Integration Framework

Business Services (Contracts) Concepts and Principles

GB942CP

TM Forum Approved Version 3.0

May, 2011

Notice

This material, including documents, code and models, has been through review cycles; however, due to the inherent complexity in the design and implementation of software and systems, no liability is accepted for any errors or omissions or for consequences of any use made of this material.

Under no circumstances will the TM Forum be liable for direct or indirect damages or any costs or losses resulting from the use of this specification. The risk of designing and implementing products in accordance with this specification is borne solely by the user of this specification.

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

Executive Summary

The intended audience for this guide book is IT managers, architects and developers. It is recognized that due to their respective needs some sections of this book are more technical than others. For those just wishing an overview then the first couple of chapters give a sufficient explanation of Business Services. This Concepts and Principles Guidebook is also intended to be used in conjunction with the open-source tooling software being developed and GB942 U User Guidelines.

This guidebook produced by the Architecture Harmonization (AH) team, focuses on the unification of the interface, information model activities with the Frameworx Business Services(also known as Contracts) across the TM Forum ranging from framework to real implementation across all aspects of Digital Media, Communications and Information business operations. This work extends and adds detail to the TM Forum Frameworx (formally known as NGOSS) lifecycle model http://www.tmforum.org/browse.aspx?catid=1912 . There is a clear need to identify the dependencies of the key Frameworx artifacts: Information Framework (SID), Business Process Framework (eTOM), Application Framework (TAM), TM Forum Interfaces (mTOP, OSS/J, IPDR, CO~OP and new interfaces as they are developed) and Business Services in a Service Oriented Architecture (SOA) context ensuring a seamless migration to the future, while adhering to a common architecture. The document shows how Business Services provide the necessary linkage between the key Frameworx artifacts.

This document provides a full definition for **Business Services** from Business and System views. A Frameworx Business Service (aka Contract) represents a specification of system capability to achieve a stated business purpose. A Business Service includes a defined interface and may also define pre- and post-conditions, semantics for using the service(s), policies affecting the configuration, use, and operation of the service. A Business Service implements one or more use cases (task level processes). The document also provides direction for the Implementation view (TM Forum Interface Program - TIP) and for tooling (Tooling Taskforce and subsequent program work).

Frameworx Business Service extends SOA to recognize that there is a relationship between specific client and server systems in the application environment. This relationship may be short-lived or long-lived based on the specification for a given Business Service. The relationship may be implemented via point-to-point interactions or via an integration framework, such as hub-and-spoke or an Enterprise Server Bus (ESB). Business Service may be state-full or stateless depending on the business need. These systems are defined by the Application Framework. SOA is also extended with granularity guidelines that derive from Business Process Framework and finally the information model that is implemented in **Business Services** is compliant with Information Framework (aka SID).

Business Services add Service Level Agreements (SLA) to the interface specification and record metrics against these SLAs. The aim is to produce an interface specification for applications that is expressed in the language of the business and can be implemented with a range of different technologies. Ideally the implementation will be compliant to a TM Forum standard, such as TM Forum Interface Program. The Business Service is technology agnostic providing the ability to harmonize disparate interface standards. The granularity, fine or larger grained, is based on the number of use cases it implements.

The concepts and principles presented in this Guide Book provide a single, adaptable, standardized solution environment that supports all necessary service operations across the entire range of properties for a converged network, operations and business continuum, maintaining alignment of key Frameworx artifacts and providing a solution environment that continuously supports application needs through growth and change in network and service capabilities and through the evolution of implementation technologies.

We expect that TM Forum will manage real world, pragmatic Business Services based on the definition presented here using tooling so that the standards can be conformance-managed and change-controlled and so that the standards can be applied directly to implementations. The current Frameworx Business Service meta-model is implemented in the Eclipse tooling as a standard Ecore meta-model.

1 Overview to GB942 Frameworx Business Services Principles and Guidelines

This section provides an overview of the key concepts and principles.

1.1 Business Service Dependency Model and Value Chain

A fundamental difference between TM Forum Business Services and Service Oriented Architecture (SOA) is that Business Services define a 'dependency model'.

SOA is based on each application advertising the IT services it supplies for use by other applications. This is an 'open market stall' approach, where, on an individual transaction basis, consumers use a directory of stall owners to pick the services they need and identify the particular a stall where that service is available. Not even market stall owners work in such a high risk way. Usually consumers will establish a relationship with a particular stall, enabling the stall holder to stock to anticipate demand.

The SOA approach hides the fact that the provider application, the market stall holder, also needs to consume resources and services in order to be able to provide its own services and the rate of consumption by the stall holder is clearly linked to the rate of demand on the stall. Hiding such critical dependencies prevents systems architects from building efficient IT solutions.

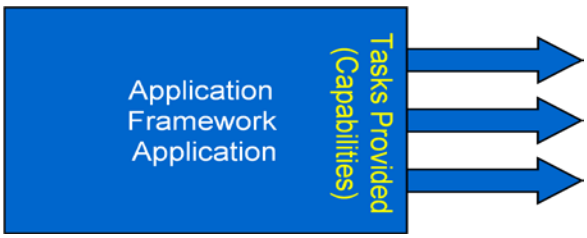


Figure 1.1-1 SOA Application with a Capability Model

Modern industry, both service and manufacturing, recognizes the role of the supply chain that links suppliers to consumers in a sequence of mutually dependent relationships. Each link in the chain derives value by being in this chain.

In contrast to SOA, TM Forum Business Services takes a 'value chain' approach, where an application not only declares the services it offers, but also declares the service it is dependent on. The approach has a 'dependency model', as well as the 'capability model' found in SOA. The result is that the systems architect can assemble a value chain of components into a solution where each application is optimally sized to deliver the demand on them. This can only be done by knowing both the dependencies and capabilities of each component available for the solution.

The TM Forum Business Service adds further business and operational realism to the SOA model by defining the granularity of the SOA service according to a business, or operational model. In TM Forum, the SOA service becomes the Business Process Framework Task.

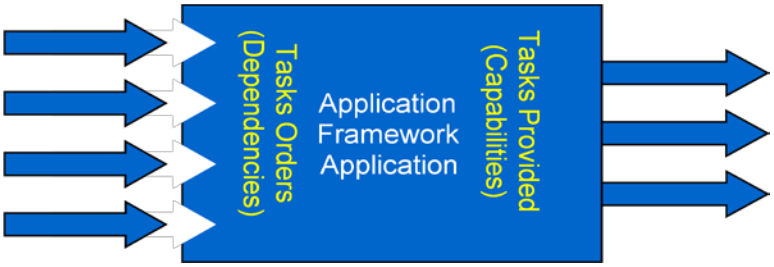


Figure 1.1-2 TM Forum Business Service Application - Dependency and Capability Model

The Business Service acts to encapsulate the business process by highlighting each application pair in the value chain and pairing capability-to-dependency for each distinct business or operational activity.

The encapsulation also has a time dimension, which means that pairing is not the transient browsing model of the market stall, but an ongoing relationship that reflects the fact that a value chain must have stability in order to deliver maximum value.

Finally, this stable, Business Service pairing enables the use of Service Level Agreements in the Business Service to not only specify what is achieved, but the manner in which it is achieved, such as volumes, performance and feedback levels that needs to be supported. Again this is only practical if the supplier has confidence about demand in the context of a value chain.

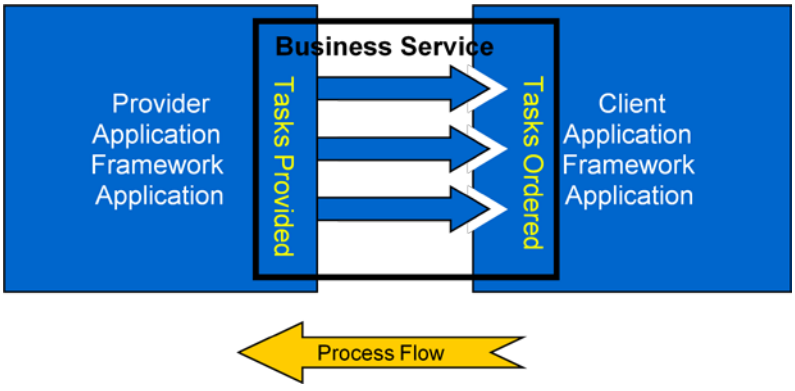


Figure 1.1-3 : TM Forum Business Service Interface

Note that the business or operational process flow is from client to provider.

The TM Forum Business Service approach is based on the need for the industry to construct systems architectures in an optimum manner. The optimization is based on the

value chain, where the overall chain is optimized, not just the individual links in the chain. The result is a shift to a solutions architecture perspective from the vendor perspective. This shift puts the focus on the Business's need for an optimized solution (the supply and value chain), not just on independently optimized application.

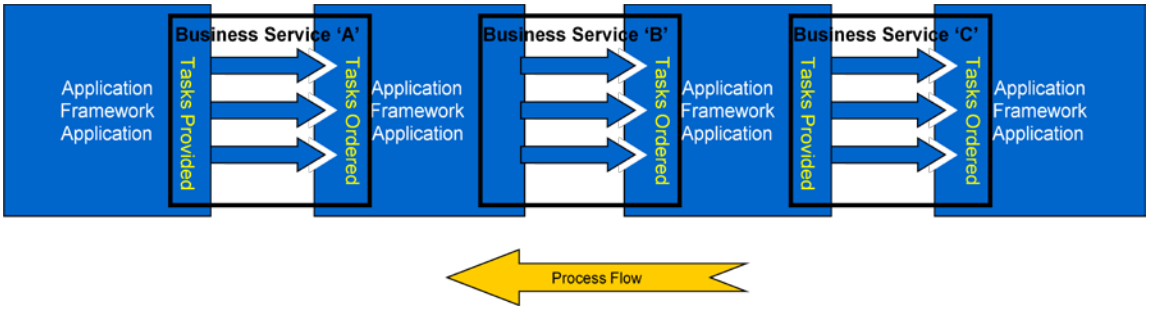


Figure 1.1-4 The Systems Architecture as a Value Chain

1.2 Business Service Model Features

The TM Forum Business Service is an (interface) technology neutral specification of interface behavior from an operational perspective. The interface neutrality is the important consideration here, but for reference this means that a Business Service specifies both the Business and System Views of the Frameworx Lifecycle. This standard is aimed at specifying an interface between two systems and includes both textual and machine readable components. The specification is layered as follows.

- The *Business Service Template* is a single, common standard artifact used to derive Business Services with a specific business purpose. This will be in Ecore and the Domain-Specific Language (DSL) will form the Business Service tool.

- *Business Service Specifications* are constructed from the template to address specific business or operational purposes between specified Application Framework applications. These will be produced using Business Service tooling.
  - Core Business Service Specifications will be standard artifacts that will be version controlled by the TM Forum.
  - User can also further specify standard Business Services to meet more particular requirements, for example, for specific network technologies. These may remain proprietary or be fed back to the TM Forum to become refinements of the standard.

- *Implementations of Business Service Specification* are (interface) technology specific implementations of the Business Service Specification. The implementation of Business Services is the remit of the TM Forum Interface Program. Note that standard Business Services can have custom implementations, which provides compatibility with existing deployments.

- *Business Service Instances are* instantiated from implementations of Business Service Specification in running applications.

The Business Service Specification will also contain features that are (interface) technology-neutral, but are required to control and guide the implementation. These are not concerned with the implementation of Business Service Tasks in the provider system, but are concerned with messaging structure and process control within the implementation. These features ensure that the implementation does not violate the principles of the TM Forum Business Service.

The following diagram, illustrates the relationships between the layers.

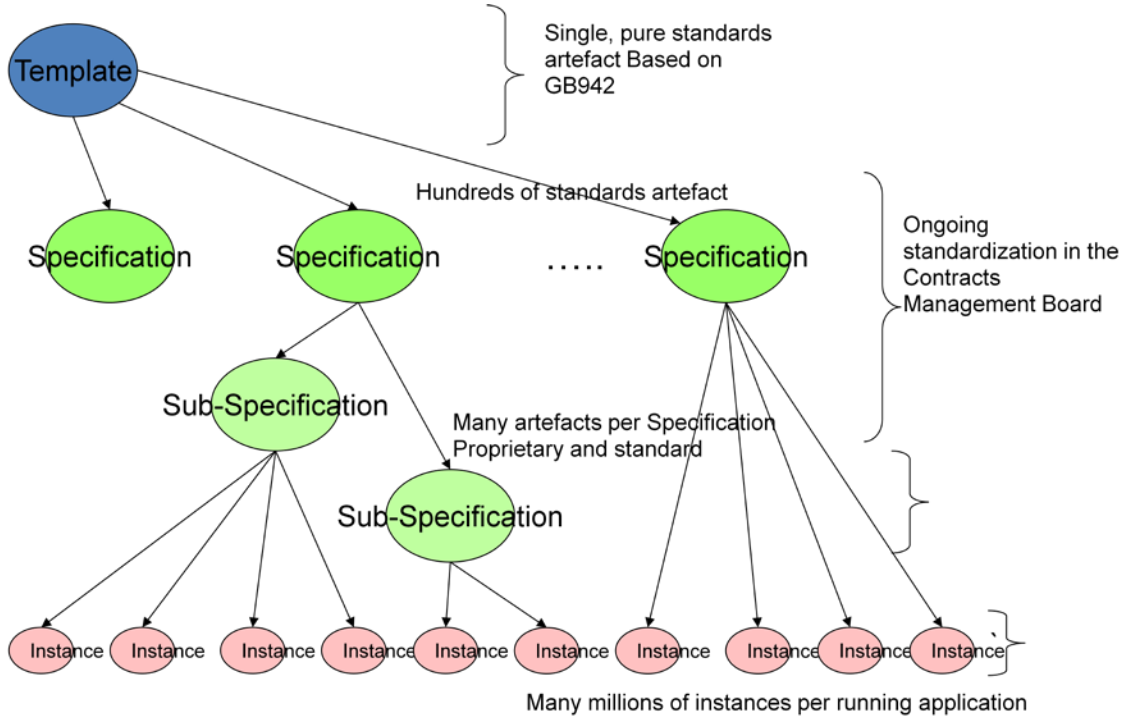


Figure 1.2-1 TM Forum Business Services Lifecycle in the Standardization Process

There are three high-level structures in the TM Forum Business Service model.

1. Business Service
2. Business Service Order
3. Task Order

The full Business Service model supplements these structures with lifecycle behaviour and dependencies, which are explicit in Business Service Specifications.
The Business Service Specification is used as part of the standardization process and is exchanged between Business Service tools.

At the instance level, both types of Orders are exchanged across the wire. They both contain information that is mastered in the Business Service Instance. The Business

Service Instance itself does not need to be exchanged over the wire, as it is derived from a standard reference, the Business Service Specification, which is available to both client and provider applications and whose structure and function is agreed by both client and provider applications at the design stage of the solution. This specification is an implementation of a standard or proprietary Business Service Specification. A copy of the Business Service Instance maybe held by the provider and client systems, but the provider system is the master.
The following further details the use of Orders to generate behaviour in the interface.

