Banking Industry
Architecture Network

BIAN
How-to Guide

Design Principles & Techniques

Organization

Authors

Role	Name	Company
BIAN Architect	Guy Rackham	BIAN

Status

Status	Date	Actor	Comment / Reference
DRAFT	October 2018		Initial draft

Version

No	Comment / Reference	Date
7.0	First edited version	October 2018

Copyright

Table of content

Table of figures

1 BIAN How-to Guide - Design Principles & Techniques

1.1 Introduction

This is the first of three documents making up the BIAN 'How-to Guide' series. It covers the BIAN design principles and techniques. The intended audience is business and technical architects. It presents the theory and concepts behind the BIAN standard. In particular it explains how the BIAN designs seek to be canonical. The content is summarized in figure 1. Some of the topics covered here from a mostly theoretical/conceptual perspective are revisited in the other documents of the BIAN 'How-to Guide' from their respective viewpoints.

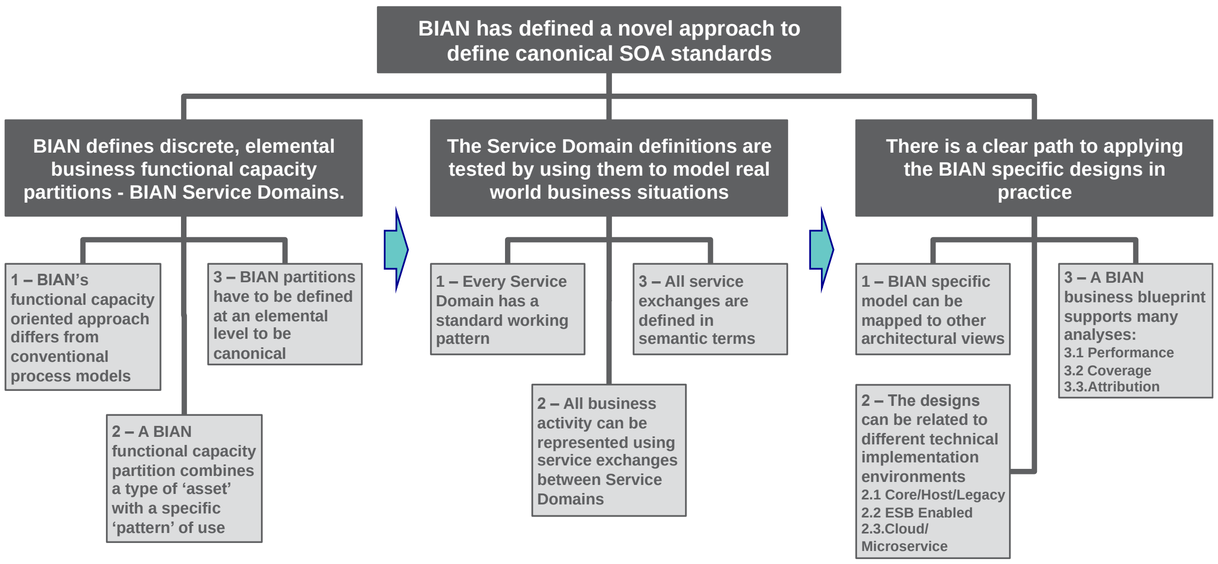

BIAN has defined a novel approach to define canonical SOA standards
BIAN defines discrete, elemental business functional capacity partitions - BIAN Service Domains.
1 – BIAN's functional capacity oriented approach differs from conventional process models
2 – A BIAN functional capacity partition combines a type of 'asset' with a specific 'pattern' of use
3 – BIAN partitions have to be defined at an elemental level to be canonical
The Service Domain definitions are tested by using them to model real world business situations
1 – Every Service Domain has a standard working pattern
2 – All business activity can be represented using service exchanges between Service Domains
3 – All service exchanges are defined in semantic terms
There is a clear path to applying the BIAN specific designs in practice
1 – BIAN specific model can be mapped to other architectural views
2 – The designs can be related to different technical implementation environments
2.1 Core/Host/Legacy
2.2 ESB Enabled
2.3.Cloud/ Microservice
3 – A BIAN business blueprint supports many analyses:
3.1 Performance
3.2 Coverage
3.3.Attribution

Figure 1: The BIAN 'How-to Guide Design Principles & Techniques

The BIAN business architecture adopts a functional capacity based view of business activity. An explanation of what this means and the reasoning behind its adoption is provided in the remainder of this first opening section. The BIAN design principles and techniques are then set out in three additional sections:

> **Section 2: BIAN Functional Capacity Partitions – the Service Domain** – BIAN's approach is based on breaking down all banking activity into a collection of discrete business functional partitions called BIAN Service Domains.
>
> **Section 3: Modeling Real World Behaviors – Service Operation Interactions** – The specification of BIAN Service Domains is tested and refined by modeling business behaviors primarily using BIAN Business Events and their associated Business Scenarios to identify service operation dependencies between Service Domains. The service operations define the Service Domain exchanges in semantic terms

Section 4: Mapping the BIAN Business Architecture to Systems Architectures – The BIAN standard is a type of service oriented business architecture (SOA). It needs to be mapped to different and more detailed systems architecture views to support solution development. Furthermore the way the BIAN model is interpreted varies for different target technical environments. This section concludes with a brief description as to how the BIAN specifications can be also used for business and technology planning and analysis.

1.2 Explaining why BIAN adopted a functional capacity based model

BIAN's goal is to define standard service operations covering the working of banks (other types of financial institution may be included in time). BIAN has adopted a specific approach to modeling business behaviors In order to define service operations that are 'canonical', or consistently interpretable in any bank. The approach captures business activity by identifying generic operational 'business functional capacity partitions' – the "BIAN Service Domains". An operational business functional capacity partition represents the ability perform some type of well-defined business need such as the capacity to maintain a contractual agreement or the capacity to execute a financial transaction.

A capacity-based model of business activity is fundamentally different from the more widely used process oriented model. The capacity model captures the ***static*** or enduring facilities/functions that collectively make up a bank. For example a typical bank will have customer contact centers, bank branches, ATM networks, trading floors, and intangible items such as relationships, knowledge and know-how. The alternate process model view captures the ***dynamic*** or temporal sequence of linked tasks that happen in response to some kind of trigger or event. For example a process model can describe the steps a bank follows when it on-boards a new customer or the steps it follows when it executes a payment transaction.

The difference between Functional Capacity (Static) and Process (Dynamic) Model Views

The two types of model are not mutually exclusive. They are simply used to represent different perspectives of a business and so highlight different aspects of the same business activity. The static functional capacity model is good to highlight the discrete working elements/resources/skills needed to respond to any likely event but does not necessarily detail the precise sequence or thresholds involved in invoking these different functional building blocks to react to any one particular business event. Conversely the dynamic process views sets out the sequence of dependent actions in a tightly linked series needed to address a specific business event but may not detail the specific functional capacities/elements responsible for completing each particular action. To properly model business activity both static and dynamic perspectives are often needed.

An analogy can be used to clarify the difference between a static/capacity model and a dynamic/process model. Consider something more tangible than a commercial enterprise – a city. A ‘static’ functional capacity model view of the city lists the types of buildings and different infrastructure that makes up the town. These elements can be lain out to create the town plan The ‘dynamic’ process model view in contrast describes examples of individual journeys made through the town by its inhabitants as they carry out their day-to-day activities.

Continuing with the city analogy the limited information needed to describe a journey between two points is simply the directions to follow – turn left, straight on, turn right etc. This description does not include the nature or purpose for visiting specific buildings on the journey. This highlights a limitation of many process models as already noted they do not necessarily formally define the responsible parties along the way in any great detail.

The BIAN standard defines discrete business functional capacity partitions – its Service Domains. These are similar in purpose to the buildings in the city plan, where each Service Domain represents a specific type of building with a discrete purpose (a school, police station, home, grocery, park, cinema etc…). BIAN Uses a number of static model views containing Service Domains for different purposes (These are described in more detail later):

- The ***BIAN Service Landscape*** – is a reference model containing one of each identified Service Domain organized in groups to help with their identification and selection
- A ***Model Bank View*** – is an enterprise blueprint assembled from a selection of Service Domains, possibly including duplicates of Service Domains where this is necessary to reflect the make-up of the particular organization. (For example if the enterprise operates multiple contact centers in different locations there will be copies of the associated Service Domains)
- A ***Wireframe*** – is a more narrow selection of Service Domains (with the available service connections between them) needed to address a particular area of business activity. For example all of the Service Domains directly and indirectly involved in customer relationship development.

BIAN also uses a representation called the ***BIAN Business Scenario***. This differs from the previous views in that it aligns loosely with a dynamic process model. Like a process model, the Business Scenario defines a linked sequence of interactions between Service Domains in response to a business event. The Business Scenario also clearly defines the specific Service Domains and service operation exchanges responsible for each action involved in the sequence.

A Functional Capacity (Static) Model is Better Suited to Defining a Standard

The city analogy also helps explain one key reason BIAN has used the capacity-based model view to define standard service operations. The capacity model of the city (the town plan) is bounded or finite in terms of its scope. Depending on the selected level of detail, the structural make-up and content of the town plan is quite stable and enduring. It should be possible to draw an unambiguous town plan that everyone can look at and agree whether or not it accurately represents the current layout of the city.

To define an unambiguous and comprehensive process model of the city would require capturing every possible ‘day-in-the-life’ journey that any of its inhabitants might wish to make. Clearly the number of process paths and possible variations needed to exhaustively capture anything that may go on in that city is practically unlimited. It would be just about impossible to document every possible one let alone have everyone agree on the validity of them all.

Process models of a business are extremely useful to capture frequently recurring behaviors that can then be streamlined and/or automated. This view would be used to determine the optimal configuration for mass transit systems in a town. But process models due to their very flexibility are not good for defining canonical standards for anything but the most commodity and predictable type of behavior.

Defining Canonical Functional Capacity Partitions

To define its standard BIAN has used functional capacity modeling to break a bank down into its constituent functional elements. BIAN has had to do this decomposition in a specific way that meets a critical objective. In order to create a canonical industry standard the constituent partitions that BIAN specifies must represent *common/generic* building blocks that any bank can select from and assemble to create their own particular ‘town plan’ equivalent of their specific enterprise. The approach BIAN has developed to identify generic functional capacity building blocks is described later in this guide.

As noted, the building blocks in the BIAN model are called BIAN Service Domains. The BIAN Service Domains define behaviors at a business architecture level. These fairly high-level definitions need to be related to more detailed systems architecture views for implementation. Some approaches for doing this mapping/translation are also outlined later in this guide.

Why Use Functional Capacities as the Building Blocks for the BIAN model

The justification for the BIAN approach described so far has explained why a static model view is most appropriate for its purposes (as opposed to a dynamic model view) and further that a functional capacity model meets this particular requirement. But are there any other static model views that could have been used?

A static model captures the enduring elements that make up the subject. The model also represents a finite or bounded dimension of the subject (that can be decomposed into its constituent parts at increasing levels of detail). It is interesting to briefly consider other static dimensions of a bank in order to confirm that the functional capacity partitioned view is indeed the best for BIAN’s purposes.

- ***Organization/roles*** – this is clearly a finite/bounded dimension but it can be volatile. It also seems that agreeing a comprehensive set of standard roles

and then also standardizing the service exchanges between them would be extremely difficult and may not also relate easily to the supporting applications/systems
- ***Locations*** – also clearly a finite/bounded dimension but clearly not a structure that can be easily related to the supporting applications/systems in any general way
- ***Finance/Capital Allocation*** – another well bounded dimension but as with locations, difficult to relate to supporting applications/systems in a standard/general manner.

From these examples it is hopefully clear that a functional capacity model is most likely to provide a standard view that can be sensibly aligned with the supporting applications and systems.

1.3 Picking the right model view for a technical solution

There are several different ways information technology can be leveraged to support business activity. It is important to match the chosen model view to the way the underlying technology solutions are intended to support the particular business activity.

As mentioned, a process model view is most useful when the goal is to design and build systems that automate/streamline repeating and well-defined activities. In very general terms, a process oriented view of a bank models it as a factory with highly structured production lines that can be automated. A common goal of the supporting technology is to maximize straight through processing (STP) in order to increase efficiency and consistency. Process models and the process-oriented systems they are used to develop are often most appropriate for the high volume transaction processing found at the core of most banks.

But there is a wide array of activities that also go on in a bank that surround this core ‘transaction factory’. This includes activities such as product/service ideation, new business development, relationship management, customer servicing and risk management. These activities are not always easily captured using process models as they do not always follow a predictable and repeating execution path. They are best modeled as collections of discrete specialist functions that collaborate as and when required in a flexible, loosely connected network.

Different model views have emerged over time to better help design systems that support this kind of business behavior. Of note is object-oriented analysis and design (OOAD) and more recently service oriented architecture (SOA). Lately some SOA concepts have been adapted/extended to support the definition of standard application program interfaces (APIs). Also there are significant similarities between the BIAN model view and micro-service architectures and some domain driven design concepts.

The BIAN model is a specific type of service-oriented architecture (SOA) applied at the level of business architecture. Its service exchanges can also be aligned to APIs at a semantic level. The interpretation of the BIAN designs into the underlying systems designs is therefore best suited to service oriented systems design. The BIAN designs can of course be interpreted for process oriented systems design but many of the advantages of service-oriented design are compromised in so doing. The mapping approach for both service and process oriented systems designs are described later in this guide.

2 Business Functional Capacity Partitions – BIAN Service Domains

This section sets out in more detail the way that the generic BIAN Service Domains are specified and how they can be interpreted in systems implementation. It is broken into five sub-sections:

1. The difference between process and functional capcity views is clarified by means of an example that uses example Service Domains.
2. The current technique for defining the discrete business role/scope of a Service Domain is outlined.
3. Every Service Domain has a common structure
4. Comparing process and capacity based solution element re-use
5. BIAN Service Domains are ‘elemental’. A technique is provided that is sometimes needed to adapt the ‘generic’ elemental Service Domain specification in deployment.

2.1 Comparing Functional Capacity and Process views using an example

As noted, the BIAN capacity model does not replace conventional process models. Both model views simply offer different perspectives of the same business activity or event. The process view of some event sets out the sequence of linked actions needed to handle the event, similar to a chef following a recipe when baking a cake. The functional capacity view of the same activity details the ingredients and kitchen equipment needed for the chef to be able to bake that cake.

When the functional components that support the activity are added into the process view they provide additional insights by formally identifying the ingredients/capacities that are used/referenced throughout a process. If every process were completely stand-alone this additional insight would have only limited value. But in a bank the functional capacity accessed in one business process will be used in different combinations in many other processes in the same way a chef can use kitchen equipment and ingredients to prepare many different recipes and many different meals.

An example showing how the same business event is represented using a process and a functional capacity view helps to highlight the additional insights provided by the capacity perspective. The business event is the processing of a month-end billing statement for a credit card customer. First a simplified process decomposition of the event is shown in the next figure:

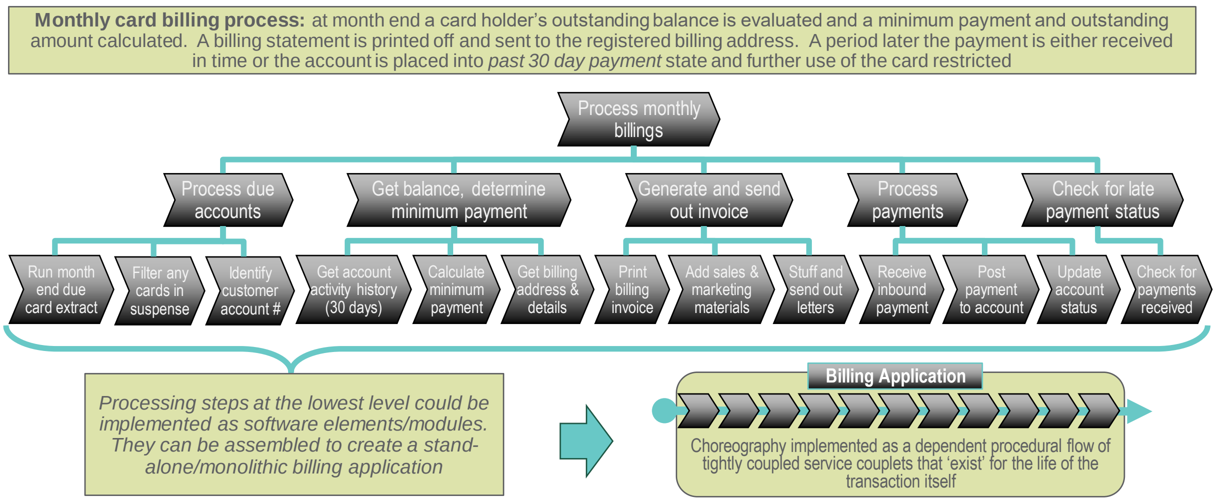

Monthly card billing process: at month end a card holder's outstanding balance is evaluated and a minimum payment and outstanding amount calculated. A billing statement is printed off and sent to the registered billing address. A period later the payment is either received in time or the account is placed into *past 30 day payment* state and further use of the card restricted
Process monthly billings
Process due accounts
Get balance, determine minimum payment
Generate and send out invoice
Process payments
Check for late payment status
Run month end due card extract
Filter any cards in suspense
Identify customer account #
Get account activity history (30 days)
Calculate minimum payment
Get billing address & details
Print billing invoice
Add sales & marketing materials
Stuff and send out letters
Receive inbound payment
Post payment to account
Update account status
Check for payments received
Processing steps at the lowest level could be implemented as software elements/modules. They can be assembled to create a stand-alone/monolithic billing application
Billing Application
Choreography implemented as a dependent procedural flow of tightly coupled service couplets that 'exist' for the life of the transaction itself

Figure 2: Process model of a card billing cycle

The process model view is typically used to break the required sequence of tasks down in ever increasing detail until the point where steps are simple enough to be coded (typically as input/process/output modules) and automated in an application. (In the example the steps are not broken down to this extent to keep it simple.)

Next, the same business event and the low level process steps are matched to the business functional capacity partitions (BIAN Service Domains) that would be involved. Five discrete partitions are involved in this simple example:

1. ***Credit/Charge Card (Fulfillment)*** - responsible for orchestrating any activities associated with the use of a credit card
2. ***Customer Agreement*** – responsible for maintaining the customer specific terms and conditions (in this case the relevant terms are their cycle date and any applicable fees and rates)
3. ***Position Keeping*** – responsible for tracking financial transactions in some form of transaction journal account
4. ***Correspondence*** – responsible for messages sent to and from customers
5. ***Payment Order*** – responsible for moving funds from one account to another.

As can be seen in the next figure the steps in the original process view can be uniquely assigned to different business functional capacities.

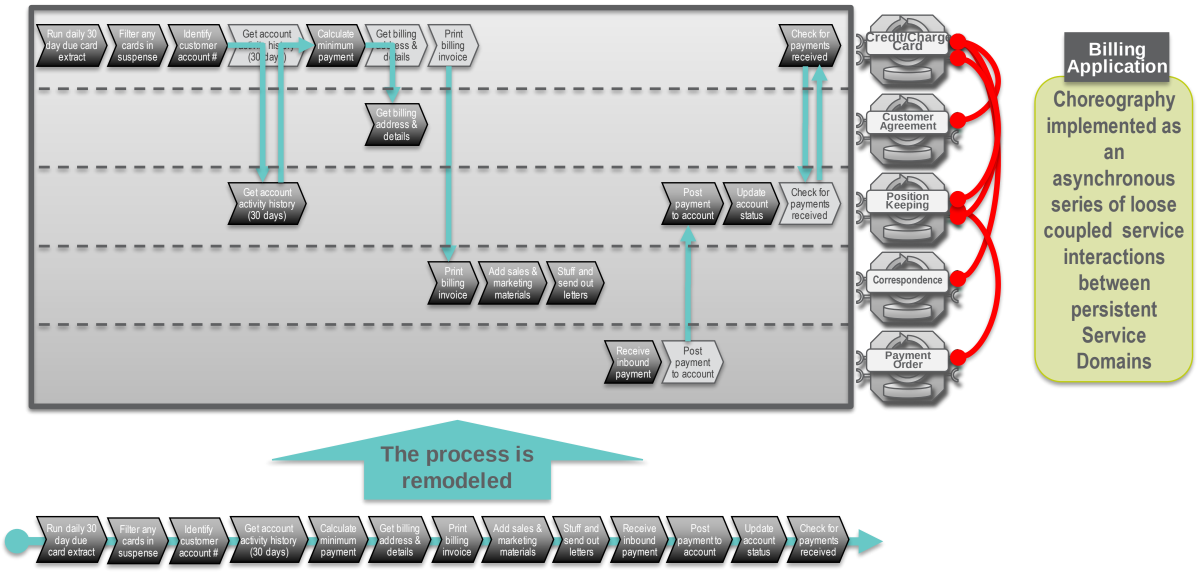

Run daily 30 day due card extract
Filter any cards in suspense
Identify customer account #
Get account activity history (30 days)
Calculate minimum payment
Get billing address & details
Print billing invoice
Check for payments received
Get billing address & details
Get account activity history (30 days)
Post payment to account
Update account status
Check for payments received
Print billing invoice
Add sales & marketing materials
Stuff and send out letters
Receive inbound payment
Post payment to account
Credit/Charge Card
Customer Agreement
Position Keeping
Correspondence
Payment Order
Billing Application
Choreography implemented as an asynchronous series of loose coupled service interactions between persistent Service Domains
The process is remodeled
Run daily 30 day due card extract
Filter any cards in suspense
Identify customer account #
Get account activity history (30 days)
Calculate minimum payment
Get billing address & details
Print billing invoice
Add sales & marketing materials
Stuff and send out letters
Receive inbound payment
Post payment to account
Update account status
Check for payments received

Figure 3: Process model aligned to functional capacities - Service Domains

The sequence of actions orchestrated in the process flow has now been re-organized as a number of service exchanges between the business functional capacities, as shown by the red connectors on the right of the original swim-lane diagram. A better way to draw the interactions between these business capacities or BIAN Service Domains is shown in the figure below where the layout is more representative of a network.

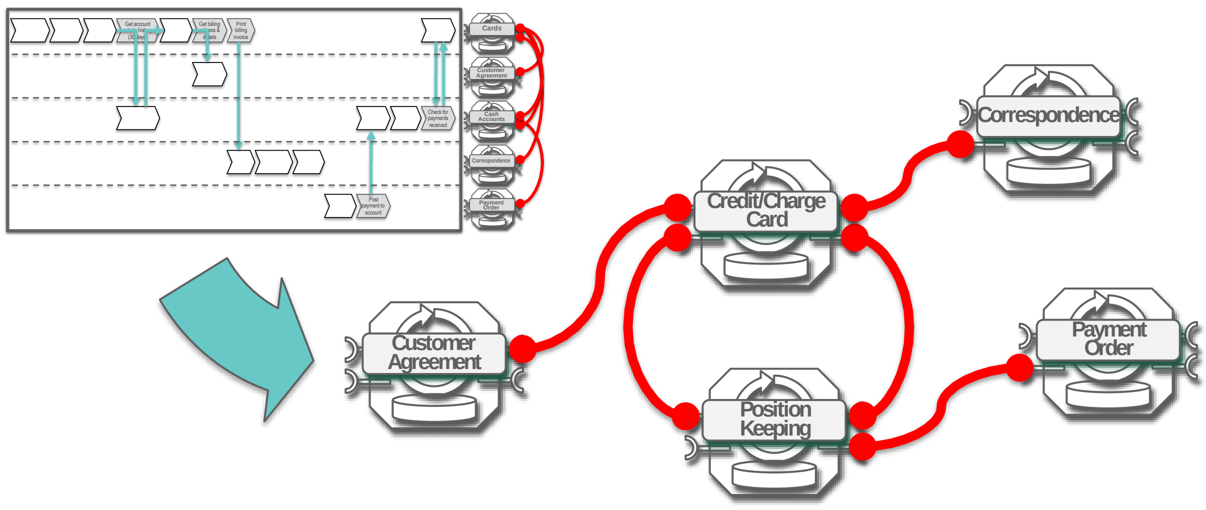


Figure 4: Billing event as a network collaboration)

The Service Domains can be implemented to act as service centers that support the pattern of collaboration needed to process month-end billing. If properly designed they can also be involved in the concurrent processing of any number of other business events in different combinations with other Service Domains.

BIAN is working to identify the complete collection of all of the Service Domains needed to support any activity that may be performed in any bank. These Service Domains are captured in the BIAN Service Landscape. As described in the next section, BIAN uses Service Domains in real world examples of business activity in order to confirm that all of the required Service Domains have been identified and to confirm their precise role and service specifications.

Business Capability Vs Business Functional Capacity Building Block

So far we have described a BIAN Service Domain using the rather unwieldy term business functional capacity rather than the more obvious business capability term that might be expected (and was used in earlier versions of this guide. There is however an important if subtle distinction between the business functional capacity partition represented by a Service Domain and an aspect of a business that is conventionally referred to as a business capability.

A Service Domain represents a discrete and generic business function or more precisely the capacity to perform some action such as maintain reference details about a customer relationship or operate a network. A more formal and complete definition of a 'business capability' describes something that the business wishes to be able to do within a defined business context for which some associated value or purpose can be defined.

Therefore in BIAN the business capability it defined to combine the capacity to perform with a specific business context. The business function performed by a Service Domain may be leveraged to support different business capabilities with different business contexts and associated value and/or purpose.

An example will help clarify this distinction. BIAN has defined a Service Domain that tracks/determines a bank's credit assessment for a customer (Customer Credit Rating). This Service Domain may be involved in many different business capabilities. Consider the two possible business capabilities:

- The capability to match products to customers
- The capability to negotiate product pricing with customers

Each of the above business capabilities can be modeled using a BIAN Business Scenario. In both cases the scenario will include a reference to Customer Credit Rating, but the value/impact of the bank having an accurate credit perspective of the customer varies between the two capabilities.

If the credit perspective is overly generous it may be the impact on product matching would be to recommend the wrong product, leading to a missed sale or worse the sale of an inappropriate product. The impact on the pricing business capability would be to offer too generous terms, clearly a different business value measurement.

Having the business capability view allows this context-based distinction to be maintained. BIAN is developing a business capability model to augment the current Service Landscape. The higher levels of this model are included in the latest release. With the capability view in order to avoid confusion the functioning of a Service Domain is more accurately referred to as a 'business functional capacity. Alternatively it can be termed a 'business capability building block' (as defined by TOGAF). The abbreviated term that will be used in the BIAN guides is a 'capacity' – which is short for business functional capacity.

2.2 Techniques used to define a Service Domains

BIAN has defined and applied a repeatable (empirical) technique for isolating and scoping out the business functional capacity represented by a BIAN Service Domain. This technique is based on the supposition that to realize commercial value from some type of asset some type of action needs to be performed upon it. This can be either to improve or sustain the asset or to then to exploit or leverage it in some way. For example you can possess the asset of a car but to gain value from it you need to both maintain/fuel the car and then operate it, perhaps as part of a taxi service.

In the BIAN technique an asset type corresponds to some tangible or intangible thing that the bank has ownership and/or influence over and has one or more inherent uses or purposes that create commercial value. Some examples can help clarify:

- A building is a tangible asset that can be used to house office functions
- Product expertise is an intangible asset that can be used as the basis for transacting business
- Market insights are intangible assets that can be used to identify business opportunities
- An ATM network is a tangible asset that can be operated and used as an access channel to deliver services to customers.

Simple tests for a well-defined asset are: that it can be purchased or acquired; owned or influenced (to foster or exploit/leverage it); and it has some associated commercial (possibly replacement) value.

All BIAN Service Domains follow this two-aspect definition of having an associated asset/entity type and a type of action/function performed to an instance of that asset type in order to create commercial value/benefit. For the asset/entity dimension BIAN has used a simple hierarchical decomposition of the full range of assets/entities, both tangible and intangible that may be found in a bank. The decomposition breaks the asset/entity types down such that at each level of decomposition the collection of identified asset/entity types is mutually exclusive (non-overlapping) and collectively exhaustive (complete) or 'MECE". Note that the structure of the decomposition itself (the different intermediate non-overlapping and collectively exhaustive categories) is not particularly important. The technique is used primarily to identify all of the different asset types. Where they happen to be found in the decomposition structure itself is not of any great significance.

The figure below shows the first few levels of the asset/entity decomposition currently used. In the decomposition you can see resource type assets (e.g. buildings/workforce) and production capacity (e.g. product delivery and call centers). In the decomposition an asset can be the capacity to perform some function that combines resources, The production capacity is an assembly of other asset types (for example you need people, equipment and know-how) but the collection of suitably configured resources to create the production capacity is defined as an additional asset here. It can be seen that at the first and second level the range of possible asset types has been divided into:

1. Production capacity made up of product/service fulfillment, distribution, combining electronic and assisted channels and sales and marketing functions
2. Centrally managed resources that includes the enterprise functions, finance and buildings and equipment
3. Relationships split into the general workforce and the wide range of external contacts
4. Intellectual property/knowledge that is split between knowing how to do things and knowing about things.

