscape**
2. **BIAN Service Domain**
- a) Functional Patterns
- b) Asset Types & Right-sizing Service Domains
- c) Control Records
- d) Behavior Qualifiers
- e) Service Operations and Action Terms
- f) Service Domain First Order Connections
- g) Service Domain Information Profile
- h) The Figure “8” Diagram
3. **Business Scenarios (examples)**
4. **Wireframe Diagrams (examples)**
5. **BIAN Semantic APIs**
6. **Service Domain Event Triggering (Proposed Extension)**

BIAN Standard

The BIAN standard combines the formal canonical designs of the Service Domains and associated service operations with a wide collection of usage examples that help illustrate the intended working of the Service Domains.

Usage Examples

Technically speaking, the usage examples (presented in the form of 'business scenarios' and 'wireframes') are not part of the formal BIAN standard. They are not intended to be prescriptive but instead serve as archetypal illustrations of how the Service Domains can interact. These examples are useful for architects to model their business requirements using the Service Domains.

The available usage examples (scenarios and wireframes) also provide technical leads for architects, offering a starting point for a design that can be adapted and extended to meet more detailed business requirements for systems development. Architects can modify the available BIAN scenarios and wireframes to suit their own needs, provided they do not amend the role or purpose of any involved Service Domains.

Access to BIAN Artifacts

Published BIAN artifacts can be found on the BIAN public site (BIAN.org).

The BIAN Service Landscape

Approximately 320 Service Domains have been identified so far by the BIAN membership, organized in a reference framework called the BIAN Service Landscape. The landscape groups Service Domains based on large business areas and further narrows them down into more defined business domains. The layout is intended to assist in understanding the relationships and structure of the Service Domains.\n\nend-of-page\n\n

---

\nnew-page\nThe image and accompanying text describe two different formats for organizing Service Domains in the BIAN (Banking Industry Architecture Network) framework:

1. Matrix Landscape Format:

- Organizes Service Domains into groups based on business areas (columns) and business domains (blocks)

- Based predominantly on technical properties of the Service Domains

- Provides a more technical view

2. Value Chain Landscape Format:

- Offers a different classification of business areas and business domains

- Depicts a more enterprise organizational view

- Designed to be more intuitive for use by business practitioners

- Generally preferred by business practitioners as it relates more readily to typical enterprise operating models or organizations

Key points:

- Both landscape layouts contain the same collection of Service Domain components

- They differ in how they group and arrange these Service Domains due to dissimilar business areas and business domain definitions

- The Value Chain layout is considered more intuitive for business users

The image (Figure 7) visually represents these two formats side by side:

- Left side shows the Matrix layout

- Right side shows the Value Chain layout

- Colored boxes and arrows highlight the relationships between Business Areas, Business Domains, and Service Domains in both layouts

The document then introduces the next section (3.2) which will cover BIAN Service Domain Specifications.

This information is particularly relevant for IT and Business consultants in the banking industry, as it provides insight into different ways of organizing and visualizing banking service architectures, catering to both technical and business-oriented perspectives.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about the BIAN (Banking Industry Architecture Network) model, specifically focusing on the BIAN Service Domain and Functional Patterns. Here's a structured breakdown of the content:

1. BIAN Service Domain

- The building block of the BIAN model

- Main challenge: Framing business requirements using collections of Service Domains exchanging services, rather than conventional end-to-end process oriented design

- BIAN Business Scenario and BIAN wireframes are helpful design artifacts for this transition

2. Service Domain Functional Patterns

2.1 Definition:

- A Service Domain is a conceptual functional design mappable to a major application module

- Core purpose: Controls the application of commercialization behavior to instances of an asset type

- Handles processes from start to finish as required by the business

2.2 BIAN Approach:

- Defines 19 general commercialization behaviors called "functional patterns"

- Every BIAN Service Domain applies one of these functional patterns to instances of its assigned asset type

3. The BIAN Functional Patterns

A table is provided listing the functional patterns and their descriptions:

- CREATE (Make plans, design, solutions):

- DIRECT: Define policies, goals & objectives and strategies for an organizational entity or unit

- MANAGE: Oversee the working of a business unit, assign work, manage against a plan and troubleshoot issues

- ADMINISTER: Handle and assign the day to day activities, capture time worked, costs and income for an operational unit

- DESIGN: Create and maintain a design for a procedure, product/service model or other such entity

- DEVELOP: To build or enhance something, typically an IT production system. Includes development, assessment and deployment

- PROCESS: Complete work tasks following a procedure in support of general office activities and product and service delivery functions

- INITIATE (Process work, operate tooling for support and product/service delivery):

  - OPERATE: Operate equipment and/or a largely automated facility

  - MAINTAIN: Provide a maintenance service and repair devices/equipment as necessary

  - FULFILL: Fulfill any scheduled and ad-hoc obligations under a service arrangement, most typically for a financial product or facility

  - TRANSACT: Execute a well bounded financial transaction/task, typically involving largely automated/structured fulfillment processing

  - ADVISE: Provide specialist advice and/or support as an ongoing service or for a specific task/event

  - MONITOR: To monitor and define the status/rating of some entity

  - TRACK: Maintain a log of transactions or activity, typically a financial account/journal or a log of activity to support behavioral analysis

- REGISTER (Catalogue and enroll):

  - CATALOG: Capture and maintain reference information about some type of entity

  - ENROLL: Maintain a membership for some group or related collection of parties

- EVALUATE (Perform test, checks and analysis):

  - AGREE TERMS: Maintain the terms and conditions that apply to a commercial relationship

  - ASSESS: To test or assess an entity, possibly against some formal qualification or certification requirement

  - ANALYSE: To analyse the performance or behavior of some on-going activity or entity

- PROVIDE:

  - ALLOCATE: Maintain an inventory or holding of some resource and make assignments/allocations as requested

The image is labeled as "Figure 8 - BIAN Functional Patterns with Descriptions".\n\nend-of-page\n\n

---

\nnew-page\nBased on the image provided, I can structure the content as follows:

Functional Patterns and Generic Artifacts in BIAN Service Domains

For developers, the functional pattern provides a clear indication of the core functionality provided by the Service Domain. To make functional patterns more easily interpreted, a 'generic artifact' is associated with each functional pattern. This simply translates the action of executing the behavior into something more concrete (basically converting the behavior from verb to noun form). The generic artifact describes the type of artifact/document that is used/produced when tracking the actions of the service domain as it completes its execution from start to finish.

The BIAN Functional Patterns and their Generic Artifacts

A table is presented with two columns: Functional Pattern and Generic Artifact. The table lists 20 pairs of functional patterns and their corresponding generic artifacts:

1. DIRECT - Strategy
2. MANAGE - Management Plan
3. ADMINISTER - Administrative Plan
4. DESIGN - Specification
5. DEVELOP - Development
6. PROCESS - Procedure
7. OPERATE - Operating Session
8. MAINTAIN - Maintenance Arrangement
9. FULFILL - Arrangement
10. TRANSACT - Transaction
11. ADVISE - Advice
12. MONITOR - State
13. TRACK - Log Record
14. CATALOG - Directory Entry
15. ENROLL - Membership
16. AGREE TERMS - Agreement

17. ASSESS - Assessment

18. ANALYSE - Analysis

19. ALLOCATE - Allocation

Service Domain Asset Types & Right-sizing Service Domains

The other key facet that defines the functional scope of the Service Domain is the asset type that it acts upon. An asset in this context is something of inherent value/purpose that the bank owns or at least has some influence over. Assets can be tangible things like computers and buildings or they can be far less tangible things such as relationships, knowledge and knowhow.

[Note: There is a handwritten annotation "data" with an arrow pointing to the last sentence, possibly indicating that data is another example of a less tangible asset.]\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about BIAN's asset classification and Service Domains. Here's a structured breakdown of the content:

1. BIAN Asset Classification

- BIAN breaks down assets into 250-300 discrete asset types
- The classification includes the "capacity to perform" a business action as an asset type
- Examples: service customers' needs, fulfilling banking products and services

2. Top Level BIAN Asset Types (Figure 10)

The asset types are categorized into four main groups:

a. Production Capacity

- Product Delivery Capacity
- Distribution Capacity

b. Central Resources

- Enterprise Resource
- Finances

c. Relationships

- Workforce
- External Parties

d. Intellectual Property

- Knowhow
- Knowledge

Each category is further broken down into subcategories and specific asset types.