Introduction to Orders, both Business Service and Task

Orders should be seen as instructions. They are not intended to be *primary* data items, but are derived from the Business Service data and used to convey instructions from client to provider applications and return results from provider to client. The Order instances and the results are permanently stored in the record of the appropriate Business Service instance, which are the *primary* data structures.

Each Order instance is a document, or data structure that is created by the client application and sufficiently populated by the client to enable the provider to process the Order. The Order data structure will not be completely populated at this stage, as the provider will also add information to it.

As far as the operation of the interface is concerned, two types of interaction results.

1. During the course of the provider processing, unsolicited status updates are passed from provider to client application as appropriate and as specified by the Business Service SLAs.
2. On completion of the Order processing, the Order document, or data structure is returned by the provider to the client application with the specified return fields completed. The provider application will also have updated the Business Service record in order to maintain a permanent record of that specific interaction within the remit of the Business Service.

Note that the *implementation* of this exchange mechanism may separate out request and response data, but the effect from an implementation technology neutral perspective remains the same.

In general, Orders can be sent from client to provider application without prior Order processing completing. Any necessary sequencing is achieved by the use of preconditions. For example, the processing of an Order may have a precondition dependent on the prior successful completion of other Orders, as in the case that a service cannot be *deactivated* unless it has been successfully *activated*.

Orders have a specified action. For example, the Business Service Order can be used to *instantiate* a new Business Service instance, or to *suspend* a running instance. Similarly a Task Order can be to *execute* a Task or to *abort* a running Task.

Note that the Business Service Template will specify all of the actions applicable to any Business Service Order or Task Order. Subsequent Business Service Specifications will define appropriate tasks (actions) specific to them, but the actions associated with a Business Service or a Task are common to all Business Service processing. Full details

are found in the formal TM Forum Business Service model contained within the Business Service tooling activities. (REF tooling task force work)

Preconditions determine whether an Order can be processed by the provider application.

1.2.1 Business Service Orders

Business Service Orders are used by the client application to control the Business Service instance as a whole. The first Business Service Order must have an *instantiate* order action that will cause the provider application to consider the preconditions of the Order and instantiate a new Business Service instance if allowed. Subsequent Business Service Orders will control the lifecycle of the Business Service through to completion or *annulment*.

The candidate set of Business Service Order actions are:

- instantiate
- suspend
- resume
- annul
- modify SLA features
- archive.

Note that Business Service completion is a function of the provider application and cannot be forced by the client through an Order. The client can only force an annulment.

Task Orders

Task Orders are used by the client application to control the Task execution within the remit of an existing Business Service instance. The first Task Order must have an *execute* Order action, which will cause the provider application to consider the preconditions of the Order to determine whether the Task can be executed by the provider application. Subsequent Task Orders will control the lifecycle of the Task through to completion or *abort*.

The candidate set of Task Order actions are:

- execute
- suspend
- resume
- abort
- modify SLA features

Note that Task completion is a function of the provider application and cannot be forced by the client by a Task Order. The client can only force an abort.

Task Orders can be bundled together or with Business Service Orders so long it is meaningful. A typical example would be to issue *instantiate* Business Service Order and an *execute* Task Order in the same package.

The following is an overview of the interaction between client and provider applications for a Business Service instance.

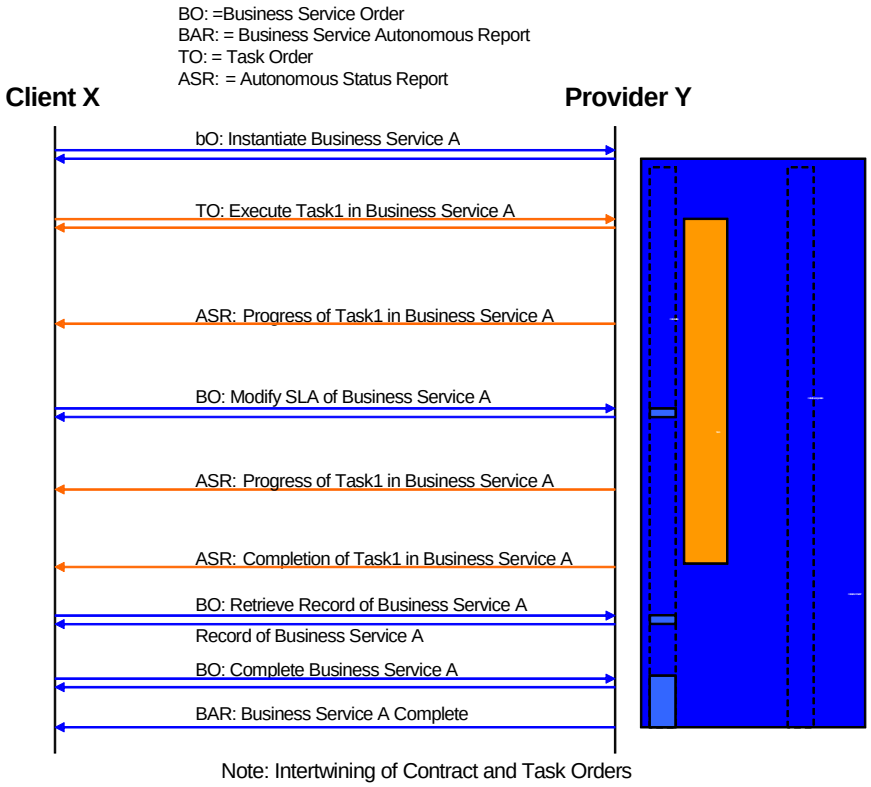
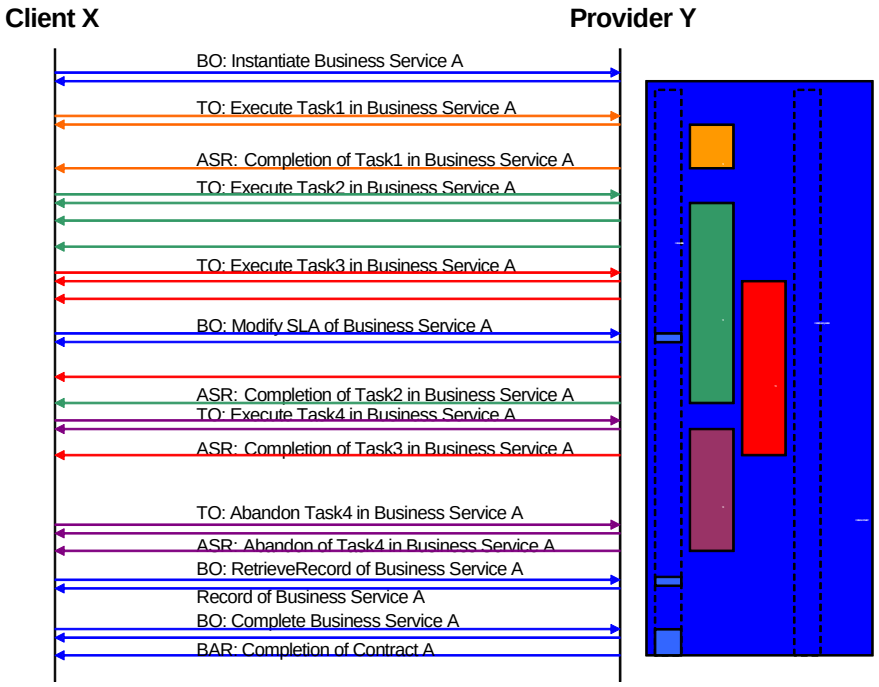


Figure 1.2-2 Client provider interaction

The next diagram illustrates the expected complexity possible within a single Business Service instance.



Note: Overlapping tasks and intertwining of Contract and Task Orders

Figure 1.2-3 Client provider interaction overlapping tasks

1.2.2 Preconditions

Each Business Service Order action and each Task Order action has its own set of preconditions that must be true in order for the Order to be carried out by the provider. The preconditions are also used as a means of supplying data from the client to the provider application (ref. Definition of precondition where this is formally stated).

The method used in the TM Forum Business Service is to specify the preconditions in a logical expression, consisting of zero or more logical clauses. The expression is evaluated by the provider application and if it resolves to true, then the Order can be processed. If it resolves to false, then the Order fails with a 'preconditions failure status' and an explanation as to which clause or clauses were responsible for the failure.

Each clause can reference the global inventory model, entities, entity attributes and attribute values. Furthermore the clauses can reference the state of Business Service as found in the Business Service record. For example a clause may reference the completion status of a prior Task execution instance.

The following is an important principle and guideline.

It is a guiding principle of TM Forum Business Services that the precondition, particularly Task precondition clauses should remain simple, ideally restricted to data lookups. There is a danger that a Business Service could be specified where the preconditions require significant processing, in effect acting to ensure that the Task can be completed by the provider system. This means that the business-as-normal processing has migrated to the Business Service precondition test, rather than where it should be, which is in normal processing by the provider to execute the Task. This is not the intention of Business Service preconditions

The preconditions should act as a simple gate, or sanity check, to ensure that the provider system is able to start processing the Order. It is entirely acceptable that during the subsequent processing, the provider application finds it cannot successfully complete the Task and has to return a failure status.

1.2.3 SLA Features

SLA features determine how the provider system behaves with regard to the Business Service. It enriches the Business Service with metrics that are essential for the operational use of the interface and application, controlling features such as performance, feedback, volumes, security and so on.

The SLA feature consists of a triple:
{metric, required value/range, actual value/range}

For example {availability, 99.999%, 99.9993%} – within SLA

Both the Business Service and each Task will have its own set of SLA features. Certain SLA features are common to all Business Services and are specified in the Business Service Template. However it is expected that each Business Service Specification will define SLA features that are relevant to the particular business purpose and subject that the Business Service is specified for.

There may be some SLA features that govern the Task as a whole that apply to all Order actions and there may be other SLA features that are particular to an individual Order Actions. The system should be capable of supporting both types.

Standard SLA metrics include:
- Overall Time to execute Order
- Feedback level
- Time to respond to Order action (separate SLA feature for each action)
  e.g., time to instantiate Business Service, time to execute Task…

The Business Service record is used to keep track of the actuals against the required SLA metrics values. This gives a powerful source of Business Intelligence to analyze the operational behaviour of the solution.

1.3 Implementation Handover

The implementation of Business Service Specifications is not determined by the specification. Although the Business Service specifies a specific client and provider application, it does not specify how Business Service Orders are exchanged between those applications and neither point-to-point, nor bus architectures are ruled in or out. The TM Forum Business Service architecture will fit into whatever enterprise integration solution it is required to do so, and the only constraints are based on the need to meet the specified SLAs.

A separate activity in TM Forum coordinated as part of the Interface Program will provide specific implementation forms of the Business Service and Task Orders so as to maximize interoperability between applications in a systems assembly. At the point of publication of this document the Interface Program work is not yet complete.

2 Use Cases, Business Process Framework Processes, Business Services and SOA Services

This section describes the relationship between use cases, Business Process Framework processes, and tasks used within Business Services to show how Business Services complete the TM Forum Frameworx by binding the key Frameworx artifacts together.

The following diagram is taken from Business Process Framework User Guidelines (GB921 U, ver1.1 Feb 2007). It illustrates how the process hierarchy is implemented by specific types of process at different layers. This is a good place to start to see how **Business Services** fit to Business Process Framework, however, it should be noted that the principles described here may need to be interpreted for the particular business problem being addressed. The levels in Business Process Framework are not completely uniform in their granularity for all business functions and it may be necessary to re-interpret the levels discussed here to suit the level of detail provided by Business Process Framework for the particular business function being considered.

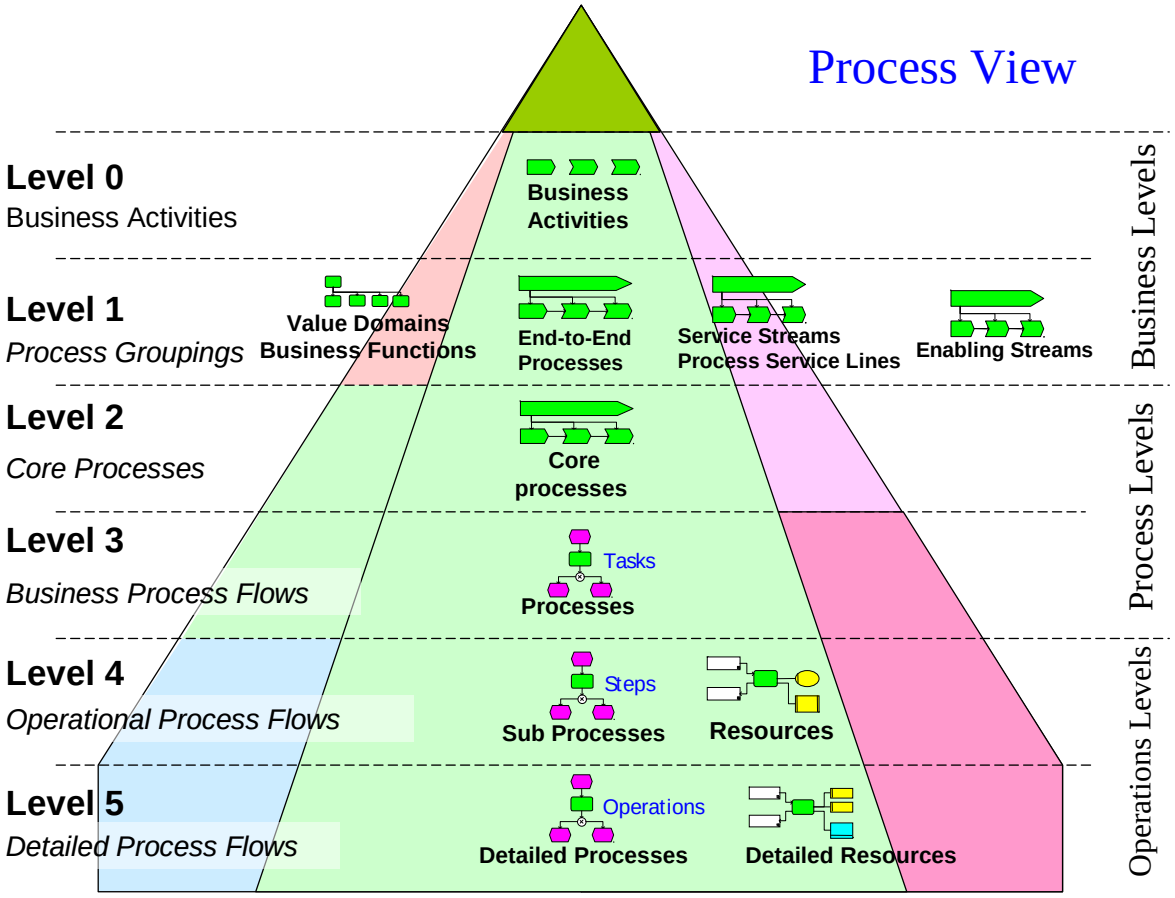


Figure 1.3-1 - Process Hierarchy: Processes and Resources

Given the proviso, the interpretation put on this diagram by this project is as follows.