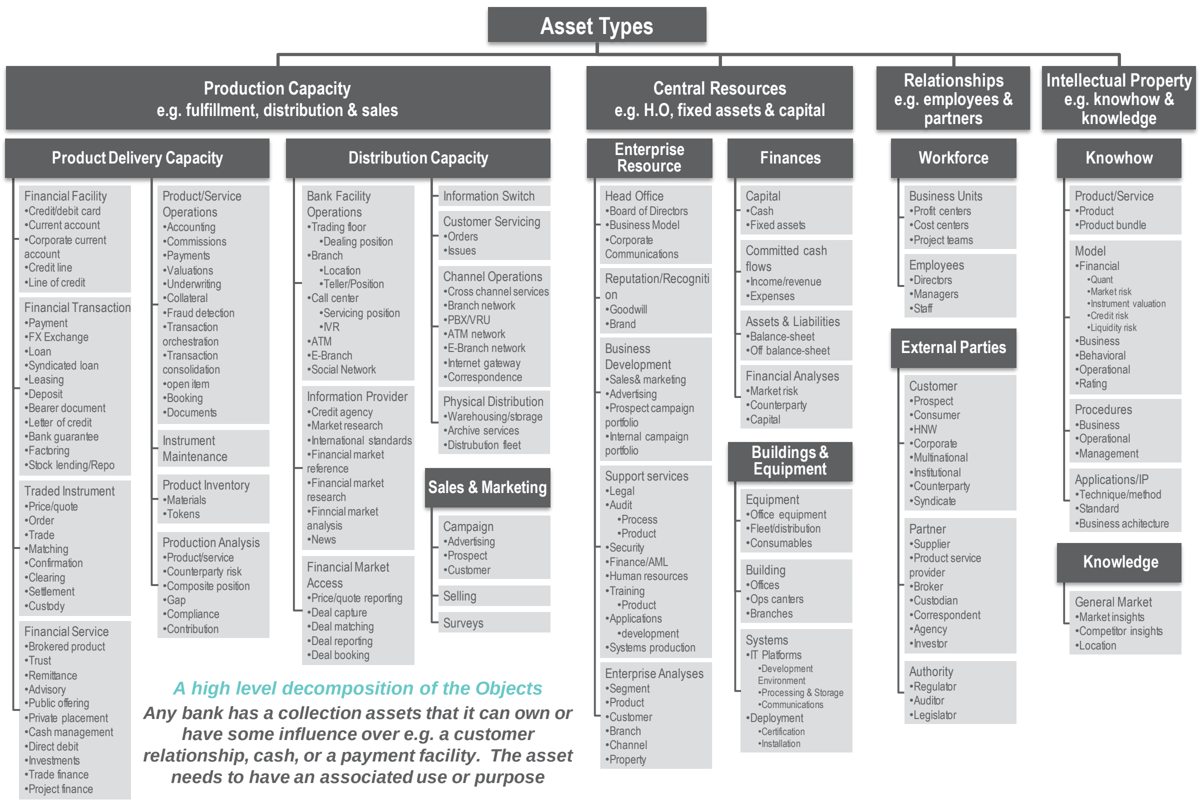

Asset Types
Production Capacity e.g. fulfillment, distribution & sales
Central Resources e.g. H.O, fixed assets & capital
Relationships e.g. employees & partners
Intellectual Property e.g. knowhow & knowledge
Product Delivery Capacity
Financial Facility •Credit/debit card •Current account •Corporate current account •Credit line •Line of credit
Financial Transaction •Payment •FX Exchange •Loan •Syndicated loan •Leasing •Deposit •Bearer document •Letter of credit •Bank guarantee •Factoring •Stock lending/Repo
Traded Instrument •Price/quote •Order •Trade •Matching •Confirmation •Clearing •Settlement •Custody
Financial Service •Brokered product •Trust •Remittance •Advisory •Public offering •Private placement •Cash management •Direct debit •Investments •Trade finance •Project finance
Product/Service Operations •Accounting •Commissions •Payments •Valuations •Underwriting •Collateral •Fraud detection •Transaction orchestration •Transaction consolidation •open item •Booking •Documents
Instrument Maintenance
Product Inventory •Materials •Tokens
Production Analysis •Product/service •Counterparty risk •Composite position •Gap •Compliance •Contribution
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 •Credit agency •Market research •International standards •Financial market reference •Financial market research •Finncial market analysis •News
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 •Distrubution fleet
Sales & Marketing
Campaign •Advertising •Prospect •Customer
Selling
Surveys
Enterprise Resource
Head Office •Board of Directors •Business Model •Corporate Communications
Reputation/Recogniti on •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 canters •Branches
Systems •IT Platforms •Development Environment •Processing & Storage •Communications •Deployment •Certification •Installation
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 •Legislator
Knowhow
Product/Service •Product •Product bundle
Model •Financial •Quant •Market risk •Instrument valuation •Credit risk •Liquidity risk •Business •Behavioral •Operational •Rating
Procedures •Business •Operational •Management
Applications/IP •Technique/method •Standard •Business achitecture
Knowledge
General Market •Market insights •Competitor insights •Location
A high level decomposition of the Objects
Any bank has a collection assets that it can own or have some influence over e.g. a customer relationship, cash, or a payment facility. The asset needs to have an associated use or purpose

Figure 5: Asset type decomposition - top levels

In order to define unique and discrete business capability partitions, the decomposition of asset/entity types is continued to the lowest level where elements retain 'unique business context'. This novel concept is explained using an analogy with building architecture in the next figure:

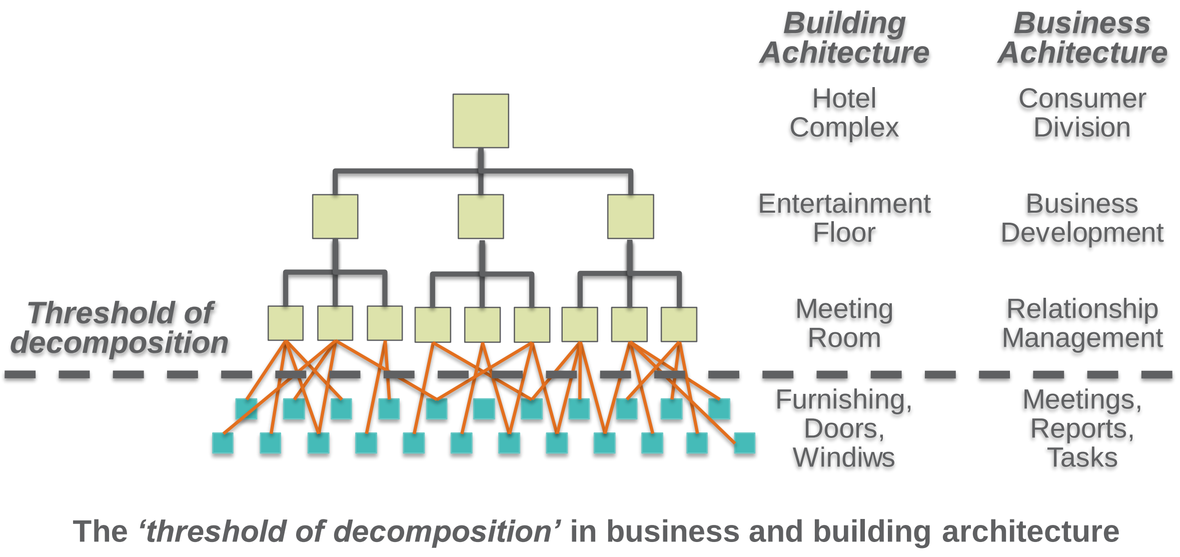
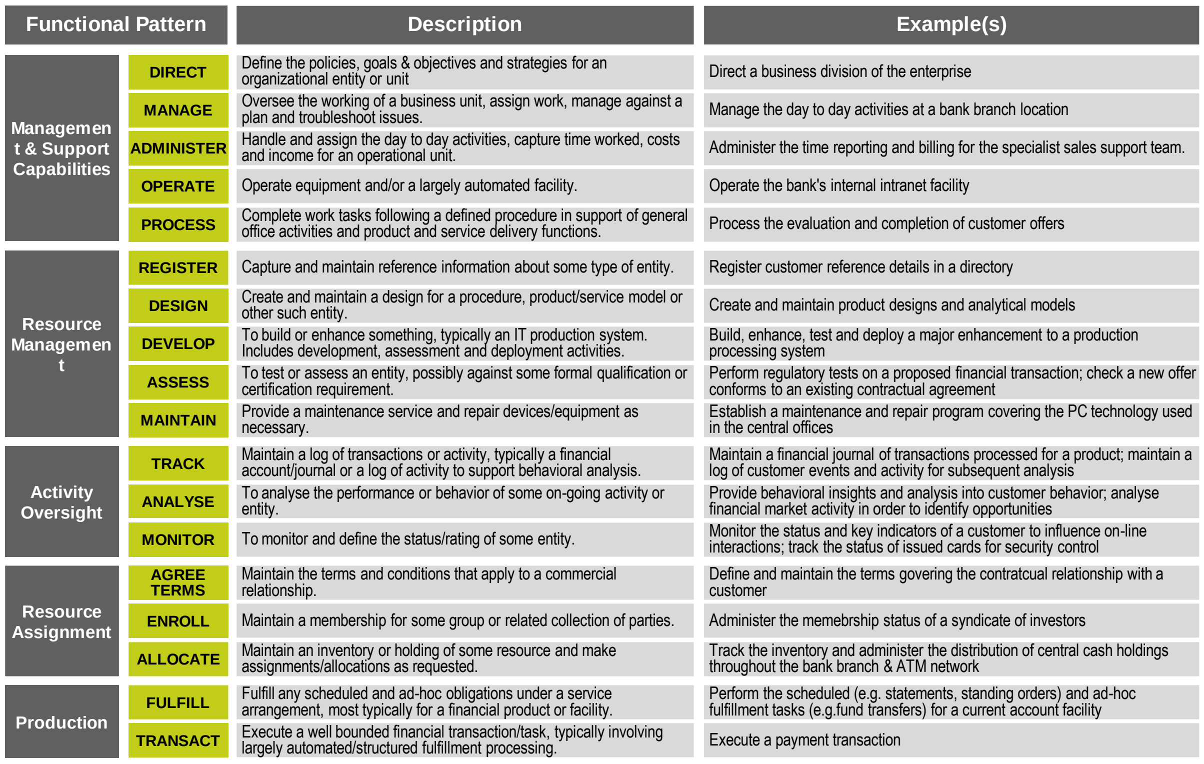


Figure 6: Unique business context example building Vs business architecture

As is shown in the above figure, once the asset/entity is decomposed below some threshold it becomes utility in nature; meaning it can be sensibly duplicated to perform its own particular function independently in many different business contexts. In order to ensure a BIAN Service Domain fulfils a unique/discrete business role (in a single valid business context) it is necessary that its associated asset type can be found in the decomposition hierarchy in a position immediately above this threshold.

The second aspect defining the Service Domain's role addresses the particular function it performs on the asset/entity. Based on an iterative review of Service Domains, BIAN has identified 18 generic 'functional patterns' as listed in the next figure. One of these patterns is selected as the dominant action performed by the Service Domain on instances of its associated asset/entity.

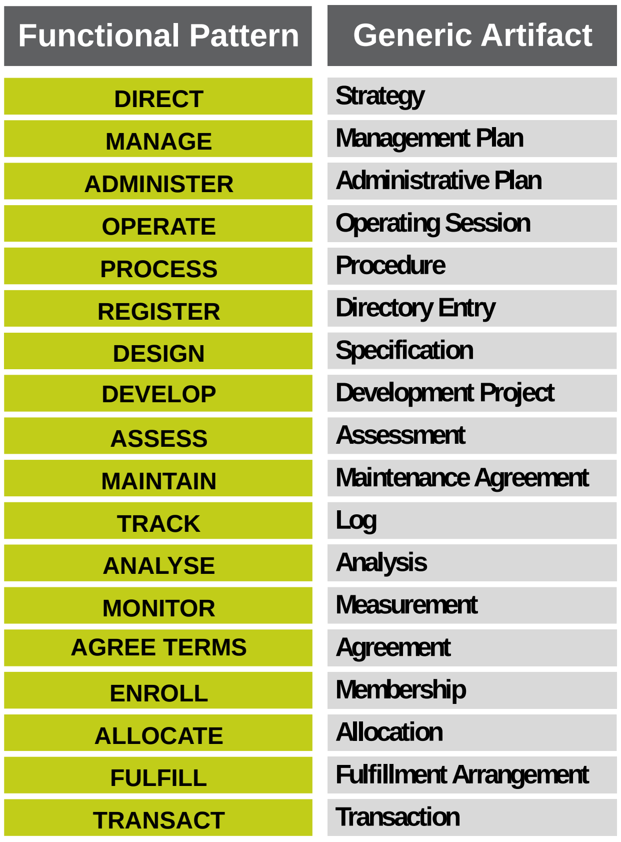
It can help to consider some example asset types from the previous section and run them down the list of functional patterns to see where a combination matches the business functions that a bank may require. For example consider a customer relationship as the asset type and then work down the list to see where the functional pattern combines well with the customer relationship to form a discrete business functional capacity partition. Some of the obvious functional patterns that match include: MANAGE (Customer Relationship Management), REGISTER (Party/Customer Data Management), ANALYSE (Customer Behavioral Insights & Customer Credit Rating), AGREE TERMS (Customer Agreement).



Functional Pattern		Description	Example(s)
Managemen t & Support Capabilities	DIRECT	Define the policies, goals & objectives and strategies for an organizational entity or unit	Direct a business division of the enterprise
	MANAGE	Oversee the working of a business unit, assign work, manage against a plan and troubleshoot issues.	Manage the day to day activities at a bank branch location
	ADMINISTER	Handle and assign the day to day activities, capture time worked, costs and income for an operational unit.	Administer the time reporting and billing for the specialist sales support team.
	OPERATE	Operate equipment and/or a largely automated facility.	Operate the bank's internal intranet facility
	PROCESS	Complete work tasks following a defined procedure in support of general office activities and product and service delivery functions.	Process the evaluation and completion of customer offers
Resource Managemen t	REGISTER	Capture and maintain reference information about some type of entity.	Register customer reference details in a directory
	DESIGN	Create and maintain a design for a procedure, product/service model or other such entity.	Create and maintain product designs and analytical models
	DEVELOP	To build or enhance something, typically an IT production system. Includes development, assessment and deployment activities.	Build, enhance, test and deploy a major enhancement to a production processing system
	ASSESS	To test or assess an entity, possibly against some formal qualification or certification requirement.	Perform regulatory tests on a proposed financial transaction; check a new offer conforms to an existing contractual agreement
	MAINTAIN	Provide a maintenance service and repair devices/equipment as necessary.	Establish a maintenance and repair program covering the PC technology used in the central offices
Activity Oversight	TRACK	Maintain a log of transactions or activity, typically a financial account/journal or a log of activity to support behavioral analysis.	Maintain a financial journal of transactions processed for a product; maintain a log of customer events and activity for subsequent analysis
	ANALYSE	To analyse the performance or behavior of some on-going activity or entity.	Provide behavioral insights and analysis into customer behavior; analyse financial market activity in order to identify opportunities
	MONITOR	To monitor and define the status/rating of some entity.	Monitor the status and key indicators of a customer to influence on-line interactions; track the status of issued cards for security control
Resource Assignment	AGREE TERMS	Maintain the terms and conditions that apply to a commercial relationship.	Define and maintain the terms govering the contratcual relationship with a customer
	ENROLL	Maintain a membership for some group or related collection of parties.	Administer the memebrship status of a syndicate of investors
	ALLOCATE	Maintain an inventory or holding of some resource and make assignments/allocations as requested.	Track the inventory and administer the distribution of central cash holdings throughout the bank branch & ATM network
Production	FULFILL	Fulfill any scheduled and ad-hoc obligations under a service arrangement, most typically for a financial product or facility.	Perform the scheduled (e.g. statements, standing orders) and ad-hoc fulfillment tasks (e.g.fund transfers) for a current account facility
	TRANSACT	Execute a well bounded financial transaction/task, typically involving largely automated/structured fulfillment processing.	Execute a payment transaction

Figure 7: Functional pattern table

The functional patterns describe a type of function that is performed. To add clarity to the role of the functional pattern, BIAN introduced an associated design element. This is the ‘Generic Artifact’. A generic artifact describes a document or record of some form that may be associated with the execution of the functional pattern – to keep track of and record activity from start to end. The table below shows the different Generic Artifacts defined for each Functional Pattern:



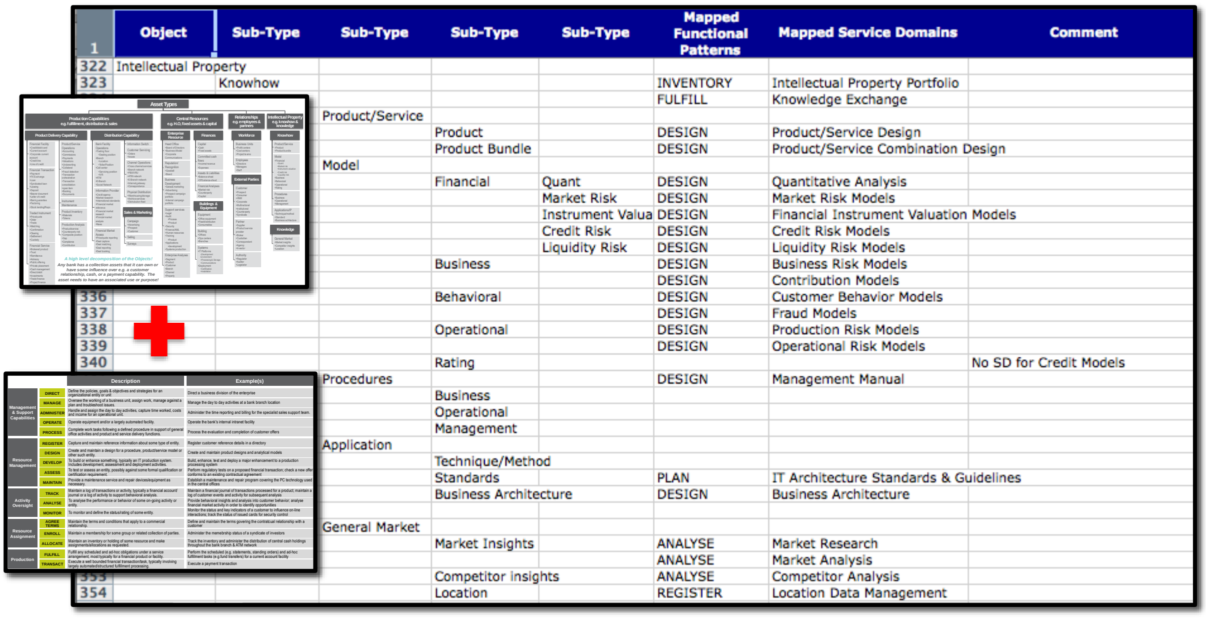
Functional Pattern	Generic Artifact
DIRECT	Strategy
MANAGE	Management Plan
ADMINISTER	Administrative Plan
OPERATE	Operating Session
PROCESS	Procedure
REGISTER	Directory Entry
DESIGN	Specification
DEVELOP	Development Project
ASSESS	Assessment
MAINTAIN	Maintenance Agreement
TRACK	Log
ANALYSE	Analysis
MONITOR	Measurement
AGREE TERMS	Agreement
ENROLL	Membership
ALLOCATE	Allocation
FULFILL	Fulfillment Arrangement
TRANSACT	Transaction

Figure 8: Generic Artifacts

The reason Generic Artifacts have been added is to better describe the working of the Service Domains by referencing something a little more tangible. The generic artifact is combined with the asset type acted on to define a Service Domain's *control record*. The mechanics of a Service Domain and the role of its control record are explained in more detail later but in summary an instance of a control record is created each time the Service Domain fulfils its business role and is used to track activity for the full life-cycle.

For example a Service Domain that handles the asset type 'customer relationship' and performs the functional pattern 'Agree Terms' with the associated generic artifact 'Agreement' has a control record 'Customer Relationship Agreement'. An instance of the control record (Customer Relationship Agreement) is created and maintained for each customer by the Service Domain for as long as they are known and of some interest to the bank. Another example using a more tangible asset type is an ATM network that is acted on by the 'Operate' functional pattern with the generic artifact 'Operating Session' resulting in the control record: ATM Network Operating Session.

BIAN maintains the associations between the asset/entity decomposition, the functional patterns and the associated inventory of identified Service Domains and their control records in the BIAN repository. An Excel based extract of this information is shown below.

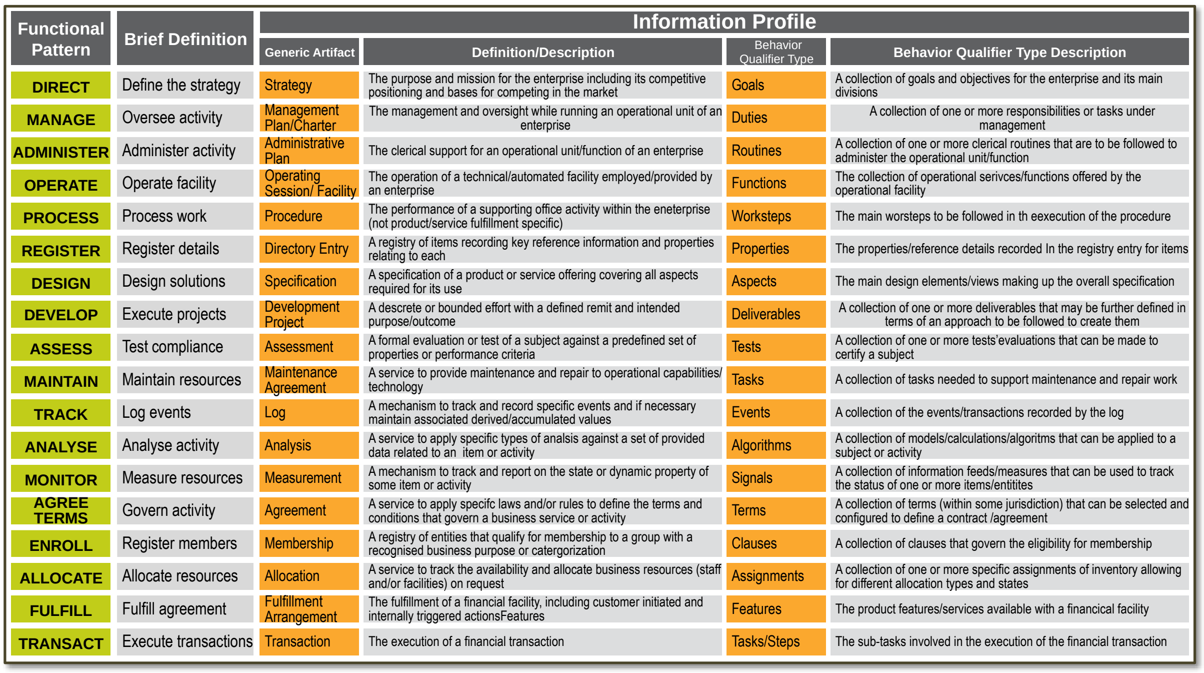


1	Object	Sub-Type	Sub-Type	Sub-Type	Sub-Type	Mapped Functional Patterns	Mapped Service Domains	Comment
322	Intellectual Property							
323		Knowhow				INVENTORY	Intellectual Property Portfolio	
						FULFILL	Knowledge Exchange	
			Product/Service					
				Product		DESIGN	Product/Service Design	
				Product Bundle		DESIGN	Product/Service Combination Design	
			Model					
				Financial	Quant	DESIGN	Quantitative Analysis	
					Market Risk	DESIGN	Market Risk Models	
					Instrument Valua	DESIGN	Financial Instrument Valuation Models	
					Credit Risk	DESIGN	Credit Risk Models	
					Liquidity Risk	DESIGN	Liquidity Risk Models	
				Business		DESIGN	Business Risk Models	
						DESIGN	Contribution Models	
336				Behavioral		DESIGN	Customer Behavior Models	
337						DESIGN	Fraud Models	
338				Operational		DESIGN	Production Risk Models	
339						DESIGN	Operational Risk Models	
340				Rating				No SD for Credit Models
			Procedures			DESIGN	Management Manual	
				Business				
				Operational				
				Management				
			Application					
				Technique/Method				
				Standards		PLAN	IT Architecture Standards & Guidelines	
				Business Architecture		DESIGN	Business Architecture	
			General Market					
				Market Insights		ANALYSE	Market Research	
						ANALYSE	Market Analysis	
353				Competitor insights		ANALYSE	Competitor Analysis	
354				Location		REGISTER	Location Data Management	

Figure 9: Asset decomposition Excel extract

Behavior Qualifiers

In order to provide sufficient detail to the specification of a Service Domain's service operations and governed information the behavior characterized by the Service Domain is further broken down. Based on the Functional Pattern a 'behavior qualifier type' is defined and this is used to list behavior qualifiers specific to the Service Domain. The definition of behavior qualifiers is then used as necessary to clarify the working of the Service Domain and its offered services to provide more precision to their purpose. The behavior qualifiers can also be the basis for defining a more detailed specification of the information governed by the Service Domain. The behavior qualifier types for the Functional Patterns are shown in the table below:



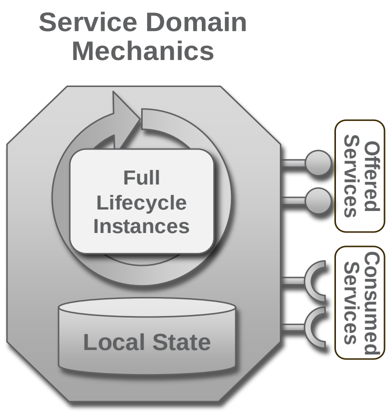
Functional Pattern	Brief Definition	Information Profile			
		Generic Artifact	Definition/Description	Behavior Qualifier Type	Behavior Qualifier Type Description
DIRECT	Define the strategy	Strategy	The purpose and mission for the enterprise including its competitive positioning and bases for competing in the market	Goals	A collection of goals and objectives for the enterprise and its main divisions
MANAGE	Oversee activity	Management Plan/Charter	The management and oversight while running an operational unit of an enterprise	Duties	A collection of one or more responsibilities or tasks under management
ADMINISTER	Administer activity	Administrative Plan	The clerical support for an operational unit/function of an enterprise	Routines	A collection of one or more clerical routines that are to be followed to administer the operational unit/function
OPERATE	Operate facility	Operating Session/ Facility	The operation of a technical/automated facility employed/provided by an enterprise	Functions	The collection of operational serivces/functions offered by the operational facility
PROCESS	Process work	Procedure	The performance of a supporting office activity within the eneterprise (not product/service fulfillment specific)	Worksteps	The main worsteps to be followed in th eexecution of the procedure
REGISTER	Register details	Directory Entry	A registry of items recording key reference information and properties relating to each	Properties	The properties/reference details recorded In the registry entry for items
DESIGN	Design solutions	Specification	A specification of a product or service offering covering all aspects required for its use	Aspects	The main design elements/views making up the overall specification
DEVELOP	Execute projects	Development Project	A descrete or bounded effort with a defined remit and intended purpose/outcome	Deliverables	A collection of one or more deliverables that may be further defined in terms of an approach to be followed to create them
ASSESS	Test compliance	Assessment	A formal evaluation or test of a subject against a predefined set of properties or performance criteria	Tests	A collection of one or more tests'evaluations that can be made to certify a subject
MAINTAIN	Maintain resources	Maintenance Agreement	A service to provide maintenance and repair to operational capabilities/ technology	Tasks	A collection of tasks needed to support maintenance and repair work
TRACK	Log events	Log	A mechanism to track and record specific events and if necessary maintain associated derived/accumulated values	Events	A collection of the events/transactions recorded by the log
ANALYSE	Analyse activity	Analysis	A service to apply specific types of analsis against a set of provided data related to an item or activity	Algorithms	A collection of models/calculations/algoritms that can be applied to a subject or activity
MONITOR	Measure resources	Measurement	A mechanism to track and report on the state or dynamic property of some item or activity	Signals	A collection of information feeds/measures that can be used to track the status of one or more items/entitites
AGREE TERMS	Govern activity	Agreement	A service to apply specifc laws and/or rules to define the terms and conditions that govern a business service or activity	Terms	A collection of terms (within some jurisdiction) that can be selected and configured to define a contract /agreement
ENROLL	Register members	Membership	A registry of entities that qualify for membership to a group with a recognised business purpose or catergorization	Clauses	A collection of clauses that govern the eligibility for membership
ALLOCATE	Allocate resources	Allocation	A service to track the availability and allocate business resources (staff and/or facilities) on request	Assignments	A collection of one or more specific assignments of inventory allowing for different allocation types and states
FULFILL	Fulfill agreement	Fulfillment Arrangement	The fulfillment of a financial facility, including customer initiated and internally triggered actionsFeatures	Features	The product features/services available with a financial facility
TRANSACT	Execute transactions	Transaction	The execution of a financial transaction	Tasks/Steps	The sub-tasks involved in the execution of the financial transaction

Figure 10: Behavior Qualifiers Types

The BIAN Semantic API initiative has developed candidate behavior qualifiers for a selection of Service Domains and these can be found in the latest release. More detailed descriptions of the behavioral qualifiers can be found in the How To Guide – Developing Content

2.3 All BIAN Service Domains Share a Common Operational Structure

All BIAN Service Domains have the same operational structure. As already described the high level role/purpose of a Service Domain is to enable some type of business function (defined as a functional pattern) that is applied to instances of some type of asset/entity. One occurrence of this role being executed for a full life-cycle or from start to finish is managed using an instance of the Service Domain's *control record*. The full life-cycle can be further defined by identifying the main externally visible states that the control record instance and/or the Service Domain may transition through. For example if the role of the Service Domain is to maintain reference details for a supplier it must do this from the time the supplier is first identified, through all possible working arrangements/states until the supplier has no further contact with the bank. If the role of the Service Domain is to operate an ATM network it is responsible for the network activation, any subsequent network re-configuration, all ATM transaction execution support and the eventual termination of the ATM Network operating session.