Full break-down of asset types:

Top Level BIAN Asset Types

Asset Types

- **Production Capacity**
  - e.g., fulfillment, distribution & sales
- **Product Delivery Capacity**
  - Financial Facility
    - Credit/debit card
    - Current account
    - Corporate current account
    - Credit line
    - Line of credit
  - Financial Transaction
    - Payment
    - FX/Exchange
    - Loan
    - Syndicated loan
    - Leasing

  - Deposit
  - Bank document
  - Letter of credit
  - Bank guarantee
  - Factoring
  - Stock lending/Repo
- Traded Instrument
  - Price/quote
  - Trade
  - Matching
  - Confirm
  - Clearing
  - Settlement
  - Custody
- Financial Service
  - Brokered product
  - Trust
  - Remittance
  - Advisory
  - Public offering
  - Private placement
  - Asset management
  - Direct debt
  - Investments
  - Project finance
- **Product/Service**
  - Operations
  - Accounting
  - Commissions

  - Payments
  - Valuations
  - Underwriting
  - Collateral
  - Fraud detection
  - Transaction
  - Transaction condition
  - Transaction selection
  - Booking
  - Documents
- **Instrument Maintenance**
- **Product Inventory**
  - Materials
  - Tokens
- **Production Analysis**
  - Production cost
  - Counterparty risk
  - Composite position
  - Gap
  - Compliance
  - Settlement

Distribution Capacity
- Bank Facility
  - Operations
  - Trading floor
  - Dealing position
  - Branch
  - Location

  - Teller/Position
  - Call center
  - Servicing position
  - IVR
  - ATM
  - E-Branch
  - Social Network
- Information Provider
  - Agency
  - International standards
  - Credit market
  - Financial market
  - Financial market news
  - Financial market analysis
- Financial Market Access
  - Price/quote reporting
  - Deal capture
  - Deal matching
  - Deal reporting
  - Deal booking
- Information Switch
- Customer Servicing
  - Orders
  - Issues
- Channel Operations
  - Cross channel services
  - Branch network
  - PBX/VRU
  - ATM network

  - E-Branch network
  - Internet gateway
  - Correspondence
- Physical Distribution
  - Warehousing/storage
  - Archive services
  - Distribution fleet
- Sales & Marketing
  - Campaign
  - Advertising
  - Prospect
  - Customer
  - Selling
  - Surveys

Central Resources

e.g., H.Q, fixed assets & capital

Enterprise Resource

- Head Office
  - Board of Directors
  - Business Model
  - Corporate Communications
- Reputation/Recognition
  - Goodwill
  - Brand
- Business Development
  - Sales & marketing
  - Advertising

  - Prospect campaign portfolio
  - Internal campaign portfolio
- Support Services
  - Legal
  - Audit
  - Process
  - Product
  - Security
  - Finance/AML
  - Human resources
  - Training
  - Product
  - Applications development
  - Systems production
- Enterprise Analyses
  - Segment
  - Product
  - Customer
  - Branch
  - Channel
  - Property

Finances
- Capital
  - Cash
  - Fixed assets
- Committed Cash Flows
  - Income/revenue
  - Expenses

- Assets & Liabilities
  - Balance-sheet
  - Off balance-sheet
- Financial Analyses
  - Market risk
  - Counterparty
  - Capital

Buildings & Equipment
- Equipment
  - Office equipment
  - Fleet/distribution
  - Consumables
- Building
  - Offices
  - Ops centers
  - Branches
- Systems
  - IT Platforms
    - Development
    - Environment
    - Processing & Storage
    - Communications
    - Deployment
    - Certification
    - Installation

Relationships

e.g., employees & partners

Workforce

- Business Units
  - Profit centers
  - Cost centers
  - Project teams
- Employees
  - Directors
  - Managers
  - Staff

External Parties

- Customer
  - Prospect
  - Consumer
  - HNW
  - Corporate
  - Multinational
  - Institutional
  - Counterparty
  - Syndicate
- Partner
  - Supplier
  - Product service provider
  - Broker
  - Custodian
  - Correspondent
  - Agency
  - Investor

- Authority
  - Regulator
  - Auditor

3. A high level decomposition of the Objects
  - Banks have collections of assets they can own or influence
  - Examples: customer relationship, cash, payment facility
  - Assets need to have an associated use or purpose

4. Right Sizing Service Domains
  - Right sizing is crucial for defining 'elemental' Service Domains
  - A bank either needs the entire Service Domain or not at all
  - Service Domains can't be split to adopt only a subset of core functions
  - Elemental Service Domains are necessary for the BIAN standard to be canonical
  - Consistent interpretation in different deployments is important
  - Applying only part of a component's specification may compromise its standard nature

This structure provides an overview of BIAN's approach to asset classification and the importance of properly defining Service Domains in the banking industry.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Right Sizing Technique

The right sizing technique used to define Service Domains is quite complex. It is not necessary for technical leads and architects to learn to apply this technique themselves, as all Service Domains they need should already be fully specified. The technique is described in BIAN architectural guidelines and training materials. An outline of the technique is included for reference as Attachment B to this guide.

Decomposition of Assets

The right sizing technique involves decomposing the types of assets to the lowest level at which they retain unique business context or ownership. Below this level of granularity, functions that act on the assets become more utility in nature.

Example of Classification Decomposition

For example, a classification decomposition of the bank's production systems can be made to the level where individual systems are identified, such as:

- The network

- Contact center support application

- Internal office network

Further decomposition of these systems would start to identify their constituent processing features. In many cases, these features are not uniquely assignable to a single party in the enterprise's organization.

Utility Features

An example system listed above is likely to include some function to register a new entity (which would be a new ATM device, a customer servicing application, and an employee reference respectively for the listed examples). This 'register' is more of a utility feature common to all or at least most systems. Operating an 'entity registration function' would not be uniquely assignable as a business function in the way that operating an ATM network is.

Operational Vs Utility Re-use

There are two different types of re-use that are both important but apply in very different contexts.

Assignable Role

An assignable role performed by a Service Domain – for example, Customer Management and Document Services (which handles the classification and distribution of important documents) represents a re-usable operational function. Operational re-use involves parts of the business using shared services provided by other specialized business units to gain access to their capabilities.

Finer Grained Functions

Finer grained functions making up Service Domains may include recurring functions that define reusable utilities. For example, a frequently performed product pricing function can be implemented as a standard software utility. This utility can then be built into multiple product processing systems. It makes great sense to define and implement solutions for commonly executed functions for consistency and to avoid re-writing code. However, this is a different type of re-use.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Service Domains

- **Definition**: Service Domains define discrete operational business capabilities that can be assigned to a responsible party in the organization and reused by other parts of the business as they undertake different business activities.

Utility Functions

- **Definition**: Utility functions define standard actions/behaviors that can be encoded in reusable code modules (using software procedure libraries, micro-services, etc.). The deployed instances of utility functions execute completely independently of each other.

Important Distinction

- The important distinction between Service Domain operational capability reuse and utility reuse is revisited in more detail in later sections of this guide.

Service Domain Control Records

- **Definition**: The combination of the generic artifact and asset type defines a Service Domain’s record. The control record specification is of particular interest as it comprises the main business information governed by the Service Domain.

- **Content**: It can contain a very diverse collection of information, including all the information needed to control any information that might be referenced and also any information that is utilized by the Service Domain as it completes a full cycle of its work.

- **Example**: An indication of the type of information that might be found in a control record is shown in an example control record for the Party Authentication Service Domain that handles the identity of a customer.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents information about the BIAN Control Record for the Party Authentication Service Domain. Here's a structured breakdown of the content:

1. Figure 11 - Excel Extract of Service Domain Control Record

   This table provides detailed information about various attributes and their descriptions related to party authentication in banking services.

2. BIAN Standard Control Record Information

   - The BIAN standard provides initial control record information definitions for Service Domains.

   - These can be filtered and expanded for specific implementation projects.

   - For object-oriented design, the control record can be considered as a type of 'class'.

3. Control Record Behavior Qualifiers (Section 3.2.4)

   - Early experience using BIAN Service Domain's service operations revealed challenges in accessing control records:

     - Accessing the complete control record in a single service exchange did not always provide a sufficiently narrow business context.

     - This could lead to ambiguous definitions for service operations.

   - Example given: A service operation to 'execute' an action on a customer's current account could have multiple intended uses in different business contexts (e.g., executing a payment or a deposit).

   - The text suggests that a further level of detail breakdown is necessary to address this issue.

The document appears to be a guide for IT and Business consultants in the banking industry, focusing on the standardization and implementation of service domains and control records within the BIAN (Banking Industry Architecture Network) framework.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about BIAN's (Banking Industry Architecture Network) approach to structuring service operations in banking systems. Here's a structured breakdown of the content:

1. Behavior Qualifier Types

- BIAN uses 'behavior qualifier types' to break down the work performed by Service Domains.

- Each functional pattern has a specific behavior qualifier type.

- The behavior qualifier type defines how the pattern of behavior can be subdivided into finer-grained activities.

- It retains the core behavioral characteristics of its associated functional pattern.

- The overall work done by a functional pattern is made up of a collection of the same type of work done by its finer-grained behavior qualifiers.

- This design implements a fractal pattern.