Level 3 provides fully articulated and specific business processes with a real world business goal. It is correct to refer to these as 'process elements' because calling a process 'end-to-end' is a relative value judgment, depending on the perspective of the observer. However, these processes define significant and meaningful business activities. These processes are composed of Tasks and can be represented by Use Cases

At Level 4, the expectation is that software, or manually operated systems, will be able to deliver the function without further decomposition to an *exposed and explicit business process.* In this sense they are considered to be atomic Tasks that are combined into business processes at the higher level. From a software implementation perspective, these Tasks are SOA services provided by a system. In this way, Business Process Framework is used to constrain the granularity of SOA in Frameworx. Tasks are grouped together into Business Services in order to support a specific business purpose between clients and providers.

Below Level 4, further decomposition is considered to be an internal function of the system, which may be custom or standardized. A Task may be implemented by one or more Operations of the system, but this is an IT implementation issue, not a business process issue. The business functions and differentiation need to be captured at the higher levels of Business Process Framework for Business Process Framework to have value.[1]

Above Level 3 there are groupings of processes into business areas.

The Frameworx Business Service inherits many of the properties of the Use Case, but adds specialist artifacts required by the Business Service. This reflects the transition from business processes to system functions in Business Process Framework. A Level 3 process element may make use of one or more Business Services and a Business Service may be used by one or more business processes. A business process may make use of a number of applications, but it can be described by a Use Case, as described later. Furthermore, Tasks have an independent existence to Business Services as they are Business Process Framework entities. The assumption is that any Task (e.g. Add Card) can be used in more than one Business Service (e.g. Planning, Service Provisioning and Repair Business Services).

This establishes a relationship between Business Services and Business Process Framework processes, where Business Process Framework is used to determine the Tasks contained in a Business Service. Higher level Business Process Framework processes flow through Business Services. However, Business Services act to unify all four of the primary Frameworx artifacts, including Application Framework and Information Framework. The following diagram illustrates the relationships.

[1] Business Process Framework adds value by capturing the standard processes at layers 1, 2 & 3. The authors are not yet clear on how differentiating processes are managed by Business Process Framework

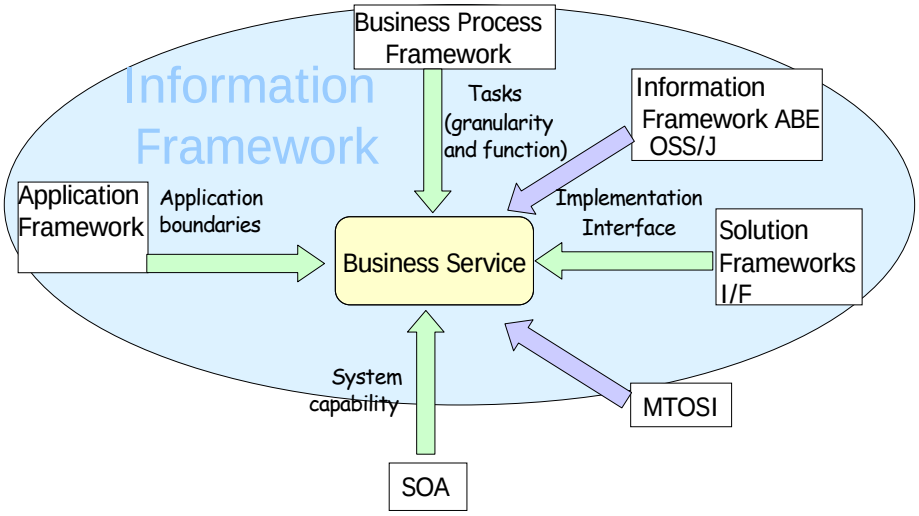


Figure 1.3-2 - Unifying Frameworx Artifacts

A Business Service specifies the requirements for a business interaction between two systems or applications. It does this by considering a managed entity, or entities, over an extended period, such as the lifetime of that entity (but not necessarily so). There is a potential confusion in terminology here as it is possible to refer to 'applications' as specified by Application Framework or 'systems, as specified by the Frameworx System Lifecycle View. The situation is open to personal preference, but in this document, the Frameworx Lifecycle is considered marginally more expressive than Application Framework terminology due to the considerations in Figure 1.3-3 - Unifying Frameworx Artifacts, below.

In one case, Business Services specify the relationship between applications. This is the role of the System Lifecycle View of the Business Service (System Business Service for short). Application Framework provides the application map and Business Services provide the relationship between the applications. The Tasks and the Business Services are specified with a data model that conforms to Information Framework. Finally, the TM Forum Interface Program is responsible for the Implementation Lifecycle View of the Frameworx Business Service. This will populate the System View with implementation artifacts that will specify the interface at the IT level.

In another case, Business Services are internal to an application and represent capability/functionality necessary to form a complete application. The current focus of the Architecture Harmonization team and the focus of this guidebook are on the first case - the relationship between applications.

Furthermore, Tasks have an independent existence to Business Services as they are Business Process Framework entities The assumption is that any Task (e.g. Add Card) can be used in more than one Business Service (e.g. Planning, Service Provisioning and Repair Business Services).This now gives a dependency relationship between these key artifacts that grounds each of the specifications by the need to support the others. This is powerful for the industry as it enables the TM Forum standards to iterate towards completion as each artifact is enhanced to support the requirements of the others.

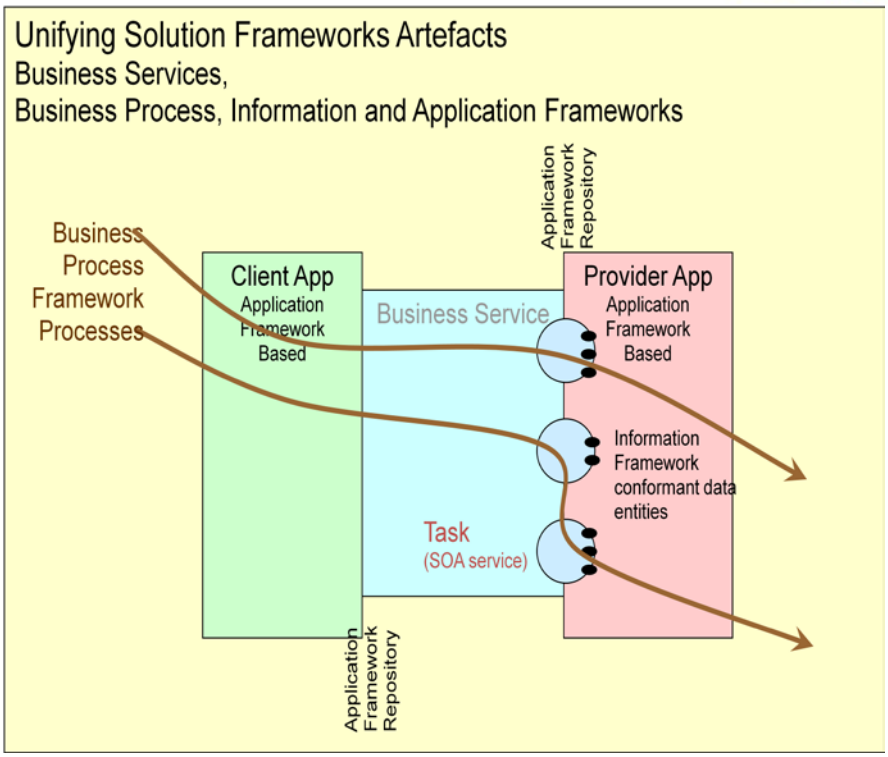


Figure 1.3-3 - Unifying Frameworx Artifacts

2.1 Frameworx Lifecycle Views of Business Services

Reference has already been made to the Frameworx Business Service having different lifecycle views. The following will describe this more fully.

At its most general, a Frameworx Business Service is designed to have a similar role to a real world business contractual relationship in that it relates two trading entities. This may be customer and provider, provider and vendor or trading entities that are internal to the provider. This brings out the additional value of a Business Service over traditional interfaces as it not only describes functions over an interface, but a trading relationship with associated service level agreements (SLAs) and lifecycle.

This most general case is captured in the Business View of a Business Service as it captures the business relationship between the client and provider, without reference to systems. As a result, the scope of a Business View Business Service can be very broad, for example specifying the entirety of the relationship between customer and provider. In this case it reverts to the actual Business Service between the two.

However, the Business View has a second use; it can also be the business level description of a Business Service that is in a later and more specific lifestyle stage. This becomes clear as we consider the subsequent lifecycle stages of a Business Service.

The System View of a Frameworx Business Service specifies the relationship between applications, as specified in Application Framework. Note that these systems need not be computer systems, but may be manual functions. Application Framework specifies functional blocks with a specific business purpose, but without implementation details. With this level of formality, it is desirable to represent the Business Service in machine readable form, so the System View will be in a format such as WSDL. However, it is also necessary to describe the business relationship that the Business Service represents to a human Solutions designer and this would be found in the Business View of the same Business Service.

Moving forward in the lifecycle, the Implementation view adds technical implementation details to the System View of the Business Service. This becomes necessary when the system is implemented by a specific software solution. The Deployment View Business Service captures the additional details when specific software systems are integrated into a specific deployment scenario.

Each subsequent lifecycle view of a Business Service builds on the previous by adding increasingly specific detail and applying to increasingly specific situations.

The following table summarizes the situation. The columns represent the different Lifecycle Views that a Business Service can be applied to; from pure business only to full deployment.

The rows represent the increasing detail of standards artefacts required to support each type of Business Service. So a Business Service that is only applied to the Business View needs only business level artifacts and applies to generic trading entities. At the other extreme a Business Service taken from the Deployment View will need the business level artifacts that specify the business purpose of the Business Service, the system artifacts that relate to Application Framework, the implementation artifacts that determine the IT for the interface, (taken from TM Forum Interface Program), as well as the artifacts that are specific to the deployment. Such Business Services apply to deployed interfaces. The table represents the intermediate cases as well.

		Lifecycle Stage of Business Service			
		Business Service			
		Relevant trading entities			
Applicable View	**Business View**	Trading entities	Application Framework Applications	Software Solutions	Deployed Solutions
	System View	n/a	Application Framework Applications	Software Solutions	Deployed Solutions
	Implementation View	n/a	n/a	Software Solutions	Deployed Solutions
	Deployment View	n/a	n/a	n/a	Deployed Solutions

Figure 2.1-1 – Trading Entities and Lifecycle Views applicable to each lifecycle stage of a Business Service

The Business Service Template will accommodate all four lifecycle views of a Business Service. The extent to which the template is completed will depend on the lifecycle stage of the Business Service that is being specified. In addition the representation of each stage in the lifecycle reflects both the tooling and the role of each stage, as follows.

- Business View – textual description of Business Service behavior
- System View – meta-data defined in –UML or XML- are used to represent a Business Service created within design guides and schemas using design and configuration management tools.
- Implementation View – Business Service Tasks descriptions are enhanced by specific SOA service descriptions of standardized interfaces from MTOSI, OSS/J and the output of the TM Forum Interface Program. Custom implementations are also possible for compatibility.
- The definition of the Deployment View is deferred to TM Forum Interface Program.

2.2 Use Cases and Level 3 Business Process Elements

Use cases describe, through a series of interactions, an agreement between the stakeholders (actors) of a system and the system itself. System is used in a very broad sense and does not imply only a computer system, or a single system. Note that initially in the business view, the system view is not yet decided and so the business use case (as used here to assist with the Business Service specification) focuses on interactions between business entities (for example, two enterprises). The agreement described in the use case defines the expected behavior of the system along with any material, services, and other items or functionality that are exchanged between the stakeholders (users) and the system. It also describes, through reference to other use cases, different scenarios that can arise and the behavior represented by the different scenarios.

A business use case defines the steps to achieve a business goal, services to be provided by the solution, and the high level methods of operation. As the system view develops, a use case may be further decomposed to simplify/focus on goals to be achieved. Mapping a system view use case to specific target technologies brings rise to one or more implementation view use cases. The deployed solution represents functioning use cases within a defined application area.

Business Process Framework processes are ideal candidates for use cases, particularly when the definition of a business use case is considered.

Use Cases are used to describe business processes whereas Frameworx Business Service's primary purpose is to specify software system interaction.

Figure 2.2-1 shows the decomposition of a representative Business Process Framework Level 2 process element. Note that Business Process Framework process elements are normally applied within process flows to show how the individual process elements work together to achieve some larger business purpose. These process flows can be "end-to-end" in which case they show an overall view that typically extends from some external trigger (say, a customer order request) through the internal process steps that utilize the relevant sequence of individual Business Process Framework process elements to deliver a business result (say, a fulfilled customer order). Process flows can also have a more limited extent; say focusing on just part of some end-to-end view, because the intent is only to examine the area of process activity covered by the more limited flow. Process flows can be specified at any desired level of process decomposition, Level 1, 2, 3, etc, according to the degree of detail chosen as appropriate for the analysis in hand.

At Level 3 and below, process elements take on the preferred naming convention for use cases, providing a subtle hint of the relationship between use cases and Business Process Framework processes.

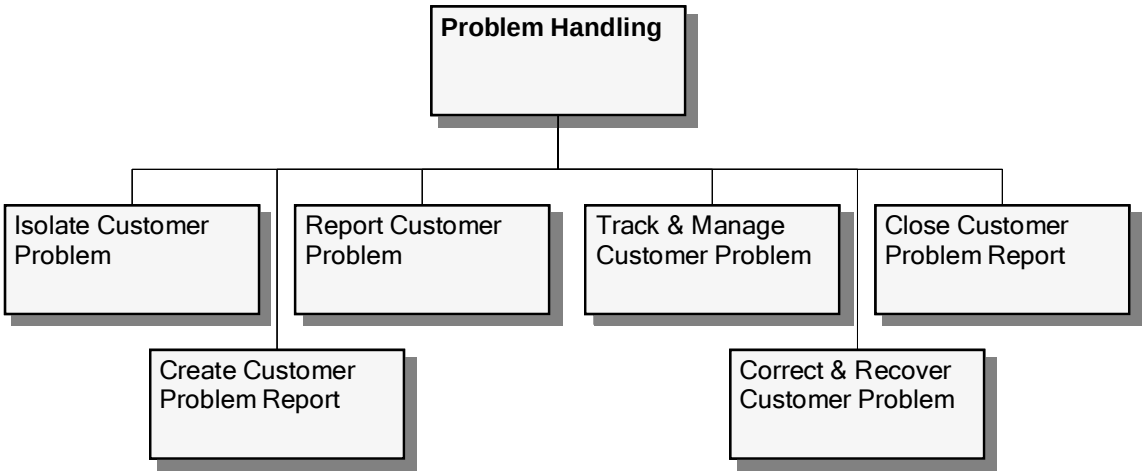


Figure 2.2-1 - Problem Handling Level 3 Processes

Candidate use cases don't stop at Level 3. Figure 2.2-2 - Isolate Customer Problem Level 4 Processes, shows that Level 4 process elements are also good candidates for use cases.