The BIAN Service Domain is a conceptual design that defines a funtional partition of banking activity:

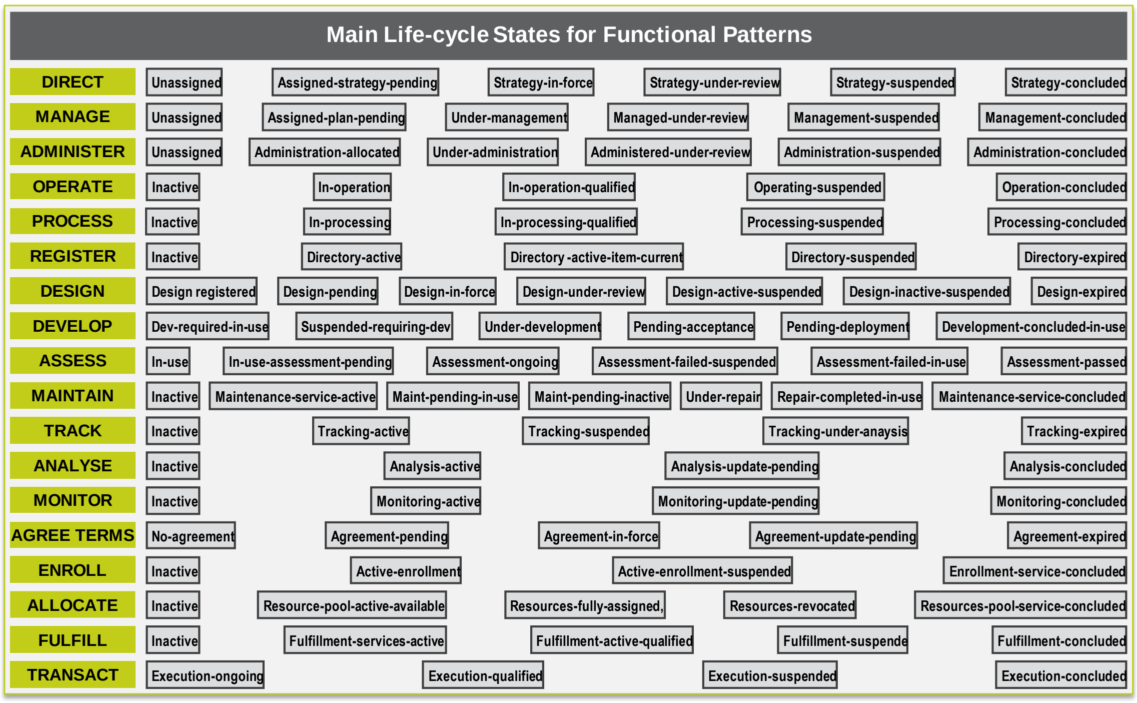
- ***a unique and discrete operational business capability***
- ***all interactions are through offered and consumed services***
- ***it fulfills its business role for the complete lifecycle, for every occurrence***
- ***can act as a stand-alone entity, can even be externally sourced***

The BIAN Service Domain

Figure 11: Structure of a Service Domain

Service Domain Operational States

Later in this guide and in the second guide of the series, BIAN How to Guide – Developing Content, the use of functional patterns, states and standard service operations used to access Service Domains are described in more detail. The types of externally visible states that a Service Domain may pass through depends on its particular functional pattern. Service operation calls may only be allowed when a Service Domain is in one or more of a pre-defined set of states and the service call may itself result in a state change for the individual control record instance or the overall Service Domain. Some examples of external states for selected functional patterns for reference are as follows:



Main Life-cycle States for Functional Patterns

DIRECT	Unassigned	Assigned-strategy-pending	Strategy-in-force	Strategy-under-review	Strategy-suspended	Strategy-concluded	
MANAGE	Unassigned	Assigned-plan-pending	Under-management	Managed-under-review	Management-suspended	Management-concluded	
ADMINISTER	Unassigned	Administration-allocated	Under-administration	Administered-under-review	Administration-suspended	Administration-concluded	
OPERATE	Inactive	In-operation	In-operation-qualified	Operating-suspended	Operation-concluded		
PROCESS	Inactive	In-processing	In-processing-qualified	Processing-suspended	Processing-concluded		
REGISTER	Inactive	Directory-active	Directory -active-item-current	Directory-suspended	Directory-expired		
DESIGN	Design registered	Design-pending	Design-in-force	Design-under-review	Design-active-suspended	Design-inactive-suspended	Design-expired
DEVELOP	Dev-required-in-use	Suspended-requiring-dev	Under-development	Pending-acceptance	Pending-deployment	Development-concluded-in-use	
ASSESS	In-use	In-use-assessment-pending	Assessment-ongoing	Assessment-failed-suspended	Assessment-failed-in-use	Assessment-passed	
MAINTAIN	Inactive	Maintenance-service-active	Maint-pending-in-use	Maint-pending-inactive	Under-repair	Repair-completed-in-use	Maintenance-service-concluded
TRACK	Inactive	Tracking-active	Tracking-suspended	Tracking-under-anaysis	Tracking-expired		
ANALYSE	Inactive	Analysis-active	Analysis-update-pending	Analysis-concluded			
MONITOR	Inactive	Monitoring-active	Monitoring-update-pending	Monitoring-concluded			
AGREE TERMS	No-agreement	Agreement-pending	Agreement-in-force	Agreement-update-pending	Agreement-expired		
ENROLL	Inactive	Active-enrollment	Active-enrollment-suspended	Enrollment-service-concluded			
ALLOCATE	Inactive	Resource-pool-active-available	Resources-fully-assigned,	Resources-revocated	Resources-pool-service-concluded		
FULFILL	Inactive	Fulfillment-services-active	Fulfillment-active-qualified	Fulfillment-suspende	Fulfillment-concluded		
TRANSACT	Execution-ongoing	Execution-qualified	Execution-suspended	Execution-concluded			

Figure 12: End to end states for the functional patterns

Note that these states only define the fairly simple externally visible states that the Service Domain may pass through. There will typically be far more detailed internal states for individual control records. It is also the case for some of the more complex Service Domains in particular that there will be many finer grained functional state cycles contained or encapsulated within the internal processing of the Service Domain. Additional definitions of the state behaviors of Service Domains will be addressed in later releases.

2.3.1 General Service Domain properties

The execution of the Service Domain's role for the full life cycle is tracked/managed using an instance of its 'control record'. Depending on the Service Domain's business role it may only need to handle a single occurrence of its role at any one time (using a single control record instance) or there may need to be many concurrent instances at different stages in their own life cycles. For example the Service Domain responsible for corporate strategy probably has one active control record instance, whereas the Service Domain handling Customer Agreements will have many million concurrent active instances.

Furthermore, also depending on the business role, the life cycle can be quite short (measured in seconds) or could extend over many years. The Service Domain responsible for product designs will have control record instances that last for many years, whereas the Service Domain handling customer interactions may have control record instances that last but a few seconds.

A summary of the Service Domain design properties with some explanatory examples is shown in the next figure:

Some defining Servcie Domain characteristics:

- ***A unique business purpose* – has sole responsibility for fulfilling a specific and discrete business purpose**
- ***It is elemental* – it is not an assembly of other Service Domains.**
- ***Collectively comprehensive* – all possible business activity can be modeled using Service Domains**
- ***Has a 'Control Record'* – the control record reflects its business role or purpose (does something to something)**
- ***Full Life-Cycle support* – it is responsible for all possible states of its control record**
- ***Single or Multiple Instances* – can have a single active instance or multiple active instances of its control record (e.g. a single business unit plan, or multiple customer accounts)**
- ***Short or Long Life-Span* – its life-span can be short or long lived (e.g. a customer interaction or a product design)**
- ***Service Based* – all possible business activity can be modeled as a pattern of service interactions between a suitable selection of Service Domains**

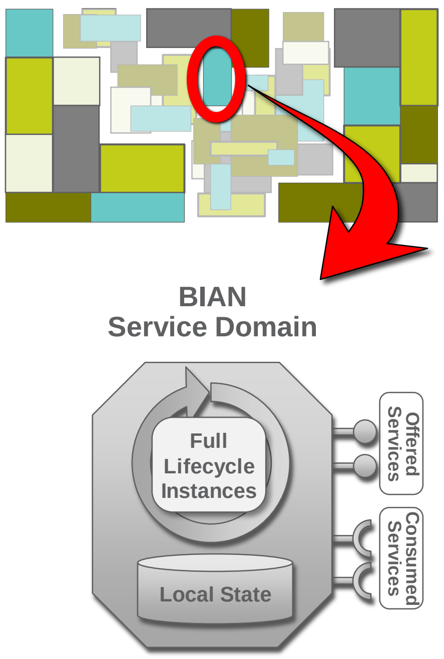


Figure 13: Design properties and Service Domain schematic

The discrete business functional capacity partition represented by a Service Domain can be considered as being broadly equivalent to an organizational unit of the enterprise that combines the 'people, process and technology'. In some cases the corresponding function may be largely automated or it may be people intensive and make limited use of supporting technologies. As BIAN's focus is on improving application interoperability, most attention is paid to the supporting systems' role for the Service Domain and the aspects of the service exchanges between Service Domains that are supported in some way through technology, as illustrated in the next figure.

The Service Domain can be considered as an operational unit combining people, processes and systems. The BIAN standard considers:

- ***Application 2 Application*** – the connections between internal business applications
- ***Semantic*** – exchange specifications are defined in descriptive (non-technical) language
- ***Implementation Independent*** – designs independent of any specific technology
- ***Externalising –*** applications defined in terms of unique, discrete and delegated/shared capabilities
- ***Canonical –*** designs that can be consistently interpreted in any bank/technical environment

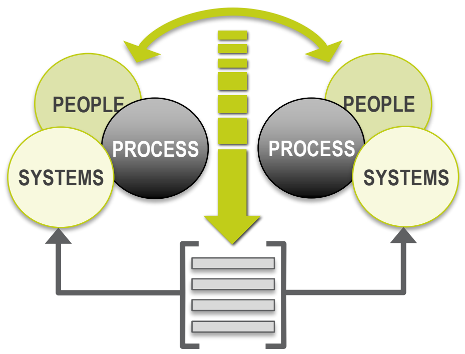


Figure 14: a Service Domain represents an organizational capability

The BIAN specification of a Service Domain includes the definition of its ‘first order events’ for a growing proportion of the landscape. The events are used to model simple ‘first order’ scenarios that show the calling and called Service Domains involved in that ‘primary’ Service Domain’s response to the event. These first order interactions are used to build up a picture of the service operation connections between all of the Service Domains.

As the Service Doman event lists are not comprehensive not all of the possible service operation connections will be identified. But those identified can be used to assemble initial wireframe views that show the main connections between a selection of Service Domains

In practice the description of the Service Domain is not always sufficient to provide an unambiguous definition of the Service Domain for use in solution design and development: additional example descriptive specifications are required. As with the Business Scenarios that provide examples of Service Domain interactions these additional descriptions are not part of the canonical BIAN standard. They simply provide examples of possible Service Domain characteristic’s that can be referenced.

The design additions are not intended to be complete and they may not always apply and may need to be interpreted differently in different implementation scenarios. Two types of design extension to the Service Domains of note are:

1. One is the Service Domain ‘feature list’ that captures prevailing functional and non-functional features of the Service Domain. An example of Service Domain feature table can be found in the third guide of the series: BIAN How to Guide – Applying the Standard.
2. The second is the definition of an information or object model that captures the information/data governed and referenced by the Service Domain. BIAN has started to develop the BIAN business object model (BOM). The BIAN BOM is an extension of the ISO 20022 Business Model with proposed extensions needed to link to the BIAN service domain and control record structures. The

BIAN BOM maps to the Service Domain control records and associated behavior qualifiers as they are defined, providing a semantic definition of the main information elements making up the Service Domain's governed information and referenced by its offered service operations

The details of the Service Domain specification are fully described later in this guide

2.3.2 The Responsibilities of Well Defined Service Domain Could be Outsourced

An informal test of a well-defined Service Domain is that a practical situation can be envisaged where its business function could be offered by a 3rd party organization. This test confirms that the partition represents a discrete business functional partition and that the service dependencies have been fully exposed. It does not require that such an outsourcing arrangement is preferential from a business performance perspective, nor that such an outsourcing arrangement would involve just one single Service Domain rather than a related collection of Service Domains. It merely poses the question: 'if necessary could a third party provide the business service defined by the Service Domain through its defined service operations?'

2.4 Comparing 'utility' and 'functional capacity' approaches to re-use in SOA

Two distinct types of re-use can be associated with service-based designs in general. One referred to here as ***utility re-use*** typically occurs for asset type instances that are found below the threshold of decomposition already described in section 2.2. Below the threshold by definition the asset type instance and its associated use does not have unique business context, meaning it can sensibly occur in two or more unrelated situations.

When an activity occurs in many different unrelated situations it often makes sense to implement the supporting systems solutions in a way that they can be re-used as a utility. Utility systems solutions are valuable and their identification and exploitation is often a goal of good systems design and implementation. Application software can frequently benefit from including re-useable utility software elements that improve functional consistency and reduce development cost/effort.

Though clearly important, this kind of utility re-use is not the main type of re-use supported by the BIAN designs. Opportunities for utility re-use are typically found in an analysis of the systems and technical architecture designs that can be linked to the BIAN business architecture view as defined later in this document.

The main type of re-use associated directly with BIAN business architecture view with its associated Service Domains is another type referred to here as ***operational capacity re-use***. This is where a unique/discrete business functional capacity partition is re-used in the broader context of handling different business events. The earlier example of the monthly credit card payment identified five such re-usable operational partitions, corresponding to the five Service Domains involved.

When using process models, business activity is typically decomposed to fine-gained actions that can be supported by simple (‘input/process/output’) software modules. These can be implemented as service enabled software utilities when they represent logic that could recur many times in different applications. When using the functional capacity model, much coarser grained operational partitions are defined and the design detail is added within them to better describe their internal working, and to the range and detail of their service specifications.

This difference results in two general interpretations of a Service Oriented Architecture (SOA). With process-based analysis the fine-grained utilities can be service enabled to create re-usable software elements that can be used to assemble new applications. With functional capacity based re-use (as used by BIAN) much coarser-grained business operational partitions (not unlike operational business units) are service enabled. This distinction is summarized in the figure below:

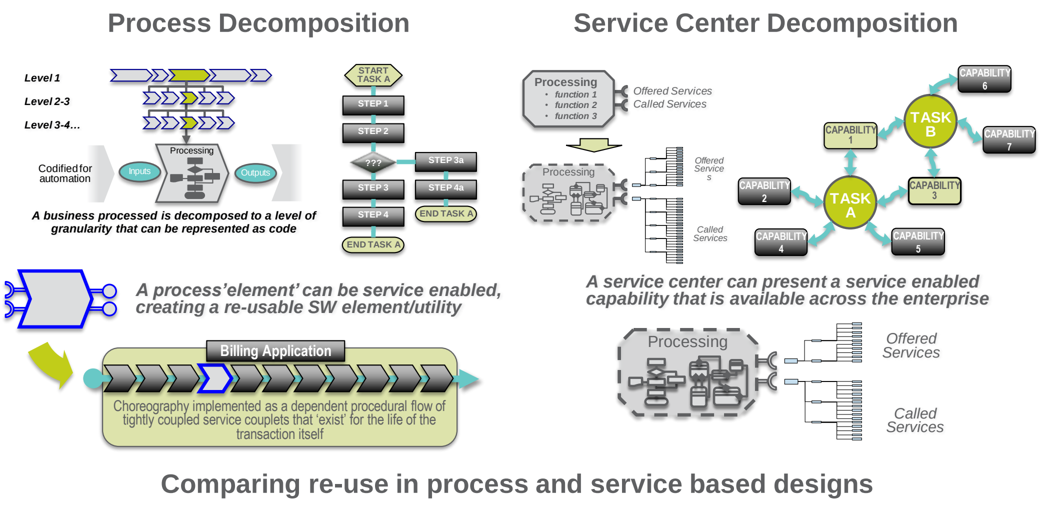

Process Decomposition
Service Center Decomposition
Level 1
Level 2-3
Level 3-4...
Codified for automation
Inputs
Processing
Outputs
A business processed is decomposed to a level of granularity that can be represented as code
START TASK A
STEP 1
STEP 2
???
STEP 3a
STEP 3
STEP 4a
STEP 4
END TASK A
A process’element’ can be service enabled, creating a re-usable SW element/utility
Billing Application
Choreography implemented as a dependent procedural flow of tightly coupled service couplets that ‘exist’ for the life of the transaction itself
Processing
function 1
function 2
function 3
Offered Services
Called Services
TASK A
TASK B
CAPABILITY 1
CAPABILITY 2
CAPABILITY 3
CAPABILITY 4
CAPABILITY 5
CAPABILITY 6
CAPABILITY 7
A service center can present a service enabled capability that is available across the enterprise
Comparing re-use in process and service based designs

Figure 15: Two types of SOA - process & functional capacity oriented

2.5 BIAN Service Domains must be ‘Elemental’

To define a canonical standard (i.e. a specification that can be consistently interpreted in different deployment situations) it is essential that the business functional scope of a BIAN Service Domain is ‘elemental’ in nature. This means that its associated function addresses a single discrete business purpose or role. If a Service Domain supported multiple roles then different combinations of these might be adopted in different deployment conditions. As a result the behavior of the Service Domain and its associated service boundary would no longer be standard/canonical.

As described in section 2.2, the approach BIAN uses to scope a Service Domain is to define it as the combination of an asset type with confirmed unique business context that is acted on by a single dominant functional pattern. This usually results in the definition of an elemental business functional capacity partition.

The exception to this is for some asset types where the decomposition boundary that defines the limit of unique business context may sensibly vary for different banks. For example for one bank the decomposition of types of customer may be appropriate to the level where corporate and consumer customer types are considered to be distinct. In another bank, the definition of a consumer customer type may need to be further categorized to differentiate between banking and card consumer types. For the first bank there would be Service Domains corresponding to functions performed on two types of customer, for the second there would need to be Service Domains covering three customer types.

Another example where the asset type decomposition hierarchy can vary by bank is in the area of product fulfillment capacity. In this area a Service Domain should support the fulfillment of a discrete product/service type. But different types of bank may wish to categorize their product hierarchies with different groupings and levels of precision. In these situations BIAN seeks to find a sensible middle path and individual banks can then adjust the BIAN model to suite their specific product categorization.

For example, consider the categorization of loan product fulfillment. Is it appropriate to define a loan as a single product type? If this is the case there would be a single Loan Service Domain that would need to support many different possible types of loans (such as mortgages, education loan, etc.) with configurable variations to its service operation calls as necessary. Alternatively there could be a collection of different fulfillment service domains for different categories of loan (such as corporate and consumer loans).

Current BIAN guidelines are that product types that have significantly different processing states/cycles and that would also usually be booked as discrete profit centers should be modeled as different Service Domains. As a result the preference in the prior example would be to define discrete Service Domains for different types of loans. These guidelines are being ratified and expanded based on practical experience as BIAN develops more content.

As noted, BIAN seeks to find a mid-point in its Service Landscape by defining Service Domains that can be easily adapted to align to the level of granularity suited to a particular enterprise. The BIAN designs can be interpreted by either concatenating or duplicating and specializing BIAN Service Domains as shown in the next figure:

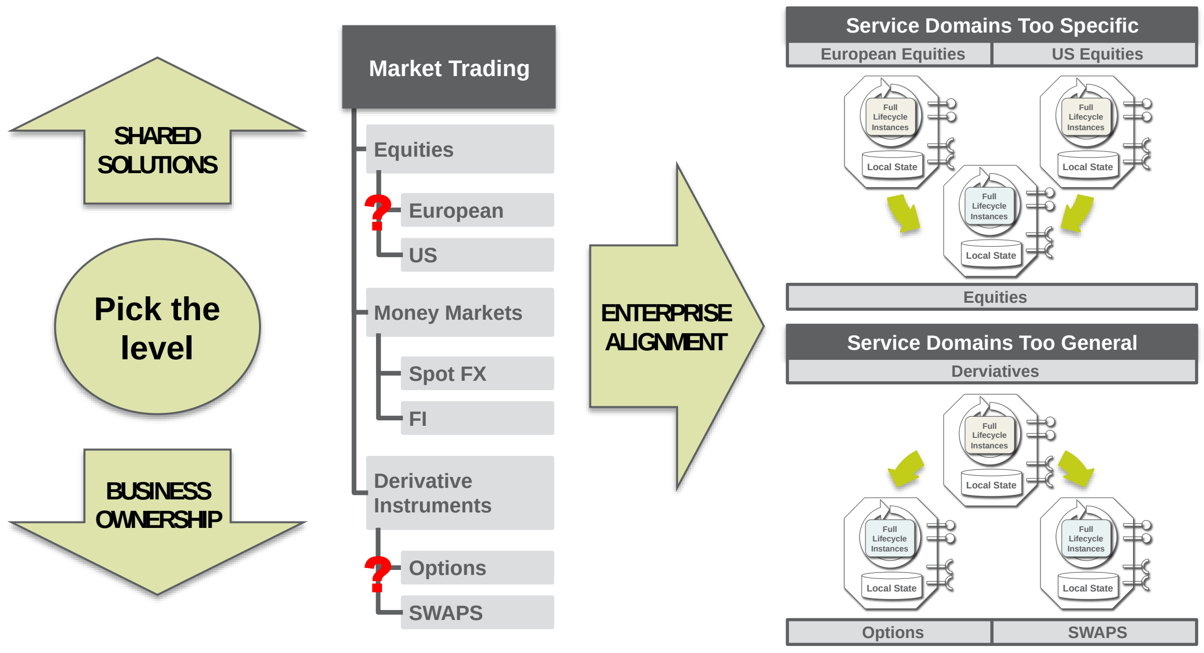


Figure 16: Rescoping a BIAN Service Domain

When two or more Service Domains are combined into a single Service Domain a review of their service operations is undertaken to merge the combination and eliminate any overlaps while retaining all of the features of the source Service Domains. When a Service Domain is duplicated and specialized the service operations are copied and then augmented with any unique features associated with the more specialized functional behaviors.

Note the approach to adapting BIAN Service Domains in deployment is covered in more detail in the third document of the BIAN 'How-to Guide - Applying the BIAN Standard.'

3 Modeling Real World Behaviors – Service Operation Interactions

The BIAN Service Domain designs are refined and further specified by considering various real-world situations and uses that are captured using business events and Business Scenarios. This section covers the concepts and techniques used to ensure the correct specification of the Service Domains and their service operation interactions.

In particular this section describes how the BIAN designs combine a ‘process aligned’ view where specific activities are modeled individually end to end using a business scenario along with a more ‘networked’ view captured in a ‘wireframe’. With the networked/wireframe view the allowed service connections between some collection of related business functional capacities (Service Domains) are captured and any business activity/event can then be traced as a ‘cascade’ of asynchronous service interactions that traverse the connected network of Service Domains as necessary using one of more business scenarios.

First Order Interactions & Business Scenarios

First order interactions are a simple form of BIAN business scenarios that is limited to modeling activity as seen from the perspective of a single Service Domain. In this context this Service Domain is defined to play the ‘primary’ role. These narrowly focused scenarios consider a business event associated with the primary Service Domain and capture the calling Service Domain that typically triggers the response and the resulting delegation calls the primary Service Domain may require to handle the event. This view captures the ‘first order’ connections between this primary Service Domain and any other involved Service Domain as it ‘orchestrates’ its response.

It is also possible that some business events are triggered internally by the primary Service Domain, for example scheduled maintenance tasks. In these cases the scenario will only show the resulting delegated first order service connections. The scenarios that capture business events in this way are useful to establish the required service connections between Service Domains that can then be referenced to create wireframe models. But they are clearly limited in scope in terms of the types and extent of business behaviors they can represent.

In order to distinguish between the business event modeling that is used to establish the first order service connections and more complex business activity the term ‘First Order Interaction’ is now used to describe these more narrowly focused business scenarios that only capture the Service Domain exchanges from the perspective of a ‘primary’ Service Domain.

The term Business Scenario is used to describe business activity that may involve more than one ‘orchestrating’ Service and more importantly has a clear business purpose or goal. This definition of the business scenario aligns with the evolving BIAN business capability discussion presented earlier in Section 2.1 of this guide:

- The First Order Interactions for a Service Domain establish its required service connections. These connections provide an aspect of the definition of the Service Domain as a business functional capacity partition
- A business scenario that includes one or more orchestrating Service Domains and one or more associated events places the Service Domain(s) in the context of business activity with a clearly defined purpose or goal as might be described as a business capability

These relevant design concepts are described in more detail in this section. They are broken down as follows:

1. **Business Events** – For each Service Domain a collection of representative Business event are identified. These can be externally triggered by a calling Service Domain or may be triggered by the internal working of the Service Domain causing it to call/delegate one or more services. The Service Domain’s Business Events are grouped under four established standard categories (Origination, Invocation, Reporting & Delegation) as described in more detail below

2. **First Order Interactions –** Are simple business scenarios that detail the specific service exchanges from the perspective of a ‘primary’ Service Domain in response to a business event. The specification only defines the calling service operation (if appropriate) and any delegated service operation calls that may be required for the primary Service Domain to handle the event. The business event is defined in general terms but does not establish specific business context for which value or goals can be defined

3. **Business Scenarios** – A well-formed business scenario has a clear business start and end point and a business goal or purpose that can have some form of associated performance or value measurement. Unlike a First Order Interaction, the full specification of the Business Scenario includes a more detailed description of these start/end states and the business purpose behind the scenario. This more complete definition can usually be related to the concept of a business capability as described.

   Business Scenarios will usually be more complex/comprehensive than a First Order Interaction, perhaps combining one or more business events from one or more Service Domains. But it is possible for the scope of a Business Scenario to match a First Order interaction for more simple activities. In this case the distinction is that the Business Scenario will add more detailed definition of the business context with start and end positions and a clearly defined business goal. Note: a Business Scenario should always combine one or more First Order Interactions.

4. **Pre and Post Conditions** – First Order Interactions and the more detailed Business Scenarios both have pre and post conditions. In the case of First Order Interactions the pre and post conditions simply define the allowed externally visible states of the primary Service Domain. For Business Scenarios the pre and post conditions are used to define the more specific definition of the business context for the scenario

5. **Wireframes** – a collection of Service Domains with their associated service operation connections for some area of activity or domain of the business can be represented using a ‘wireframe’ diagram. It shows the Service Domains as a loose-coupled service connected network. Business activity can then be overlain on the wireframe as a flow through the network of service operation connections involved for any particular business event/scenario

6. **Service Operations** – The individual service operation interactions between Service Domains are captured with the semantic description of these service operations. More recently BIAN has added additional detail to the specification of service operations and the breakdown of the service operation information payload. This breakdown has been aligned with a parallel effort to define how BIAN service operations are used to define semantic application programming interfaces (APIs) as described in more detail in the associated guide

In order to be able to assemble wireframes for the overall BIAN Service Landscape it is important for BIAN to complete the definition of the provisional First Order Interactions as soon as is practical. However during the latest release cycle focus was shifted to developing the semantic API approach, essentially extending the level of detail for selected Service domains and their service operations rather than continue the expansion of first order coverage across the landscape. The established first order connections do provide a foundation and additional connections will be captured as the content is further developed.

The semantic API BIAN efforts include the definition of representative business scenarios that can be used to ratify and extend the first order connections maintained in the BIAN model. The BIAN central function will continue to support the development of business scenarios and extract from these first order connections to continue to enrich the model in the background. Business Scenario specification developed to support the expanding API effort will continue to be ratified and refined by BIAN Working groups.

The combination of a wireframe providing a ’map’ of the service operation connected Service Domains with a representative collection of example business scenarios defines a high-level design specification of a business area/domain. A key use of this high level design is as a framework for defining application program interfaces (APIs) by providing the context for mapping physical systems and aligning their interfaces to the corresponding service connections. The BIAN approach API design is covered in greater detail in an associated BIAN how to guide. The same high-level design specification can be used in other types of solution evaluation and development as set out in another How To Guide in this series.

First Order Interactions and Business Scenarios

In release V6.0 BIAN published a large number of the comparatively simple first order business scenarios matched to service domain business events. These scenarios have been retained. But with the pivot to semantic API development the focus changed to assembling more complex and specific business scenarios (using established first order interactions where available). The continued development of these more complex scenarios to support the continued expansion of the API coverage across the landscape can be anticipated for the near term. The central team will ensure these scenarios leverage established first order connections and add any newly identified first order connections to the BIAN model.

The business scenario definition work will also be aligned to the BIAN business capability definition initiative as appropriate. This will hopefully help include more business context/detail to define the start and end position, business context/purpose and value associated with the business scenario.

Enforcing Service Oriented Design Disciplines

A key design principle for service orientation is that the entity calling a service should not need to know anything about how that service is provided either in terms of the internal logic of the service provider or any down-stream service dependencies it may have. These features should be fully ***encapsulated*** within the service provider.

The use of First Order Interactions when assembling Business Scenarios can help enforce good encapsulation design principles. In practice the ability of a service provider to fulfill offered services will frequently be dependent on services that they delegate to other service providers. There are many design options that can support the concept of encapsulation for example:

1. **Constraining the Offered Service** – the offered service when implemented in production should include performance goals. So for example the committed response time for an offered service can be set at a level where any internal processing and/or downstream dependencies are fully provisioned. I.e. the Service Domain does not commit to offering something that may not be delivered

2. **Internal Decoupling** – the offering Service Domain can be designed to decouple its downstream dependencies as much as possible. For example if a Service Domain relies on another service provider to provide key information, it can arrange to have this information in advance or in a frequently refreshed form so that it can respond to service requests without having to make a ‘nested’ or dependent delegation call before it can respond

The First Order Interactions that are used to assemble business scenarios can be referenced in more detailed implementation design cycles to identify the linked dependencies and adopt suitable design approaches to ensure the services are as well encapsulated as practical in implementation.

Each of the relevant behavior modeling design concepts listed above are now explained in more detail.

3.1 Modeling Service Domain Business Events

Service Domain Business Events have been used to identify example First Order Interactions that can be used to demonstrate the operation of a selected Service Domain – referred to as the primary Service Domain for the defined events. The business events are described in terms of the necessary actions as seen from the perspective of the primary Service Domain. Note that the focus of BIAN for the latest release cycle has been on extending the service domain definitions to support API specifications. As a result there has not been any significant development of the definition and application of Service Domain business events in the latest release cycle. The use of events described here is likely to be refined and continued in future releases.