2. The BIAN Functional Patterns, Generic Artifacts and Behavior Qualifier Types

- A table is provided showing the relationship between:

  a) Functional Pattern

  b) Generic Artifact

  c) Behavior Qualifier Type

  d) Example

The table includes 20 rows, each representing a different functional pattern. Some examples:

- DIRECT: Strategy (Generic Artifact), Goals (Behavior Qualifier Type), Increase market share (Example)

- MANAGE: Management Plan, Duties, Relationship development, Troubleshooting

- ADMINISTER: Administrative Plan, Routines, Time-sheet recording

- PROCESS: Procedure, Worksteps, Invoice generation

- OPERATE: Operating Session, Functions, Message capture/routing

- FULFILL: Arrangement, Features, Current account standing order

- TRANSACT: Transaction, Tasks/Steps, FX pricing, market trade, clearing & settlement

3. Application of Behavior Qualifier Types

- A single general behavior qualifier type is associated with each functional pattern.

- The actual behavior qualifiers defined for a Service Domain will be particular/specific to that Service Domain.

- Example: A Service Domain with the functional pattern 'process' has the associated behavior qualifier type 'work steps'.

- The actual work steps for a 'process' Service Domain will reflect its own specific business role.

- For instance, the work steps for the Customer Billing Service Domain would reflect how it processes a customer bill, such as customer invoice generation.

This structure helps in organizing and understanding the various components and behaviors within banking service operations according to the BIAN framework.\n\nend-of-page\n\n

---

\nnew-page\nBased on the image content, I can structure the information as follows:

Service Domain Control Records

BIAN currently breaks down Service Domain control records to define the first level of behavior qualifiers as part of the standard definition. This approach is generally sufficient to define the payload of a discrete service operation unambiguously. However, for Service Domains with extensive information or functional content, solution architects may need to define additional levels of 'sub-qualifiers' to break the control record down further for suitably focused service operations.

Example: Party Reference Data Directory Service Domain

An example of possible sub-qualifiers is shown for the Party Reference Data Directory Service Domain with its functional pattern 'catalog', generic artifact 'directory entry' and behavior qualifier type 'properties'.

The Service Domain Control Record Breakdown

Figure 13 - Party Reference Data Directory Control Record

The image shows how the Service Domain Control Record is broken down using the Behavior Qualifier Type:

1. Service Domain: Party Reference Data Directory
2. Behavior Qualifier Type: Properties
3. Behavior Qualifiers as defined by BIAN:
   - Reference: Properties are general customer reference details
   - Associations: Properties detail the customer's links and associations to other parties of interest
   - Demographics: Properties cover demographic, employment and educational background
   - Bank Relations: Properties capture any bank to customer links/relationships
4. Sub Qualifiers as defined by users (site specific):
   - Reference/Features: Properties relate to customer properties such as SSN, Passport #, Date of Birth
   - Reference/Address: Properties relate to contact details such as home address, email, phone
   - Reference/...: Properties TBD

Important Note

It is crucial that as the control record is broken down, the applicable behavior qualifier type is applied consistently to define the sub-qualifier partitions. These partitions should be defined to be discrete and mutually exclusive & collectively exhaustive (MECE) at each level of decomposition.

3.2.5 Service Domain Service Operations & Action Terms

Every Service Domain offers a collection of service operations and usually consumes or 'delegates' by calling the service operations of other Service Domains as needed to complete its work. This section considers the Service Domain's offered service operations.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Introduction

This guide is designed for IT and business consultants working within the banking industry, focusing on the BIAN (Banking Industry Architecture Network) standards and practices.

BIAN Action Terms

Definition and Purpose

BIAN has defined a general set of “action terms” that characterize the purpose of a service. These action terms are intended to be non-overlapping and to encompass all main types of service exchanges supported by any Service Domain.

Current Set of Action Terms

The current set of BIAN action terms includes definitions and examples that illustrate their application.

Revisions and Updates

Consultants familiar with the BIAN standard will note that the action terms have undergone minor revisions and additions since the release of Service Landscape Version 7.0. These changes are based on early feedback from members implementing the BIAN Semantic APIs.

Alignment with REST Specifications

The updates to the action terms have been required to better align BIAN service operations with REST point specifications, which are described in more detail later in this guide.

Conclusion

This guide serves as a resource for understanding the BIAN action terms and their relevance in the banking industry, ensuring that IT and business consultants can effectively implement and utilize these standards in their work.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

BIAN Action Terms

The BIAN action terms can be grouped into four main categories:

1. **Service Operation Influence**

- Action Terms: activate, configure, feedback

- Description: These terms act on or influence the operation of the Service Domain overall as a service center.

2. **Creation of Control Record Instances**

- Action Terms: create, initiate, register, evaluate, provide

- Description: These terms result in the creation of a new control record instance, triggering a new life-cycle.

3. **Modification of Existing Control Record Instances**

- Action Terms: update, control, exchange, capture, execute, request, grant

- Description: These terms act on an existing control record instance, typically invoking some function and/or changing or updating its state.

4. **Information Updates Subscription**

- Action Terms: retrieve, notify

- Description: These actions obtain or subscribe to information updates for one or more control record instances without changing the state of the instance. This distinction is important for Command Query Responsibility Segregation (CQRS) type deployments.

Service Operations

The service operation defines a service dependency between the Domains. It does not presume any specific choreography or protocol. For example, in implementation, the service exchange could be:

- A one-way exchange of information or an instruction, possibly with a simple acknowledgment.
- A complex iterative dialogue where the request is refined based on exchanged details.
- A response that could be immediate or delayed, requiring monitoring from both the caller and provider.

The BIAN service operation details the exchange of information, which can include instructions and responses, but does not track the actual movement of physical items other than by implicit descriptions (e.g., the movement of physical resources).

Default Service Operations

In defining the service operations for the Service Domains, BIAN has identified sensible combinations of action terms that apply to different patterns. For the 19 functional patterns and 17 action terms, BIAN currently provides a default set of service operations.

- **Note**: These defaults may not apply in all deployments, and there may be practical scenarios where additional service operations are required that do not correspond to the defaults. The mapping serves as a starting point for architects to consider and is also intended to define the service operations reflected in the BIAN Semantic API Portal.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Default Action Terms

Overview

In the context of default mappings within service domains, several interesting patterns emerge that warrant discussion.

Key Observations

1. **Service Domain Control**:

- The action terms that govern the overall Service Domain—namely activate, configure, and feedback—are applicable across all Service Domains, irrespective of their functional patterns.

2. **Creation of Control Record Instances**:

- Among the five action terms that lead to the creation of a new control record instance, only one is relevant to any specific functional pattern. While this is expected, the nature of the control record instance archetype created varies significantly for each of the five applicable action terms.

3. **Active Control Record Actions**:

- Most action terms that interact with an active control record (or a subordinate behavior qualifier) instance can be applied across all functional patterns, with a few notable exceptions.

4. **Retrieve and Notify Action Terms**:

- The retrieve and notify action terms are universally applicable, regardless of the functional pattern. These terms can be utilized at the Service Domain level, as well as at the control record and behavior qualifier levels, to extract or report different types of information.

Conclusion

While it may appear that the same action term yields fundamentally different results across various Service Domains, this variation is primarily due to the distinct operational characteristics inherent to each Service Domain. The consistency in the application of action terms is maintained, but the responses differ based on the underlying operational frameworks.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Operation Specifications

- **Service Operation Specifications**:

- The actual service operation specifications for a Service Domain in the current release of the standard are defined using the applicable action term and optionally a behavior.

- Each service operation’s payload is specified as an organized list of semantics covering the key business information provided and returned for the service.

- The precise format as applied for the BIAN Semantic APIs compliant with the architectural style is set out in more detail later in this section.

- **Historical Context**:

- In earlier releases of the BIAN Standard, the Service Domain service operations were defined with four attribute types: Identifiers, Depiction, Instructors, and Analysis.

- This has since been replaced with a more practical and comprehensive format for the BIAN Semantic APIs.

Service Domain First Order Connections

- **Definition**:

- First order connections for a Service Domain capture any identified service interactions required between it and both calling and called Service Domains.

- Each first order connection defines a service dependency between a single calling and called Domain that uses one available service operation (i.e., one action term and optionally one behavior qualifier).

- **Completeness of Connections**:

- The list of first order connections for a Service Domain as captured in the BIAN standard will never be complete, as the connections are discovered empirically through business activity.

- It is likely that some viable business behaviors may not be anticipated. The known first order connections are useful for architects as they outline the connections required to handle different business requirements and can help define the overall scope and purpose of the Service Domain.

- **Association with Business Events**:

- First order connections can be associated with a (first order) business event. The event serves as the external trigger that causes a call to the Service Domain’s offered services.

- Different Service Domains may call the same offered Service, and each of these represents its own first order connection and associated business event.