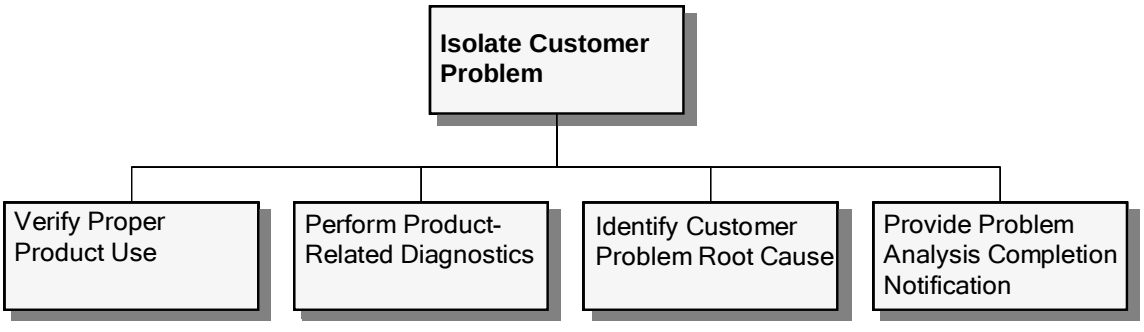


Figure 2.2-2 - Isolate Customer Problem Level 4 Processes

The identification of candidate use cases could proceed to lower levels of decomposition depending on the complexity of the process and the extent to which a process is automated. When a process step can be provided by a support system or the network, there is little value in standardizing its further decomposition. Its implementation is better seen as proprietary to the vendor and an opportunity for the vendor to provide differentiated value.

Note that identifying use cases by focusing in this way on a specific Business Process Framework process element, at the chosen level of decomposition, is an excellent starting point. As the goal is to define a Business Service, and this is positioned at some interaction boundary between business entities (in the business view), and then systems (in the system view), typically an individual process element will be "exposed" at this interaction boundary (in some cases, depending on the scope of the Business Service, there may be more than one process element that is "exposed" in this way). As the analysis develops, and the use case is forming to define the interactions, then a process flow becomes a very useful tool to establish the business context in which the Business Service will operate.

The interactions addressed in a business use case can therefore be expressed through a process flow, which may be end-to-end if this is helpful, or more limited, as indicated previously. However it is scoped, the process flow should be defined to bridge the interaction boundary at which the Business Service will be positioned, so as to provide insight into how the Business Service relates to other activity within each of the interacting entities. Note that only some of the process interactions in the flow are actually exposed at the interaction boundary where the Business Service is formed, so the other interactions (i.e. away from the interaction boundary) that are identified in the flow will not strictly form part of the eventual Business Service definition, but they do provide vital insight on the requirements for the context and behavior of the Business Service.

2.3 SOA Services and Business Services

.2.3.1 Service and Business Service Granularity

Understanding the granularity of SOA Services can be a stepping stone to understand the relationship between **Business Services** and Services. It would be a good thing if there was a relationship and that it was simple to describe.

When designing services from the ground up, it is helpful to categorize services according to a set of preexisting service models. These models already establish a proposed context and boundary within which the service can be modeled. Common service models include application services, entity-centric business services, and task-centric business services. These establish the utility-centric, entity-centric, and task-centric service contexts, respectively, and can be viewed as services that possess the best degrees of cohesion and coupling.

Task-centric context are required for services modeled to encapsulate process logic or use case steps. In this case, the thread that ties together the grouped logic or steps is a specific activity being automated by the service logic. Therefore, the use of verbs in service names is common. For example, a task-centric service may be called Isolate Customer Problem or Close Service Order, if that accurately represents the task's scope. This centricity exhibits the most preferred degrees of cohesion and coupling.

Another task-centric context for a Business Service is one that combines tasks associated with multiple, related entities. For example, the Create Subnetwork Connection Business Service implements tasks associated with the creation of the connection and associated termination points.

The utility-centric context is found in application services involving operations that encapsulate cross-cutting functions, such as event logging, exception handling, or notification. These reusable services need to be labeled according to a specific processing context, agnostic in terms of any particular solution environment. For example, a utility service might be named Notify. This centricity exhibits intermediate degrees of cohesion and coupling.

An entity-centric context is established in a business service that represents a specific business entity, such as a Customer Problem or Service Order. The labeling of entity-centric business services is often predetermined by the entity name. For example, a service may simply be named Invoice or Customer. This centricity exhibits the next best degrees of cohesion and coupling.

A final context is a Composite Business Service that is comprised of other task- and/or entity-centric Business Services. For example, a Customer Order Management Business Service may represent a composite of the Customer Order Business Service and the Product Business Service entity-centric Business Services based on their combined use by the Business Process Framework Order Handling process.

This all implies that the concept of an operation (method) on a business entity may be too fine grained to be considered a service.

With these centricities in mind, it appears that a task-centric service corresponds nicely to a Business Service that implements one or more use cases. The fine-grained approach to Business Services/use cases/services would assume that that a single lowest level use case/Business Process Framework process is implemented within a Business Service and that Business Service corresponds to a task-centric service.

Considering that Business Process Framework L2 and lower level processes manage the life-cycle of a small number of Information Framework Aggregate Business Entities (ABE), larger grained Business Services/services would implement more than one Business Process Framework-based use case.

At the upper level of granularity would be a Business Service that manages the entire life of an Information Framework ABE. This would be equivalent to an entity-centric service. There are already examples that have been provided/developed of entity-centric Business Services. They typically take on the name Manage X, where X is the name of an Information Framework ABE, such as Customer Problem or Service Order.

Based on this, it could be imagined that the name of such a Business Service could be just X or X Business Service, similar to the naming convention for services.

2.3.2. Additional SOA Service and Business Service Relationships

Another stepping stone to understanding the relationship between a SOA Service and a Business Service is to consider a Business Service as a specialization, enhancement, and simplification of a SOA service.

- Specialization – The SOA Reference model is industry agnostic and thus presents a generalized model that states a requirement for a process model and an information model. While not required for Business Services, the TM Forum does provide specific, or specialized, models, contained within the Process Framework, the Information Framework (including the Information Framework and the MTNM models). The Forum also provides a blueprint of Business Services within the Integration Framework and the Interface program's APIs.
- Enhancement – Business Service User Guidelines (GB942U) contains granularity guidelines, implementation guidance, and tool requirements. Additionally, an open source Business Service tool is under development that enables the specification and generation of Business Services. A Business Service extends/enhances SOA to recognize that there is a relationship between specific client and server systems in the application environment. This relationship may be short-lived or long-lived based on the specification for a given Business Service.
- Simplification – The contexts (centricities) of Business Services represents a simplification of the SOA Service concept. There are many contexts for SOA Services – task, process, entity, policy, utility, and others. The three centricities presented in this guidebook are adequate to support most requirements. Process and Policy can be viewed as specialized entity- and/ or task-centric Business Services.

3 The Frameworx Business Services

The specification of the Frameworx Business Service has previously been addressed from several aspects.

Early work on a documentary definition template based on the work of the Architecture activity in TM Forum053b is captured in **TM Forum410** and has been used to generate some example Business Services that were concentrated in the Business View and also the System View.

The TM Forum053b work was evolved into a UML representation of the Business Service Business View and was published within the Information Framework material as **GB922-1c**. Later work on the implications of moving into the System View and beyond was published as **TR138**. In parallel with this, work in the **mTOP** area and in the **Harmony Catalyst** (TMW, Nice 2007) created more insight into the Business Service specification, particularly moving on into the Implementation View. For mTOP, work has been done on a UML meta model for Business Services by the mTOP Methodology team, resulting in the MTOSI Service Description Meta Model. This has much in common with the Harmony Catalyst Business Service specification and the work in this document is intended to bring all three artifacts together.

The material set out below follows on from all this, and tries to present an integrated view of the Frameworx Business Service specification. The aim of this is to bring together elements of all the Frameworx Views into a unified model that can support any of the Views as desired.

At this stage, the model shown below is still evolving, but represents the current perspective on the overall Frameworx Business Service structure. Note that this version of the material concentrates on the SOA-oriented approach for using Frameworx Business Services.

3.1 Business Service Benefits/Differentiators

A Business Service specifies the requirements for a business interaction between two systems (see Introduction). It does this by considering a managed entity, or entities, over an extended period, such as the lifetime of that entity (but not necessarily so). It brings together the set of tasks (services) that the provider system is required to perform, expressed in terms of the Communications Service Provider (CSP) activities, such as 'create SNC across device'[2], or 'insert card into device'. In this way, **Business Services** are the final stage in Service Orientated Architecture (SOA), where SOA stops being an IT feature and starts to represent real world CSP activities that are provided by management systems. We have used CSP 'Tasks', rather than IT 'Services', in the Business Service specification to represent this transition.

The sequence of Tasks and their outcomes represent the lifecycle history of the Business Service. This is the Business View. The Business Service moves to the

[2] Sub Network Connection

Implementation View when technology specific Remote Procedure Calls are added to the Business Service to implement the Tasks.

In this way, a Business Service provides the context and purpose for a set of interactions between two systems in a way that fully specifies the requirements for implementation. Importantly, this specification of the meta-data is independent of the integration technology, so can be applied to all technologies.

Business Services reduce the risk and cost of integration by supplying standardized specifications off-the-shelf for each CSP activity and supplying sufficient detail to fully specify the CSP business process. Specifying an interface purely from low level APIs is complex, costly and high risk, which the SOA methodology does not fully address.

Business Services can be further refined for the specific needs of each CSP and each implementation if required. This is done by sub classifying the standard Business Service.

A Business Service is targeted at a specific business purpose, but in principle any given interface will support a number of different business purposes. The CSP is free to select which Business Services it will implement based on the specific purpose of the integration.

In addition, because the Business Service is integration technology neutral[3], the Business Service provides the perfect mechanism to harmonize different interfaces standards. This is important as new standards appear for emerging technologies. It also accelerates the process of standardizing emerging technologies, by providing pre-built business frameworks. A significant proportion of existing interaction patterns is transferable to new technologies.

By taking a business-level approach, SLA metrics can also be specified for the interaction, which are meaningful to the CSP business, such as 'time to complete'. The Business Service will specify these metrics and their required values.

Business Services are managed and reusable, much more so than individual APIs can be, because they specify common business requirements over the interface, rather than individual interface commands. This means that interfaces are specified 'top-down' according to business requirements, rather than 'bottom-up' by vendors offering a broad range of APIs, many of which are not used.

However, where standardized interface technologies are used, such as OSS/J or MTOSI, the Business Service becomes a fully standardized interface specification, enabling plug and play integration.

Business Services capture an extended relationship between client and provider covering a number of tasks over the subject's lifecycle. This enables greater autonomy between the systems and in particular, allows the provider system to implement the most efficient way of managing the subject or subjects. This not only reduces the complexity and risk of an over-specified interface, it also encourages vendors to add greater differentiation and value to their products.

[3] Integration Technology neutral is to denote the use of optional middleware stacks;

3.2 Business Service Requirements/Constraints

The definition of Business Services used here have been developed from the desire to represent interactions with the following features

- A Business Service is between a single client (initiator) and a single provider system in an established system solution design. However, Business Services may be reused by other client and provider systems.
- Business Services may capture an extended or ongoing relationship between the systems, as opposed to small scope, point interactions typical of APIs, or a short-lived relationship based on the specification of the Business Service.
- Business Services express a collaborative interaction, where a Business Service will be initiated without the necessity for all the preconditions for Business Service completion to be met. It is sufficient that the preconditions to initiate the Business Service are met. This characteristic is required if the Business Service is an extended interaction. This underlies that Business Services may be long-lived and state-full.
- A Business Service will be progressed through its lifecycle as the preconditions for lifecycle transitions are met. Both lifecycle steps and preconditions are explicit in the Business Service. The Transaction model is not specified in a Business Service the only declarative specification is the guarded post condition on completion of a Business Service.
- A Business Service will have a subject and will typically last for as long as the subject. It will capture the SLA metrics associated with the subject as well as any metrics specified for the Business Service itself. Subsequent releases of this specification will incorporate features from the SLAM and SQM programs.
- Business Services are scoped to satisfy business requirements, as opposed to small scope IT interactions. They provide the business context for the IT interactions.
- Business Services capture a delegated relationship, where the provider determines the manner in which the business requirements will be satisfied. The Business Service makes explicit only those aspects of the interaction that are not delegated, but are explicit requirements of the client.

These features express the business requirements of CSPs for network and systems integration. They reflect higher level requirements from the CSP to be able to roll-out new capability rapidly, with low cost and low risk, so as to be able to exploit innovations in the industry to deliver lower cost, improved customer experience and product innovations.

These features also meet the requirements of both network and software vendors to produce differentiated and innovative products that can be easily integrated into CSP solutions, so accelerating sales.

3.3 Business Service Overview

The role of the Business Service is to specify the business and system processes that take place between the client and the provider, those processes being specified by Business Process Framework Tasks.

The Figure 3.3-1 – Business Service Instance between Client and Provider illustrates a Business Service implementation, where the Tasks have been implemented with Operations taken from the interface specification of the provider system. Note that these commands are executed on the provider system, using the provider's implementation technology. The client system needs only to understand the integration technology that connects the client and the provider applications. Note that the Business Service does not determine, specify or constrain the nature of the integration solution, which may be point to point, via an enterprise bus or through other adapters.

Where an Enterprise bus is used, it is possible for the client to request a Business Service instantiation from a number of candidate providers. However, once a Business Service instance has been established between a client and a provider application, considerations such as transactional and data integrity implies that the Business Service executes its full lifecycle between that pair of applications.

3.3.1 Business Service Order

The Business Service Order is a request sent from the client application to the provider application to initiate action in the provider. An order can be sent to a provider application or an existing Business Service instance running in the provider.

In the first case, it will be an order for a new Business Service instance, for a specific business purpose, with a specific subject or subjects in the information model implementation. The business purpose is specified by the name of the Business Service template to be instantiated and the subject/subjects will be part of the parameters passed in the order to the provider application. There are likely to be other parameters, such as the required SLA values. If the preconditions of the Business Service template are met, then a new Business Service is instantiated. The Business Service instance Id and possibly other values are returned to the client.

In the second case, the Order will be to run a Business Service Task within a Business Service instance. The order specifies the Business Service by using the Business Service instance Id (supplied by the provider when the Business Service was instantiated) and the Task is specified by another parameter in the order. There will be other parameters supplied by the order, such as the required SLA values for the Task execution. If the preconditions of the Task are met, the provider application will execute the Task and return the 'output parameters' specified by the Task back to the client application. This will be the final status of the execution.

It is expected a client system to only execute one Business Service Order at a time. This is to make control and behaviour explicit without needing an additional layer of orchestration within the Business Service itself. Otherwise the Business Service turns into an application, rather than an interface specification. (Orchestration must be the responsibility of the client app, where it can maintain control of behaviour. If the client wants the provider to 'do lots of things', this is achieved by establishing a Business

Service that specifies lots of things, some of which may take place without the client having to control it).

Note that an explicit Business Service Order from the client is not necessary to execute a Task. The Task may have preconditions (such as a time-out) that can be met without the client needing to send an order.

However, we have discussed the allowed behaviour of orders for multiple Tasks in GB942. For example if there are multiple subjects specified in the Business Service (e.g. multiple sub network connections for an Activation Business Service) then an order to execute a Task may require a Task to be executed for each instance of subject and then success criteria and sequencing need to be defined.