There are some wide ranging business activities that may trigger a range of concurrent business events in many different Service Domains that may or may not be directly connected. For example when a bank establishes a new customer relationship many different parts of the bank may need to respond and these responses may not always be fully synchronized/coordinated across the enterprise. To fully handle such an activity several First Order Interactions would need to be defined: one for each Service Domain that is impacted by the arrival of the new customer.

The working assumption motivating the definition of First Order interactions is that any and all required service operation connection between Service Domains can be matched to underlying business events and service operation exchanges at the first order. As BIAN defines more business events and their associated interactions the BIAN model will contain an increasing proportion of all possible/sensible service operation connections in its ‘database’. These service operation connections can then support the derivation of wireframe models and support the assembly of more sophisticated business scenarios.

The business events identified for a primary Service Domain are grouped under four categories. These categories called ‘responsibility types’ are also used to classify service operations. The business events listed for each Service Domain provide examples of activity that triggers a response on the behalf of the primary Service Domain. The different responses can be further related to the way the Service Domain handles the control record instances it manages to govern its execution:

> ***Origination*** – origination events result in the creation of a new control record instance (or for Service Domains that handle some persistent operational activity can configure/re-configure that activity). For example register a new

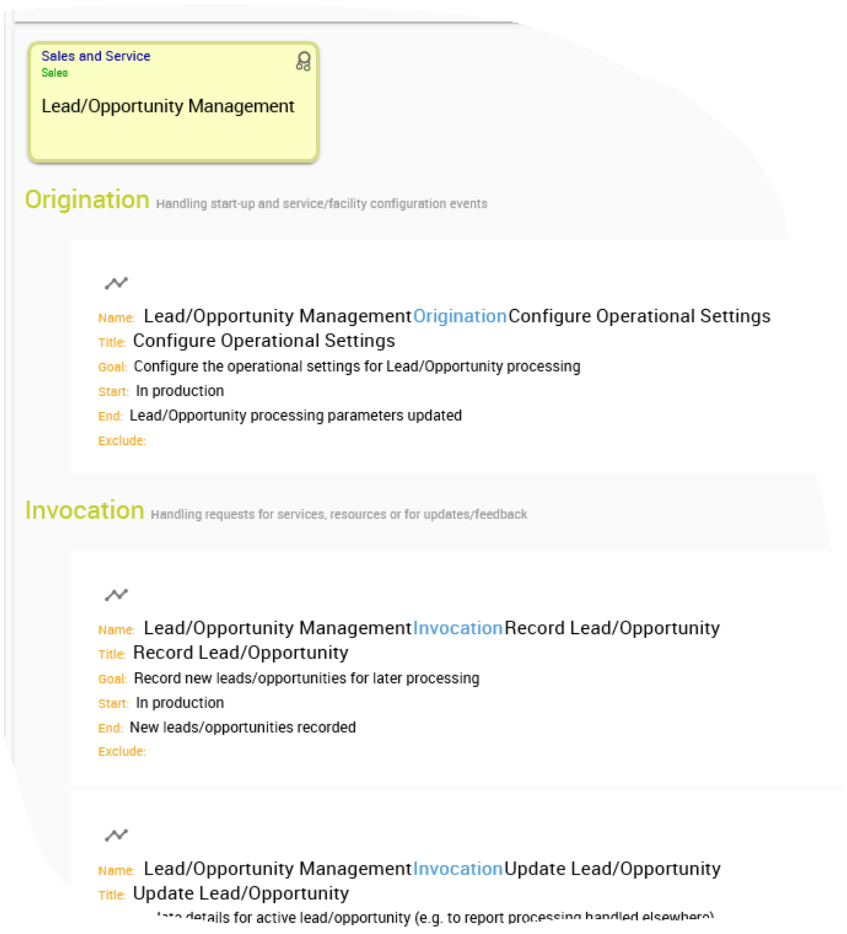
customer agreement or reconfigure the priorities for the guideline compliance business unit/function.

Invocation – invocation events initiate processing activity associated with an existing control record instance. For example update the details of an existing customer agreement, perform a check that a proposed activity conforms to the terms of that agreement or book a transaction against a financial facility.

Reporting – reporting events include both reporting requests and pre-configured notification services where the primary Service Domain provides details of its active control record instances and/or any historical and analytical views it might maintain. For example provide the current terms of an active agreement or an analysis of updates made to customer agreements over a period. Reporting events differ from Invocation events in that there is no work done or content change made to the control records themselves

Delegation – delegation events are internally triggered by the primary Service Domain. They may arise from some internal scheduled processing or may be needed to support the handling of an offered service. The events capture a dependent service operation call to some supporting Service Domain that is needed for the primary Service Domain to fulfill its role. For example the Customer Agreement Service Domain may delegate to Regulatory Compliance to obtain compliance requirements and to subsequently confirm compliance. Note: the BIAN designs assume that there is no formal connection between an offered service and any delegated service calls that may result – they are treated as fully decoupled activities

The purpose of a Business Event is to identify an example business trigger that causes the focus Service Domain to act. The business event can then be modeled using a simple form of business scenario: the ‘first order interaction’ that defines the service operation exchanges. An example collection of business events is shown for a Service Domain in the next figure (note this view is a screen shot taken from BIAN tooling)


Sales and Service
Sales
Lead/Opportunity Management

Origination Handling start-up and service/facility configuration events

Name: Lead/Opportunity ManagementOriginationConfigure Operational Settings
Title: Configure Operational Settings
Goal: Configure the operational settings for Lead/Opportunity processing
Start: In production
End: Lead/Opportunity processing parameters updated
Exclude:

Invocation Handling requests for services, resources or for updates/feedback

Name: Lead/Opportunity ManagementInvocationRecord Lead/Opportunity
Title: Record Lead/Opportunity
Goal: Record new leads/opportunities for later processing
Start: In production
End: New leads/opportunities recorded
Exclude:

Name: Lead/Opportunity ManagementInvocationUpdate Lead/Opportunity
Title: Update Lead/Opportunity
...ate details for active lead/opportunity (e.g. to report processing handled elsewhere)

Figure 17: Example Business Events for a primary Service Domain

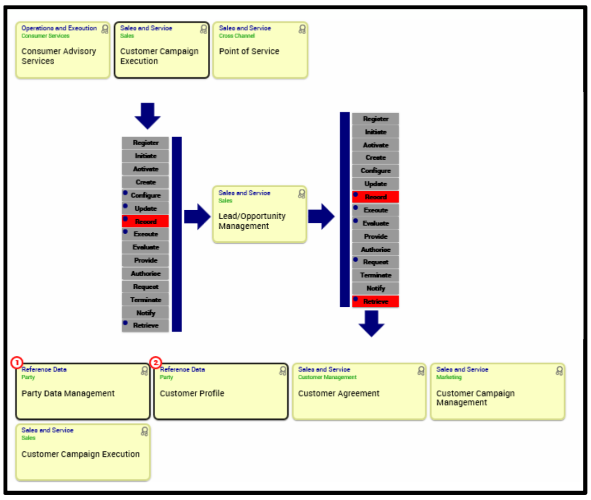
3.2 Modeling First Order Interactions

Each business event that is identified for a focus Service Domain is then used to define a First Order Interaction that as noted earlier is a constrained/limited type of business scenario. The First Order Interactions have a derived name that concatenates three elements: The focus Service Domain’s name, the category of the business event and the descriptive name of the business event. So for example the First Order Interaction associated with the ‘new customer’ business event as handled by the Customer Agreement Service Domain has the derived name: ‘CustomerAgreement|Origination|NewCustomer’.

Every First Order Interaction is modeled as a simple constrained business scenario/first order interaction. This is currently represented as a business scenario in the BIAN repository with the current release. As noted below as BIAN develops more sophisticated business scenarios a more formal distinction will be made between these First Order Interactions and the more comprehensive business scenarios. The interactions include the following elements:

- ***Calling Service Domain (Optional)*** – the interaction may optionally identify a 'calling' Service Domain. This is a Service Domain that calls the offered service of the primary Service Domain that is associated with the business event. There can in practice be more than one candidate Service Domain to call the service operation associated with the business event and if necessary several First Order Interactions could be developed. For example the request to set up a new customer agreement described earlier might come from different operational centers (and their associated Service Domains) in a bank. The calling Service Domain is an optional aspect of the First Order Interaction as there may be business events that are internally triggered by the primary Service Domain
- ***Primary Service Domain*** – the main aspect of the First Order Interaction captures the likely sequence of delegated calls made by the primary Service Domain in response to the called service operation or its own internal triggers
- ***Called Service Domains*** – the First Order Interaction will list one or more Service Domains that the primary Service Domain calls on/delegates to in order to fulfill its responsibilities associated with the business event. As noted any secondary service exchanges that these called Service Domains may need to make are not captured in the First Order Interaction as these represent second order/'dependent' connections.

BIAN has developed tooling support to help with the definition and reference to First Order Interactions. The screen capture below shows the way the BIAN Workbench tool captures a First Order Interaction as a simple business scenario. The complete user guide for the BIAN Workbench tool is covered in a separate publication.



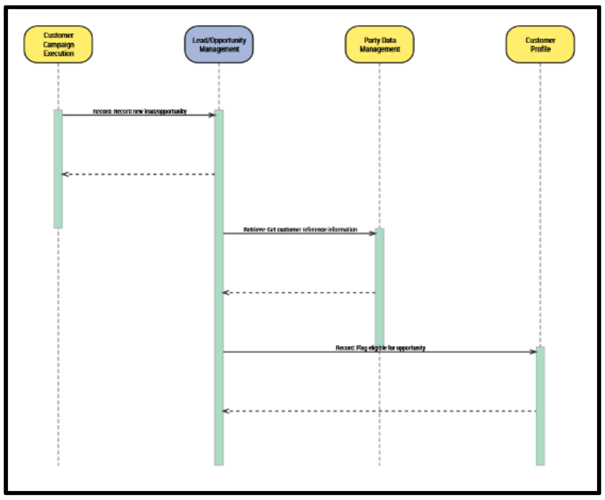

Customer Campaign Execution
Lead/Opportunity Management
Party Data Management
Customer Profile
Record: Record new lead/opportunity
Retrieve: Get customer reference information
Record: Flag eligible for opportunity

Figure 18: Example First Order Interaction

3.3 Modeling Business Scenarios

As already noted as BIAN expands its coverage of Business Scenarios a more formal distinction will be made between the simple business scenario layout used to capture the large number of first order interactions for all Service Domains and the more comprehensive Business Scenarios in future releases. The updated Business Scenario definition will add more business context, in particular detailed descriptions of the start and end business condition/situation, a more detailed definition of the business goals/purpose and possibly some form of impact/value measurement. This more involved definition aligns with the work being done to add a business capability perspective to the BIAN model.

A Business Scenario is intended to represent an archetypal business behavior. A number of qualifications can be made of the derived Business Scenarios that can be found in the BIAN service landscape:

- ***Not Canonical*** - the Business Scenario is not canonical (it does not define a standard pattern of behavior). It simply provides a realistic example of business behavior that is used to provide context for defining the service operational characteristics of the Service Domains involved
- ***Not Prescriptive*** - the Business Scenario is not intended to define a specific pattern or sequence of behavior that should be followed, nor does it necessarily attempt to be exhaustive/complete in its content.
- ***Not Discrete*** – the pattern of service operation interactions captured using a Business Scenario for one business situation can and often will overlap with other Business Scenarios

Business Scenarios are used to establish the service operation connections between a suitable collection of related Service Domains that are involved in the execution of some kind of business requirement or need. It models an archetypal pattern of service operation exchanges between all of the involved Service Domains. Unlike a process representation that defines the precise logic and sequence of tightly coupled tasks, the BIAN business scenario simply identifies the Service Domains involved and highlights likely service exchanges that could occur between them. It does not presume the sequence nor the timing and protocol for the exchanges, it merely represents that some form of business interaction between the business capacity functions represented by the Service Domains is possible.

The scenario does not mandate the use of any particular service operation in any specific situation that it may present. Furthermore a BIAN business scenario need not be exhaustive in terms of the participation of service domains it reveals, the steps it includes or its start and end conditions – it is merely providing meaningful business context to help describe the particular service exchanges that it includes. The original purpose of the business scenario is to discover and clarify service operation connections/exchanges between Service Domains by providing the example context of a real world business situation.

Despite the many limitations just described, the BIAN Business Scenario has been found to be a powerful tool for interpreting and applying the BIAN standard. As a result a more structured approach to defining the scope and coverage of Business Scenarios is being adopted in BIAN. The use of the Business Scenario has evolved from being a tool to identify and ratify service connections to being a way to present how Service Domains interact to support specific business activity.

BIAN Working Groups and other BAN initiatives continue to define collections of representative business scenarios that are needed to define the main activities/working of aspects of the business. A current area of focus is using Business Scenarios along with other BIAN artifacts to support API design and development.

Various tools and formats are used within BIAN to capture and present business scenarios. Below a Powerpoint version is used to outline the standard design content.

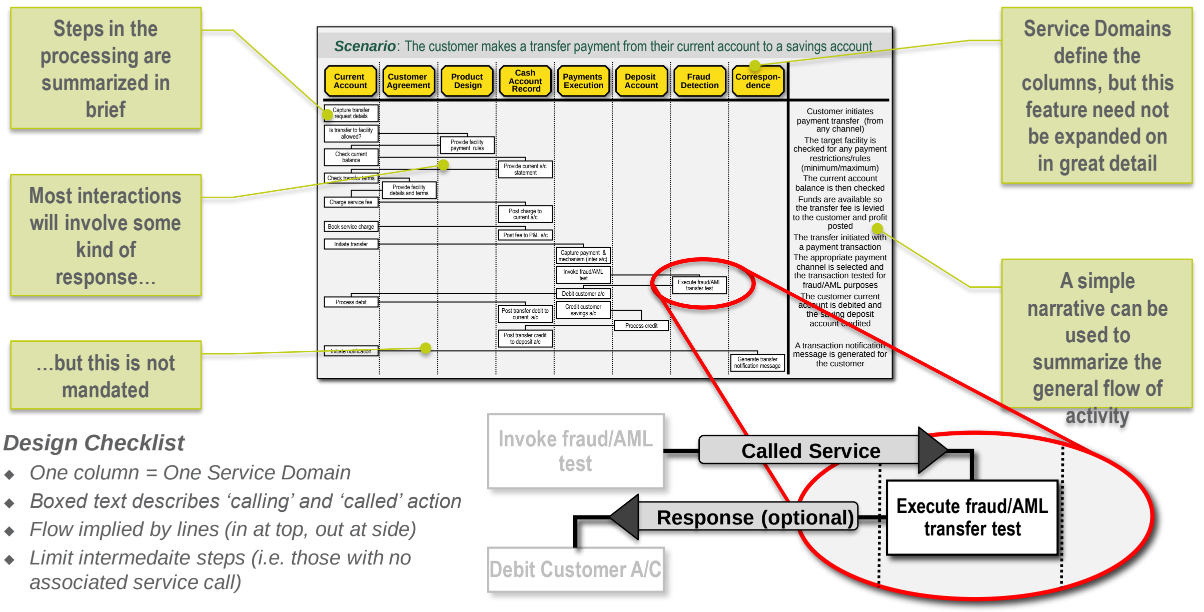

Steps in the processing are summarized in brief
Most interactions will involve some kind of response…
…but this is not mandated
Scenario: The customer makes a transfer payment from their current account to a savings account
Current Account
Customer Agreement
Product Design
Cash Account Record
Payments Execution
Deposit Account
Fraud Detection
Correspon-dence
Capture transfer request details
Is transfer to facility allowed?
Provide facility payment rules
Check current balance
Provide current a/c statement
Check transfer terms
Provide facility details and terms
Charge service fee
Post charge to current a/c
Book service charge
Post fee to P&L a/c
Initiate transfer
Capture payment & mechanism (inter a/c)
Invoke fraud/AML test
Execute fraud/AML transfer test
Debit customer a/c
Process debit
Post transfer debit to current a/c
Credit customer savings a/c
Process credit
Post transfer credit to deposit a/c
Initiate notification
Generate transfer notification message
Customer initiates payment transfer (from any channel)
The target facility is checked for any payment restrictions/rules (minimum/maximum)
The current account balance is then checked
Funds are available so the transfer fee is levied to the customer and profit posted
The transfer initiated with a payment transaction
The appropriate payment channel is selected and the transaction tested for fraud/AML purposes
The customer current account is debited and the saving deposit account credited
A transaction notification message is generated for the customer
Service Domains define the columns, but this feature need not be expanded on in great detail
A simple narrative can be used to summarize the general flow of activity
Design Checklist
One column = One Service Domain
Boxed text describes 'calling' and 'called' action
Flow implied by lines (in at top, out at side)
Limit intermedaite steps (i.e. those with no associated service call)
Invoke fraud/AML test
Called Service
Execute fraud/AML transfer test
Response (optional)
Debit Customer A/C

Figure 19: An example business scenario with guidelines

Note the figure above does not include the additional design content that will be added in future releases including pre and post states, goals and performance measures. There has also been a refinement to the business scenario to support the API initiative. This refined business scenario representation is shown in the corresponding API How to Guide.

The main distinction between a business process and a BIAN Business Scenario has been described in the previous Section of this guide. However this comparison has been applied for business processes that are defined the level of granularity where the behaviors can be sensibly mapped to service operations between Service Domains. One strength of process modeling is that it can be applied at varying levels of detail where as BIAN business scenarios are constrained to being defined only at the level of service operation exchanges between Service Domains. The way process models at other levels of detail relate to the BIAN view are as follows;

- **Processes at a higher level of detail** – there are few obvious situations where a business activity or event is so high level that it can't be captured as a collection of one or more Business Scenarios. One example would be where a process view is used to capture business exchanges/agreements between commercial entities in the financial services industry

- **Processes captured at the same level** – for defining business requirements it is common to find process definitions at about the same level of detail as the typical BIAN Business Scenario. The role of the Service Domain can be loosely compared to 'Actors' in some process model formats. But as noted, the tightly coupled sequence of the process has no equivalent in the Business Scenario
- **Processes at a finer level of detail** – process models can also be used to define much finer grained activities as might be found within the internal logic of a Service Domain. BIAN does not presume any particular internal logical design for a Service Domain. The best technical design approach should be selected based on the Service Domain's desired operational characteristics. For example multi-threaded process design versus a state driven object based design.

Distinguishing between transactional and reference and reporting traffic

Many Service Domains need to access reference business information that is governed by other Service Domains in the execution of their role and also provide performance reporting to management. For example, the Service Domain maintaining customer contract details needs to be kept up to date with changes in the reference information for that customer (such as their legal domicile). Sometimes this reference information is exchanged within the flow of a transaction as might ba captured in a business scenario – for example checking a customer's available balance before making a payment. But a significant portion of this type of information exchange occurs in the background based on implied or explicit notification agreements established between Service Domains.

The BIAN Business Scenarios developed to date have concentrated on modeling transactional activity and exchanges. The addition of business events and First Order Interactions has highlighted other types of service operation exchanges. These exchanges support the background coordination of reference information, such as standards, policies, guidelines and budgets. In addition they support the exchange of performance analysis and other management reporting flows that may be needed to govern business operations

Depending on the particular role of a Service Domain its service exchanges with other Service Domains will typically include transactional exchanges but may also involve some proportion of background reference and management reporting information exchanges. The business events and First Order Interactions include reference and management reporting exchanges in addition to the more familiar transactional traffic and will help identify these less obvious, background interactions.

3.4 Pre & Post Conditions

The definition of a business scenario can specify associated pre and post conditions. For the simple First Order Interactions this definition is limited to defining the allowed externally visible states of the Service Domain. This provides some basic insight into when a service exchange may or may not be allowed.

The definition of pre and post conditions will increasingly be used to establish the business context for more complex business scenarios. One particular use of pre and post conditions will be to highlight dependent sequences of interactions between events and their associated Business Scenarios. For example noting at the conclusion of the handling one event that another particular event will typically follow. This helps establish the start and end position of a scenario and can also be useful to clarify the related flow of activity between business scenarios by highlighting the dependency in the post condition description and corresponding pre-condition description.

For example: customer offer processing when successful will typically initiate (delegate) product set-up in the product fulfillment Service Domain once the offer processing has been completed. This will be captured as a delegated service call with an associated business event for the Customer Offer Service Domain as can be highlighted with an associated description in the post condition field of the offer processing scenario.

3.5 Wireframes

Wireframe models have been described in earlier versions of the BIAN How To Guides. As BIAN designs are increasingly being used in design specifications wireframes are proving a useful mechanism to provide a stable framework for mapping Service Domains to systems solutions.

The wireframe diagram captures the known or ‘allowed’ service operation exchanges available between a suitable selection of Service Domains. It presents a stable framework or a ‘static’ structure. A Business Scenario on the other hand describes a dynamic behavior. It shows the flow of Service Domain service operation exchanges resulting from some business event or request. Business Scenarios can be traced as a ‘point-in-time’ flow of service interactions that traverses the wireframe.

The value of the wireframe is that it represents the scope of business capabilities in a stable format that can be readily related to some area of the business. For example a wireframe can be defined to cover the responsibilities of a business unit or mapped to the functional coverage of a business application. The required dynamic behaviors can then be captured using as many business scenarios as necessary that can be overlain on the wireframe to confirm the service exchanges and operation of the Service Domains meet behavioral requirements.

A variation of the wireframe perspective called a ‘Service Domain Cluster’ is described later in this guide. A cluster also groups Service Domains as they might map to a business unit or application but also allows for Service Domains to have different implementation roles within the cluster. In this way the cluster can be used to define the service boundary for a physical business application. This service boundary then provides a framework for defining the application implementation integration requirements.

The final format of the BIAN wireframe view has not been finalized. In current uses a less formal notation is used where the links between Service Domains that represent a service operation connection simply need to connect to any point on the boxes representing the Service Domains.

In later releases a more formal representation may be considered where the connection point to the Service Domain corresponds to key properties of the service exchange. The connection point on the Service Domain element is determined by the type of service operation (as characterized by the service operation 'action term') and whether the service operation is offered or is a call/delegation. The positioning also reveals whether a new control record instance is created as a result of the service operation call or whether the exchange acts on existing control record instances.

This additional detail is expected to be useful when BIAN extends the specification of internal states for Service Domains. The figure below summarizes the general positioning of service operation connections to a Service Domain in the more formal version of a wireframe

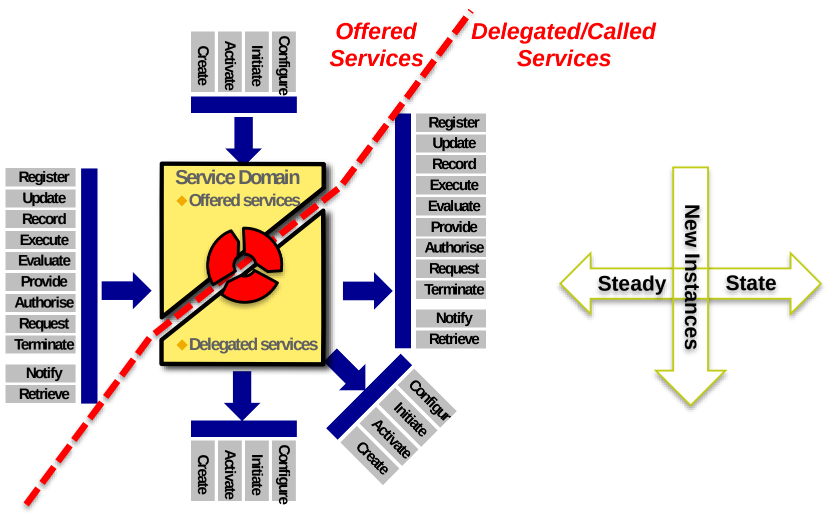


Figure 20: Service Domain's general Service Connections

The links between Service Domains in the wireframe correspond to an allowed service operation exchange. The connection is labeled with the associated service operation 'action' term (as defined later in this section) and the arrow points from the calling to the called Service Domain. (As noted, in the current release the notation of the wireframe allows the service operations can connect to any point on the calling and offering Service Domain – a notation that may be revised in future releases)

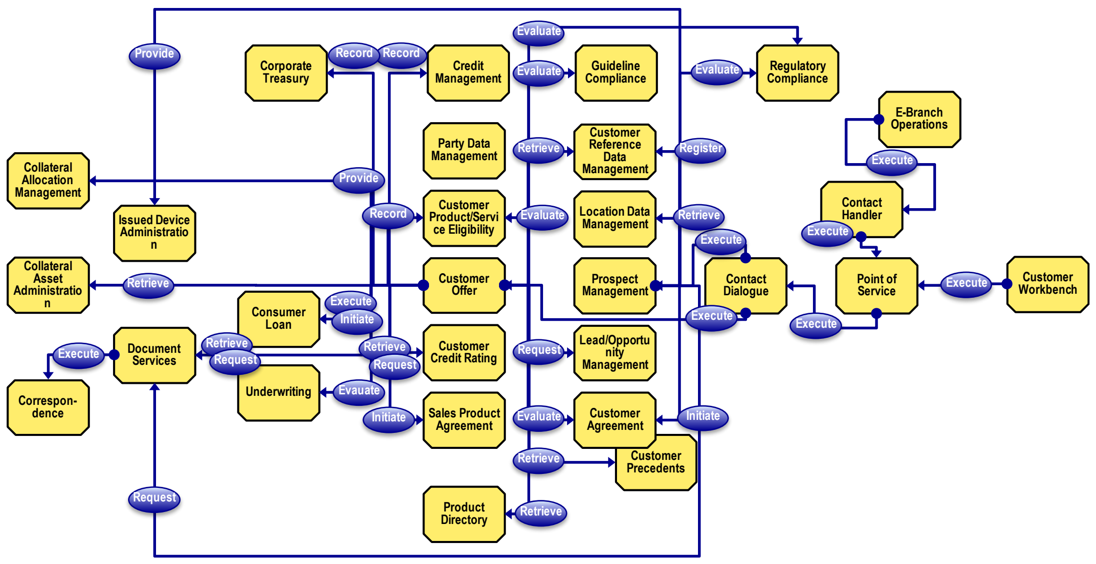


Figure 21: Wireframe example - informal connections

For information purposes only, a wireframe where the service operation connections align with the type of service operation connections described earlier looks as follows:

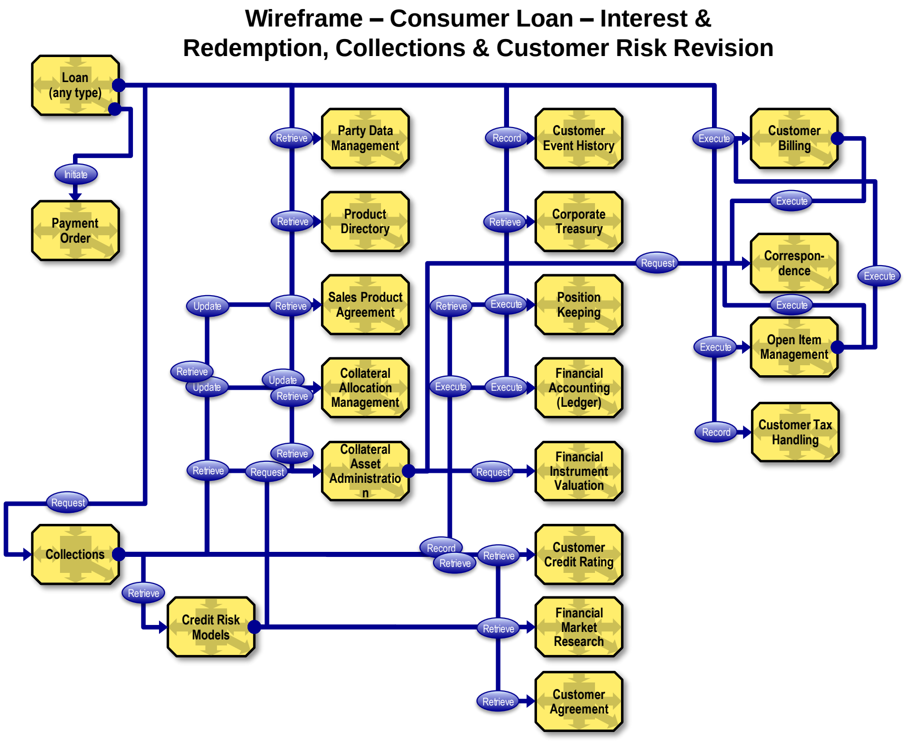


Figure 22: Wireframe example - formal connections

The value of the wireframe model is that it defines a static or stable structure that is more easily related to the organization or business applications of a bank. The Business Scenarios remain a key aspect of the standard by describing the allowed behaviors that are supported and these can be mapped as appropriate to a wireframe model view.

3.6 The Semantic Definition of Service Operations

The core of the BIAN industry standard is the semantic definition of the service operations offered and consumed by Service Domains. BIAN’s service operation specifications are intentionally ‘implementation agnostic’ - the specification includes nothing that relates to any possible technical implementation. This includes communication protocols and the likely message exchange choreography. A Service Domain exchange that is captured in a BIAN service operation as a semantic description of information provided and information returned may in practice result in a broad range of physical exchanges not limited to:

- A simple one-way delivery of information possibly with acknowledged receipt
- A simple request for information with a timely complete response
- An interactive dialogue to progressively narrow in on the required information by developing increasingly detailed query criteria
- Any of the above types of information request that results in a delayed response at some point in the future
- Any of the above types of exchange that results in the allocation of some facility or resource
- Any of the above types of exchange that is accompanied by the physical movement of goods or other resources.

When using the BIAN semantic service operations to specify an interface or service interaction it is necessary to determine the operational characteristics of the many exchanges. This definition will be site specific and so no operational properties of the exchange are defined, simply the make-up of the business information that is exchanged and any associated service fulfillment dependencies.