- For example, many different Service Domains may request a customer’s current balance using the same ‘retrieve’ service operation offered by the Current Service Domain. However, each call will be for its own specific purpose, representing a unique first order connection.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about service exchanges in banking systems, particularly focusing on "nested" service exchanges. Here's a structured breakdown of the content:

1. Service Domain Interactions

- Service Domains process service requests through internally scheduled activity
- They may delegate actions to other Service Domains
- First order connections capture both offered and delegated service operation exchanges
- No formal link between offered services and delegated service calls is maintained to adhere to SOA encapsulation principle

2. BIAN Business Scenarios

- First order connections are used to assemble BIAN business scenarios and wireframe views
- BIAN aims to capture all first order connections for Service Domains in the standard model
- These connections are discovered through different requirement modeling efforts

3. Second Order or 'Nested' Service Exchanges

- Useful for modeling Service Domains that need to perform roles concurrently
- Particularly applicable to Service Domains handling customer interactions with the bank

4. Example of 'Nested' Service Exchanges

- The image presents a BIAN Business Scenario titled "Customer Initiated Case"
- It shows a sequence of service interactions across different bank functions:
  a. Advanced Voice Service Operations
  b. Contact Handler
  c. Party Authentication
  d. Session Dialogue

e. Servicing Order

f. Current Account

g. Customer Case

- The diagram illustrates three levels of service exchanges:
  - 1st Level
  - 2nd Level
  - 3rd Level

- The scenario includes steps such as:
  - Customer calling the contact center
  - Customer authentication
  - Initiating a service request
  - Selecting a disputed charge processing servicing order
  - Checking transaction history
  - Creating a customer case to handle the dispute

This example demonstrates how different service domains interact in a layered or "nested" manner to handle a complex customer interaction, showing the practical application of the concepts discussed in the text.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Business Scenario Overview

In the business scenario, the Advanced Voice Services Service Domain, which operates the PABX, interacts with the Contact Handler Service Domain. This, in turn, calls the Session Dialogue Service Domain, which subsequently calls the Servicing Order Service to process the customer’s request. As a result of their integrated start/end services, three levels of nesting are required in the scenario.

Service Domain Information Profile

The control record instances already described contain the primary information for developers that is maintained by the Service Domain. Control records are typically accessed by the service operations for most transactional activities.

Information Profile Components

The Information Profile describes the complete make-up of business information governed by any Service Domain when implemented as a stand-alone service center. The information profile consists of:

1. **Service Domain Management Information**

- Information used in the control and management of the Service Domain as a service center, including:

- Local copies of referenced information

- Accessed/delegated service details

- Resource administration

- Service domain activity and performance records

- Offered service definition and service configuration settings

2. **Collective Views and Analyses**

- Collective views and analyses of the collection/portfolio of control record instances, including:

- Usage

- Performance

- Historical analysis as might be required

3. **Individual Control Record Content**

- The content of individual control record instances, further broken down using behavior qualifiers as necessary.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents "The Information Profile at the top level" as described in Figure 17. This profile is structured into several main sections:

1. Service Domain Resource Plans & Activity Records
   - Service Reference/Name
   - Servicing Unit Budget & Organization
   - Servicing Unit Resource & Activity Plans
   - Unit Activity & Performance Analysis Reports

2. Reference Details
   - Reference Data Sources and Update Schedule
   - Reference Data
   - Reference Data Usage

3. Configuration & Status
   - Service Definitions
   - Service Configuration & Status
   - Service Access and Usage Records

4. Work Space Records

   This section contains multiple Control Records

5. Control Record
   - Control Record Portfolio Analysis & Reporting
   - Individual Control Record Content
     - Artifact Reference Details
     - Set-up/Parameters
     - Transaction/Activity Records

The accompanying text explains that when the control record is broken down using the behavior qualifier type, the resulting partitions have the same characteristics as their parent partition, applying a fractal pattern. This property is described as particularly useful because it allows the

action term for a service operation to be applied consistently to the control record, a behavior qualifier partition, or any further sub-qualifier partitions.

The text also notes that the use of the behavior qualifier type provides a mechanism for adding increasing precision in terms of defining the scope of the referenced information within the control record.

This structure and explanation demonstrate how the Information Profile is organized at the top level, providing a framework for understanding and managing service domain information in a banking context.\n\nend-of-page\n\n

---

\nnew-page\nThe Service Domain embeds business information in a Fractal Pattern

Figure 18 - The Fractal Nature of the Information Profile

The image illustrates the structure of a Service Domain, showing how information is organized in a fractal-like pattern. The main components are:

1. Service Domain Resource Plans & Activity Records
2. Reference Details
3. Configuration & Status
4. Work Space Records
5. Control Records (multiple instances)

The diagram shows how these components are further broken down into smaller units, demonstrating the fractal nature of the information structure.

At this stage BIAN has defined Service Domain specific descriptions of the make-up of the control records with their individual behavior qualifiers. This is the current focus for the BIAN membership because it defines the payload of the service operations most widely used in the API specifications for transactional banking activity. For completeness BIAN has also provided general descriptions/checklists for the type of information that can be anticipated covering the operational control and activity/performance analysis of the Service Domain that is used in more general

management and control. All of the information definitions are semantic and subject to the following qualifications and limitations:

- Only Covers Mainstream Behavior – the information definitions relate to the prevailing mainstream functions performed by the Service Domain – they are intended to be indicative such that they support an unambiguous definition of the core role/purpose of the Service Domain and its service operation exchanges. They do not attempt to be exhaustive for example covering regional variations or more advanced/specialized distinctions. Furthermore, all activity considered is 'happy path' so error processing and exceptions are not generally considered

- Only Provide High Level Semantic Definitions – BIAN is a business architecture level conceptual specification and as such its information attributes are defined using fairly high-level semantic descriptions. In some cases, the level\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Overview of BIAN Specifications

Implementation Level Granularity

- The detail provided by BIAN approaches implementation-level granularity.

- Attributes defined by BIAN need to relate to more detailed information structures and definitions by architects/developers.

- Example: BIAN defines an ‘account statement’ with properties like ‘period covered’ and ‘types of transaction included’. This is clear in terms of business requirements but lacks detailed specifications for a physical statement report.

BIAN’s Descriptive Definitions vs. Standard Data Formats

- BIAN information attributes are defined semantically/descriptively.

- BIAN does not provide formal data definitions, ensuring implementation agnosticism.

- Example: The attribute ‘customer reference’ provides a unique reference to a bank customer, which can be interpreted consistently. The realization of this reference in specific implementations is not defined, allowing for flexibility in data standards.

Mapping to ISO20022 & Other Standards

- BIAN does not aim to compete with existing industry standards.

- Current focus is on mapping semantic information attributes to the ISO20022 Business Model.

- BIAN defines its own conceptual object model to cover areas not complete in the ISO model and will map to other existing standards as appropriate.

BIAN Specification Components

The BIAN specification comprises three related information descriptions:

1. **Service Domain Information Model**

- Comprises semantic attributes that make up the Service Domain's information profile, primarily focusing on control record definitions.

2. **BIAN Business Vocabulary**

- Provides descriptions of different information attributes, adopting industry-accepted definitions where available.

3. **BIAN Business Object Model**

- Maps information attributes to conceptual business objects for definitional consistency.

Encapsulation Property of Service Domains

- The encapsulation property results in two overlapping views of business information:

- A high-level conceptual information vocabulary that is passed as the payload of services.

This structured approach provides clarity on BIAN's specifications and their relevance to IT and business consultants in the banking industry.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about the BIAN (Banking Industry Architecture Network) Information Model and Service Domain design. Here's a structured breakdown of the content:

1. BIAN Information Model

- Goal: To provide a complete and consistent set of definitions at a detailed level

- Purpose: Enable unambiguous and consistent mappings to implementation level data standards and physical specifications

- Challenge: Addressing gaps and limitations in existing information/data standards across the industry

- Ongoing Development: BIAN will continually add detail and coverage based on practical experience

- Current Status: Available semantic information definitions may be limited to high level/generic descriptions

2. The Figure "8" Diagram

- Concludes the section on Service Domain design

- Main properties of Service Domain design are illustrated

3. BIAN Service Landscape

- Contains all currently identified Service Domains

- Equated to the Periodic Table of Elements (visual comparison provided)

4. Key Properties of all Service Domains:

a. A discrete business functional partition (not a process step)

b. Peer collection covers all business activity (elemental)

c. Acts as an operational service center

d. Can combine people, procedures & systems

e. Capable of being outsourced (one 'sizing' test)

f. Does 'something' to 'something' for the full life-cycle

g. Handles single or multiple instances for a short or long life-span

5. Visual Representation

- Figure 19 shows the Service Domain Key Properties

- Includes a diagram of the BIAN Service Landscape and its relation to a BIAN Service Domain

This information is crucial for IT and Business consultants in the banking industry to understand the BIAN framework and its approach to standardizing banking architecture and operations.\n\nend-of-page\n\n

---

\nnew-page\nThe document provides an overview of BIAN (Banking Industry Architecture Network) Service Domains and their design elements. Here's a structured breakdown of the content:

1. Service Domain Characteristics
   - All Service Domains have the same basic design pattern
   - They perform an action on something from start to finish as often as required
   - Operational characteristics span a broad range

2. BIAN Design Artifacts
   - Multiple design artifacts link together to provide the overall Service Domain specification
   - Captured in the BIAN 'figure 8' diagram
   - Architects and developers typically don't need to reference most of these detailed artifacts
   - Used within BIAN to generate semantic API specifications
   - Provide high-level service operation descriptions found on the BIAN Semantic API Portal

3. The Figure 8 Diagram
   - Links the Design Elements of a BIAN Service Domain
   - Central element: BIAN Service Domain
   - Connected elements:
     - Functional Patterns
     - Asset Types
     - Control Record
     - Behavior Qualifiers
     - Service Operations
     - Action Terms

4. BIAN Business Scenarios

- Help explain the business role/purpose of Service Domains
- BIAN provides various design artifacts with examples of their use
- A BIAN Business Scenario models the handling of a single business event
- First of the design artifacts to explain Service Domains

The image illustrates the "Figure 8 Diagram" (labeled as Figure 20 in the document), showing how different design elements connect to form a comprehensive Service Domain specification in the BIAN framework.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Business Scenarios in BIAN

Definition of a Business Scenario

A business scenario is a representation of a possible set of service interactions that occur between a collection of Service Domains as they handle a specific event.