The Business Service Order is analogous to an order for Service and indeed, a Service order could be seen as a specific, early example of the generic Business Service Order

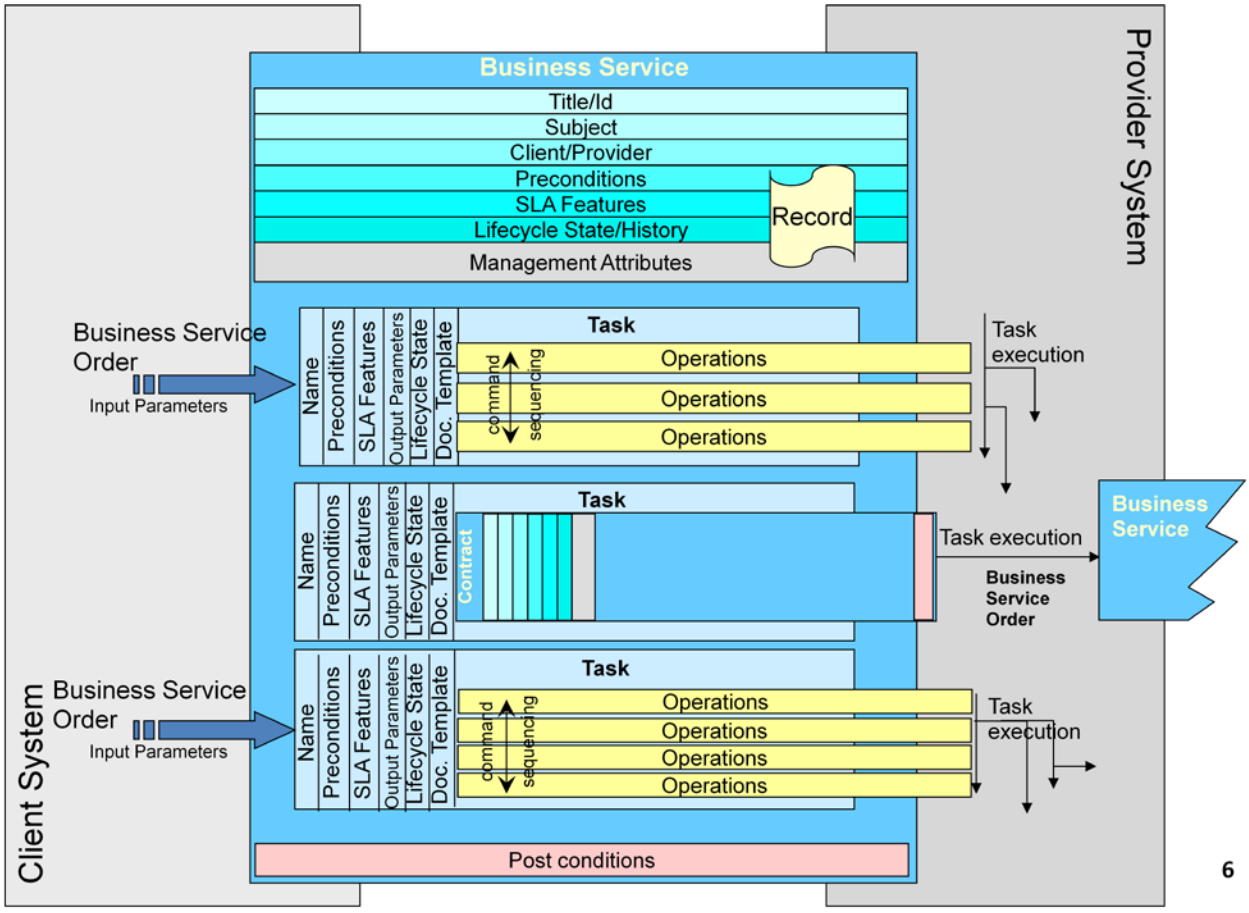


Figure 3.3-1 – Business Service Instance between Client and Provider

The diagram illustrates some features of Business Service usage. It is an overview and a detailed description of the Business Service Template is provided later in section 2.7.3.

1. The Tasks may or may not be triggered by an explicit Business Service Order
    a. Where there is an Business Service Order, there may be further preconditions
    b. Where there is no Business Service Order, the Business Service is triggered by some change in state in the provider.

2. When a Task is triggered, it will generate activity in the provider system, which may result in the provider system acting as a client to establish a further Business Service with a downstream system acting as a provider.

3. The Task may be a requirement for a sub-Business Service. When such a Task is triggered, the provider acts as the client system to establish a new Business Service with another system acting as the provider for the Business Service. In this situation, a return path is established from the second provider direct to the original client using the output parameters of the Task

3.4 Business Service Management – Main Phases

Library of Business Services

- Managed by the TM Forum to capture all Business activities
- Has Task fully specified
- Has Task implementations fully specified with TM Forum Interface Program standard interfaces
- Created by members to suit market needs

Design a solution

- Business Service specialized for the specific pair of systems that will exchange the Business Service
- Technology specific operations are written to implement the Task in the native language of the Provider (But if the systems are TM Interface Program compliant, the operations can be taken directly from library)
- Provider system is extended to run the Business Service if is not already compliant
- This may be through 3rd party mediation (or adapter) system, which will act as an external adapter to the provider system.

Run a solution

- Business Services are instantiated in both client and provider.
- Business Service instances are typically run on the provider system, which executes the Tasks using commands native to the provider. Note that the client system does not need to record the commands, so its representation of the Business Service instance remains independent of technology. This means that the client may have a different native technology to the user.

3.5 Business Service Orders and Task Choreography

In the following, a 'Contact instance' is used to describe a running Business Service instantiated between live client and provider systems. This will be an instance of a Business Service for a specific business purpose (which might be referred to as a 'type' of Business Service).

An initiating Business Service Order is used by the client system to request a new Business Service instance with the provider. Subsequent Business Service Orders are used to initiate and manage Task executions in the instantiated Business Service. This is the explicit way to progress a Business Service through its lifecycle. A Business Service Order is a data package that passes (over the wire) from client to provider, but it is best considered as a verb, or instruction, rather than a first-class entity in itself. A Business Service Order can be considered to be a generalized form of the traditional 'Service Order' and is open to similar kinds of discussion. In this description of Frameworx Business Services, the Order is taken to be a package of data that is used as an instruction from the client to the Business Service provider, targeted at a specific Business Service instance, where it is used to trigger a Task, or it is used to create a new Business Service instance. The two activities are distinct and the type of information that is conveyed by the Business Service Order is different for each activity. At the Business Service specification, two *pro forma* documents need to be specified, one to manage a Business Service instance and one to manage the Task executions in a Business Service instance It is possible to combine an order for a Business Service instantiation with Task execution instructions, but separate sets of information still need to be provided as the provider requires information and a pre-condition test for the Business Service instantiation as distinct from the Task execution

When the provider application receives a Business Service Order for the instantiation of a Business Service, the provider need not instantiate the order immediately. The provider must respond positively to the request, but may delay instantiating the Business Service until it is required for processing a Task execution order. The information for the Business Service record still needs to be stored by the provider and it may be best to do this by immediately instantiating a Business Service.

The client application can annul a Business Service if the pre-conditions are met. The Business Service itself may specify for how long the Business Service instance is held in provider application; otherwise the provider itself would be responsible for archiving the instance data.

Where there is more than one subject specified by a Business Service instance, for example a Business Service for multiple network connections, the Task runs for each subject and a separate record is kept of the input and output for each subject. The Business Service Order will specify the Task to be evoked and will supply data for the input parameters of the Task. The choreography options are as follows.

The Business Service Order is for one or more subjects of the Business Service. Where there is more than one subject, the Task runs for each subject and a separate record is kept of the input and output for each subject.

The Business Service Order will specify the Task to be evoked and will supply data for the input parameters of the Task.

The Business Service Order for multiple subjects falls into three categories

1. Single Subject. The set of entities represented are to be treated as a single subject and only one Task is triggered for the set of entities and only one set of input parameters are provided. For example the subject may be some part (e.g. the ATM layer), or the entire network model for a data synchronization Business Service.
2. Parallel Processing. The set of entities are each of the same class and each is treated as a separate subject. Tasks are triggered for each subject simultaneously if the preconditions to the Task are fully met for the given subject.
3. Sequential Processing. The set of entities are each of the same class and each is treated as a separate subject. Tasks are triggered for each subject sequentially in the order in which they are presented in the Order if the preconditions to the Task are fully met for the given subject. The Task for the next subject in line will only trigger if the previous subject has completed its Task. The previous Task may either succeed or fail, but whether the subsequent Task triggers will depend on the completion mode, described next.

Note that subjects are only specified in the Business Service Order that creates a Business Service instance. Subsequent Orders to an existing Business Service will apply to all subjects of the Business Service separately in the manner described above.

In addition, for Business Service Orders with multiple subjects (2&3 above), the Order can specify two completion modes of behavior for each Task execution instance.

1. Best Efforts, where all it is not necessary to successfully complete Tasks for all subjects. The Order will return a status to the client indicating the partial success if it occurs.
2. Atomic, where the Tasks for all subjects must complete successfully and if one or more Tasks fail, then all other Tasks are rolled back and the Order will return a failed status to the client. If any roll back fails, the Order returns a failed status to the client indicating a corrupted state has resulted in the provider.

The model is specified locally to each Task in the Business Service specification. This limited set of behavior for Business Services with multiple subjects is designed to allow sufficient flexibility for bulk transactions, without leading to the situation where complex sequencing and dependencies are implemented in the Business Service itself. Such functionality should be kept within the client system, where the processing logic and the full information context are available to enable complex processing.

An example of a Business Service Order could be: Business Service Order (Cancel, problemID, explanation) sent to the appropriate ‘Create and Resolve Customer Problem’ Business Service instance and triggering the Cancel task.

3.6 Provider System Behavior

Clearly, whilst the specification of the Business Service Order and Task Order imply Provider system behavior there are no constraints placed by this specification on how that behavior should be implemented. However, to further understand the choreography and the role of the provider system it is useful to construct stylized sketches of the provide system behavior.

The following sketches are not intended to be complete, accurate or represent the best approach but are intended to highlight some of the aspects of behavior that would be expected to be implemented.

The sketches consider the implementation of the elements of the solution handling Task Orders as part of a Business Service that has already been created. The Business Service Order processing would be essentially similar. The first sketch considers the element of functionality on the Provider system that deals with initial processing of a Task Order and evaluates the correctness of the Order with respect to the content of the Order and the current state of the Provider system.

Figure 3.6-1 Task specification skectch a

The second sketch considers the element of functionality that houses the task execution on the Provider system and deals with processing of internal Task execution events, with logging and with handling of events generated as a result of Task Order actions on a running task that occur as a result of the functionality

represented by the first sketch. The sketch was constructed with the assumption that the Execution function evaluates the correctness of the Task Order detail and partitions the Task Order content across separate operations/methods to the appropriate degree depending upon whether the Business Service has a single simple subject or multiple subjects with associated complex model fragments (e.g. where the Business Service is task-centric and associated with multiple related entities). Where there are multiple subjects and/or complex model fragments the execution will provide support for parallel processing or sequential processing dependent upon the specification of the Task.

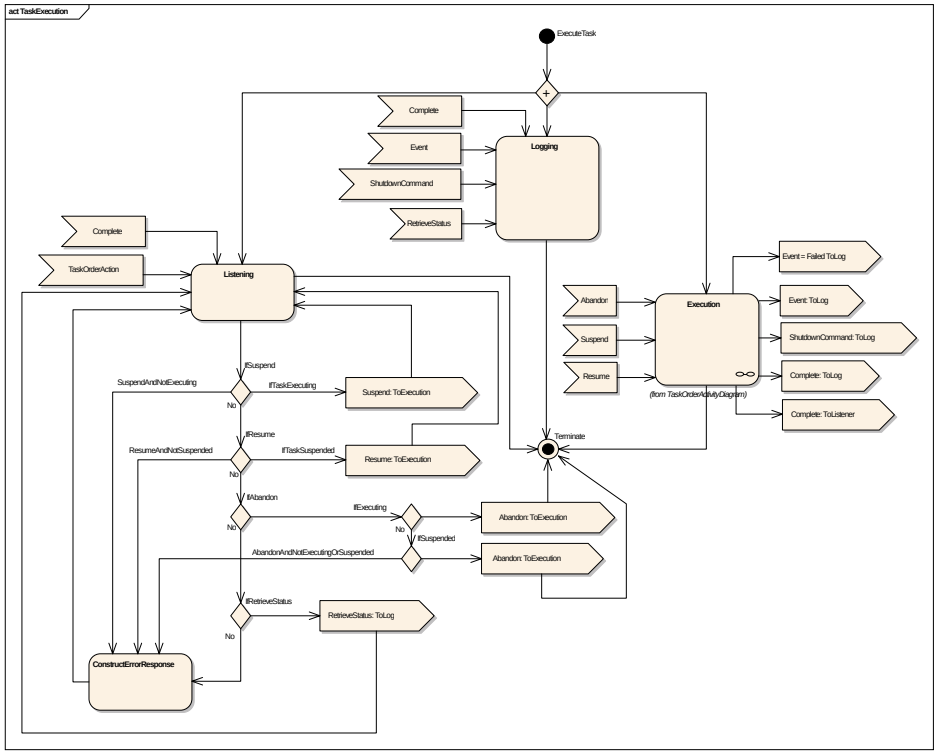


Figure 3.6-2 Task specification skectch b

3.7 Business Service Adapters

The roles of client and provider are asymmetrical and this is reflected in the interfaces. The client system needs to be able issue Business Service Orders and receive responses from the provider as a result. The provider system needs to execute the Business Service Tasks and supports the majority of the Business Service functionality, as illustrated below.