A general guideline used within BIAN when defining a Service Operation is that to be complete, the semantic service operation information content should include all of the main business concepts involved in an unambiguous way such that a basic technical design can be developed to implement the exchange without further significant business input (other than to perhaps provide clarification of detail). However, the BIAN description is only intended to describe the mainstream features of a service operation: those that are anticipated to apply in most deployments. So for example additional business input would be needed to specify unique/differentiating, advanced or location specific needs.

It is assumed that the BIAN designs will be referenced by individuals already expert or at least highly familiar with the subject area. The intention of the BIAN specification is to establish the business purpose and core information exchanged through the service operations and to clarify the functional capability partitioning represented by the Service Domains. It is not the intention of the BIAN standard to provide an exhaustive specification and explanation of requirements (as might be expected in an educational or reference definition).

In time BIAN may augment the main/common Service Domain and service operation information content definitions to also capture optional extensions/specializations that could be relevant based on qualifications such as geopolitical context, scale/performance requirements and/or, level of sophistication. At this time BIAN's focus is on defining only the common/mainstream specifications.

Some specializations may also be needed to deal with different implementation situations and technical environment considerations including the integration of major packages, proprietary standards and technologies. These implementation level details may be also recorded using some appropriate mechanism and linked or mapped back to the BIAN standard for reference and reuse purposes. However these particular specifications would not be part of the BIAN standard which is intended to remain implementation (and commercial product) agnostic.

The structure of a BIAN service operation specification is explained in more detail in the second document of the BIAN 'How-to Guide - Developing Content.' The specification includes three primary design considerations/concepts that are outlined in this guide:

1. Allowed types of exchange – these have been discontinued
2. Standard service operation parameter types – each service operation is made up of a standard collection of parameters
3. Service operation standard action terms – a standard list of allowed action terms defines the main types of service operations
4. At a minimum the BIAN service operations include a checklist of the types of information exchanged. In the latest release the checklists are replaced with a more comprehensive list of attributes for selected Service Domains. Furthermore there is an ongoing collaboration between BIAN and ISO to map the content to the ISO20022 Business Model
5. Checklist Information is referenced in the BIAN Vocabulary – all BIAN specific terms including the definition of checklist items is included in the BIAN vocabulary tool

These are set out in more detail below:

1. Allowed types of exchange – in previous releases BIAN defined four types of exchange that characterized the operational dependency between the involved Service Domains. In practice it has been found that the same service operation can sensibly be implemented using several of these types of exchange depending on local conditions. As a result the classifications add little insight and can be misleading. The classifications have been discontinued. The selection of the appropriate protocol for the service operation exchange is now a task that has been moved to the implementation level design. For reference the exchange types that have been discontinued are outlined below:

A two-way exchange – the calling Service Domain expects the response immediately so that it can continue with its work.
A request with an anticipated delay in the response – the calling Service Domain anticipates that the response will take some time and so will continue with other work and monitor for the expected response.
A hand-off notification – the calling Service Domain once passing on the necessary detail has no further operational interest in what the

called Service Domain does (no response expected other than possible acknowledgement of receipt).
Provision of previously subscribed-to updates – the calling Service Domain has at some point subscribed to updates from the called service domain.

2. Standard service operation parameter types – a BIAN service operation has four parameter types that capture the information making up the payload of the service operation. The same four parameters describe the content of the call and response aspects of the exchange:

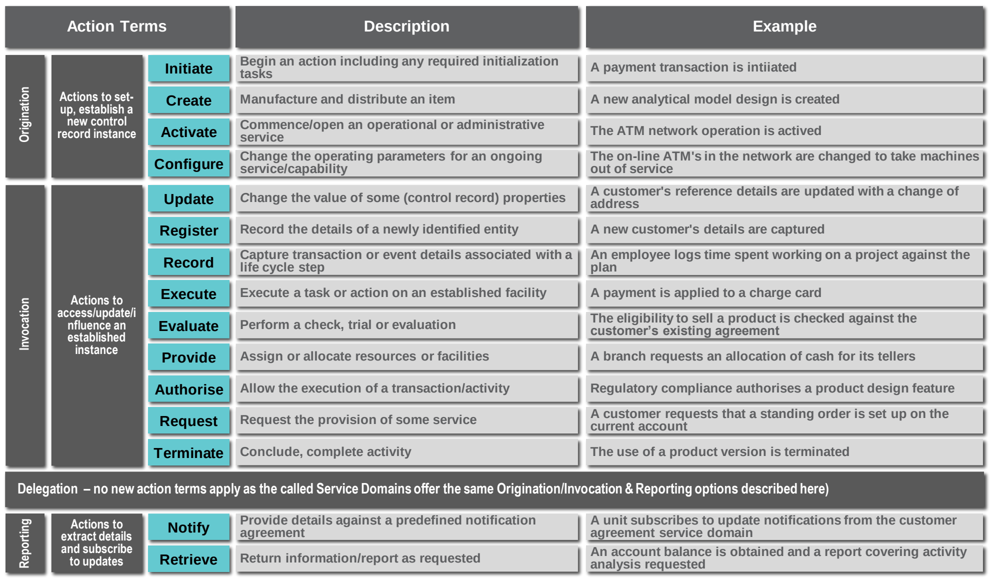
Identifiers – define the information items that can be used to isolate the control record instance or collection of instances that are being accessed.
Depiction – represents the information content of the control record that may be provided or returned.
Instructors – defines 'parameters' governing the requested action such as timing, priority and any action selection options (this can include the specification of reporting/query details).
Analysis – references any tracked/derived values associated with one or some combination of control record instances that provide historical and or analytical views (as would be maintained by the called Service Domain).

The main content of the service operation will be a selection from one or more control record instances that are governed by the offering Service Domain. The way the control record maps to the service operation varies slightly depending on the type of service operation as indicated by its action term. In general terms the control record content maps as follows:

a. ***Identifiers*** – are extracted information items taken from the control record that can be used individually or in combination to uniquely identify the instance.
b. ***Depiction*** – contains extracts or the complete content of one or more control record instances – this is typically the main payload of most service operations.
c. ***Instructors*** – have no connection with the control record content other that possibly making some reference for selection/filtering purposes.
d. ***Analysis*** – does not reference control record content, but historical and analytical views maintained by the Service Domain in addition to handling control records. The analysis typically provides operational and analytical insights covering the operation of the Service Domain.

In addition to the input and output parameter descriptions the Service Operation lists the allowed pre and post states of the Service Domain. These are described in more detail in the How To Guide – Developing Content. At this stage the use of Service Domain states is very limited. These service operation fields have been retained as it is anticipated more detailed state analysis may be included in later releases of the BIAN standard.

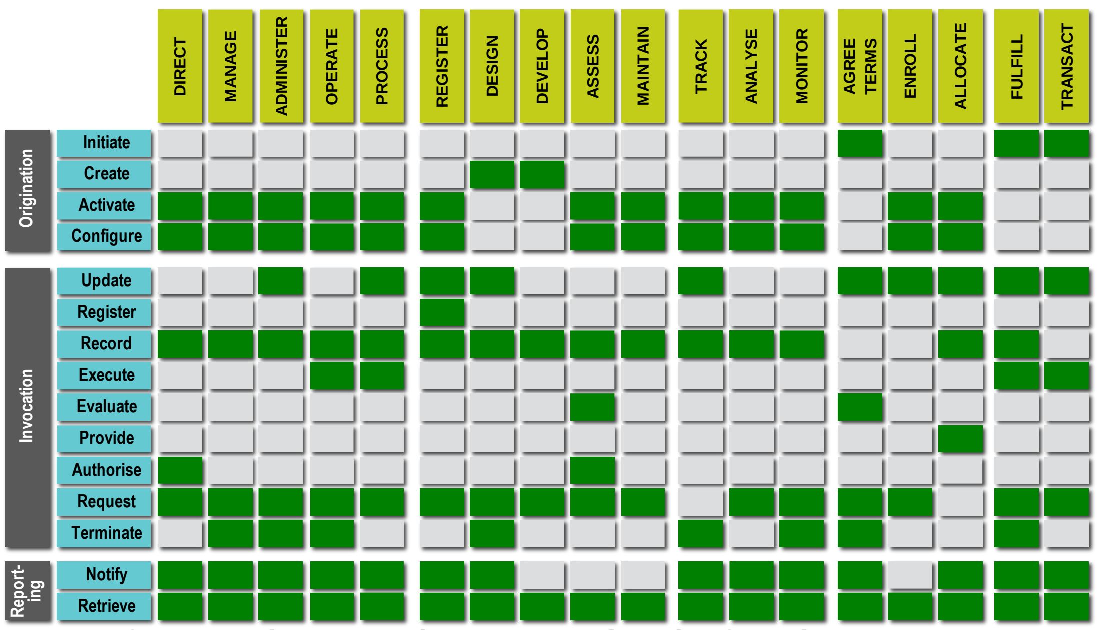
3. Service operation standard action terms – Because all Service Domains have a common operational structure (they implement the full life cycle of some pattern of behavior on an instance of some type of asset) it is possible to define a standard collection of action terms that characterise the different possible types of service operation offered by a Service Domain.



Action Terms			Description	Example
Origination	Actions to set-up, establish a new control record instance	Initiate	Begin an action including any required initialization tasks	A payment transaction is intiiated
		Create	Manufacture and distribute an item	A new analytical model design is created
		Activate	Commence/open an operational or administrative service	The ATM network operation is actived
		Configure	Change the operating parameters for an ongoing service/capability	The on-line ATM's in the network are changed to take machines out of service
Invocation	Actions to access/update/influence an established instance	Update	Change the value of some (control record) properties	A customer's reference details are updated with a change of address
		Register	Record the details of a newly identified entity	A new customer's details are captured
		Record	Capture transaction or event details associated with a life cycle step	An employee logs time spent working on a project against the plan
		Execute	Execute a task or action on an established facility	A payment is applied to a charge card
		Evaluate	Perform a check, trial or evaluation	The eligibility to sell a product is checked against the customer's existing agreement
		Provide	Assign or allocate resources or facilities	A branch requests an allocation of cash for its tellers
		Authorise	Allow the execution of a transaction/activity	Regulatory compliance authorises a product design feature
		Request	Request the provision of some service	A customer requests that a standing order is set up on the current account
		Terminate	Conclude, complete activity	The use of a product version is terminated
Delegation – no new action terms apply as the called Service Domains offer the same Origination/Invocation & Reporting options described here)				
Reporting	Actions to extract details and subscribe to updates	Notify	Provide details against a predefined notification agreement	A unit subscribes to update notifications from the customer agreement service domain
		Retrieve	Return information/report as requested	An account balance is obtained and a report covering activity analysis requested

Figure 23: Action terms, descriptions and examples

Different selections of these standard action terms sensibly apply to the different functional patterns of the Service Domains. A default mapping is shown in the summary table below. Note that the precise working of any specific Service Domain may require changes to this default mapping. Some action terms may not apply, other may be required and in some cases more specialized services can be defined. This is explained in more detail in How-to Guide – Developing Content.



		DIRECT	MANAGE	ADMINISTER	OPERATE	PROCESS	REGISTER	DESIGN	DEVELOP	ASSESS	MAINTAIN	TRACK	ANALYSE	MONITOR	AGREE TERMS	ENROLL	ALLOCATE	FULFILL	TRANSACT
Origination	Initiate														🟩			🟩	🟩
	Create							🟩	🟩										
	Activate	🟩	🟩	🟩	🟩	🟩	🟩			🟩	🟩	🟩	🟩	🟩		🟩	🟩		
	Configure	🟩	🟩	🟩	🟩	🟩	🟩			🟩	🟩	🟩	🟩	🟩		🟩	🟩		
Invocation	Update			🟩		🟩	🟩	🟩				🟩			🟩	🟩	🟩	🟩	🟩
	Register						🟩												
	Record	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			🟩	🟩	
	Execute				🟩	🟩												🟩	🟩
	Evaluate									🟩					🟩				
	Provide																🟩		
	Authorise	🟩								🟩									
	Request	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		🟩	🟩	🟩	🟩		🟩	🟩
	Terminate		🟩	🟩	🟩			🟩				🟩		🟩	🟩			🟩	
Reporting	Notify	🟩	🟩	🟩	🟩	🟩	🟩	🟩				🟩	🟩	🟩	🟩		🟩	🟩	🟩
	Retrieve	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Default service operations mapped to the functional patterns
(Green box indicates a match)

Figure 24: Action Terms mapped to Functional Patterns

4. Service operations select from a standard information profile – With the latest release BIAN has added significant detail to the semantic information content of the Service Domains that is accessed through its service operations for a selection of Service Domains. This new content has been generated in coordination with an initiative to support API design using the BIAN standards. This updated content will progressively replace the current checklist based 'information profile' definitions that describe the type of information governed by a Service Domain that is used to select the service operation information content.

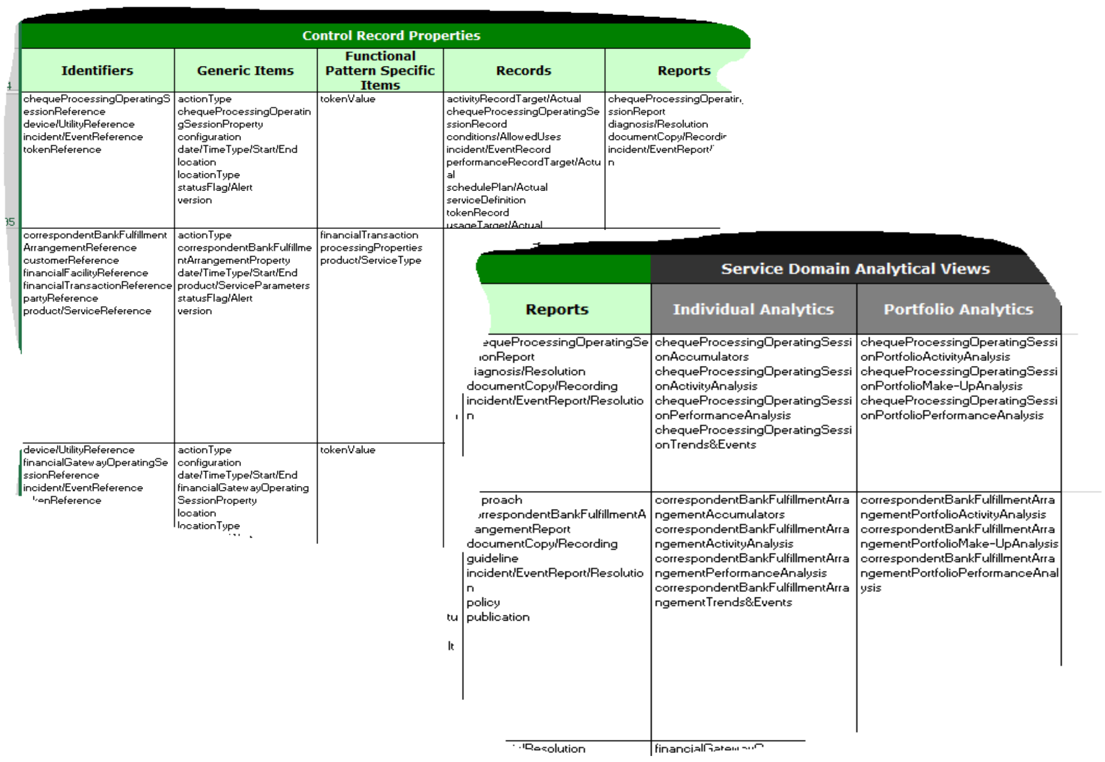
The way more detailed content is being added to the Service Domains and service operations is described more fully in the BIAN How to Guide – Developing Content. Here the information profile approach that produces the original 'checklist' based service operation content that is being replaced is briefly described.

The Service Domain's 'information profile' includes the information make-up of the Service Domain's control record and any historical and analytical views the Service Domain may maintain for one or the complete portfolio of its active control record instances. The control record portion breaks down into three different types of information:

- ***Information Items*** – are references to singular properties or measures such as a date of birth, amount or location. Information Items can be used as indexes to select a particular control record and they typically map directly to data elements or simple data structures

- ***Information Records*** – are structured collections of information items that might make up a transaction record or data entry form for example
- ***Information Reports*** – represent any free-form/unstructured information such as a free-form report or a scan of a document

BIAN defined an initial check-list of representative types of content that can be selected from to define the Service Domain's information profile. Selections from this overall list were then made specific to the different Service Domain functional patterns. Definitions of the candidate information elements can be found in the BIAN vocabulary.



Control Record Properties				
Identifiers	**Generic Items**	**Functional Pattern Specific Items**	**Records**	**Reports**
chequeProcessingOperatingSessionReference device/UtilityReference incident/EventReference tokenReference	actionType chequeProcessingOperatingSessionProperty configuration date/TimeType/Start/End location locationType statusFlag/Alert version	tokenValue	activityRecordTarget/Actual chequeProcessingOperatingSessionRecord conditions/AllowedUses incident/EventRecord performanceRecordTarget/Actual schedulePlan/Actual serviceDefinition tokenRecord usageTarget/Actual	chequeProcessingOperatin ssionReport diagnosis/Resolution documentCopy/Recordir incident/EventReport/ n
correspondentBankFulfillmentArrangementReference customerReference financialFacilityReference financialTransactionReference partyReference product/ServiceReference	actionType correspondentBankFulfillmentArrangementProperty date/TimeType/Start/End product/ServiceParameters statusFlag/Alert version	financialTransaction processingProperties product/ServiceType		
device/UtilityReference financialGatewayOperatingSessionReference incident/EventReference kenReference	actionType configuration date/TimeType/Start/End financialGatewayOperatingSessionProperty location locationType	tokenValue		

	Service Domain Analytical Views	
Reports	**Individual Analytics**	**Portfolio Analytics**
equeProcessingOperatingSe ionReport iagnosis/Resolution documentCopy/Recording incident/EventReport/Resolution	chequeProcessingOperatingSessionAccumulators chequeProcessingOperatingSessionActivityAnalysis chequeProcessingOperatingSessionPerformanceAnalysis chequeProcessingOperatingSessionTrends&Events	chequeProcessingOperatingSessionPortfolioActivityAnalysis chequeProcessingOperatingSessionPortfolioMake-UpAnalysis chequeProcessingOperatingSessionPortfolioPerformanceAnalysis
proach rrespondentBankFulfillmentA angementReport documentCopy/Recording guideline incident/EventReport/Resolution policy publication	correspondentBankFulfillmentArrangementAccumulators correspondentBankFulfillmentArrangementActivityAnalysis correspondentBankFulfillmentArrangementPerformanceAnalysis correspondentBankFulfillmentArrangementTrends&Events	correspondentBankFulfillmentArrangementPortfolioActivityAnalysis correspondentBankFulfillmentArrangementPortfolioMake-UpAnalysis correspondentBankFulfillmentArrangementPortfolioPerformanceAnalysis
/Resolution	financialGatew	

Figure 25: Information Profile extract

An extract from the Service Domain's information profile can next be mapped to the parameters of the service operation. This mapping varies depending on the service operation action term. A standard mapping template was been used to create the service operation profile content.

As a result, the default service operations each contain a checklist list of information content descriptions that has been selected from the overall checklist to reflect both the functional pattern of the host Service Domain and the type of service it provides as defined by the service operation action term.

As mentioned this standard information profile that was defined to generate descriptive content for service operations is being progressively replaced with more detailed and specific information descriptions in coordination with the BIAN Semantic API initiative. With the latest V7.0 release 67 selected Service

Domains and their service operations have been updated with more detailed definitions.

The new service operation information content is also being mapped to an extended version of the ISO20022 Business Model, creating the BIAN Business Object Model. Examples of the more specific service operations can be found in the latest Service Landscape release and the associated BIAN API development portal

5. Checklist Information is referenced in the BIAN Vocabulary – As noted earlier, all of the BIAN specific terms including definitions of the checklist items are included in the BIAN business vocabulary.

3.6.1 Specialized service operations

BIAN defines a standard set of default service operations for a Service Domain based on its particular functional pattern where each service operation is characterized by one of the selected action terms. For some of the more complex Service Domains, those involved in product fulfillment in particular, the standard default collection of service operations is insufficiently detailed at the level of the action terms.

For these more complex Service Domains there can be some contained or embedded operational features that merit their own direct service based access. In this case the most appropriate type of service operation (based on the action term) is duplicated and each duplicate specialized with a qualifier term representing its more specific purpose. The qualifier term used in these cases is called a ‘behavior qualifier’ defined for the Service Domains functional pattern as mentioned in Section 2.2. of this guide. The approach is more fully described in the BIAN How to Guide – Developing Content.

An example clarifies the approach. For the Service Domain that handles fulfillment of the Current Account facility the ‘request’ action term is used to access the range of services that are on offer. The Current Account Service Domains functional pattern is ‘Fulfillment’ and the behavior qualifier type for fulfillment is ‘features’ referring to the different product features that are supported. One of the behavior qualifiers defined for current account handling would be support for ‘standing order’s.

Typically the Instructor field of the service operation could be used to select the specific type of request. However in the case of a major product fulfillment feature such as a standing order a design decision can be made to define a specialized ‘request’ service operation that acts on the standing order functional element specifically.

The result is a service operation as follows:
‘requestCurrentAccountArrangementStandingOrder where the ‘standing order behavior qualifier has been tagged to the end of the service operation name to define its more specific purpose. A broad range of extended service operations with the

associated behavior qualifiers has been defined under the API initiative for selected Service Domains. As noted, this extended definition is explained in more detail the BIAN How to Guide – Developing Content.

As BIAN develops more detail specifications for the API initiative more specialized service operations will be identified. The formal guidelines for when more detailed specialized service operations are defined have not been finalized at the time of writing. A key influence is the make-up of the underlying information message. When services exchanges with the Service Domain using the same action term but in different usage scenarios result in significantly different information content this is a good indication that a specialized service operation using the associated behavior qualifier is required.

4 Interpreting the BIAN Standard in Implementation

The BIAN standard is defined at the business architecture level. As explained in Section 1 of this guide it also adopts a service-based perspective (a service oriented architecture SOA). In order to leverage the BIAN standard its business level representation needs to be related to different systems architecture views for systems solution design and deployment. The systems architecture views required will vary depending on the deployment approach and the target technical environment.

BIAN members tend to apply the BIAN standard in two fundamentally different deployment situations. In one the BIAN standard is used to help define well-bounded or ‘targeted’ system implementation solutions. In the other the BIAN standard is used to create a much broader “enterprise blueprint” that can be used for a wide range of business and systems planning and analysis activities.

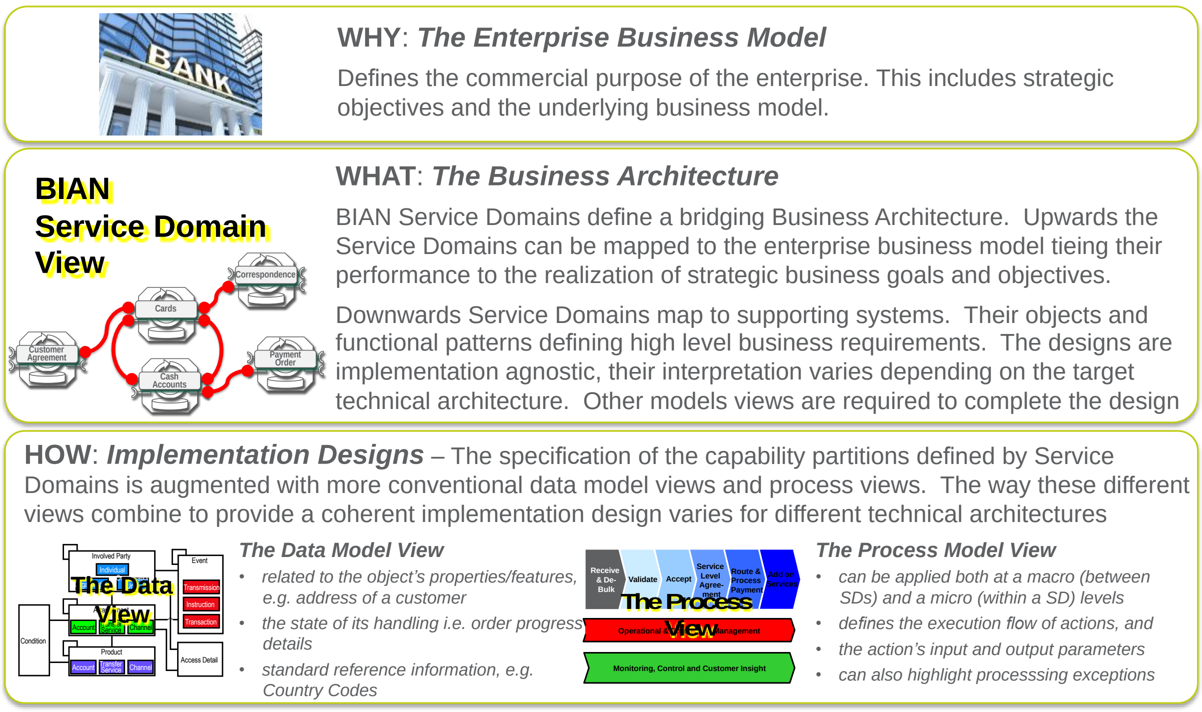
In this section the interpretation of the BIAN standard is covered as follows:

1. Relating the BIAN business architecture to underlying application/systems architectures – some general considerations
2. Clustering Service Domains
3. Mapping the BIAN standard to other industry standards
4. Applying the BIAN standard in three different technical implementation environments.
    - Conventional (legacy/core) application renewal/rationalization
    - Host/ESB integration and application assembly
    - Loose coupled distributed/cloud and micro-service architectures
5. Using the BIAN standard to define an enterprise blueprint that can be used for business and systems planning and analysis.

Note that in this document the design concepts and principles are explained in general terms. The specific guidelines and techniques used to deploy the standard are repeated in more detail the third document of the series: ‘How-to Guide - Applying the BIAN Standard.’ Furthermore BIAN has produced several more formal descriptions of how the BIAN specification can be related to more traditional architectural model views and various standards in other documents that can be found on the BIAN website.

4.1 Mapping BIAN’s Business Architecture to Systems Architectures

The BIAN Service Landscape (which refers collectively to the high level BIAN reference framework, Service Domain and service operation specifications) is a business architecture. It can be positioned as a bridging mechanism linking an enterprise’s overall business strategy and associated business model to the underlying implementation level technical solution specifications. This positioning is shown with some broader explanatory context in the next figure:


WHY: ***The Enterprise Business Model***

Defines the commercial purpose of the enterprise. This includes strategic objectives and the underlying business model.

WHAT: ***The Business Architecture***

BIAN Service Domains define a bridging Business Architecture. Upwards the Service Domains can be mapped to the enterprise business model tieing their performance to the realization of strategic business goals and objectives.

Downwards Service Domains map to supporting systems. Their objects and functional patterns defining high level business requirements. The designs are implementation agnostic, their interpretation varies depending on the target technical architecture. Other models views are required to complete the design

HOW: ***Implementation Designs*** – The specification of the capability partitions defined by Service Domains is augmented with more conventional data model views and process views. The way these different views combine to provide a coherent implementation design varies for different technical architectures

The Data Model View

- *related to the object's properties/features, e.g. address of a customer*
- *the state of its handling i.e. order progress details*
- *standard reference information, e.g. Country Codes*

The Process Model View

- *can be applied both at a macro (between SDs) and a micro (within a SD) levels*
- *defines the execution flow of actions, and*
- *the action's input and output parameters*
- *can also highlight processsing exceptions*

Figure 26: BIAN - the link between business and technical architectures

The BIAN Service Domain represents a business functional capacity partition that combines people, procedures and supporting systems. As mentioned, BIAN's focus is on the definition of the technology-enabled aspects of a Service Domain's operation and the service interactions between Service Domains. The connection to the supporting systems solutions is made by relating the high-level BIAN Service Domain and service operation specifications to supporting systems designs. These (typically more detailed) supporting systems specifications can be captured in many different forms. Furthermore different model views are appropriate for different types of technical environment.

Considerations for the connection between BIAN and the underlying systems architectures are covered here in three ways:

1. How may the high level BIAN specifications be extended?
2. How are the Service Domain boundaries mapped to applications?
3. Relating information and data to service based design.

.
The following frequently used terms are defined for their use in this guide:

- **Business Model** – defines the commercial rationale for an enterprise – for example its performance goals, market segmentation, bases of differentiation, product and service coverage and organizational structure
- **Business Architecture** – one or more model views of business activity that can be used to structure and optimize the enterprise's functioning and to

specify the requirements for supporting resources such as personnel, buildings and equipment and information technology
- **Business Requirements** – different model views of intended business behavior in a format and level of detail needed to define implementation requirements for supporting resources – information technology in particular
- **Systems Architecture** – different model views of the supporting information technologies (including application logic, data and technical environments/ platforms)
- **(Business) Application** – a term loosely applied to a computer program that supports a coherent collection of business activities. A Business Application can be an assembly of **Application Modules**
- **(Computer) System** – a term loosely applied to the combination of a Business Application and the supporting technology infrastructure needed to operationalize the application
- **(Business) Service Operation -** the term BIAN uses to refer to the external access mechanism used to access the capabilities offered by a Service Domain (or called by a Service Domain for it to delegate/gain access to another Service Domain). Note that this is a more constrained definition than sometimes associated with this term.