Properties of a BIAN Scenario

A BIAN scenario has the following properties:

1. **Bounded**:

- It should have a clearly defined business goal/objective with an associated start and ending position.

2. **Meaningful**:

- It includes sufficient content in terms of the Service Domains and their service operation exchanges to represent a coherent example of business activity for a business practitioner to review.

3. **Non-Prescriptive**:

- It presents a sensible sequence/flow of interactions as an archetypal flow, but this sequence and the thresholds/triggers for service exchanges are not mandatory, just viable examples.

4. **Loose Coupled**:

- Though the scenario may read as a linked sequence of exchanges, this serial coupling is not imposed. A service link shown between two Service Domains in the scenario simply indicates that a service dependency exists between them in the context of this business event. How and when this exchange is implemented and any start/end dependencies are not defined.

5. **Non-exhaustive**:

- A scenario does not attempt to define all required/possible service exchanges. Its intent is to clarify some specific role/behavior of the selected Service Domains by providing an example. For this reason, it is usual when defining business activity in an area of interest to use a collection of several overlapping business scenarios.

6. **Non-redundant**:

- Once a specific exchange pattern has been captured in one scenario within a collection, this pattern can be excluded from the other scenarios for simplicity. It can be cross-referenced if needed to avoid confusion.

Elements Generally Avoided in Business Scenarios

Certain elements can be captured in a business scenario but are generally avoided:

1. **Conditional and Multi-path Flows**:

- Most business scenarios will not include conditional/multi-path flows. If there are different options to be defined, these are usually better captured as multiple scenarios within the overall collection.

2. **Second Order Exchanges**:

- Most business scenarios limit the ‘nesting’ of dependent service calls (i.e., where a called Service Domain is shown to depend on making a further delegated service call to respond). Nesting technically breaches the fundamental principles of encapsulation in service-oriented design.

However, there are situations, particularly when modeling real-time customer interactions, where these properties need to be shown for practical implementation purposes.

Conclusion

The business scenario model is effective as a discussion mechanism to define and clarify requirements with business practitioners. It also helps to clarify the specific roles of Service Domains in the context of business activities.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information about BIAN (Banking Industry Architecture Network) Business Scenarios and their relation to Service Domains. Here's a structured breakdown of the content:

1. Introduction to Business Scenarios
   - Similarities with conventional process models
   - Example of a mortgage application redrawn as a BIAN Business Scenario
   - Key difference: sequence of exchanges is not tightly coupled in the scenario

2. Mortgage application captured as a BIAN business scenario
   - Figure 21 shows an example Mortgage Business Scenario
   - The diagram illustrates:
     - BIAN Business Scenario: Customer Mortgage Application
     - Bank service domains (yellow/orange boxes) across the top
     - Sequence of interactions represented by arrows and circles
     - Explanatory notes on the right side describing the process steps

3. Service Domains involved in the scenario:
   - Customer Offer
   - Party Reference Data Directory
   - Product Directory
   - Credit Administration

- Collateral Asset Administration
- Underwriting
- Document Services
- Mortgage Loan

4. Process flow (as described in the notes):
- Customer contact details captured and offer created
- Customer's credit assessment obtained
- Property details captured and valued
- Underwriting decision made
- Offer and related documents classified and recorded
- Mortgage facility initialized

5. Key aspects highlighted from the example:
- Service Domains (yellow/orange boxes) each have their own dedicated column
- The archetypal service exchange flow runs from top to bottom

The image emphasizes that this BIAN Business Scenario approach allows for a more flexible representation of business processes compared to traditional, tightly-coupled process models. It demonstrates how different service domains interact in the context of a mortgage application process.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Service Domain Interaction

- **Service Domain Activity Duration**:
- The vertical blue box indicates the duration the Service Domain is active in the scenario.

- **Service Exchange/Dependency**:

- The horizontal arrows indicate a service exchange or dependency between the Service Domains. The arrow points to the called Service Domain (i.e., the one providing the service that has been delegated to by the calling Service Domain).

- **Call and Response Representation**:

- A circle at the calling end of the arrow indicates that the exchange represents both the call and the response. This is applicable for all exchanges in this example. More complex scenarios can include nesting of calls with the response coming later in the service flow.

- **Service Operation's Action Term**:

- The purple ellipse on a service exchange arrow includes the service operation's 'Action Term'. This property is fully described later and characterizes the nature of the service call. This notation can also include an additional behavior qualifier field when applicable.

- **Service Exchange Description**:

- The service exchange text describes the purpose of the interaction in general terms in the context of this scenario.

- **Overall Processing Flow**:

- The narrative in the column on the right outlines the overall flow of the processing from start to finish.

Business Scenarios

- **Development of Systems Solutions**:

- When developing a new systems solution or mapping to one or more legacy systems, a set of Business Scenarios is used, each addressing a particular event or business requirement of interest.

- **Number of Business Scenarios**:

- As a guide, fifteen to twenty Business Scenarios might be defined to cover the key requirements of a targeted business area (such as payments or customer servicing). The required number of scenarios clearly depends on the scope and complexity of the application design.

BIAN Wireframe

- **Service Connections**:

- The BIAN Wireframe shows the available (first order) service connections between a collection of Service Domains. A Service Domain may utilize more than one of the services offered by another Service Domain (for example, requesting an action to be performed or simply retrieving status information from the same Domain).

- **Wireframe as a City Map**:

- The wireframe is analogous to a city map that shows the allowed service connections connecting the Service Domains. A business scenario is then one example of a journey that traverses this map using its particular service connections or paths.

- **Dynamic Model View**:

- A business scenario is referred to as a ‘dynamic’ model view because it details the actions taken over time for some event. The BIAN Wireframe, conversely, is a model view of the Service Domains as it depicts their persistent available connections regardless of any timing or specific event/activity.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents a simplified wireframe perspective for a mortgage application scenario, demonstrating how a complex business process can be reorganized into a more straightforward visual representation.

Figure 22 - Simple Wireframe for the Mortgage Application Scenario

The diagram shows two parts:

1. On the left, a complex BIAN Business Scenario chart for Customer Mortgage Application.

2. On the right, a simplified wireframe version of the same process.

The simplified wireframe consists of:

- A central "Customer Offer" node

- Connected to several Service Domains represented by yellow/orange boxes:

- Party (Reference Data Directory)
- Product Directory
- Credit Authorization
- Collateral Asset Administration
- Underwriting
- Document Services
- Mortgage Loan

Each connection is represented by a purple ellipse, indicating the nature of the service exchange.

The text explains that:

- Yellow/orange boxes are Service Domains
- Arrows indicate service connections (pointing to the service provider from the caller)
- Purple ellipses indicate the nature of the service exchange (referencing the associated "action term")

The key properties of the BIAN wireframe are:

1. Non-exhaustive: The wireframe only needs to show available service connections used in the associated collection of business scenarios. Some versions may show additional or all available connections, but these can become unwieldy.

2. Arbitrary Scope: The selection of Service Domains and associated connections is informal. It's permissible to add or suppress connections and include/exclude Service Domains to clarify a particular viewpoint.

The text notes that wireframes assembled for a broad collection of Business Scenarios can become quite complex, and arranging the Service Domains to avoid too many crossed connections can be challenging.

This wireframe approach is typically used to capture the main processing/event requirements for an area of interest, assembling all Service Domains and service connections used in a collection of

Business Scenarios. Additional connections or 'boundary' Service Domains can be added to round out the wireframe where helpful.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents a complex wireframe diagram titled "The customer servicing wireframe with the mortgage offer process highlighted". This diagram illustrates various interconnected service domains involved in customer servicing with a focus on the mortgage offer process. The wireframe is designed to balance including all required service connections while maintaining readability.