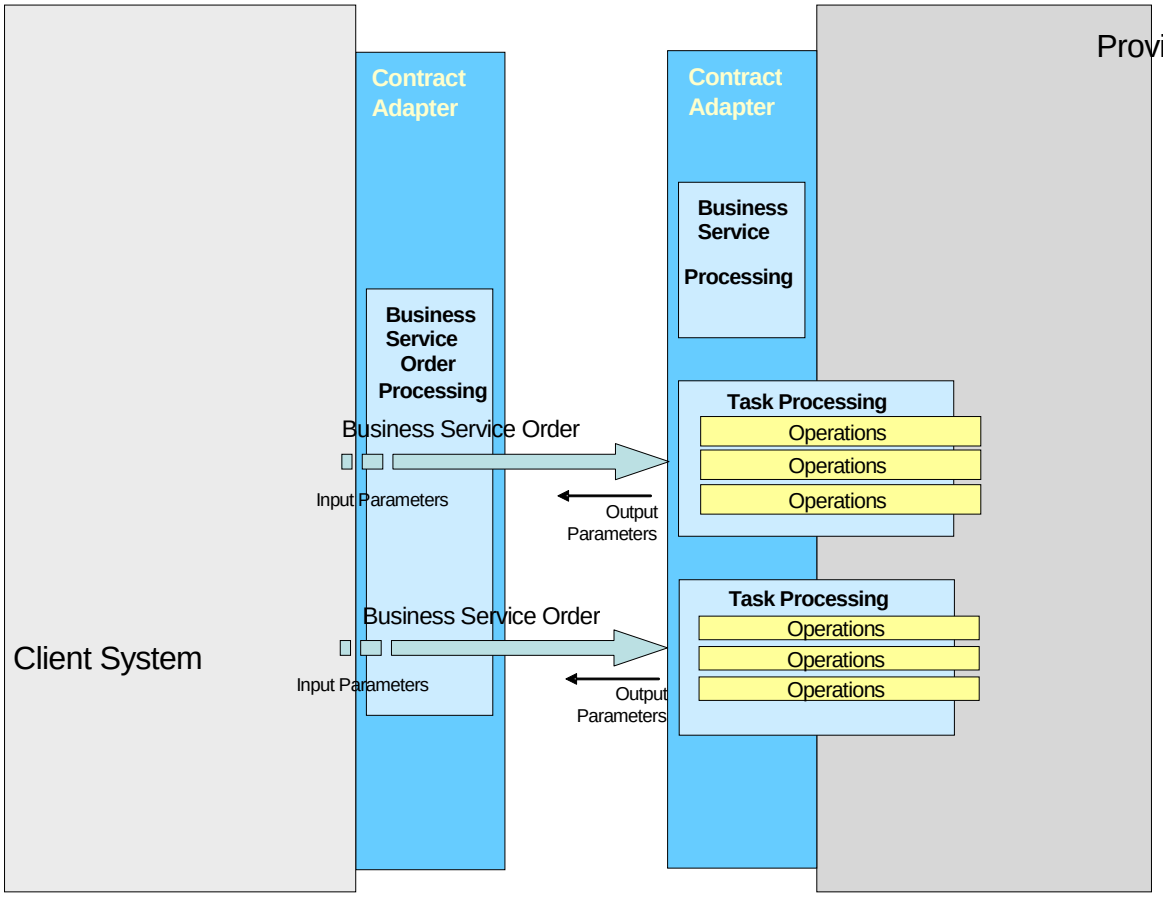


Figure 3.7-1 – Business Service Implemented with Adapters at both Client and Provider

It is also possible to have a separate Business Services adapter between systems that are not Business Services compliant, as illustrated below. Here an intermediate system adapts between the two systems that can have different APIs and transport technologies. The value of this solution is that the adapter is standards based and hence reusable.

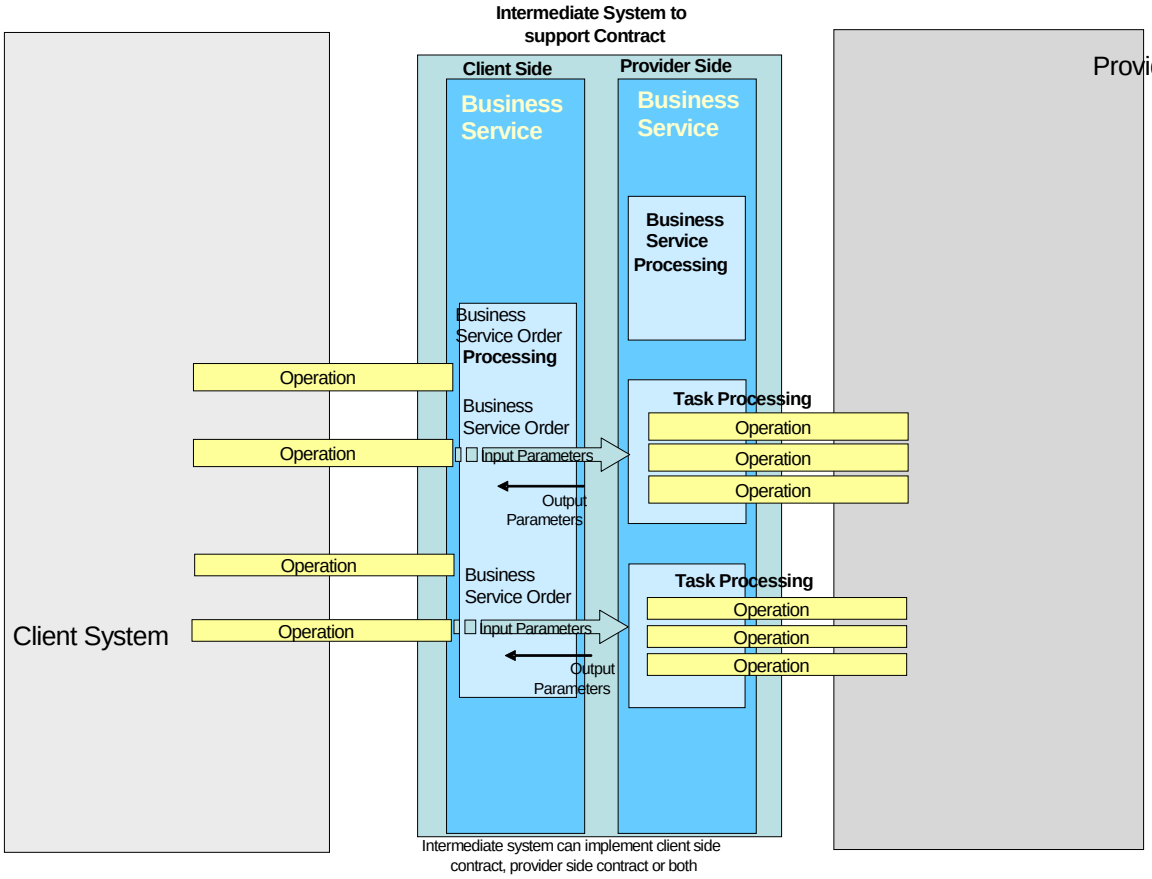


Figure 3.7-2 – Business Service Implemented with a separate Adapter

3.8 Evolving the Frameworx Business Service Template

This section builds up the Business Service Template from existing models.

3.8.1 A Starting Point

This section describes a view of the structure, relationships and intended content for a Frameworx Business Service. The vehicle for this is a UML view that represents a meta-model for specific Frameworx Contacts that may be used by TM Forum or others to define Business Services that meet specific business needs through Use Case analysis, etc.

Thus, it is expected that an individual Frameworx Business Service addressing a specific business purpose - a Frameworx Business Service focused on interaction between two companies about fault handling around a resource supplied by one company for use within the other - becomes an instantiation of this Business Service Template into a specific UML model for that Business Service. To avoid any confusion, in an application, such a specific Business Service – say, here, a "Resource Fault Handling" Business Service - is then itself instantiated as an implemented Business Service that would be used between two specific companies to manage fault handling around the specific resource(s) they share. This "Resource Fault Handling" Business Service might be further refined many times by different companies to meet their needs, and of course the Frameworx Business Service Template will give rise to the definition of many specific Business Services, for this Resource Fault Handling case as well as many other business needs.

Here, then, we are focused on the overall meta-model that guides definition of many Business Service types, which then can produce instances of these in implementation, Note that the practicalities of scoping, defining and maintaining specific Business Services are addressed elsewhere in this document.

The Business Service Template is used to support all four Frameworx lifecycle views and in order to achieve this; the template specifies the additional content required for each new Frameworx lifecycle phase. In this way, all four lifecycle views are represented by the same Business Service Template, with more of the content of the template being relevant as the lifecycle progresses from Business View, through System, Implementation to Deployment View.

To begin with, the material developed in TR138 and in other works is that we use the meta-model to identify and position the elements that influence the Business Service behavior and operation, and to highlight how these interrelate. Note that when using this to help the design of specific Business Services, significant creativity and design input is required to make appropriate choices on the specific components that will be relevant. To see this, consider Figure 3.8-1 - An initial System View of the Frameworx Business Service (from TR138) below, which is drawn from TR138 and provides a high-level view of how aspects such as a Process view (here, 'SystemProcess') apply. When designing a specific Business Service using this view, it will be necessary to decide which actual process elements and process flows are relevant to the case (i.e. the Use Case) in hand. The meta-model gives no direct guidance on this selection mechanism, but simply identifies that there is a process influence and indicates how this can be linked in. The implication is that the designer must make intelligent choices that include and exclude some possible process components, to show how particular areas of process relate to a specific Business Service design. This point is stressed since there may be confusion on what is "automated" in the design activity because of the use of such models, and it is important to recognize the importance of the design choices, and the designer, in this.

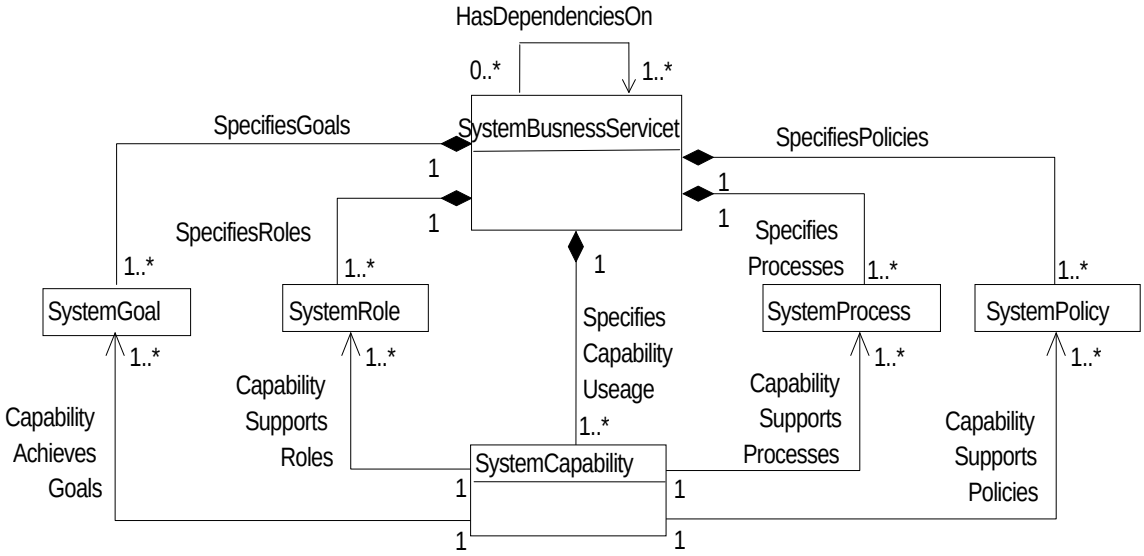


Figure 3.8-1 - An initial System View of the Frameworx Business Service (from TR138)

3.8.2 Overall Frameworx Business Service View

If we generate a full lifecycle view that is a broadening of the System View in Figure 3.8-1, a first conclusion is that no conflicts are envisaged in using this view to address all four Frameworx Views (Business, System Implementation, and Deployment). Figure 3.8-2 below, which as a first step simply generalizes the System View model, can therefore be taken to represent a full Frameworx Lifecycle view of the Frameworx Business Service. Note however that this is a high-level view and further refinement is described later.

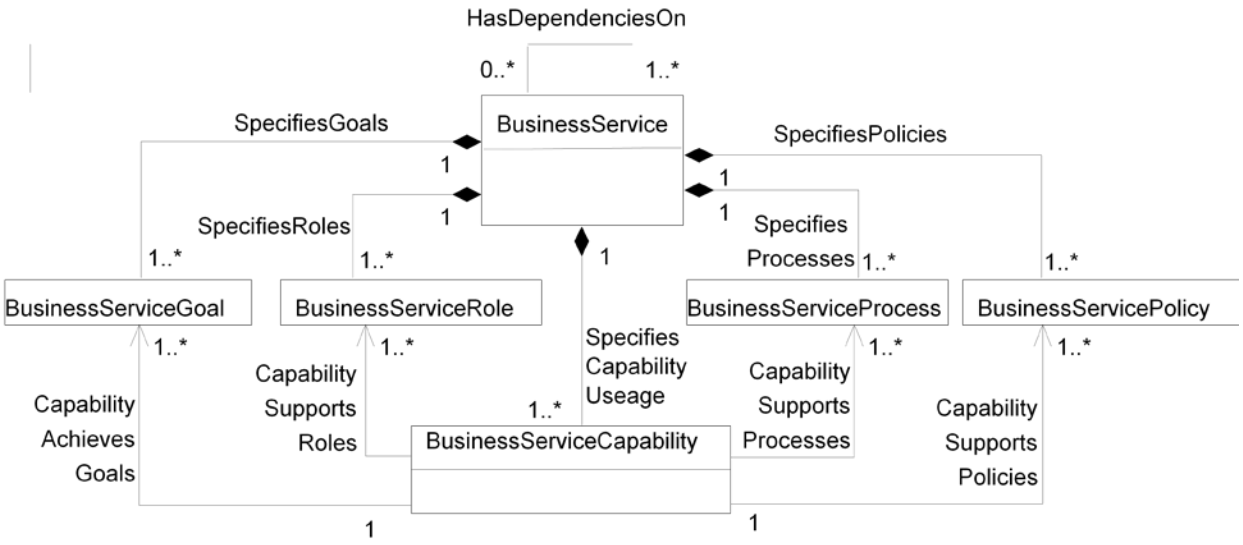


Figure 3.8-2 - An Overall Lifecycle View of the Frameworx Business Service

In understanding this, we can interpret elements of this structure (and others that emerge in further detail below) as follows:

Goal - statement of the intended or desired solution outcome

Role - statement of the part played by each of the key participants involved in utilizing the solution

Process - description of the flow of behavior of the solution

Policy - description of the governance and/or decision-making behavior of the solution

Entity - description of the information/data required to support the solution

Capability - points of interaction with the offered solution functionality

Trading Lifecycle – description of the ongoing relationship between the trading entities (as formalized in the Business Service, from initiation through to termination)

As understanding and specification of an individual Frameworx Business Service proceeds around the Frameworx – i.e. design – lifecycle an important focus is on the last of these items, the Capability. This is meant to represent the raw functionality of the Business Service, and so is a design focus for how the other overall influences identified in the Figure funnel into the actual Business Service behavior.

Capability (here, Business ServiceCapability) will be used within a specified context and will operate within defined constraints to achieve a stated goal – so, a definition of an overall scope for a system could be a capability operating in a context with defined constraints to achieve a goal. These additional elements can be interpreted as:

Context - the environmental aspects and expectations of the solution

Constraint - the restrictions and/or limits imposed on the operating environment of the solution

Note that both Context and Constraint can be expressed through the other elements expressed previously – for example they can be related to the Goal and hence introduced via the definition of this as a specific Business Service as defined. Or, they might be enumerated within pre and/or post conditions (see later) that apply. There might also be an aspect injected through Policy, as business context and constraints. Moreover, Context and Constraint may evolve or change as one works around the Frameworx Lifecycle, with an implementation constraints that are only introduced with the Implementation View. Because of this, it is suggested that Context and Constraint are not separately identified within the UML, but instead are considered as aspects of the other elements, such as Goal, that were identified earlier.