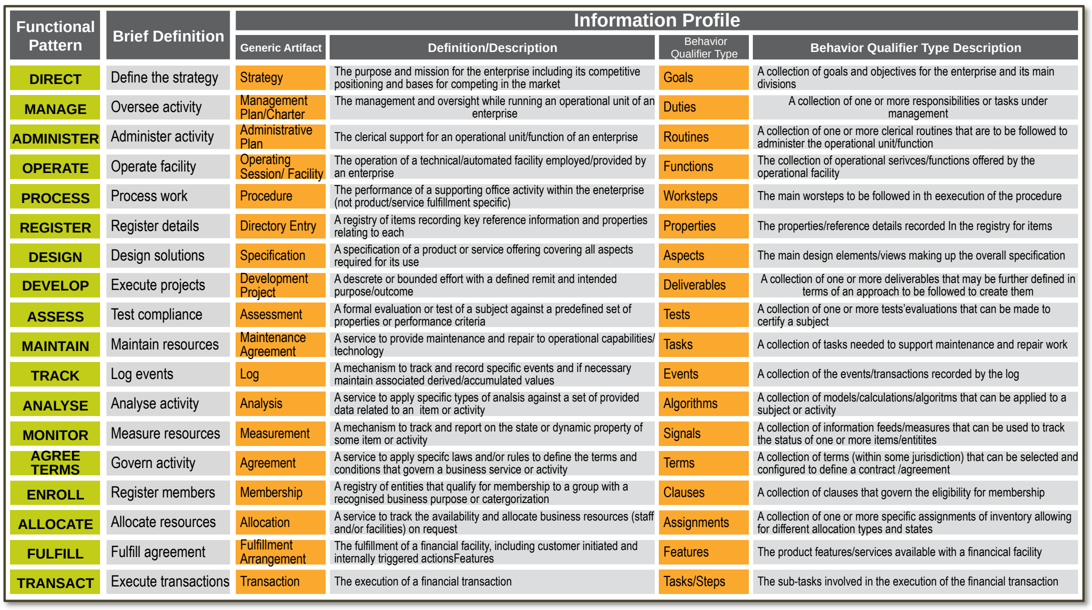
4.1.1 Extending the detail of the BIAN specification

With the BIAN Semantic API initiative and the associated development of the BIAN BOM BIAN adds another level of design specification to the Service Domains and their service operations across the Service Landscape. Examples of this additional detail are included in the latest release as noted earlier.

The finer detail is achieved by breaking down the functional pattern behaviors into finer grained elements called ‘behavior qualifiers’. The approach is more fully outlined in the How to Guide – Developing Content. In this section some of the main aspects and implications are summarized.:

Functionality – the BIAN Service Domain role description, control record definition and operational states provide a limited outline of the business functionality of the Service Domain. In practice it has been found that it is often necessary to augment the BIAN specification with checklists of the main prevailing functional and non-functional features. The features clarify the role and can help relate the Service Domains to business systems. In the past simple feature checklists have been developed and though these checklists are not part of the formal BIAN standard - they can provide informal definitional clarity by example. Service Domain feature checklists are described in more detail in the How-to – Applying the BIAN Standard.

As an alternative to these informal feature tables the extended definitions of the Service Domains are based on breaking down the Service Domains functional pattern into its underlying *behavior qualifiers.* This results in a more structured checklist of features. The table below lists the behavior qualifier types for each of the functional patterns. Specific behavior qualifiers need to be defined for each Service Domain corresponding to the type matched to its functional pattern:



Functional Pattern	Brief Definition	Information Profile			
		Generic Artifact	Definition/Description	Behavior Qualifier Type	Behavior Qualifier Type Description
DIRECT	Define the strategy	Strategy	The purpose and mission for the enterprise including its competitive positioning and bases for competing in the market	Goals	A collection of goals and objectives for the enterprise and its main divisions
MANAGE	Oversee activity	Management Plan/Charter	The management and oversight while running an operational unit of an enterprise	Duties	A collection of one or more responsibilities or tasks under management
ADMINISTER	Administer activity	Administrative Plan	The clerical support for an operational unit/function of an enterprise	Routines	A collection of one or more clerical routines that are to be followed to administer the operational unit/function
OPERATE	Operate facility	Operating Session/ Facility	The operation of a technical/automated facility employed/provided by an enterprise	Functions	The collection of operational serivces/functions offered by the operational facility
PROCESS	Process work	Procedure	The performance of a supporting office activity within the eneterprise (not product/service fulfillment specific)	Worksteps	The main worsteps to be followed in th eexecution of the procedure
REGISTER	Register details	Directory Entry	A registry of items recording key reference information and properties relating to each	Properties	The properties/reference details recorded In the registry for items
DESIGN	Design solutions	Specification	A specification of a product or service offering covering all aspects required for its use	Aspects	The main design elements/views making up the overall specification
DEVELOP	Execute projects	Development Project	A descrete or bounded effort with a defined remit and intended purpose/outcome	Deliverables	A collection of one or more deliverables that may be further defined in terms of an approach to be followed to create them
ASSESS	Test compliance	Assessment	A formal evaluation or test of a subject against a predefined set of properties or performance criteria	Tests	A collection of one or more tests'evaluations that can be made to certify a subject
MAINTAIN	Maintain resources	Maintenance Agreement	A service to provide maintenance and repair to operational capabilities/ technology	Tasks	A collection of tasks needed to support maintenance and repair work
TRACK	Log events	Log	A mechanism to track and record specific events and if necessary maintain associated derived/accumulated values	Events	A collection of the events/transactions recorded by the log
ANALYSE	Analyse activity	Analysis	A service to apply specific types of analsis against a set of provided data related to an item or activity	Algorithms	A collection of models/calculations/algoritms that can be applied to a subject or activity
MONITOR	Measure resources	Measurement	A mechanism to track and report on the state or dynamic property of some item or activity	Signals	A collection of information feeds/measures that can be used to track the status of one or more items/entitites
AGREE TERMS	Govern activity	Agreement	A service to apply specifc laws and/or rules to define the terms and conditions that govern a business service or activity	Terms	A collection of terms (within some jurisdiction) that can be selected and configured to define a contract /agreement
ENROLL	Register members	Membership	A registry of entities that qualify for membership to a group with a recognised business purpose or catergorization	Clauses	A collection of clauses that govern the eligibility for membership
ALLOCATE	Allocate resources	Allocation	A service to track the availability and allocate business resources (staff and/or facilities) on request	Assignments	A collection of one or more specific assignments of inventory allowing for different allocation types and states
FULFILL	Fulfill agreement	Fulfillment Arrangement	The fulfillment of a financial facility, including customer initiated and internally triggered actionsFeatures	Features	The product features/services available with a financical facility
TRANSACT	Execute transactions	Transaction	The execution of a financial transaction	Tasks/Steps	The sub-tasks involved in the execution of the financial transaction

Figure 27: Functional Pattern and Behavior Qualifier Types

Service Operations – the core of the BIAN standard and the way the functional partitioning is defined is through describing the service operations offered and consumed by Service Domains. The service operations outline the service dependency between two Service Domains, listing the information exchanged in semantic terms. In many cases, where available, the BIAN service operation descriptions can be mapped to one or more systems level message definitions that can enable an application to application exchange. A repeatable mapping approach is outlined described later in this guide.

BIAN does not define the operational nature of the service operation exchange in any detail as this will typically vary greatly in different deployment situations. Is it worth noting that the BIAN designs do not require a service-based implementation. For example the boundary implied by a service operation exchange could be realized as a 'hard-wired' application to application interface when appropriate.

With the progressive addition of the BIAN BOM as BIAN extends the Service Domain definitions across the landscape, coupled with BIAN's Semantic API initiative more detailed service operation specifications are being defined. As already noted, these more detailed service operation definitions are progressively replacing the default checklists that were included in an earlier release

Business Information – the Service Domain control record can be used to help define business information use and governance needs. Furthermore the way information is partitioned in service-based designs can mitigate problems of scope and consistency. This aspect is considered in more detail in Section 4.1.3 below.

4.1.2 Mapping BIAN Service Domains to Business Applications

The BIAN business architecture defines logical, generic, functional building blocks that can be used to better partition, design and assemble business applications. There is an implicit assumption that these functional building blocks will be implemented using designs and technical solutions that exploit service oriented architecture (SOA) techniques. But as described later in this guide the BIAN designs can be applied in many different technical environments not necessarily limited to service based designs. Note that a BIAN working group is currently developing the approach for mapping BIAN to different vendor systems architectures. When this work is complete this section of the guide is likely to be revised significantly.

The BIAN Service Domain partitions business activity in conformance with two key considerations: one, the partition must be unique and discrete and two, the partition must be elemental. An elemental business functional capacity partition though it may be succinctly defined at the business architecture level can require an extremely complex collection of underlying applications and technology. BIAN defines Service Domains with a brief description of their business functionality and more importantly semantic definitions of the business service operations they offer and consume. Staring from the latest release an additional level of detail is being progressively added across the BIAN service landscape that improves the ability to map the BIAN specification to underlying systems solutions.

When mapping the Service Domain to a systems architecture view the functional partitions represented by a Service Domain must be aligned in some way to the supporting business applications. A frequent assumption is that there is a simple one to one mapping between Service Domains and well-scoped business applications. In practice this is rarely the case.

It is more common to find a business application that contains a large collection of Service Domains and sometimes a Service Domain may represent a sensible collection of multiple business applications. Furthermore there are two specific situations where a business application or application module may recur in many Service Domains.

The main mapping patterns are as follows.

> **One-to-One** – the simplest is where the scope of the Service Domain aligns precisely to the coverage of the supporting application. Possible examples include Customer Agreement, ATM Network Operations and Product Directory. In these cases the functionality of the Service Domain defines the core functioning of the application and the offered and consumed service operations match the main (service) interfaces that the application needs to support to be integrated into the overall application portfolio.

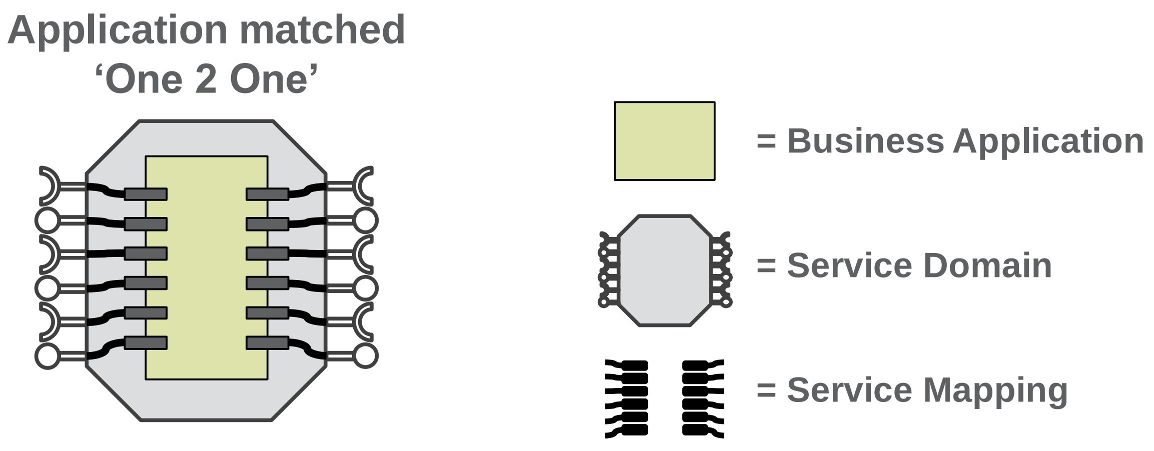


Figure 28: Service Domain to Business Application one to one

One-to-Many – less commonly it may be that the Service Domain defines a business function that typically brings together a collection of business applications. This can occur when the function exploits an array of tooling and support that may be implemented by different specialized applications. Possible examples include Systems Development and Product and Model Design Service Domains. The assumption is that the constituent applications can be assembled in a manner that is largely transparent to the outside world. They can be contained or integrated in a way that supports a coordinated external service boundary and any connections/dependencies between them are fully encapsulated within the boundary of the Service Domain.

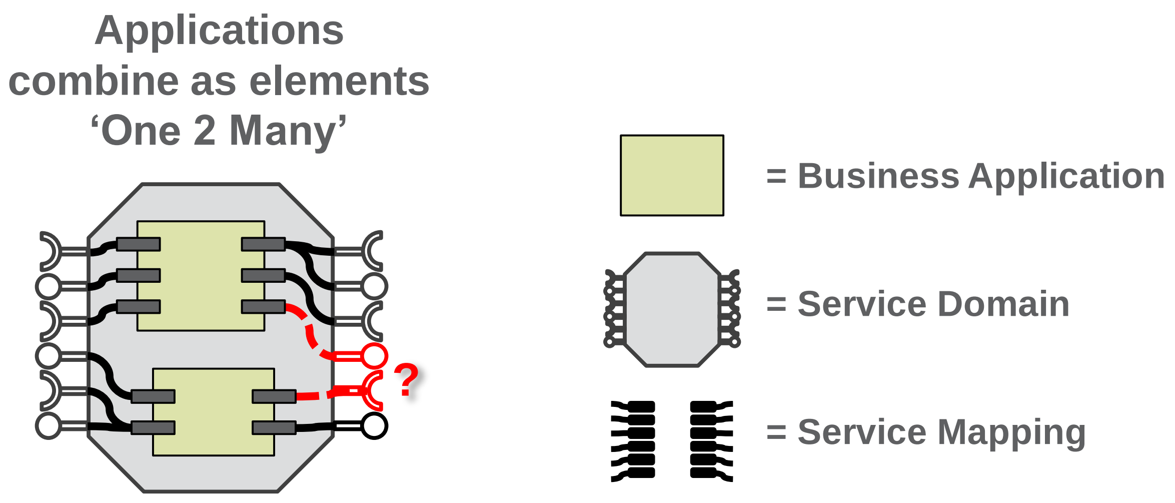


Figure 29: Service Domain to Business Application one to many

Many-to-One – by far the most common mapping will be where several Service Domains are contained within a business application. Possible examples include the majority of Product Fulfillment applications and Contact Center Applications. Most enterprises are familiar with the issue of redundancy in the application portfolio - it is not unusual for an enterprise to have many competing applications that perform essentially the same function as a result of silo’ed development and business acquisition. The issue of redundancy goes beyond this however. The typical stand-alone application contains a significant proportion of functionality and if it were properly engineered could be shared by multiple applications. By mapping Service Domains against business applications it is possible to identify partitions that are truly unique to an application and those that are candidates for shared solutions. This type of mapping is discussed further in the context of different technical environments later in this guide.

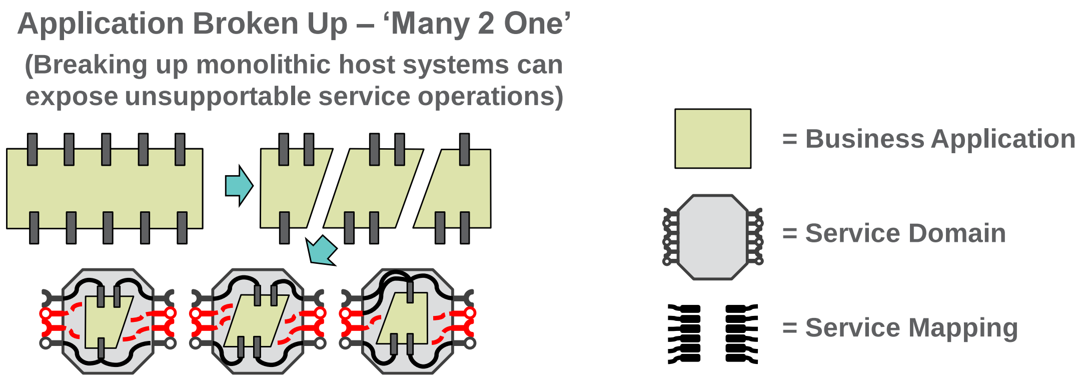
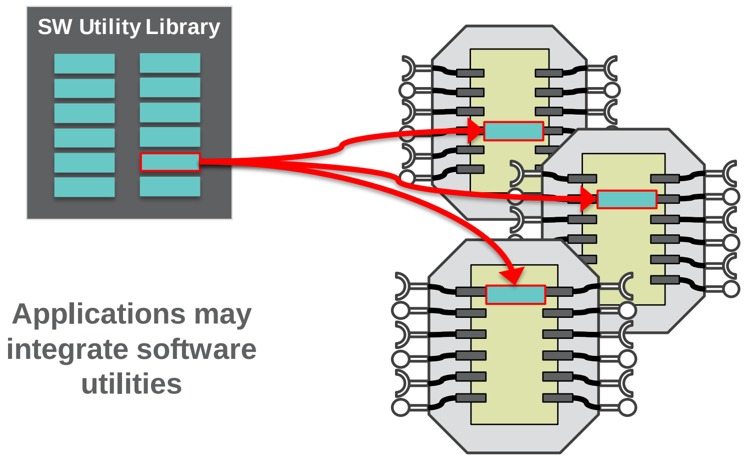

Application Broken Up – ‘Many 2 One’
(Breaking up monolithic host systems can expose unsupportable service operations)
= Business Application
= Service Domain
= Service Mapping

Figure 30: Service Domain to Business Application - many to one

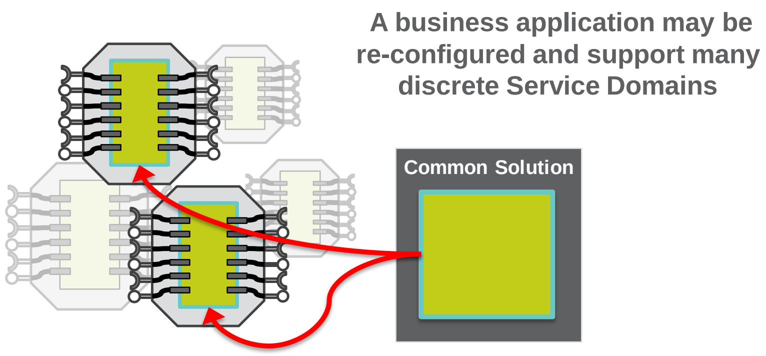
Many-to-Many – in some cases application modules may recur in many Service Domains as repeated utility functions. Examples of utility application modules include financial model components and generic productivity tools. This mapping/association is not an integral aspect of the business architecture view. It relates specifically to the design of the supporting systems that may re-use many utility functional elements. It is mentioned here because the ability to identify opportunities for utility application modules and to trace their re-use across the application portfolio is important. This can be captured as an overlay on a BIAN business architecture representation.



How SW utility solutions relate to Service Domains

Figure 31: Utility application module re-use

Repeated Solutions – the final mapping option is another specific to systems implementation that can also be captured as an overlay on a BIAN business architecture view. This is where a systems solution for one Service Domain can be reconfigured and used to support another Service domain independently. An example of this is a product fulfillment solution that can be deployed to support different variations of products – a Loans solution may be easily re-configured to support many different loan products.


A business application may be re-configured and support many discrete Service Domains
Common Solution

How common SW solutions relate to Service Domains

Figure 32: Configurable solution re-use

Many of the different mapping options outlined above are revisited in the section that describes implementation in different technical environments later in this guide. To conclude this sub-section a few key points are reiterated for clarity:

1. ***BIAN adopts a service oriented paradigm to model business behavior*** – for the reasons explained in Section 1 of this guide the BIAN business architecture adopts a service based paradigm. Supporting systems architectures may also adopt a service oriented approach but this is not

mandatory – the BIAN designs can be mapped to more conventional process and monolithic architectures to provide specific insights – examples are described later in section 4.4 of this guide
2. ***The BIAN Service Domain defines a logical functional capacity partition*** – the partition represented by the Service Domain is a unique and discrete business functional capacity partition. It can be thought of as a high-level design pattern. It is quite possible that this logical design will be repeated many times in physical deployment – examples are discussed later in section 4.5 of this guide
3. **Service Domain service operations can be mapped to different types of systems interfaces** – though the prevailing assumption is that a Service Domain exchange as defined by a service operation will be implemented as one or more messages in a service oriented architecture this is not mandatory as noted above. A service operation may correspond to a hard wired interface or in more extreme circumstances (typically for performance reasons) may be eliminated in practice through the use of shared database technology (typically with separate logical data views)
4. **The BIAN Service Domain may support business information governance** – the BIAN Service Domain partitions may be used to resolve business information and data governance issues. This topic is addressed briefly next.

4.1.3 Relating Business Information to Data in Service Oriented Design

This sub-section briefly summarizes how business information and the underlying data representations are scoped and handled in the BIAN service-based design. It is first necessary to clarify how some terms are used here in this guide:

- **Business Information** – refers to business concepts and details that can be described in simple narrative terms. Business information defines what needs to be known to describe something or in order to outline some necessary action.
- **Semantic Vocabulary** – BIAN maintains a vocabulary of key business information terms that defines their meaning and relates them to equivalent or similar terms in other selected industry standard vocabularies
- **Business Object Model** – BIAN is currently developing a business object model that is built on and extends the industry standard ISO 20022 Business Model. The BIAN BOM provides a detailed decomposition of the business information governed by each Service Domain and made available through its service operations
- **Data/Data Structure** – refers to a machine-readable data representation of one or more business information terms. Note that a data view is typically far more detailed and that there may be many competing/overlapping data views/formats for the same business information term
- **Information/Data Scope** – defines the extent or boundary of the context for which the definition of a business information term and/or its associated data representation is valid/agreed.

These definitions are needed to explain a useful property of the BIAN service-based design with respect to information and data governance. In a more conventional process view of business activity the scope of the information referenced (and any underlying data representations) typically spans the complete end-to-end process. Consider for example a mortgage application and fulfillment process. When the customer first completes the application form they will provide details about themselves, the mortgaged property and agree the payment terms and conditions that apply. This information will be referenced and updated and transactional activity logged through all subsequent stages of processing until the mortgage is finally repaid. It is hard to identify any specific stages in the processing of the loan where any aspect of the associated business information may not be needed for some reason.

These business information requirements will be translated into more detailed data structures and elements for the supporting mortgage loan processing business application. As (for the sake of this example) the complete end-to-end mortgage process is supported by one business application, the same data views of the business information can sensibly be adopted by all users/interested parties. In summary in conventional process based solutions the scope of the business information tends to be broad and the mapping to the underlying data structures is the same for all interested parties.

It is obviously very sensible to be able to support broad access to common business information and to have widely applied data standards. However in practice this is not always easy to achieve across application portfolios that may combine many bespoke developments and different commercial packages. It is even more difficult to coordinate between different enterprises. It is therefore interesting to note that these two properties – broad scope and common data definitions/mappings can sometimes be relaxed to some extent in service based designs.

Consider the scope of business information in service based design using the example of the month-end credit card billing process first seen earlier in this guide. In the example each service domain retains control over the control record instances that handle the fulfillment of its particular business function for the full life cycle. For example, Customer Agreements maintains the customer’s agreed terms and conditions from start to finish. The only information exchanged with other Service Domains is extracted reference views of the agreement (and perhaps some control related requests).

It is possible to determine for each Service Domain what portion of the business information that it uses to support its role needs to be externally visible and what portion of that business information relates to specific internal processing that can be hidden or encapsulated away from all other Service Domains. In the example of the Customer Agreement there is a limited public view of the Customer Agreement and a more complete array of business information needed to set-up, verify and maintain the agreement behind the scenes.

The scope assessment of the overall business information can therefore be divided into public business information for which the definition needs to be agreed by the parent/owning Service Domain and all other Service Domains that may access its services and the more comprehensive internal business information that only needs to be seen/accessed by its ‘parent’ Service Domain.

Next consider the required degree of precision with which a business information item is mapped to a representative data structure. This consideration is rarely exposed in more conventional process based designs but is an important property of service operation exchanges. The concept can be clarified using two examples at either end of the specification resolution spectrum.

A Credit Card fulfillment Service Domain needs to post the transaction details for a card transaction against the transaction journal (maintained by the Position Keeping Service Domain). It is fairly obvious that every field (amount/date/currency/account number etc.) needs to be strictly defined in machine readable, formatted fields.

At the other end of the range, Campaign Execution has identified a new prospect that is not currently a bank customer and wishes to notify Prospect Management to follow up. In this situation there are many possible ways (information items) to identify the individual and to describe the new business prospect. Furthermore the precise data formatting of the underlying data fields can be more loosely matched. It does not matter if one side of the exchange happens not capture middle names or has different address field lengths. Sufficient information can hopefully be exchanged to have a good chance of making the business connection.

In summary there will be many service domain exchanges that require similar levels of consistency to data scoping and formatting rigor to the more conventional process based implementations. But there are areas of the Service Landscape where the scoping of shared business information definition and values is reduced to a significantly more focused ‘public’ semantic vocabulary. Also where differences in the internal data representations of these semantic business terms in the respective applications of the supporting Service Domains do not constrain their ability to exchange services.

These information and data properties are a key aspect for achieving loose coupling between service-based capabilities in highly networked technical environments such as the world wide web, the cloud and micro-service architectures and most recently through mechanisms such as application program interfaces (APIs).

An additional observation can be made with respect to business information governance. The BIAN Service Domains are defined to represent non-overlapping business capability partitions that collectively cover all possible aspects of banking. Each Service Domain has a defined control record that represents the information used to manage one instance of the Service Domain performing its role for a complete life cycle. It follows that the business information associated with the collection of one or more control records for a Service Domain (and any derived analytical views) is also discrete and that the combination of these discrete information partitions covers all possible business information.

As a result the Service Domains can be used to define business information partitions that can be used to scope out business information and the underlying data governance responsibilities for the Service Domain's supporting applications. Note that this partitioning of the business information and its supporting data views does not preclude that one Service Domain may require access to the data governed by another Service Domain. This information will be accessed through appropriate service operation exchanges and the retrieved information once interpreted as necessary becomes information that is owned and governed locally.

It is also the case that two Service Domains may govern discrete business information that happens to share a common data representation and even the same current value. For example the Customer Agreement Service Domain may maintain a name and address for a contract that has the same format and value as the mailing address maintained by the Correspondence Service Domain. Though they may use the same data format and even have the same value, these two data items represent different business information as they have different contexts, meaning and purpose.

It is for the above reasons that the BIAN Business Object Model is being defined with a structure that aligns with the Service Domain control records.

The Control Record can be modeled

The business information view of a Service Domain's control record can be modeled using any suitable entity or object data approach to define its structure and content. Such a view is used to help define the information payload of a Service Domain's service operations. In an earlier release BIAN defined a standard collection of types of information that might make up the content of a control record based on the Service Domains functional pattern. This generic list provides a checklist from which a suitable selection can be made and adapted for each individual Service Domain.

In the significant majority of cases the Service Domain's control record defines a single 'primary' object or entity with a potentially complex make-up of more detailed elements as can be represented in a conceptual data model. In more recent releases an additional design property has been used to specify a Service Domain – the 'behavior qualifier type'. The behavior qualifier type indicates how the functional pattern of a Service Domain can be broken down into finer elements. Based on this decomposition a more detailed structure of the function and data can be derived. The way behavior qualifiers have been defined and applied is explained in more detail in the BIAN How To Guide – Developing Content.

In addition to the greater level of specification for the Service Domain BIAN has also started to define a Business Object Model, that builds on and extends the ISO20022 Business Model industry standard. The BIAN BOM aligns to the Service Domain control records and provides a significantly greater level of specification of the associated business information than available in previous releases. The BIAN BOM is being be expanded to cover the BIAN Service Landscape in conjunction with the ongoing BIAN API initiative.

For some Service Domains the underlying information model can have a more complex structure than others. This is particularly the case where the Service Domain handles asset types that include some form of relational/hierarchical property. Examples of Service Domains that have an embedded hierarchical property are:

1) Service Domains for a legal party (a legal entity can be made of other legal entities
2) Service Domains for a product (a product can be an assembly of other products)
3) Service Domains for Risk Management Service (risks can be composite views of other risks).

Service Domains with a hierarchical structure require additional service operations to manage the internal hierarchical relationships. For example in the case of the legal party there is one service operation to add a new party and then an additional service to add a new party relationship.

4.2 Clustering Service Domains

BIAN defines a cluster to be a simple mechanism for grouping together Service Domains so that they can be viewed as a related collection in some form of representative structure. This grouping may represent either business model perspective to represent a ‘segment’ as defined by TOGAF to be a business unit, profit/cost center, division or enterprise. Or grouping for systems purposes for example to represent the scope of a business application. Though BIAN is actively looking as mapping BIAN to business model views the clustering concept has yet to be applied in great detail.

BIAN guidelines for applying clustering for systems purposes define Service Domain groupings that map to the underlying business/systems application architecture. The cluster can be a form of wireframe with some additional structure and definition needed to better relate to the business application.

Known service connections between Service Domains, based on First Order Interactions described earlier in this guide provide good initial insights into the likely range of interactions between selected Service Domains. When a selection of Service Domains is defined to generate a cluster the service operation dependencies can be used to define the external service boundary of that cluster/application.

The key difference between a cluster view and the standard wireframe is that the cluster view takes into account scoping considerations for Service Domains that apply when the conceptual/logical designs of the Service Domains are translated into the more physical implementation designs. In particular the standard wireframe only allows a single copy of any Service Domain. In implementation there are often practical reasons for duplicating Service Domains across multiple physical applications.

There are two levels at which this duplication can be considered. At the highest level there may be duplication business operations, for example customer contact centers may be deployed in different regions. More relevant to the clustering concept is the lower level type of duplication where a Service Domain is needed in more than one business application typically for performance and integration purposes. Three roles have been defined to recognize this possible need to duplicate a Service Domain in implementation across the application portfolio. The roles are:

- ***Core*** – The Service Domain's implementation exists only in whatever business application this cluster represents. Any and all service-based references to this Service Domain must be supported by the external service boundary of the cluster/application. (As must all of its delegated service operation dependencies). For example the Service Domain Current Account Fulfillment would be a core Service Domain in the Current Account Processing Application cluster...

- ***Proxy*** - Represents a capability that is likely to be repeated in other clusters, and is included in the cluster to provide a local 'view'. In such a case it could be the master version meaning all other instances need to reference this instance for their needs (as with the Core role), or it could be a slave, meaning it needs to synchronize with the master instance elsewhere through suitable 'background' services. For example Service Domain Party Data Management could be a slave proxy service domain in the Current Account Processing Application cluster.