Key components of the wireframe include:

1. Customer Management: Including Customer Relationship, Advanced Analytics, E-Branch Operations, and Contact Handler.

2. Channel Management: Covering Channel Activity Analysis and Channel Activity History.

3. Product and Offer Management: Highlighting Customer Offer, Product Directory, and Mortgage Loan.

4. Compliance and Risk: Including Regulatory Compliance, Fraud Models, and Fraud/AML Resolution.

5. Operations: Covering Document Services, Transaction Authorization, and Card Capture.

6. Customer Engagement: Including Customer Agreement, Customer Profile, and Customer Product/Service Eligibility.

The diagram uses color-coding to emphasize certain elements, with red outlines highlighting key components in the mortgage offer process.

The text below the diagram explains that for more complex projects, multiple wireframes can be used to highlight different areas and aspects of development.

The document then introduces a concept: "Combined scenario and wireframe views clarify the components for development". It explains that having both dynamic (business scenario) and static (wireframe) models is useful for fully understanding the service-centered design. In systems development, the business scenario confirms key business requirements, similar to process-oriented design. The Wireframe provides a stable blueprint for defining the scope of development.

This approach helps IT and Business consultants in the banking industry to visualize and understand the complex relationships between different service domains and processes, particularly in the context of mortgage offerings and customer servicing.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Overview

This guide provides insights into the mapping of applications with the BIAN Service Domains, focusing on the functional partitions of applications and the use of APIs for internal exchanges.

Mapping Applications to Service Domains

- Applications, whether existing or to-be-developed, can be mapped to the BIAN Service Domains.

- This mapping allows for a clear understanding of the major functional partitions within the application.

Use of Wireframes

- Wireframes can be utilized to visualize the interfaces that can be established via APIs between applications.

- The guide will expand on the use of wireframes in the context of different technical considerations.

BIAN Semantic APIs

REST Mapping and BIAN Semantic API Portal

- The BIAN standard is implementation agnostic, supporting the use of BIAN Service Domain partitions and service operations as a framework for service-based architectures.

- BIAN definitions have been mapped to the REST architecture style, which is currently the most popular approach for API development in the banking sector.

API Mapping Assumptions

- When mapping APIs to BIAN, it is assumed that the BIAN Service Domain aligns with the application boundary of the API.

- The operations of the Service Domain constitute the collection of program interfaces (PIs) described by the API.

- A semantic API in BIAN terms consists of the collection of service operations offered by a Service Domain, formatted for developer enhancement and extension.

REST Architecture

State Transfer (REST)

- REST has been developed specifically for creating services, defining constraints to ensure performance and efficiency for applications communicating over the internet.

- The REST approach provides access to resources using a predefined set of operations, ensuring stateless interactions (no client information is persisted between requests).

- Resources are identified with a URI, and requests yield responses that return values related to the resource in the payload, which can be formatted in various ways (e.g., JSON, HTML, XML).

Common HTTP Methods

- The most common protocol used for service request operations is HTTP, with specific methods applied in BIAN mapping:

- GET

- POST

- PATCH

- DELETE

Architectural Constraints

- BIAN defines six constraints for compliance, which relate to the BIAN design. The specifics of these constraints and their implications will be summarized in the following sections of the guide.

This structured guide aims to assist IT and Business consultants in understanding the integration of BIAN standards within banking applications, focusing on the use of APIs and REST architecture for effective service delivery.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

1. Architectural Principles

1.1 Client-Server Architecture

- Separation of concerns (no assumed link between client and server data).

- BIAN Service Domains can fully conform to a client-server architecture in implementation.

1.2 Statelessness

- No client context is stored on the server.

- BIAN Service Domains support SOA design principles which can conform to statelessness where practical.

1.3 Cacheability

- Responses can support caching (handles sequential responses sensibly and responses are re-useable).

- BIAN is intended to support SOA design principles, and selective cacheability can be fully supported in implementation.

1.4 Layered System

- The client has no awareness of intermediary layers between it and the host.

- BIAN conforms to SOA design principles, particularly encapsulation, which can be handled in implementation.

1.5 Code on Demand

- The response can embed executable logic.

- BIAN does not preclude that service exchanges can include executable logic, typically being an implementation consideration for front-end applications.

1.6 Uniform Interface

- Comprises four more specific constraints:

- Resource identification in request.

- Manipulation through representations.

- Self-descriptive messages.

- Hypermedia as the engine of application state (“HATEOAS”).

- BIAN service operation definitions do not constrain the adoption of any of these service implementation features as might be appropriate.

2. Summary of BIAN Standard

- The BIAN standard is a conceptual business model that defines service in semantic terms in a manner that is implementation agnostic.

- BIAN specifications are usually deployed using service-oriented architectural (SOA) approaches, specifically representing activity with this goal in mind.

- SOA concepts align well with those imposed by the REST architectural style and are not incompatible with implementation in any significant way.

3. Archetypes

- The BIAN to REST mapping involves a Service Domain control record instance, which essentially represents the accessed resource.

- REST defines four resource archetypes: documents, store, and controller.

- To ensure the Service Domain control record is interpreted correctly, it helps to align the different BIAN generic artifact types for the patterns to these four REST archetypes.\n\nend-of-page\n\n

---

\nnew-page\nThe image provides information on REST archetypes and their mapping to BIAN generic artifact types. Here's a structured breakdown of the content:

REST Archetypes:

1. Documents

  - A singular resource concept

  - Referenced using a conventional hierarchical naming structure

  - Example: http://api.example.com/building-management/office-buildings/{building-Id}

  - State representation typically combines feature values of the instance

2. Collections

- Represents a managed/directory collection of resources
- Determines when to create a new resource instance on request
- Example: http://api.example.com/building-management/office-buildings

3. Store

- A client managed resource repository
- Does not create new resource instances but enables a collection
- Example: http://api.example.com/cart-management/users/{id}/carts

4. Controller

- Handles a procedural concept
- Acts like an executable function with parameters and inputs/outputs
- Example: http://api.example.com/cart-management/users/{id}/cart/checkout

BIAN Generic Artifact Types Mapping:

The image includes a table that maps BIAN generic artifact types to REST archetypes. The table has the following columns:

- Functional Pattern
- Generic Artifact
- Mapped RESTful Archetype
- Example URI

Some key mappings include:

- Strategy, Management Plan, Administrative Plan, Specification, Development Project, Agreement, Assessment, Analysis, and Allocation are mapped to Document
- Operating Session, Maintenance Arrangement, Fulfillment Arrangement, Transaction, Advice, State, and Log are mapped to Controller

- Directory Entry and Membership are mapped to Collections

The table provides specific example URIs for each generic artifact, demonstrating how they would be structured in a RESTful API context.

This mapping shows how BIAN's functional patterns and generic artifacts can be implemented using REST architectural principles, providing a standardized approach for designing APIs in the banking industry.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents information on three types of external access approaches for banking systems, along with a diagram illustrating these approaches. Here's a structured breakdown of the content:

1. Direct to Core
   - External access governed by a gateway implementing basic customer authentication
   - Gateway connects directly to the host system
   - Often re-uses existing external interface for web-based or contact center servicing

2. Wrapped Host
   - Includes access gateway plus front-end capability to address host system shortfalls
   - Supports host migration and repurposing efforts
   - Features:
     - Coordinates access with multiple systems for complex transactions
     - Resolves master-slave data conflicts
     - Optimizes host access sessions
     - Provides advanced information look-up and caching
     - Supports functional extensions

3. Component Architecture
   - Comprehensive set of controls to manage external access

- Allows direct connection to internal bank capabilities
- A wireframe of the external access platform is shown in the diagram

The diagram titled "Three Types of External Access" illustrates:

- Type 1: API Gateway (shown on the right side of the bank)
- Type 2: Service Wrapper (shown in the middle of the bank section)
- Type 3: External Access Framework (shown on the left side)

The diagram depicts:
- Customer and 3rd Party Provider at the top
- An API Management layer
- The BANK section, which includes:
  - API Gateway with Authentication and Resource Access Services
  - Service Wrapper with Host Migration components
  - Business Applications
  - An External Access Framework (labeled as Autonomous Service Center Business Applications)

The image notes that most banks will likely need to support a combination of all three types of access approaches.