3.8.3 Frameworx Business Service Structure

The Frameworx Business Service, then, has a number of influences and linkages as indicated previously. These must be analyzed and defined for a specific Business Service to show, for example, what specific process elements are relevant when considering the Business Service scope and behavior, and what process interactions apply when looking at the interaction boundary (i.e. between two businesses, stems, products, etc) at which the Business Service applies.

However, it is also important to look within the Business Service and understand how it is structured to support the required external interactions and behavior. To assist

with this, the Business Service, via the Capability, is considered to support one or more Tasks, which represent individual activities (typically, visible as individual methods or operations accessible when using the Business Service) that may be combined or orchestrated to achieve some overall activity.

Figure 3.8-3 also generalizes the System View model seen in TR138, and thus is a further step towards representing a full Frameworx Lifecycle view of these aspects of the Frameworx Business Service, with further refinements described later. Figure 3.8-3 illustrates how the Business Service, through the Business Service Capability, is structured as Tasks (here, Business ServiceTasks). This is an implementation neutral representation and we do not imply that **Business Services** utilize any specific implementation at this stage e.g. J2EE. The Frameworx Business Service refers to entities from Information Framework. A generic Business Service would apply to any discipline where there is a process model, an information model and an applications map, i.e. it need not be Business Process Framework, Application Framework & Information Framework).

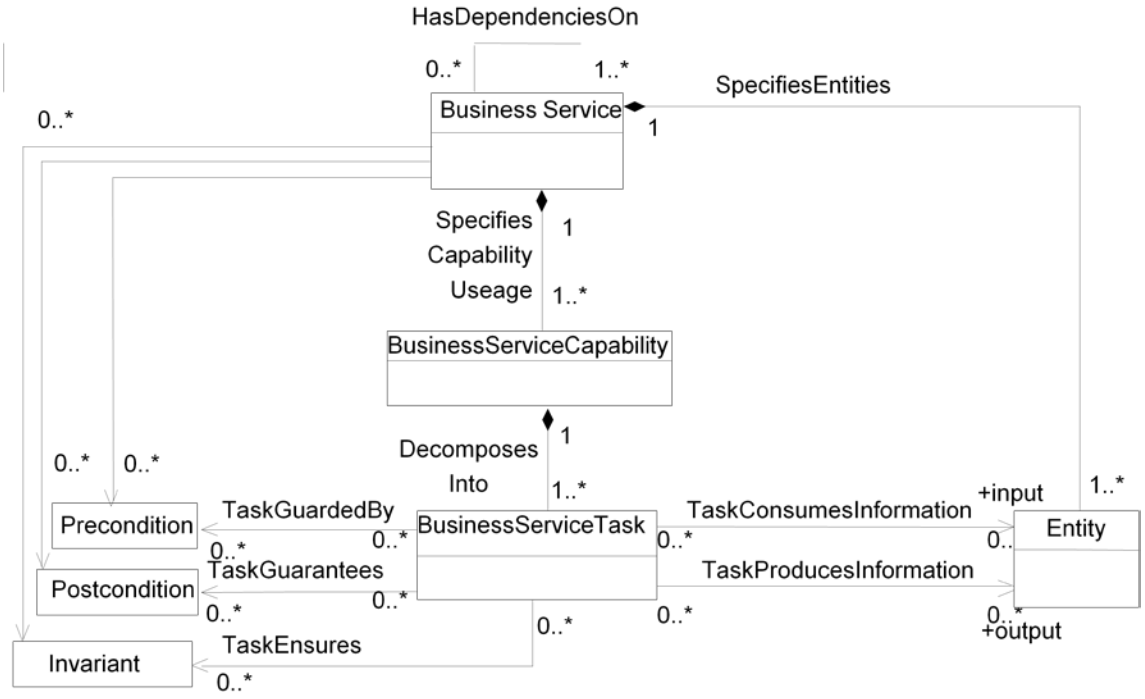


Figure 3.8-3 - The Frameworx Business Service, Capability and Task

As we consider how individual Tasks are used in an actual Business Service development, additional aspects are required, as shown in Figure 3.8-4. Here elements from the Harmony Catalyst and the MTOSI Service Description Meta Model have been incorporated into the broader view.

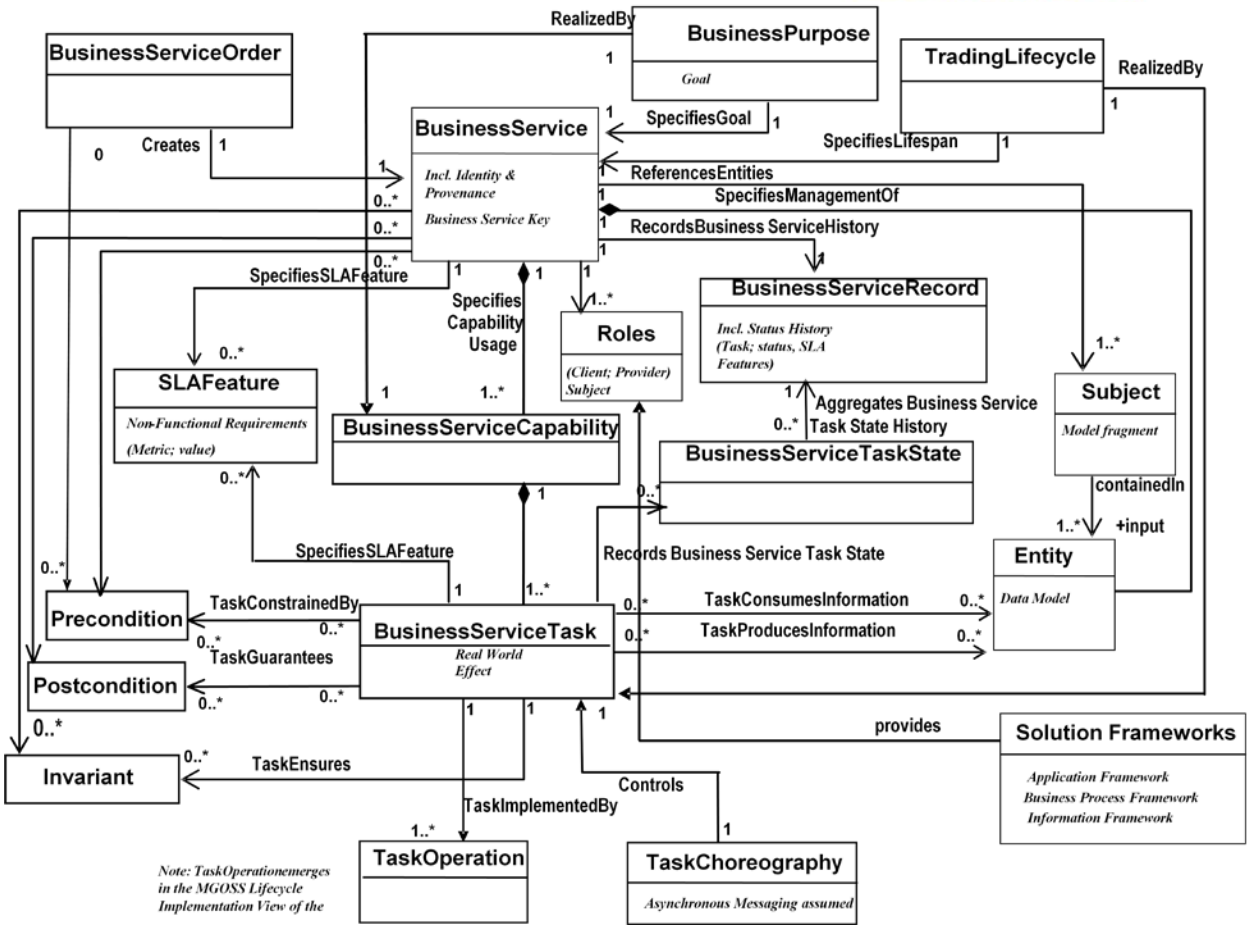


Figure 3.8-4 - Expanded View of Frameworx Business Service and Task

The Business Service scope can be identified through the needs of the overall trading relationship and its lifecycle (here, Trading Lifecycle) that give rise to the opportunity for using a Business Service to support this between two enterprises, etc.

Since the view here is that the capability of a Business Service can be visualized as a set of interlinked Tasks, in operation it is important to be able to maintain a Business ServiceRecord that captures state and other information about the Business Service. The Business ServiceRecord depends on the individual states of the individual Tasks, and so links with Business ServiceTaskState. The Record not only tracks the execution lifecycle of a running Business Service instance, but also tracks the SLA metrics of the execution. This provides valuable business intelligence as to the performance of the software solution and the business. Note that the Record does not replicate the data that may be inventoried as part of the normal functioning of either the client or provider system.

The Business ServiceRecord is structured into Sections. When a Business Service instance is created between two running systems, a first Section is opened in the Record to record the precondition values of the Business Service instance and other information, such as time of creation and instance identifiers for the applications.

When Tasks are fired, subsequent Sections are opened for each Task execution, recording the Task precondition values and other information such as time of firing. When the Task completes the same section is updated with Output values, SLA metric actuals (e.g. time taken to execute), completions status (succeeded, failed, etc.) and reason for failure. When the Business Service is ceased, a final Section is added with the Business Service post conditions. In this way the Record records the execution history of the Business Service.

It may be useful to have a separate Section in the Business ServiceRecord that simply lists the sequence of Tasks executed and the completion status of the Task, in order to provide a snapshot of the Task lifecycle history. For example a 'Create and Maintain Service" Business Service may have a section

- Inquire Service Task – Succeeded
- Reserve Service Task – Succeeded
- Configure Service Task – Failed (network data mismatch)

3.8.4 Business Service Dynamics

It is intrinsic to the approach here that, depending on the circumstances, a solution designer may require a given Business Service to employ a chosen set of Tasks in a particular sequence or orchestration. This orchestration is not explicitly represented in the Business Service, as this would turn the Business Service from an interface specification into a workflow engine. The explicit business workflow is rightly the function of the calling client system, where it is open and explicit. However, given the explicit orchestration is in the client, it is necessary to provide a mechanism for the client to task out activity in the Business Service, and the Business ServiceOrder provides this mechanism by selecting Tasks and defining their initiation/sequencing/completion conditions. Note that an initial Business ServiceOrder may be created without linkage to identified Business Services at that point. Later, Business ServiceOrder can be associated through the Business ServiceKey with one or more Business Services.

Note that there is implicit orchestration within the Business Service. This results from the Tasks being fired by changes in data, status or timing and not necessarily by an explicit Business Service Order. This mechanism is a further aspect of the loose coupling between client and provider system, enabling the provider to proceed through the execution lifecycle of a running Business Service instance without direct control from the client.

Regardless of whether the Task execution is explicit through a Business Service Order or implicit, the Record tracks the Task, its status and the SLA metrics associated with its execution.

Another necessary aspect is to represent directly the service commitments, and this is modeled in the form of SLAs (here, SLAFeature) for the Business Service and/or individual Tasks, that apply in use.

The following diagram shows the sequencing of events involved in Business Service-based interfacing.

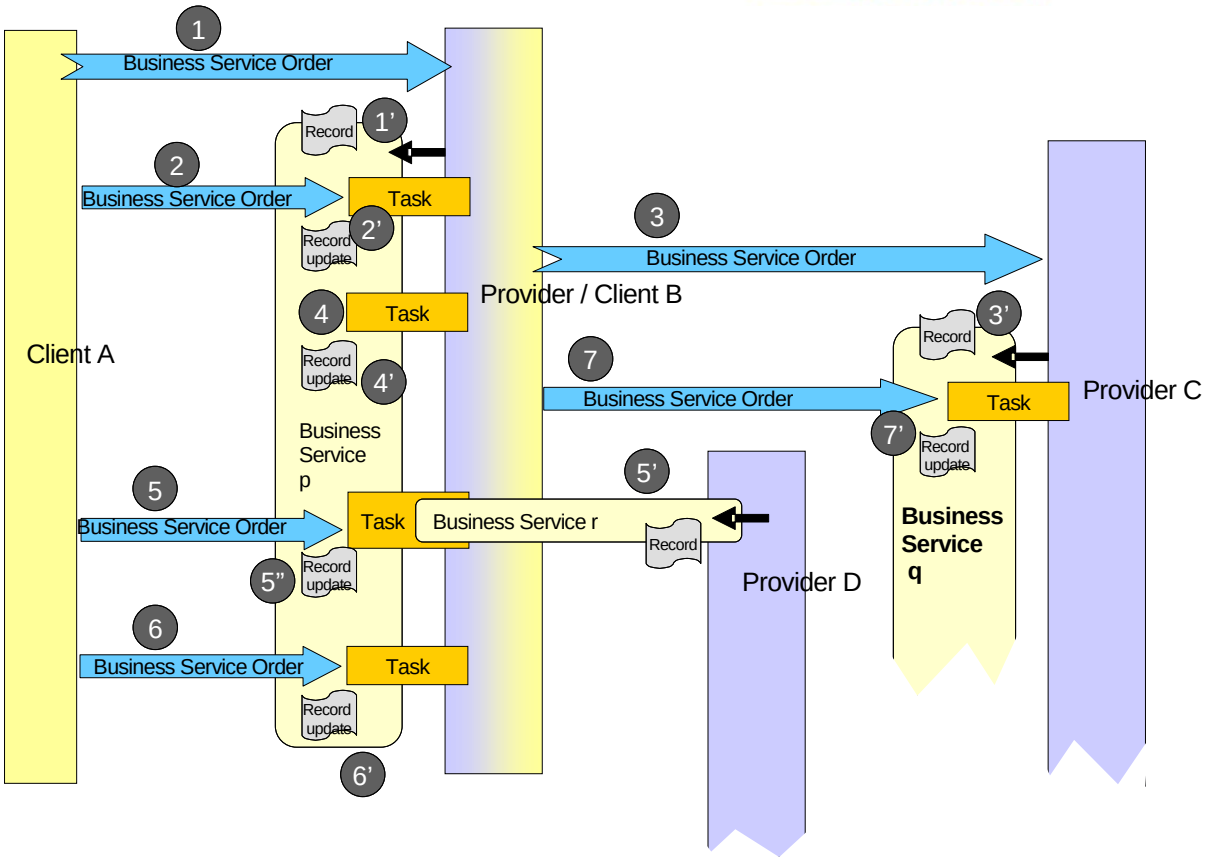


Figure 3.8-5 - Expanded View of Frameworx Business Service and Task

Step 1 Initial Business ServiceOrder from client to provider system to establish a Business Service (p) for a specific business purpose, for one or more specified subjects

Step 1' if the Business Service preconditions are met, the Provider system (B) instantiates a new Business Service and initializes a Business ServiceRecord with precondition data and administration information. The client system (A) receives a status update indicating the status of the new Business Service.

Step 2 Client sends Business ServiceOrder to initiate a Business ServiceTask within the specified Business Service.

Step 2' if the Business ServiceTask preconditions are met (true), then the Provider system (B) executes the Business ServiceTask. On completion the end status and SLA metric actuals are added to the Business ServiceRecord. If the preconditions are not met, the Business ServiceTask fails to fire. In either case a status update is sent to the Client (A). The Client may receive status updates during the execution of the Task. The frequency and detail of the updates are set by an SLA feature of the Business ServiceTask or Business Service.