- ***Utility*** – Like the proxy Service Domain role, the application cluster contains a non-unique instance. But in the case the local instance operates in a fully standalone manner - it does not need to synchronize or even be aware of other similar SD instances elsewhere. For example Position Keeping (the transaction journal) is a utility Service Domain instance in the Current Account Processing Application cluster

The figure below is an informal example of a cluster of service domains for a retail banking application showing the different roles within the cluster the and main external service dependencies:

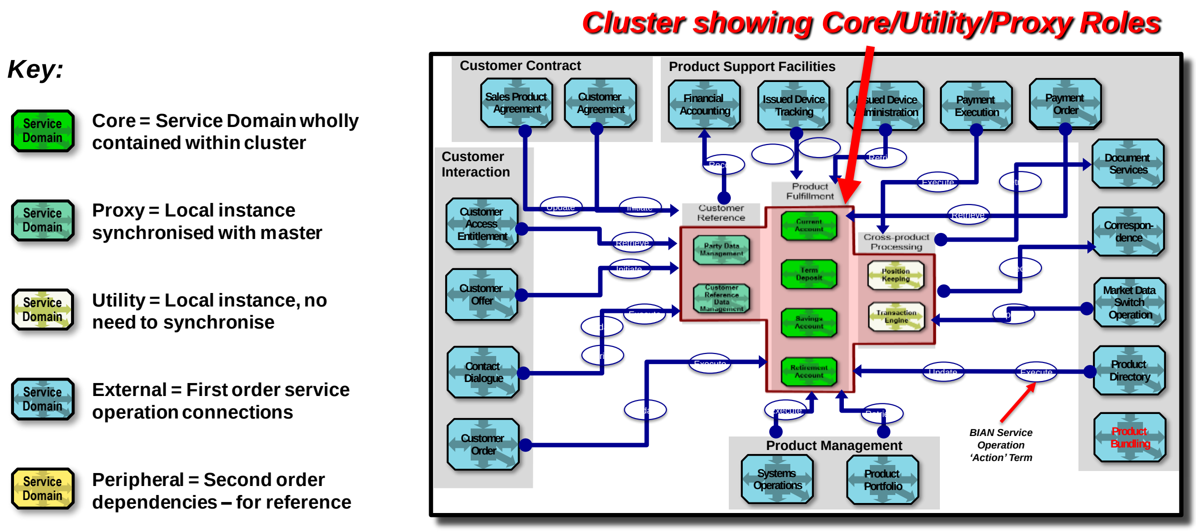


Figure 33: Example Cluster for a Retail Banking Application

4.3 Mapping BIAN to Other Industry Standards (e.g. IFX, ISO20022)

An earlier 'Capstone' Project with Carnegie Melon University supported by PNC Bank and BIAN developed a repeatable approach for mapping BIAN service operations to the message specifications in IFX and ISO 20022. The findings are fully document in a report (BIAN_CMU_Final_Report.pdf) that is available on the BIAN website (www.BIAN.org). This work built on an earlier joint project between BIAN and IFX that explored the ability to map BIAN service operations to implementation level message standards. This report (BIAN-IFX-PoC-Webinar-Dec-2013-Final.pdf), is also available on www.BIAN.org).

Both initiatives followed a similar general approach to mapping messages to the BIAN service operations. This used the standard design structure of the Service Domain and its associated control record to narrow in on possible matches. In the CMU initiative the standard service operation 'action terms' were also used (these were not available for the earlier exercise). The main steps in a repeatable mapping approach (that combines both efforts) are described below. Note it may help to refer back the descriptions of Service Domains and service operations earlier in this guide to fully understand the explanation:

- ***Business Scenario Selection*** – a suitable collection of BIAN Business Scenarios is used to agree the business context for the mapping. From the Business Scenario views the target Service Domains and their associated service operations are isolated
- ***Service Domains and their Control Records are identified*** – for each involved Service Domain, the associated control record is used to identify the primary asset type/entity acted on by the Service Domain
- ***Asset Type Aligned Message Selection*** – Candidate messages are selected from the target message standard based on matching the primary asset type to their data/object model. In general terms standards such as IFX often define and categorize messages that provide access to data views of major business objects. An example object would be a customer (relationship)
- ***Functional Pattern Based Filtering*** – the second facet of a Service Domain's control record is its 'functional pattern'. This can be used to further filter the selected messages and/or message content based on a sub-set data view of the asset type. Continuing with the example of the customer relationship, BIAN defines a Service Domain that applies a functional pattern 'Agree Terms' to the customer relationship. This can be used to narrow the customer relationship messages and details of interest to just those associated with their contract
- ***Mapping Action Terms to Matched Messages*** – BIAN service operations use one of a predefined list of action terms. These action terms can align to the types of messages in the target standard
- ***Matching Message Payload*** – the final step uses the semantic content descriptions of the BIAN service operations and matches this with the payload

of the candidate messages to further filter out messages and message content.

The steps described above may be applied in different combinations depending on the target message standard being mapped. Practical experience to date has confirmed that the semantic descriptions of the BIAN service operations are adequate to support a fairly rigorous mapping to established message standards.

The extended Service Domain specification with behavior qualifiers provide additional detail to support message mapping that could be factored into the above approach. This aspect is being considered as an aspect of the ongoing BIAN Semantic API initiative and any findings will be included in future versions of this guide.

4.4 Other Mapping Considerations

BIAN currently maintains a comprehensive UML metamodel of its designs. This is an extension of the established ISO20022 metamodel. The BIAN metamodel is fully documented in the BIAN MetaModel guide that is available on www.BIAN.org . The BIAN model is maintained in a UML based repository that allows members to download the content into their own tooling environments with limited adaptations.

BIAN's policy is to align with prevailing industry standards where possible in order to avoid creating duplicate/competing definitions. This includes adopting naming conventions/notations and aligning with the most popular tooling when this is possible. Where there are competing standards BIAN will attempt to reconcile or at least document incompatibilities'.

BIAN is currently evaluating the following standards and supporting tooling/techniques to improve alignment:

- Open Notations: Archemate & possibly SoaML
- Open reference models: TOGAF, SOA, RefArch, BIZBOK
- Vocabularies/Ontologies: ISO20022, OMG/FIBO.

Others may be added to the list based on the recommendation of BIAN members.

4.5 Applying BIAN Designs in Technical Environments

The BIAN standard is intended to be implementation agnostic. To be of practical value there need to be repeatable approaches for interpreting the BIAN designs in the main prevailing technical environments found in the banking industry. More detailed guidelines for the concepts outlined in this section can be found in the third document of the BIAN 'How-to Guide - Applying the BIAN Standard.'

4.6 Translating BIAN 'down the stack'

Before briefly considering three specific different technical environments there are some general comments as to how the BIAN high level conceptual designs can be interpreted 'down the stack' for the supporting application and infrastructure levels. A Proof of Concept Initiative was recently completed within BIAN that sought to explain and enhance the mapping from BIAN to the traditional architectural views (business, application, information and technology). The result of this work is presented in a summary report that can be found on the BIAN website. Its findings and recommendations will be reflected as appropriate in later versions of this guide.

4.6.1 Translating at the Application Level

In section 4.1.2 above there is a detailed list of the different ways a BIAN Service Domain partition can map to the underlying business applications and their constituent application modules. Points of clarification to add here are:

- A BIAN Service Domain may align to a single business application, several Service Domains may be covered by a business application or a single Service Domain may be supported by a collection of business applications
- It is more likely that at some level an application module within a business application will align most closely to a Service Domain
- It is not necessary that the mapped business application or application modules be implemented with a service enabled external interface. The exchanges can be realized by many types of technical exchange mechanism.

The main purpose of the Service Domain is to define a logical business functional capacity partition that can be used as pattern to better structure the application logic to avoid fragmentation and duplication and improve encapsulation. Some examples of the way the Service Domain concept can be leveraged at the application level include:

- **Specialization** – because the Service Domain supports one 'elemental' purpose its implementation can be optimized for that specific behavior. Conversely more complex designs that support multiple behaviors are often forced to adopt technical and operational compromises
- **Externalization & Re-use** – The Service Domain design partitions make it clear when actions should be delegated to other specialized Service Domains ('Externalized') – allowing the Service Domain to focus on its own specific role and for greater re-use of functionality between Service Domains. This important concept is explained in more detail with examples later in this Section
- ***Service Enablement*** – though it is not required it is anticipated that the Service Domains are typically implemented to act as service centers in a service oriented architecture (SOA). The many advantages of SOA are well documented elsewhere
- **Loose Coupling and Multi-Threaded** – features typically associated with SOA, the Service Domain partitions are well suited to an implementation

where the required collaborations between service domains are fully asynchronous and defensive (i.e. handle delayed/erroneous responses). Also where the Service Domain is able to handle multiple concurrent streams of activity.

- **Encapsulation** – because the Service Domains business role is discrete and it performs its role from start to finish for every occurrence the partition tends to 'encapsulate' business information and logic. The Service Domain can 'hide' complexity that is not relevant to those that consume its services. This property is particularly useful for highly distributed environments like the cloud

There is much about the BIAN Service Domain that aligns to the design principles associated with a micro-service architecture. This topic is explored in detail in the BIAN Semantic API How to Guide. Finally because the business role addressed by a Service Domain is enduring (see Section 2.1) it is often possible to build out and integrate the capabilities of a Service Domain incrementally. It may also be able to add new features as new practices evolve without destabilizing established services, extending the shelf-life of business applications significantly.

4.6.2 Translating at the Infrastructure Level

A Service Domain and its mapped application module(s) can be further related to the supporting technical infrastructure. For simplicity in the explanations that follow it assumed that a single application module has been mapped to the associated Service Domain.

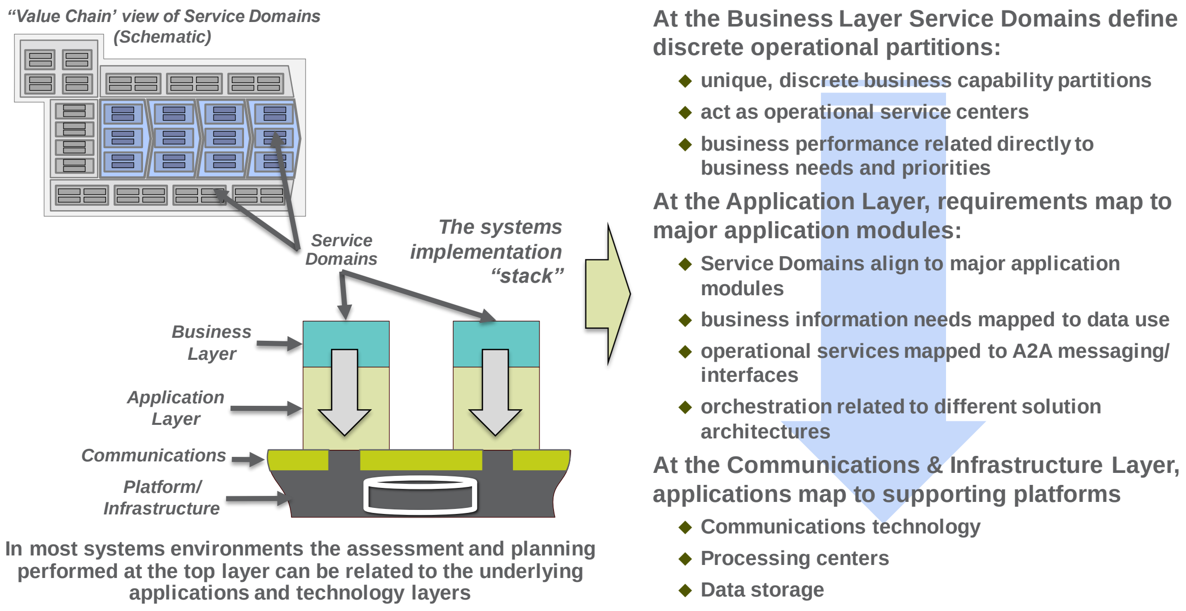
Each Service Domain will have its own non-functional profile in terms of its data storage access and processing requirements that need to be supported by the technical infrastructure. Most technologies provide comprehensive instance partitioning facilities/virtual machines so multiple Service Domains and their associated application modules can operate independently on the same platform allowing for Service Domains/application modules with similar operating profiles to be supported on shared technical infrastructures with the required performance profile.

Some optimization options can be considered for supporting the communications traffic between application modules as represented by the high level Service Domain service operations. Where the volume and frequency of the exchanges is modest most data exchange/communications mechanisms can be considered as is appropriate for the particular technical environment.

Where the exchange is high volume/high frequency it is possible to configure the infrastructure to facilitate the exchange. For example the logical service exchange between two discrete conceptual functions as represented by Service Domains can be mapped to a shared database with each having controlled access to their respective views of the data. In practice this would eliminate the need for a physical service based data exchange to realize the conceptual service interaction. Other technical communications options may be found.

4.6.3 Translating Summary

The mapping of Service Domain conceptual designs down the stack is summarized in the next figure:



In most systems environments the assessment and planning performed at the top layer can be related to the underlying applications and technology layers

At the Business Layer Service Domains define discrete operational partitions:

- unique, discrete business capability partitions
- act as operational service centers
- business performance related directly to business needs and priorities

At the Application Layer, requirements map to major application modules:

- Service Domains align to major application modules
- business information needs mapped to data use
- operational services mapped to A2A messaging/ interfaces
- orchestration related to different solution architectures

At the Communications & Infrastructure Layer, applications map to supporting platforms

- Communications technology
- Processing centers
- Data storage

Figure 34: Mapping Service Domains down the stack

4.7 Applying Service Domains in Different Technical Environments

The Service Domain mapping approach is outlined for three general technical environments. But before describing these it helps to consider two discrete aspects of the Service Domain specification as these aspects play different roles in different technical situations. A Service Domain can be considered as the combination of its functional 'core' and a service boundary or 'wrapper' that handles the orchestration of the service based connections with other Service Domains as represented below:

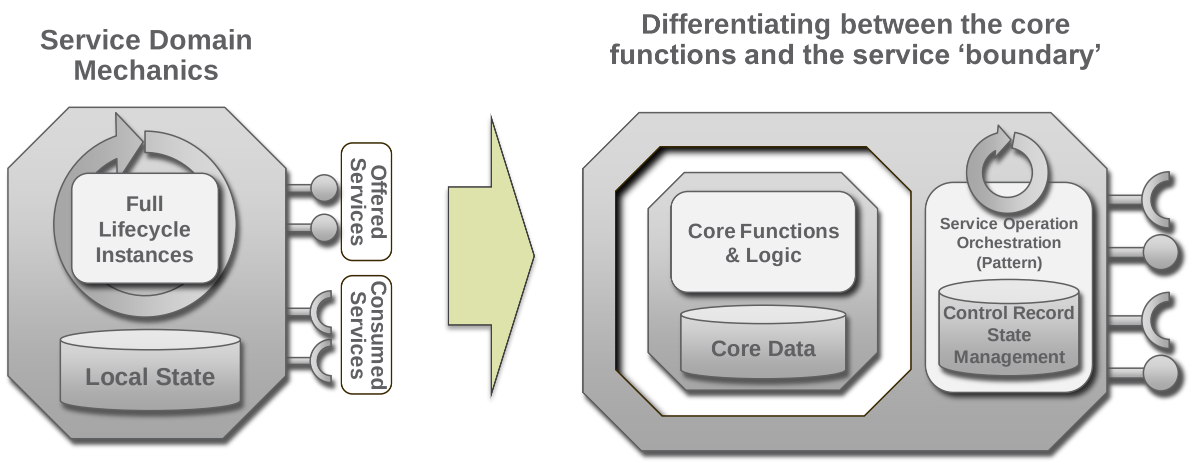

Service Domain Mechanics
Full Lifecycle Instances
Local State
Offered Services
Consumed Services
Differentiating between the core functions and the service 'boundary'
Core Functions & Logic
Core Data
Service Operation Orchestration (Pattern)
Control Record State Management

Figure 35: Service Domain split into two key components

The functional core contains the internal processing logic of the Service Domain, needed to fulfill its offered services. BIAN does not attempt to fully define the internal working of Service Domains so the description of this functionality is limited.

A simple table is sometimes used to capture prevailing functional and non-functional features for a Service Domain as described in more detail in the third document of the How-to Guide – Applying the Standard. These tables are not part of the formal standard but as with Business Scenarios they help with the correct interpretation of the standard by providing examples and context. Service Domain Feature tables can be developed and used to define requirements and to map and compare application options. In the latest release the definition of the Service Domain functionality has been extended with the introduction of 'behavior qualifiers' as described earlier and in more detail in the How to Guide – Developing Content.

The way the Service Domain specifications can be interpreted for systems solution development is considered for three different technical implementation environments:

- Conventional (legacy/core) system rationalization
- Host renewal/ESB integration and application/system assembly
- Loose coupled distributed/cloud and micro-service solutions.

It is likely that all three technical environments and maybe 'hybrid' variants will exist in combinations in many of the larger banks today.

4.7.1 Type 1 - Conventional (legacy/core) system rationalization

The BIAN Service Domain standard partitions can be used to assess, realign and repurpose the transactional mainframe based host systems. The BIAN Service Domains define non-overlapping functional partitions and the required connections/interfaces between them. The Service Domains' specifications can be used to create a framework that is then used to assess the coverage of host systems.

The functional ‘footprint’ of the hosts systems can be overlain on the Service Domain framework by matching their functional coverage. The Service Domain Feature tables mentioned earlier can provide a simple checklist to help with this exercise as can the more recently added behavior qualifiers where they are available. Because the Service Domains define discrete, non-overlapping functional partitions, the core system mapping quickly reveals gaps, redundancy and misalignment (where a system suited to one purpose is over-extended to support other purposes) in the overall application portfolio, as shown in the figure below.

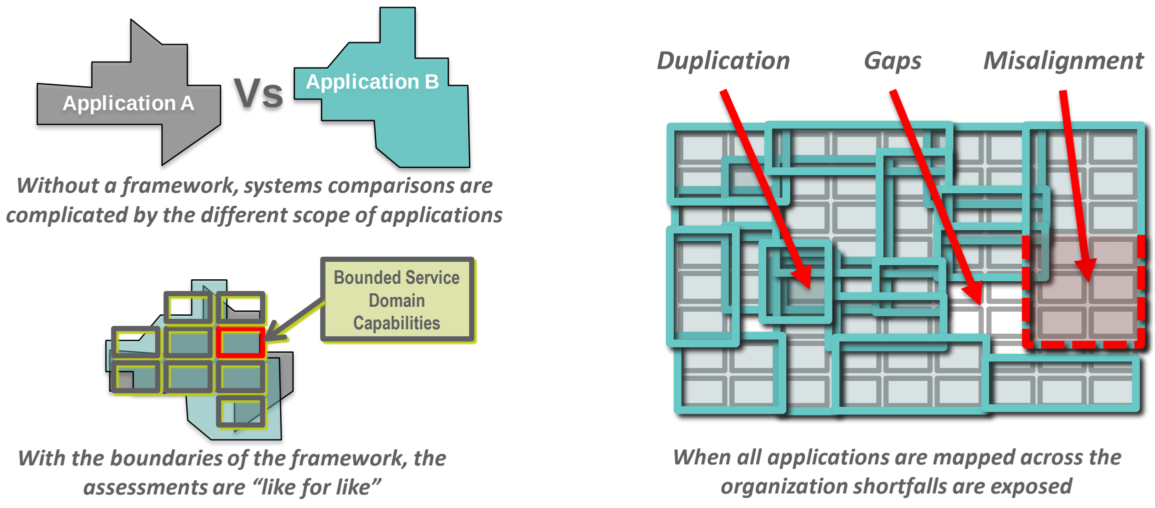

Application A
Vs
Application B
Without a framework, systems comparisons are complicated by the different scope of applications
Bounded Service Domain Capabilities
With the boundaries of the framework, the assessments are “like for like”
Duplication
Gaps
Misalignment
When all applications are mapped across the organization shortfalls are exposed

Figure 36: Using Service Domain partitions to do comparisons

When competing host applications/systems are compared using the Service Domain framework it is possible to do a more structured ‘like for like’ comparison by considering their respective coverage one Service Domain at a time. The framework is useful for getting a ‘big picture’ view of the application portfolio. It is used to rationalize duplicated capabilities – a situation that is common in enterprises where there has been siloed development and/or many company acquisitions over the years.

The individual Service Domain partitions can then be used to define these overlaps in more detail to help identify and eliminate fragmentation of the functionality and data within the application portfolio. Consider the example of multiple product fulfillment business applications – a consumer banking application, a credit card fulfillment application and a mortgage processing application.

Assume that these product fulfillment applications have been implemented independently and currently operate largely as stand-alone operational capabilities. As a result there will significant functional duplication such as customer reference information, customer access and servicing capabilities and customer transaction and accounting records.

At this stage the focus of the assessment is to determine to what extent this duplication of function and data leads to fragmentation and consistency problems (solution re-use is considered later). The assessment is performed for each Service Domain in isolation, considering all 'mapped' business applications – the three product applications listed above for this simple example. The impact can be assessed for the two key facets (function and data) as follows:

- **Duplicated Functionality** – to what extent will the user or customer experience different functionality when it would be expected/preferred that the experience should be identical or similar. If in this example a customer has all three products, how awkward is the experience of accessing each through a different application interface? Are there obvious synergies in terms of common or shared actions that could be exploited?
- **Duplicated Business Information/Data** – to what extent is the same business information/data duplicated across all business applications leading to problems of consistency and synchronization. If a customer changes their address on one application will it get reported to the others?

The point of these assessments is to determine the extent to which the duplicated functionality and data needs to be 'synchronized' in order to maintain the integrity of the overall business operation. These simple examples are only intended to clarify the basic concepts for using Service Domains to assess and rationalize existing applications. More detailed examples and approaches can be found in the third guide of the series: How-to Guide – Applying the Standard.

4.7.2 Type 2 – Host renewal/ESB integration and application/system assembly

The second technical environment considered here builds on the insights gained from mapping existing host systems to the Service Domain framework. Moving on from identifying where duplicated data and function needs to be synchronized this second environment is geared to developing common or shared solutions that eliminate the duplication altogether. This approach is referred to as 'externalization' within BIAN. The main stages in the externalization procedure is shown in the following figure: At a finer level of detail the mapping can be used to identify the opportunity to externalize capabilities matched to Service Domains that are built into standalone systems, but which could be shared by multiple applications. The concept of externalization is shown schematically in the figure below:

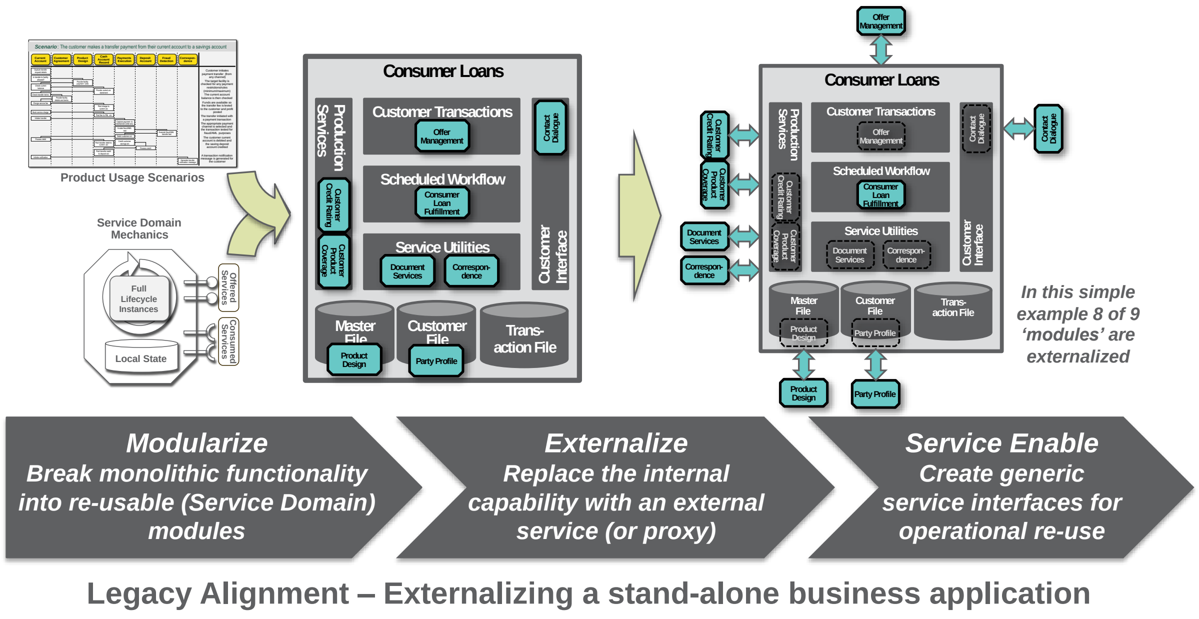


Figure 37: The stages of externalization

The above figure includes an informal systems architecture view of a stand-alone loan fulfillment application. A loan on-boarding Business Scenario has been used to identify some Service Domains and position them within the systems architecture. A more comprehensive collection of Business Scenarios would be needed to identify all of the major components but this sample is sufficient for the purpose of this example.

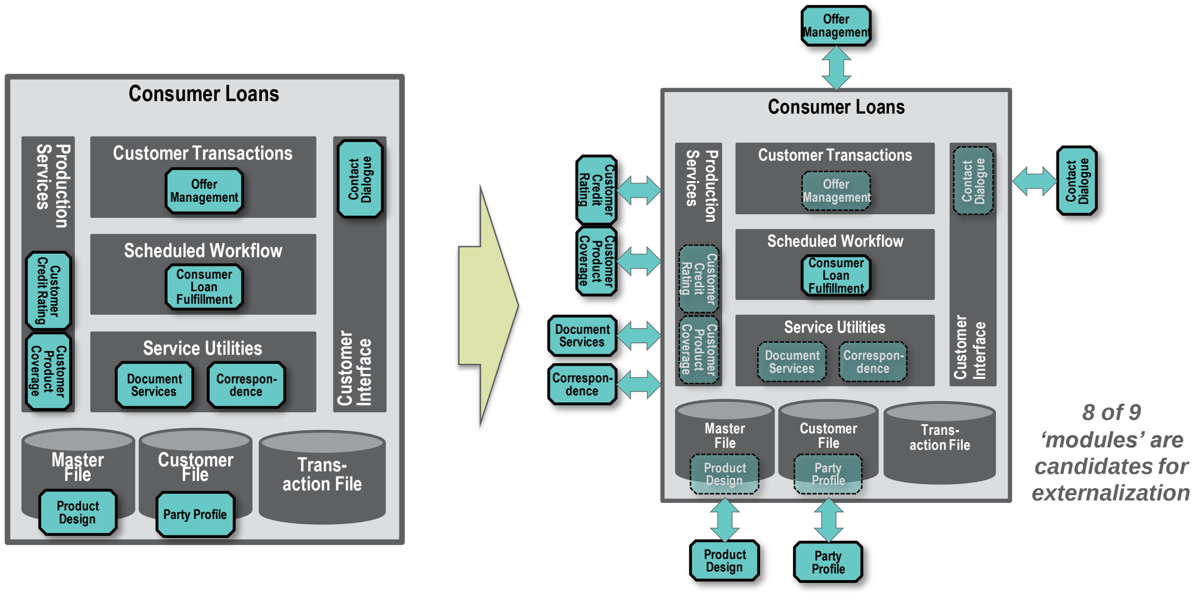


Figure 38: More detail of Loan externalization

In the right-hand view the Service Domains have been classified as being one of two flavors: do they offer functionality that could sensibly be used in another business application? Or are they likely to be used in only this particular business application. Not surprisingly the majority (eight of nine in this example) fall under the first category. As a result the functionality represented by these Service Domains is a candidate for being externalized. In other terms they could be supported in some way by a common/shared solution. In this case we assume this sharing is achieved through service enablement.

The source of the common/shared solution could be selected from the existing host systems if it is possible to isolate and service enable that specific partition of its functionality. Alternatively a new solution can be acquired or developed and progressively integrated across the application portfolio by shutting down the existing internal solution partitions and adopting/embedding the replacement shared service enabled solution. The approach is described in more detail in the third guide of the series – How-to Guide – Applying the Standard.

In the simple stand-alone loans application example the significant majority of the Service Domain partitions were externalization candidates. Observations to date are that this is also the case rather than the exception for a significant portion of production business applications. Accordingly there is an opportunity to achieve significant operational efficiencies and simplification through well architected and implemented operational capability re-use. One way to realize this is through the implementation of an enterprise service bus (ESB).

4.7.3 Configuring an Enterprise Service Bus (ESB)

As just described using the Service Domain framework as a guide, host systems can be broken into service enabled application modules. A technical mechanism that can then be implemented to support service based access to these shared service enabled capabilities is the enterprise service bus (ESB).

BIAN Service Domains define a comprehensive and non-overlapping collection of the required business functional capacity partitions for any bank and each Service Domain has its own unique collection of offered service operations. As a result the combination of selected Service Domains and their service operations can be used to define the service directory for a bank's ESB. The host systems can be mapped to the ESB offering service based access to their functionality and resolving duplications as outlined earlier. These ESB enabled services can then be assembled to support different business applications.

The different techniques and mitigation approaches for service enabling host systems and the way applications can be assembled in an ESB enabled environment are explored in more detail in the How-to Guide – Applying the Standard.