Figure 39 is captioned as "Three Types of Access Schema"\n\nend-of-page\n\n

---

\nnew-page\nThe document discusses different access approaches for open banking, focusing on Type 3 access and comparing it with Types 1 and 2. Here's a structured breakdown of the content:

1. Type 3 Access Approach
   - Significant for banks considering open banking
   - Supports third party access while linking customer contact to front office Service Domain – Session Dialogue

- Acts as a gateway structuring access to the bank

- Can access back office transaction systems directly (like Types 1 & 2)

- Supports customer interaction with a wide array of front office capabilities

- Session Dialogue Service Domain can leverage Servicing Order Service Domain for more complex structured workflows

2. Comparison with Type 1 & 2 Approaches

- Types 1 & 2 typically connect directly to back office transaction processing systems

- This eliminates bank participation in 'front office' dialogue with customer and third party solution provider

3. Diagram: Contrasting BIAN Type 1 Vs Type 3 access for open banking interactions

- Illustrates the differences between Type 1 and Type 3 access

- Shows various front office and back office components

- Highlights External Access Framework (Type 3) and 'Standalone' API Gateway (Type 1)

- Note: Type 2 access covers legacy wrapping and migration options (limited interest in this context)

4. Key Components in the Diagram

- Front Office: Various service domains (e.g., Customer Campaigns, Relationship Management, Servicing Order)

- Back Office: Transaction Systems

- Type 3: External Access Framework with Authentication & Consent Authorization

- Type 1: 'Standalone' API Gateway with 'Keyhole' access to back office systems

5. Session Dialogue and Servicing Order

- Session Dialogue supports menu-driven "canned" transactional access

- Servicing Order can optionally support more complex structured workflows

6. Type 3 External Access Framework

- Main elements can be seen in a wireframe (not provided in the image)

- Reveals range of capabilities needed to control access of customers and external third parties to internal bank workings

The document emphasizes the importance of the Type 3 access approach in the context of open banking, highlighting its flexibility in supporting both direct back-office access and enhanced front-office customer interactions.\n\nend-of-page\n\n

---

\nnew-page\nThe image presents a complex diagram titled "The Service Domains Handling External Access (Type 3 Wireframe)" which is part of the BIAN External Access Framework. Here's a structured breakdown of the content:

1. Introduction

- Banks are exploring new business models, including collaboration with FinTechs and open banking approaches.

- Support for external coordination will be crucial.

- The BIAN External Access Framework is a draft application of the BIAN standard.

- It's being applied to a proof of concept initiative at the time of publication.

- More complete specification of the framework is documented elsewhere.

2. Diagram Components

The diagram is divided into several sections:

a. External Directories & Agencies

- Includes Competent Authority (TA, CP, CA) and IP Directory

b. Access Controls

- Issued Device Tracking

- Issued Device Administration

c. Authentication

- Party Authentication
- Party Directory Access

d. External Access Framework

- Authorization
- Customer Access
- Customer Servicing

e. Service Provider (TPP) and Customer (PSU)

- E-Branch
- ATM/POS Device

f. Contact Center

- Contact Routing
- Contact Handler

g. Relationship Development

- Customer Campaign Management
- Product Directory
- Customer Offer

h. Back & Front Office Servicing

- Includes various account types (Savings, Mortgage, Investment, Current)

i. Fraud Management

- Fraud Evaluation
- Fraud Resolution
- Fraud Models

j. Access History & Analysis

- Customer Event History

- Customer Activity Analysis

3. Connections

- The diagram shows numerous interconnections between these components, represented by blue lines.

- These connections illustrate the flow of information and processes within the framework.

4. Purpose

This wireframe appears to be a comprehensive visualization of how different service domains interact to handle external access in a banking system, particularly in the context of open banking and collaboration with FinTechs.

Figure 41 is labeled as the "External Access Framework Wireframe", indicating its importance in illustrating the overall structure and connections within the BIAN External Access Framework.\n\nend-of-page\n\n

---

\nnew-page\nIt seems that the text provided does not contain any content other than a footer and a page number. To assist you effectively, I would need more content from the document. Please provide additional text or sections from the PDF that you would like me to parse and structure.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Action Terms as they Relate to Functional Patterns & Control Records

Overview

This section discusses the relationship between action terms and functional patterns within the context of control records in the banking industry. It emphasizes the importance of understanding these terms to enhance operational efficiency and compliance.

Key Concepts

- **Action Terms**: Definitions and examples of action terms relevant to banking operations.

- **Functional Patterns**: How these terms fit into broader functional patterns within banking services.

- **Control Records**: The role of control records in maintaining the integrity and accuracy of banking operations.

Right-sizing a Service Domain

Introduction

This section addresses the concept of right-sizing within a service domain, focusing on optimizing resources and services to meet customer needs effectively.

Principles of Right-sizing

- **Assessment of Current Services**: Evaluating existing services to identify areas for improvement.

- **Resource Allocation**: Strategies for allocating resources efficiently to enhance service delivery.

- **Customer-Centric Approach**: Ensuring that the right-sizing process aligns with customer expectations and market demands.

Implementation Strategies

- **Data Analysis**: Utilizing data analytics to inform decision-making in the right-sizing process.

- **Stakeholder Engagement**: Involving key stakeholders in the right-sizing initiative to ensure buy-in and support.

- **Continuous Improvement**: Establishing a framework for ongoing assessment and adjustment of service domains.

This structured guide provides a clear understanding of the key topics relevant to IT and business consultants in the banking industry, focusing on action terms, functional patterns, control records, and the right-sizing of service domains.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

A. Action Terms Related to Functional Patterns

Terms as they Relate to Functional Patterns & Control Records

In some cases, it can appear that the same action term results in different behaviors when applied to different Service Domains. This is due to the Service Domains having very different underlying operational characteristics. To fully understand how the term is consistently applied, it is sometimes necessary to consider the Service Domain’s functional pattern/control record explicitly.

Case Study: Action Terms "Initiate" and "Execute"

This is particularly evident with the response to the action terms "initiate" and "execute" when applied to Service Domains with a fulfillment functional pattern compared to those with a process functional pattern.

Current Account Service Domain

- **Functional Pattern**: Fulfillment

- **Control Instances**: Current account fulfillment arrangements

- **Behavior Qualifier Type**: A fulfillment arrangement (i.e., how the current account fulfillment arrangement record is broken into parts)

- **Features**: Represents the different product features that make up the current account facility.

- **Example Qualifiers/Product Features**:

- **Interest**: Handles the array of interest rates applicable to the current account facility.

- **Payments**: Manages the setup and processing of different types of payments made from the account, including regular standing orders, direct debits, and bill pay.

Customer Billing Service Domain

- **Functional Pattern**: Process
- **Control Instances**: Customer billing procedures
- **Behavior Qualifier Type**: How the customer billing procedure record is broken into parts
- **Work Steps**: The series of steps in handling a customer billing cycle.
- **Example Qualifiers/Procedure Work Steps**:
- **Invoicing**: Handling the creation and management of the customer invoice.
- **Tracking and Reminders**: Managing the tracking and issuance of reminders.

Action Terms: "Initiate" and "Execute"

The action terms "initiate" and "execute" have very different general purposes. Below, we describe how they act on the two different Service Domains to clarify their consistent actions on control records (or their behavior qualifiers). However, the requests result in rather different behaviors due to the differing functional patterns:

- **Initiate**: Results in the creation and initialization of a new control record instance or a contained behavior qualifier instance for an existing control record.

This structured approach helps in understanding the application of action terms in different functional contexts within the banking industry.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Current Account and Customer Billing Service Domain

Current Account

- **Initiate Action Term Responses:**

- **Initiate Current Account Fulfillment Arrangement:**

- Results in the establishment and initialization of a new current account facility.

- **Initiate Current Account Fulfillment Arrangement | Interest:**

- Establishes specific interest handling features for the account.

- Typically performed as an internally orchestrated product fulfillment setup; external service calls may not always be required or supported.

- **Initiate Current Account Fulfillment Arrangement | Payments:**

- Sets up a payments capability associated with the account.

- Includes regular scheduled payments (e.g., standing orders).

- For one-off/ad-hoc payments, this call configures the capability to handle payments but does not execute the transaction itself (refer to ‘execute’ later).

Customer Billing

- **Initiate Action Term Responses:**

- **Initiate Customer Billing Procedure:**

- Triggers the customer billing process, encompassing the end-to-end processing of a customer bill.

- The Service Domain’s process logic may orchestrate all end-to-end actions, making this the only required external service call.

- **Initiate Customer Billing Procedure | Invoicing:**

- Triggers the generation of the invoice sent to the customer.

- This step typically follows automatically from the initiation of the overall process, so this specific service is unlikely to be required or supported.

- **Initiate Customer Billing Procedure | Tracking and Reminders:**

- Triggers the generation of a reminder if the payment is overdue.

- This action can be internally generated on a schedule or may involve an external service to allow other parties to trigger billing reminders.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Execute Action

The **Execute** action is defined as an action that acts on an active control record instance or one of its subordinate behavior qualifier instances as necessary. This action will invoke some automated task that is then applied to the instance.

Current Account Responses

For the **Current Account**, the execute action can result in the following responses:

- **Execute Current Account Fulfillment Arrangement**:

- Triggers an automated action that applies to the overall account, such as purging old account records.

- Note: It is challenging to define a particularly good example as most interesting product functions are covered by the underlying behavior qualifier product features.

- **Execute Current Account Fulfillment Arrangement | Interest**:

- Triggers an automated task associated with the application of interest to the account, such as applying an amended rate to a specific aspect of the account.