The ESB systems architecture is shown the following simplified schematic:

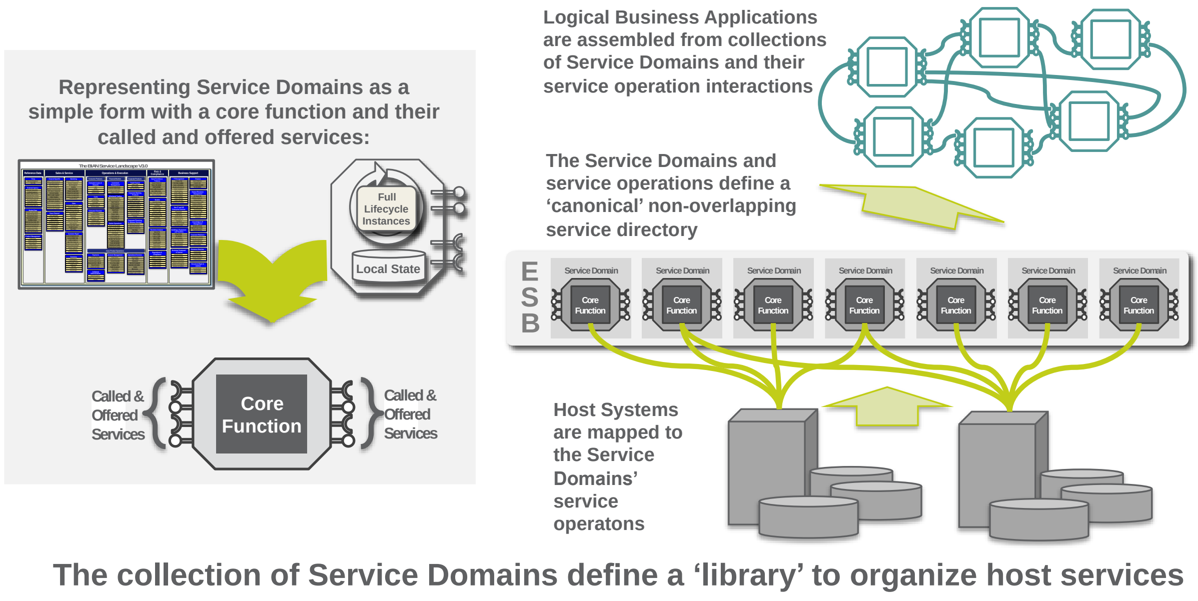

Representing Service Domains as a simple form with a core function and their called and offered services:
Full Lifecycle Instances
Local State
Called & Offered Services
Core Function
Called & Offered Services
Logical Business Applications are assembled from collections of Service Domains and their service operation interactions
The Service Domains and service operations define a 'canonical' non-overlapping service directory
ESB
Service Domain
Core Function
Host Systems are mapped to the Service Domains' service operatons
The collection of Service Domains define a 'library' to organize host services

Figure 39: ESB based application assembly

The ESB approach can be used to migrate progressively towards a business application configuration where each operational service is offered by a single source and re-used whenever needed across the overall application portfolio. Such an arrangement would, in theory at least, eliminate operational duplication and maximize business functional capacity re-use.

4.7.4 Type 3 - Loose coupled distributed/Cloud systems

Highly distributed and networked technical environments such as the Internet, the cloud and more recently micro-service architectures provide powerful implementation options for interpreting the BIAN designs. The BIAN Service Domains can be used to define service enabled business functionality that can be accessed over the network. Business applications can then be assembled using combinations of these service centers in a manner similar to the previous ESB example. In this case the service-enabled host systems acting behind the scenes through the ESB are replaced by service providers available over the network. In some technical solutions these services are implemented as 'containers'.

Business applications/systems assembled in these networked/distributed/container based service environments ideally need to be 'loose coupled' (fully asynchronous) and defensive (i.e. deal with unsolicited, delayed and erroneous responses). They are likely to use message queue and state management and triggering mechanisms in their implementation. Some example system architectural features are outlined in the third guide of the series: How-to Guide – Applying the Standard.

The way the BIAN Service Domain partitions can be matched to a highly distributed Internet, cloud or micro-service/container based technical environment is summarized in the following schematic:

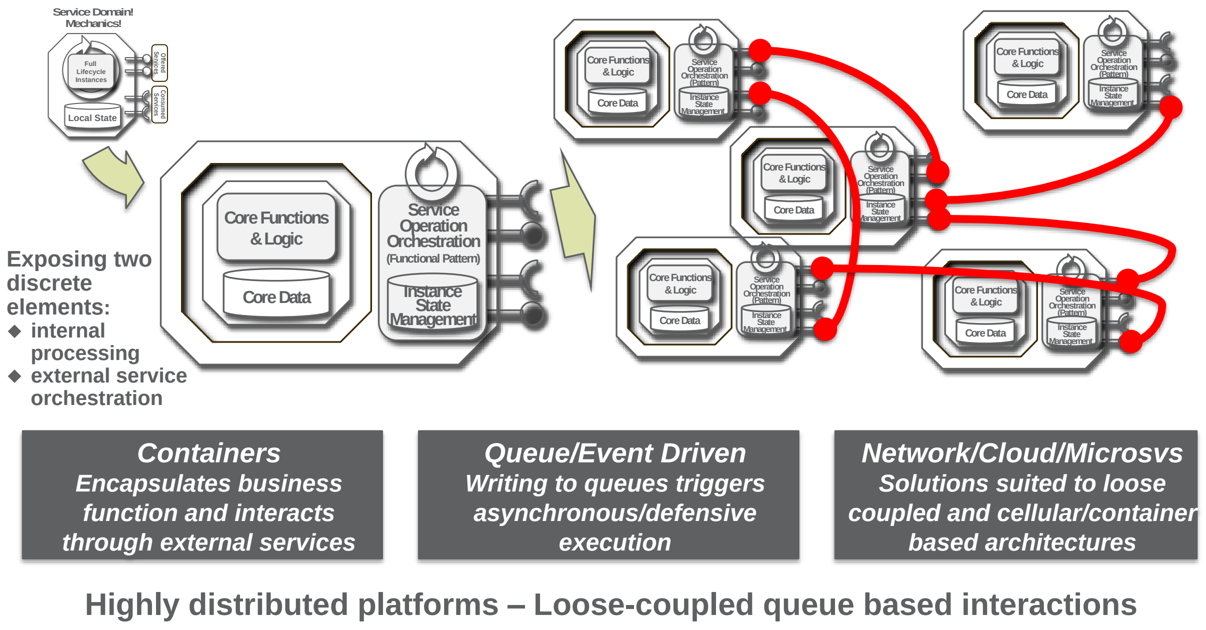

Service Domain! Mechanics!
Full Lifecycle Instances
Local State
Offered Services
Consumed Services
Exposing two discrete elements:
◆ internal processing
◆ external service orchestration
Core Functions & Logic
Core Data
Service Operation Orchestration (Functional Pattern)
Instance State Management
Containers
Encapsulates business function and interacts through external services
Queue/Event Driven
Writing to queues triggers asynchronous/defensive execution
Network/Cloud/Microsvs
Solutions suited to loose coupled and cellular/container based architectures
Highly distributed platforms – Loose-coupled queue based interactions

Figure 40: Cloud based deployment environment

The requirements and implications of service oriented application design for business information and data scoping and precision outlined earlier in Section 4.1.3. of this guide are of particular significance in this distributed environment. It is to be expected that different service centers will reside on different technical platforms with their own data interpretations of the business information exchanged through services.

Note that the service exchanges that are defined in semantic terms can combine physical movements of resources, free-form conversational dialogues but will also typically include some portion of machine representable data. A service exchange may also be implemented as a simple one or two way transfer or may involve a complex negotiation/dialogue. For the machine representable content it is necessary to translate the high level semantic service descriptions into a more detailed collection one or more data messages.

In more conventional technical environments, the application-to-application service exchange is typically driven down to the data and communication infrastructure where machine-to-machine message based interfaces are realized. The message data payload needs to conform to a commonly accepted data format/convention for these exchanges to work. This can be a significant design overhead that requires that all data fields conform to a common schema that can be consistently interpreted at the machine level.

In highly distributed environments the exchanges are initially resolved at the semantic level – agreeing in general narrative terms the required business participation and service dependencies. From there the appropriate level of precision needed for the correct interpretation of these services exchanges needs to be determined. The level of data precision and definitional alignment in the exchanged machine messages will vary for different types of service exchanges.

For the exchange of financial transaction details it can be anticipated that the required level of precision will be very high (both parties will need to share common data element specifications). But there are some exchanges, typically in areas that are less transactional and more to do with business and relationship development where such a high level of precision is not necessary for the service exchange to work. The involved parties can agree the meaning of information items at the semantic level and then their own internal data representation can be different. This looser, ‘semantic level coupling’ is shown in the following schematic.

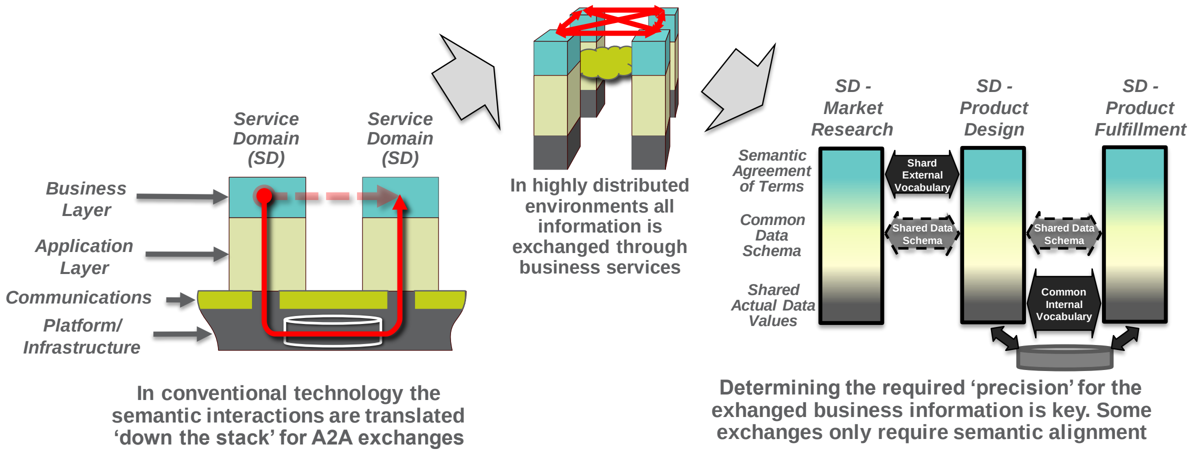

Service Domain (SD)
Service Domain (SD)
Business Layer
Application Layer
Communications
Platform/ Infrastructure
In conventional technology the semantic interactions are translated ‘down the stack’ for A2A exchanges
In highly distributed environments all information is exchanged through business services
SD - Market Research
SD - Product Design
SD - Product Fulfillment
Semantic Agreement of Terms
Common Data Schema
Shared Actual Data Values
Shard External Vocabulary
Shared Data Schema
Shared Data Schema
Common Internal Vocabulary
Determining the required ‘precision’ for the exhanged business information is key. Some exchanges only require semantic alignment

Figure 41: In the cloud communication can be semantic

For example, both parties can agree on the definition of the term ‘prospect’ as being ‘an individual that has expressed an interest in doing business’. This may then be referenced in any service exchanges between the parties. How each uniquely identifies the prospect and what specific information they might maintain etc. can be handled independently within their respective machine environments.

Commercially available cloud based banking solutions are widely used in CRM and risk management. The operational behaviors enabled by the technology are particularly well suited to the networked/collaborative nature of the operational activities in these areas. The BIAN standard can support standard solution designs that will improve re-deployment and integration activities for cloud based offerings and other highly distributed platforms and approaches including open API development and more recently micro-services.

Furthermore as noted earlier, due to the way Service Domains are designed they tend to encapsulate business information and logic. This can further reduce the complexity of the services to containing only business information and data that is commonly defined or necessarily externally visible. More complex business information and logic that is unique to the role/working of the Service Domain can remain hidden.

4.7.5 Using BIAN Service Domain partitions to define API’s

In the past BIAN published a white paper that describes how BIAN Service Domains provide a template for defining and managing Cloud based services. This can include providing application program interface (API) based access to these services. The white paper can be found at www.BIAN.org. With this release BIAN has developed extensions to the designs and sample API specifications for selected service domains as part of a broader BIAN API initiative. The BIAN APIs are available through an open source portal and BIAN will add content and detail to this portal as the API initiative extends to additional Service Domains across the landscape.

A key consideration explained in the earlier white paper still applies for API design however. It addresses how access to the services can be controlled. Many banks are looking ways to move beyond offering 'packaged' products and services to their customers to providing direct access to banking facilities. This would allow customers and third parties to integrate the bank directly into their operations somehow. Banks wanting to provide external access to their capabilities need to find a way to do this in a secure manner.

Achieving adequate security spans many levels, including the platform and application access controls (IAAS/PAAS) that are complex but for which solutions and approaches are already available or are rapidly emerging. The aspect that BIAN can provide help with is in defining the allowed use of offered services.

When a Bank enables service-based access to facilities internally (inside the bank) it can ensure that the services are being used for appropriate purposes and by people with the correct access authority. When the service is accessed externally, this is far harder to manage. Banks needs some way to ensure the services are used appropriately and do this in a way where every external service access agreement does not need to be specified and implemented individually.

One way to do this is to use the role of the Service Domain to define the service access context and use this to control its use. The figure below shows an example for a service call to access customer details using the Relationship Management Service Domain to provide the context.

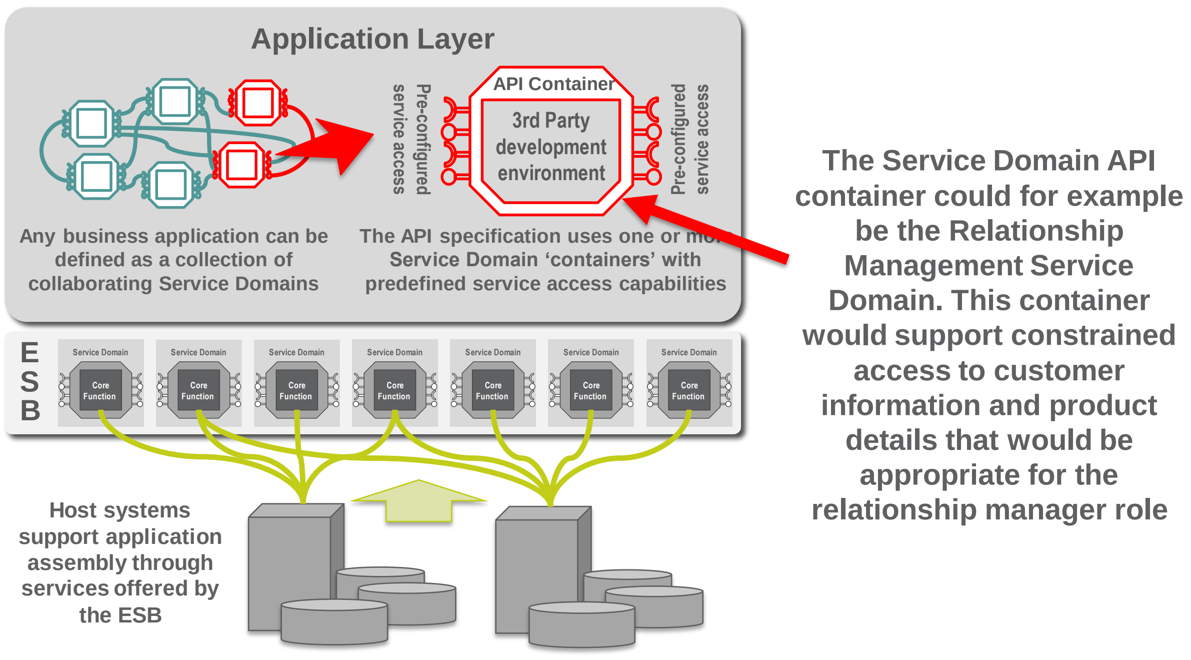

Application Layer
API Container
3rd Party development environment
Pre-configured service access
Pre-configured service access
Any business application can be defined as a collection of collaborating Service Domains
The API specification uses one or more Service Domain 'containers' with predefined service access capabilities
E S B
Service Domain
Core Function
Host systems support application assembly through services offered by the ESB
The Service Domain API container could for example be the Relationship Management Service Domain. This container would support constrained access to customer information and product details that would be appropriate for the relationship manager role

Figure 42: Cloud based service access control

The offered service API could be bundled in a software container with the pre-enabled constrained access to services allowing the external user to then develop their own logic within the container.

4.7.6 Combining Types 1-3 – Most banks have elements of all three

As noted it will be the case in most large banks that their existing application and systems portfolio includes legacy solutions spanning all three types. One additional advantage of adopting the BIAN Service Landscape is that it makes it easier to engineer solutions that combine systems across these environments.

A good example combines wrapped host systems that are accessed through an ESB with external cloud based services. A customer management application may use such a configuration to combine access to internal product fulfillment systems and external cloud based CRM functions.

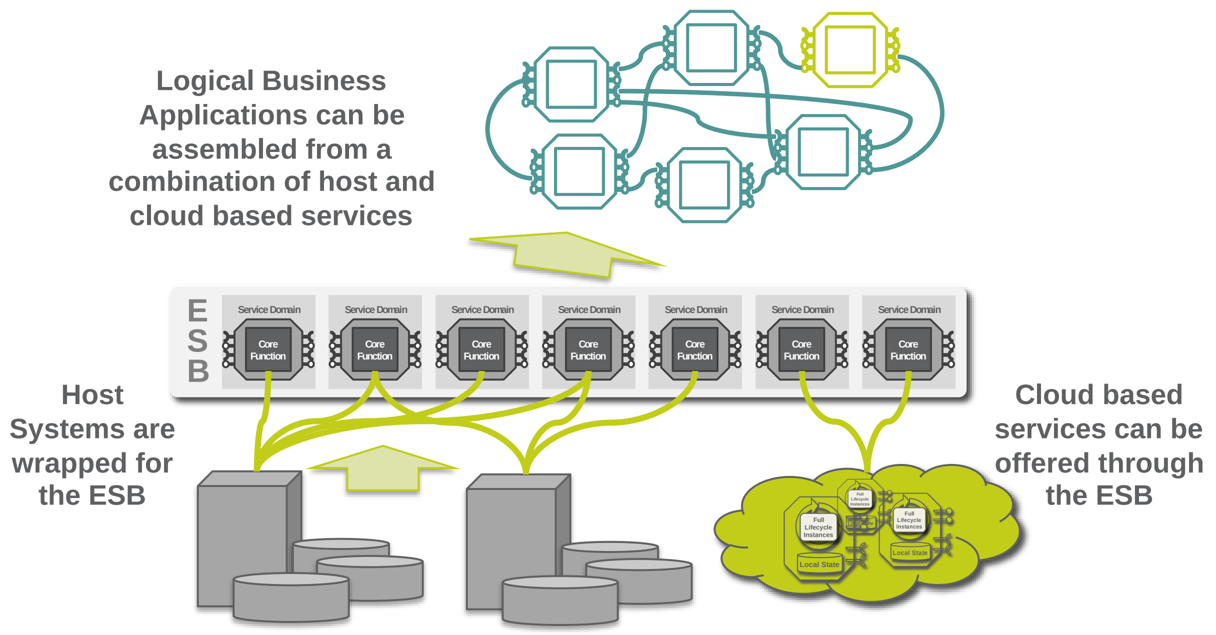

Logical Business Applications can be assembled from a combination of host and cloud based services
E S B
Service Domain
Core Function
Host Systems are wrapped for the ESB
Cloud based services can be offered through the ESB
Full Lifecycle Instances
Local State

Figure 43: Client-server BIAN design

4.8 Defining an Enterprise Blueprint for Business & Technical Analysis

A different deployment context for the BIAN designs leverages properties of the BIAN Service Domain to assemble an enterprise 'blueprint' that is useful to perform a wide range of business and technical analyses. As described earlier in this document, the BIAN Service domain models a business capability partition and the business purpose or role that is enduring. The way a Service Domain works or achieves its purpose can change as practices and enabling solutions evolve but its core business purpose does not change. Furthermore, each service domain represents a discrete/non-overlapping business capability partition and it is assumed that all service domains have or will be identified to cover all possible banking activity.

As a result, it is possible to assemble a representative map or 'blueprint' of an enterprise using BIAN Service Domains as the elemental building blocks. As long as the scope or structure of the business does not change (e.g. by entering new markets or opening in new locations) the blueprint itself should not change. The blueprint can therefore provide a stable framework supporting many types of analysis.

The approach for developing a blueprint and examples of the types of analyses are covered in more detail in the third document of the BIAN 'How-to Guide - Applying the BIAN Standard.' The concepts are outlined here in three sub-sections:

1. the steps in creating a blueprint
2. the types of analysis
3. linking between business & technical assessments.

4.8.1 Creating an Enterprise Blueprint

The procedure for assembling an enterprise blueprint can be performed to reflect current or future state – in most cases defining the future state is most useful. The three steps are outlined in the figure below:

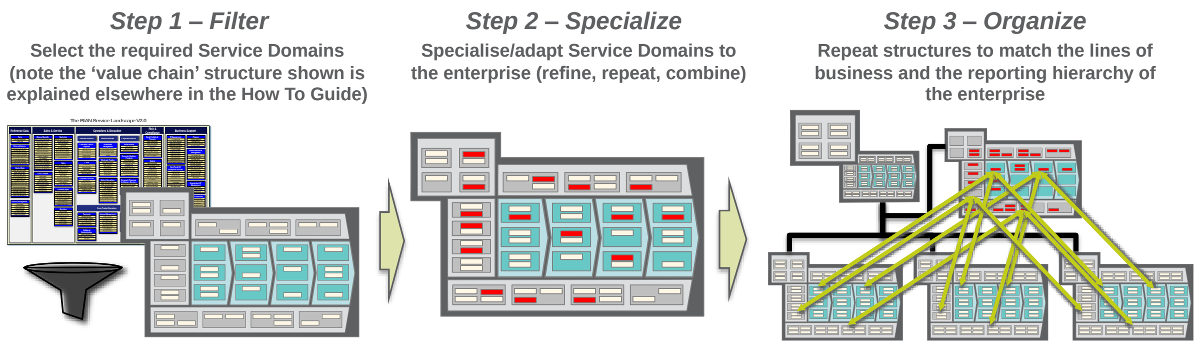

Step 1 – Filter
Select the required Service Domains (note the 'value chain' structure shown is explained elsewhere in the How To Guide)
Step 2 – Specialize
Specialise/adapt Service Domains to the enterprise (refine, repeat, combine)
Step 3 – Organize
Repeat structures to match the lines of business and the reporting hierarchy of the enterprise

Figure 44: Three stage process for defining a blueprint

The BIAN Service Landscape is intended to contain the range of Service Domains needed to support any bank. In the first step the scope of activities is used to filter out any Service Domains that are not needed, for example because they support products, services or channels that are not used in the target bank.

In the second step it may be necessary to revisit the scoping decision made by BIAN as already described in Section 2.5 of this document. As a result, selected Service Domains may be combined or duplicated and specialized. It may also be appropriate to rename Service Domains to match the prevailing terminology of the Enterprise (but note that this should not change the role of the Service Domain).

Finally the operational layout of the enterprise is mapped. This is likely to involve duplicating groups of Service Domains to reflect lines of business, geographic deployments and regional/shared operations. In addition the legal entity/reporting structure of the organization need to be captured.

Specific consideration needs to be given to shared or centralized and common or synchronized business functions. For centralized functions there will need to be two types of instance of the Service Domains. One provides the shared service and the other is a 'proxy' version in each of the supported locations to act as consolidation access points for the shared services. For example if payments processing is centralized there will be one or more shared central processing Service Domains and proxy instances at each of the local access points that connect to the central facility.

For common or synchronized business activities the configuration is slightly different. In this case there will be repeated copies of Service Domains each fulfilling a local role at each location. These local instances will then report to an additional Service Domain instance that is responsible for maintaining the regional consolidated/synchronized view. For example, risk management of a multinational may be handled by local risk management functions in different locations but with a consolidated regional or enterprise-wide risk perspective also being maintained at a regional/central location.

As noted more detail descriptions and examples of this enterprise blueprint definition process can be found in document three of the BIAN ‘How-to Guide - Applying the BIAN Standard.’

4.8.2 Analysis Supported by the Enterprise Blueprint

The BIAN Service Domains define discrete business functional capacities at a high level. Once the Service Domains have been mapped into an enterprise blueprint the service domain aligned elements of the blueprint can be augmented with additional specification details covering the target functional and non-functional requirements. Some example templates used are described in document three of this series.

The same framework can be used to support business and technical planning and analysis. Three broad categories can be considered:

1. ***Capability Overlays*** – typically using more detailed functional specifications of the Service Domains, resources such as organizational units and more commonly production systems/business applications can be mapped to reveal gaps, duplications and miss-alignments.
2. ***Performance Measures*** – target and current measurements can be defined and tracked for business and technology related aspects. In this way the blueprint can act as a management dashboard.
3. ***Feature or Property*** – different properties/characteristics can be associated with the service domain elements to guide planning decisions and analysis. For example relative cost, business criticality and more complex characteristics such as centralized/decentralized, in-sourced/outsourced. The possible attributions are practically unlimited and can be selected and calibrated to match the types of insights/decisions they support.

The range of approaches, techniques and possible used of the enterprise blueprint are extensive. Often the attributions will be applied to related groups of Service Domains clustered in Business Domains as the individual Service Domains can be too fine grained for the high level analysis. As noted more detail and examples can be found elsewhere in the How-to Guides. A summary if the above description is captured in the next figure:

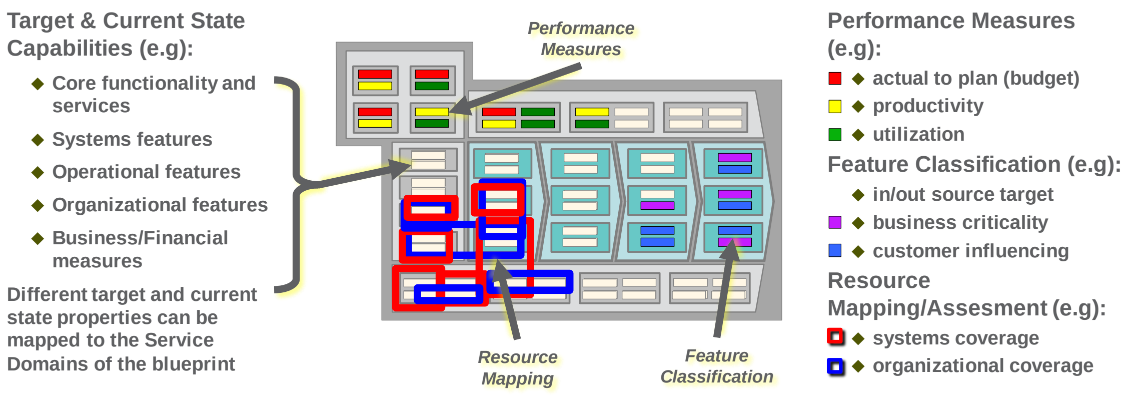
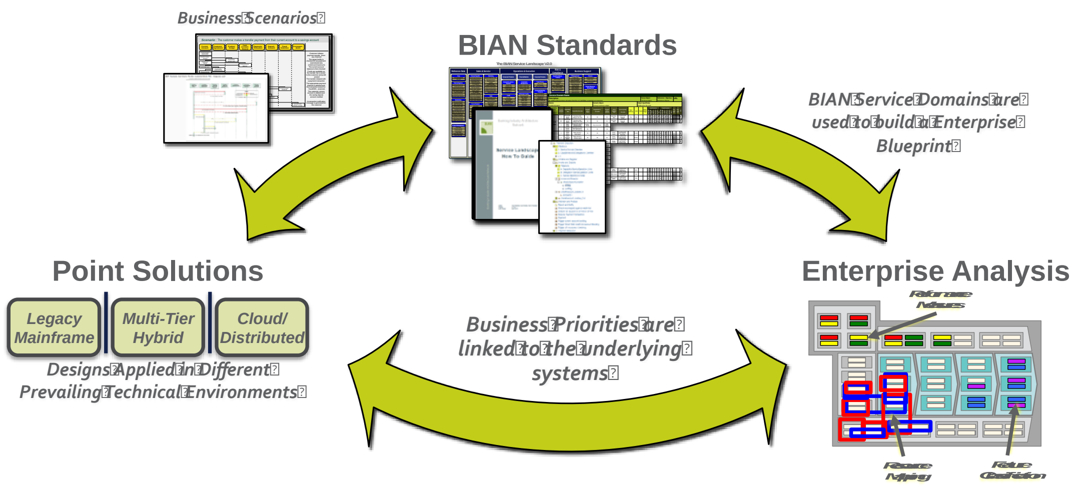

Target & Current State Capabilities (e.g):
Core functionality and services
Systems features
Operational features
Organizational features
Business/Financial measures
Different target and current state properties can be mapped to the Service Domains of the blueprint
Performance Measures
Resource Mapping
Feature Classification
Performance Measures (e.g):
🟥 actual to plan (budget)
🟨 productivity
🟩 utilization
Feature Classification (e.g):
in/out source target
🟪 business criticality
🟦 customer influencing
Resource Mapping/Assesment (e.g):
🟥 systems coverage
🟦 organizational coverage

Figure 45: Using the enterprise blueprint for planning & analysis

4.8.3 Linking Between Business & Technical Assessments

The final observation relating to the use of BIAN Service Domains to create an enterprise blueprint notes that the same elements in the blueprint map to those used to implement targeted solutions. As a result the link between the high level planning and analysis performed using the blueprint and the underlying systems is greatly simplified. As a result of the common BIAN Service Domain partitions investment decisions can be related directly to the underlying systems solutions. This linkage is captured in the figure below:


Business Scenarios
BIAN Standards
BIAN Service Domains are used to build a Enterprise Blueprint
Point Solutions
Legacy Mainframe
Multi-Tier Hybrid
Cloud/ Distributed
Designs Applied in Different Prevailing Technical Environments
Business Priorities are linked to the underlying systems
Enterprise Analysis

The two prevailing uses of the BIAN standard can be linked

Figure 46: BIAN designs help bridge between point and enterprise viewpoints

The arrow connecting the two deployment views underscores the connection.

5 Conclusion

This document covering the BIAN design principles and techniques is intended to provide business and systems architects with an explanation of the approaches BIAN has developed in order to define canonical SOA designs for the financial services industry (with initial focus on banking). It contains the current approaches used and under discussion within BIAN at the time of publication.

The document is a working document that will continue to be updated as BIAN refines these design principles and techniques based on practical experience and feedback from the membership and others in the industry. BIAN published these design principles in order to encourage an active debate on a practical approach to develop canonical SOA designs that can support the industry.