- **Execute Current Account Fulfillment Arrangement | Payment**:

- Triggers a payment transaction against some pre-configured payment feature.

- This could involve making an ad-hoc payment from the account or an instruction to override a scheduled standing order payment.

Customer Billing Responses

For **Customer Billing**, the execute action can result in the following responses:

- **Execute Customer Billing Procedure**:

- Triggers an automated action against an active billing procedure, such as an instruction to reset the billing process.

- **Execute Customer Billing Procedure | Invoicing**:

- Triggers an automated action specific to an already active invoicing work step of the billing process, such as generating a duplicate/repeat invoice.

- **Execute Customer Billing Procedure | Tracking and Reminders**:

- Triggers an automated action against an already active reminder work step, such as redirecting the scheduled reminder missives.

Summary of Actions

The examples illustrate that for Service Domains that handle an ongoing facility, such as Current Accounts, the **initiate** action is needed to set up some processes, and then the **execute** action is used to trigger related transactional events.

In contrast, for domains that are more process-oriented, such as Customer Billing, the **initiate** action typically triggers a chain of processing activities, and the **execute** action is comparatively rarely used to intervene in these active processing activities.

The response to the **initiate** and **execute** action terms is consistent across different service domains.\n\nend-of-page\n\n

---

\nnew-page\n# Guide for IT and Business Consultants in the Banking Industry

Introduction

This guide is designed to assist IT and business consultants working within the banking industry. It provides insights into the operational frameworks, service domains, and the importance of consistent terminology in banking processes.

Service Domain Responses

- **Operating Profiles**: The responses from service domains can vary significantly due to different operating profiles.

- **Control Records and Behavior Qualifiers**:

- BIAN control records and behavior qualifier types are structured to maintain consistency in the meaning of action terms.

- This consistency applies to both the overall control record instance and its constituent parts, as defined by behavior qualifiers (or sub-qualifiers).

- The behavior qualifier continues to apply the functional pattern behavior to a subset of the service domain’s function.

Conclusion

Understanding the nuances of service domain responses and the importance of consistent terminology is crucial for effective consulting in the banking sector. This guide serves as a foundational resource for navigating these complexities.\n\nend-of-page\n\n

---

\nnew-page\n# Right-sizing a Service Domain

Overview

Right-sizing a Service Domain involves a complex technique that defines the correct scope for a Service Domain. This process includes several overlapping considerations. While implementers utilize Service Domains, they are not required to define new ones, making it unnecessary for them to learn this particular technique. The details are explained in the BIAN guides, but a brief outline is provided here for general information.

Key Concepts

Functional Pattern of Behavior

Every Service Domain applies a single functional pattern of behavior throughout its lifecycle. The primary variable that determines how a Service Domain is ‘right-sized’ is the type of asset it acts upon.

Definition of Assets

In this context, an asset refers to anything the bank owns or controls. Assets can be tangible, such as buildings or technology, or intangible, such as customer relationships or knowledge. The capacity to perform functions, like a call center that services customers or facilities that deliver current account services, is also considered an asset.

Asset Type Classification

To isolate banking asset types, BIAN has established a mutually exclusive, exhaustive (MECE) asset type classification hierarchy. Asset types are categorized into sub-types until they retain business meaning and context. Below this level, finer-grained asset types become more utility-focused.

Example of Asset Classification

For instance, consider the asset representing a bank's overall capacity to handle interactions with different parties. This asset can be broken down into sub-types that address interactions with various parties, such as business partners and customers.

When attempting to further dissect the capacity to handle customer interactions, we might define finer activities like holding meetings, developing performance plans, and addressing issues. These actions are no longer uniquely assignable to a specific role (e.g., customer relationship management) but are more utility-based, as they can be performed across different parts of the organization.

Defining Service Domains

Types defined at the level just above where they become commodity in nature, and that are acted upon by a single functional pattern, define an elemental business function. For example, applying the ‘management’ functional pattern could lead to the establishment of a Customer Relationship Management Service Domain, as it has unique business functions that can be clearly assigned within the organization. Conversely, a lower-level “Party Management” Service Domain could be assigned to various responsible areas as a utility function/activity, failing to meet the design requirements for a Service Domain.\n\nend-of-page\n\n

---

\nnew-page\n

It seems that the text provided does not contain any content other than a footer and a page number. To assist you effectively, I would need more content from the document, such as titles, subtitles, or any relevant sections that you would like to be structured. Please provide additional text or specify the sections you want to focus on.\n\nend-of-page\n\n

---

Annex: important BIAN Definitions:

Commands:

Definition: Commands are explicit instructions given to a system to execute predefined operations. These instructions can trigger actions or set of actions within the system's environment.

Example: In a banking system, a command might be "Transfer Funds", which initiates the process of moving funds from one account to another.

Services:

Definition: Services are discrete functions that a system provides to support specific business activities or processes. Services encapsulate business logic and data, providing a defined output based on inputs.

Example: A loan approval service in a banking application that evaluates customer data and returns an approval or denial based on predefined criteria.

Event Agents:

Definition: Event agents are components within a system designed to detect specific events and trigger corresponding actions based on those events.

Example: An event agent in a fraud detection system might monitor transaction patterns and, upon detecting a potential fraud, trigger an alert and initiate a transaction hold.

Business Logic:

Definition: Business logic comprises the algorithms and rules that govern the processing of data for business operations. This logic defines how business objects interact with one another to carry out business processes.

Example: In an e-commerce platform, the business logic determines how discounts are applied to products in a shopping cart based on current promotions and customer eligibility.

Asset Type:

Definition: Asset Type refers to categories of items or resources that are managed within a system, each characterized by specific properties and behaviors.

Example: In a content management system, asset types might include documents, videos, and images, each with metadata such as creation date, author, and usage rights.

Service Domain:

Definition: A Service Domain is a logical grouping of services that relate to a specific aspect of business functionality, handling all aspects of the business function from initiation to completion .

Example: A "Customer Management" Service Domain in a CRM system might include services for customer data integration, customer profiling, and communication management.

Functional Pattern:

Definition: Functional Patterns describe standard operational behaviors within a Service Domain, guiding how services are structured and interact.

Example: A "Transaction Processing" functional pattern might standardize steps for validating, processing, and recording transactions.

Control Record Type:

Definition: Control Record Type classifies the records used within a Service Domain to manage and monitor the lifecycle of business operations .

Example: In a financial system, control record types might include transaction records, audit logs, and reconciliation records.

Control Record:

Definition: A Control Record stores information about each instance of a business operation within a Service Domain, tracking its progress and state throughout its lifecycle .

Example: A loan application record that tracks all changes, assessments, and decisions made during the loan processing phase.

Behavior Qualifier Type:

Definition: Behavior Qualifier Types classify the conditions or characteristics that affect the processing and management of control records.

Example: In a customer service system, behavior qualifier types might include priority levels, such as standard, urgent, or critical, which affect how customer inquiries are handled.

Behavior Qualifier:

Definition: Behavior Qualifiers are specific conditions or attributes that influence how control records are processed within a Service Domain.

Example: An "escalation" behavior qualifier that triggers additional review steps if a customer complaint meets certain criteria.

Configuration:

Definition: Configuration refers to the setup and arrangement of system components and settings to meet specific operational requirements.

Example: Configuring a network security system with specific rules for traffic filtering, monitoring, and threat detection.

Reference Data:

Definition: Reference Data includes data that is used to categorize or clarify other data within a system, often used for validation, enrichment, or classification purposes.

Example: Currency codes, country codes, and standard industry codes used within a financial trading platform to ensure consistency in transaction processing.

Annex: The BIAN Metamodel:

UML Diagram Elements

Commands

Services

Event Agents

Business Logic

Asset Type

Service Domain

Functional Pattern

Control Record Type

Control Record

Behavior Qualifier Type

Behavior Qualifier

Configuration

Reference Data

Description of Relationships

Commands, Services, and Event Agents: These are inputs to the Business Logic component. Think of them as various triggers or actions that the business logic needs to process.

Business Logic: This component is responsible for processing the commands, services, and event agents. It interacts with the Asset Type.

Asset Type: This represents the specific type of asset that the business logic is dealing with. It is connected to the Service Domain.

Service Domain: This is the central element of the diagram. It manages the interactions and dependencies of other components. It is influenced by the Functional Pattern, Configuration, and Reference Data.

Functional Pattern: This defines standard behaviors and processes within the service domain. It specifies the Control Record Type.

Control Record Type: This specifies the type of control records used to manage the service domain. It defines the Behavior Qualifier Type.

Control Record: This component keeps track of each instance of responsibility execution within the service domain for a complete lifecycle. It is associated with the Behavior Qualifier.

Behavior Qualifier: This specifies the conditions or qualifiers that influence the control records. It helps in tracking and processing within the service domain.

Configuration and Reference Data: These components provide additional context and data to the service domain, enabling it to perform its functions effectively.