Step 3 Subsequent activities in the provider system (B) initiated by the Business ServiceTask in Step 2 result in the system (B) acting as a client and initiating a new Business Service with a down-stream provider system (C) by sending to it an initiating Business ServiceOrder

Step 3' if the Business Service preconditions are met, the down-stream provider system (C) instantiates a new Business Service and initializes Business ServiceRecord with precondition data and administration information. The system now acting as the client in this context (B), receives a status update indicating the status of the new Business Service.

Step 4 Task fires in the first Business Service (p) as a result of the preconditions of the Business ServiceTask becoming true. This happens without a Business ServiceOrder from the client as the preconditions do not require such an order.

Step 4' The Provider system executes the Business ServiceTask. On completion, the end status and SLA metric actuals are added to the Business ServiceRecord. A status update is sent to the Client (A). The Client may receive status updates during the execution of the Task. The frequency and detail of the updates are set by an SLA feature.

Step 5 the client system (A) sends a Business ServiceOrder to initiate a Business ServiceTask within the specified Business Service. This Business ServiceTask is a subBusiness Service that will be established by the provider system (B) acting as a client for a down-stream provider system. D).

Step 5' if the Business Service preconditions are met, the down-stream provider system instantiates a new Business Service (r) and initializes a Business ServiceRecord with precondition data and administration information. The system now acting as the client in this context, (B) receives a status update indicating the status of the new Business Service.

Step 5" Because the new Business Service (r) was initiated by a subBusiness Service task in the original Business Service (p), the original client (A) is also updated with status information that results from the execution of that Business ServiceTask and hence is updated with status information from the new Business Service (r) with the downstream provider system (D), which would otherwise just go to the system (B), acting as the client to the new Business Service. All subsequent status data that results from the lifecycle of the new Business Service (r), which would otherwise just go to system (B), will now also be passed onto the original client system (A). This situation is therefore different from that in Step 3, because the subBusiness Service mechanism establishes an explicit feedback path to the original client (A), from the new, downstream subBusiness Service (r).

Step 6 the client system sends a Business ServiceOrder to a Business ServiceTask in the original Business Service (p). This Business ServiceTask is to close and archive the Business Service. This Business ServiceTask has preconditions that align to the post conditions of the Business Service (p).

Step 6' if the post conditions of the Business Service are met, then the preconditions of the Business ServiceTask are also met when the Business ServiceOrder is received. The order is an additional precondition for the task. The provider system will close the Business Service and archive the data. SubBusiness Services (r) will also need to be cancelled and this will require that their post conditions are met. The original Business Service (p) cannot be cancelled unless all downstream subBusiness Services are also cancelled.

Step 7 Business Service behaviors for other Business Services (q) that are not subBusiness Services continue as normal regardless of the state of the original Business Service (p). Once created, they are independent.

Task Choreography is required where a Business Service has multiple subjects. This will be typically where there are a number of the same type of subjects, for example multiple sub network connections, managed by a single Business Service instance. (reference section 2.5)

3.9 Adding Implementation View Artifacts in the Template

The previous diagram used Operations as a hand-off point to the implementation. The Operations of a real world application are used to implement the Tasks specified in the System View of the Business Service.

Note that in the Implementation View, explicit orchestration of the Operations may be required in order to implement the Task. This is not a business level workflow orchestration, but instead it is an IT level implementation issue.

The figure below is the UML for the additional artifacts introduced by the Implementation phase of the Business Service's lifecycle.

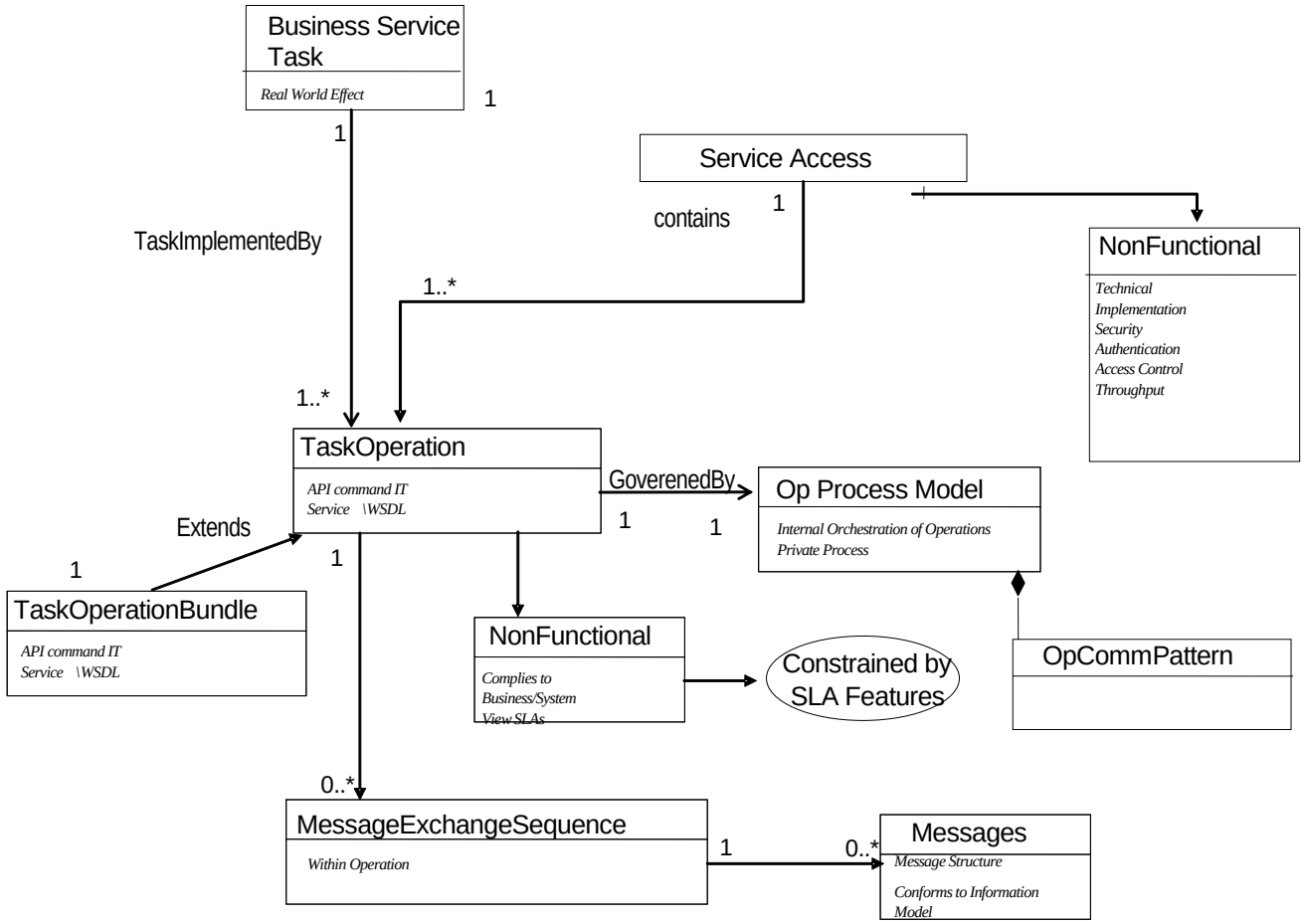


Figure 3.9-1 – Additional Implementation Artifacts to the Template

Note that the IT level aspects of the Business Service, such as the Non-functional components of the Service Access are introduced at this point. The discussion is continued in more detail in the Implementation View chapter. The Implementation View of **Business Services** will be developed by the TM Forum Interface Program.

3.10 Business Service Specification

The Business Service concepts are captured in a Domain Specific Language (DSL). The DSL is expressed in its meta-model commonly or its abstract syntax[4]. A meta-model is just a model that provides the basis for constructing another model. Although both are models, one is expressed in terms of the other; in other words, one model is an instance of or conforms to the other. The DSL for Business Services is specified as a normative model using the eclipse ECORE as its Meta-meta model. The detail specification is not detailed in this document but currently under specification. The following section outlines the core concepts and structure of the DSL. In this context we define the following:

- Business Services DSL (Business Service specification language): An ECORE based DSL that captures all the concepts and syntaxes needed by users to specific Business Services Instances.
- Business Service (Business Service): real world specification of Business services that are expressed in accordance to the underlying grammar and abstract syntax of the Business Service DSL.
- Business Service Instance (Business Service Instance): A concrete instantiation of a Business Service deployed in an OSS/BSS environment.
- Business Service Tooling (Business Service tool): a reference implementation of the DSL using the eclipse GMF graphic modeling framework. This tool allows end users specify instances of the Business Services. This is being developed as in TM Forum collaborative environment and can be accessed from there.

3.10.1 Relation to OMG

In order to better address the requirement specific to each individual Frameworx modeling activity and simplify the modeling process for the end-user, the tooling framework is aimed to be oriented towards the use of Domain Specific Languages, where each modeling domain (Information and Data modeling, **Business Services**

[4] The term abstract syntax refers to a meta-model, so it often has a corresponding concrete syntax in the form of text or diagram notation. These are referred to as textual concrete syntax and graphical concrete syntax, respectively. A textual syntax enables users to work with instance models just as they would other text-based programming languages.

modeling, Business Process modeling, etc.) uses the metamodel specific to its own application domain Figure 3.10-1.

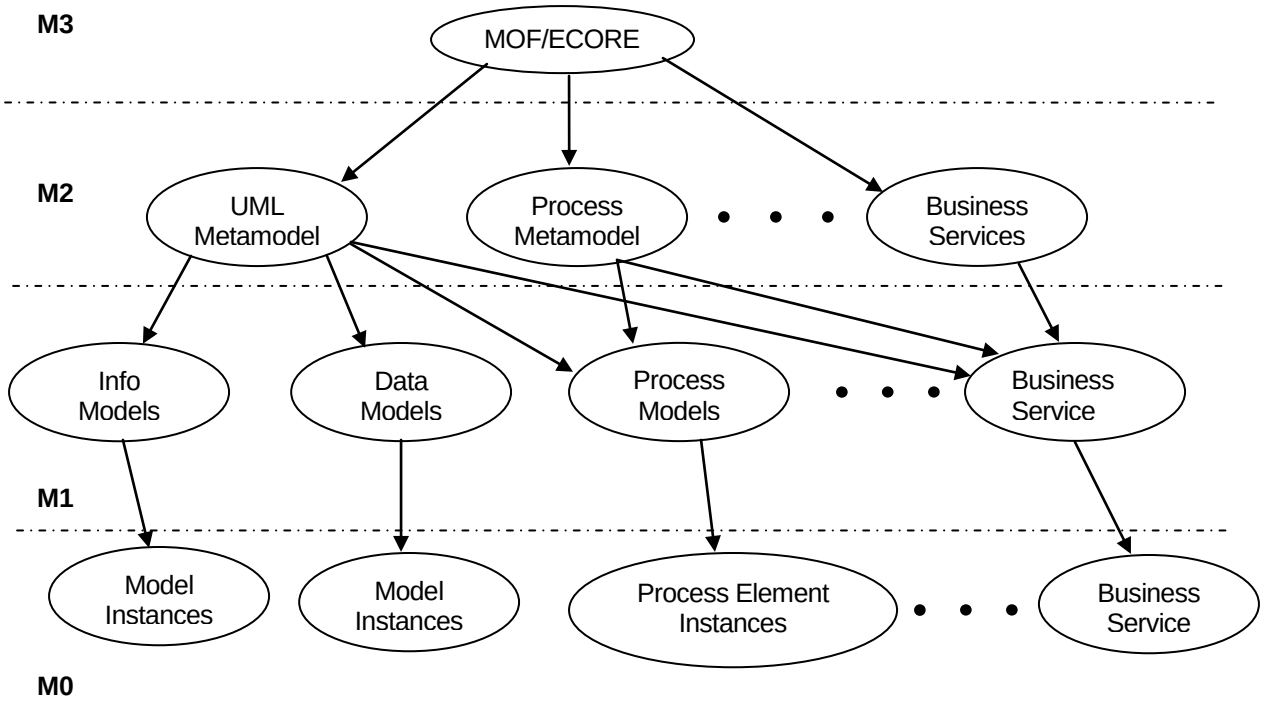


Figure 3.10-1 Frameworx Metamodel

In accordance with this approach standard UML 2.x metamodel will be used for information and data modeling. Domain specific metamodels (at M2 level) need to be developed (or, possibly, reused from other industry domains) for the modeling of Business Processes, Application Components and Frameworx Business Services.

Use of different metamodels raises an issue of compatibility between artifacts produced and consumed by different modeling activities. For example, it is necessary to use Information and Data models specifications developed using standard UML 2.x metamodel while working on **Business Services** defined using Domain Specific metamodel.

However, in spite of using different metamodels at M2 level the compatibility becomes non-issue as long as these metamodels are the instances of common meta-metamodel at M3 level (MOF and/or ECORE).

Advanced Note for Tool-Builders:

This technical section is aimed at readers who wish to understand the mechanism for building Frameworx Business Service tooling.

The terms M0 to M3 are OMG defined terms to label the various levels of models but they are also applicable to the Eclipse Domain Specific Languages (DSL's) (refer to figure 3.2)

- Meta-Meta-Model = M3

- Meta-Model = M2
- Model = M1
- Generated Code = M0

The ECORE 'language' is the Eclipse Reference implementation of a subset of part of the OMG MOF, (called Essential MOF). MOF is OMG's Meta-Meta-Model (M3), the language which defines UML, OCL, etc. Hence Ecore is also a meta-meta-model (M3).

The purpose of the M3 is to define the semantic rules for creating meta-models (M2).

The UML kernel is an example of a meta-model, this provides the semantic rules for building models in UML (e.g. Information Framework and TMF608 are models (M1)). It is the kernel that allows the user to do certain UML actions but prohibits others.

Both MOF and ECORE can describe themselves entirely (they are said to be self reflective) and hence there is no need to have a M4 (meta-meta-meta-model) to describe them.

It is possible to derive many M1 models from a M2. When a model (M1) is instantiated in a runtime it becomes a M0 instance, similarly there may be many M0 instances of each M1.

3.10.2 Business Service tooling reference implementation

Business Service Specification is supported by a reference implementation using the Eclipse open source tooling platform. Eclipse Modeling Framework (EMF) is a sub project within Eclipse that provides a Java framework and code generation facility for building tools and other applications based on a structured model. While EMF uses XMI (XML Metadata Interchange) as its canonical form of a model definition, you have several ways of getting your model into that form:

- Create the XMI document directly, using an XML or text editor
- Export the XMI document from a modeling tool such as Rational Rose
- Annotate Java interfaces with model properties
- Use XML Schema to describe the form of a serialization of the model

It is important to also identify how EMF[5] Relation to OMG MOF?

The OMG (Object Management Group) specify a MOF (Meta Object Facility) which allows for the formal specification of modeling languages such as UML, you may be wondering how EMF relates to it. Actually, EMF started out as an implementation of the MOF specification but evolved from there based on the experience we gained from implementing a large set of tools using it. EMF can be thought of as a highly

[5] EMF: Eclipse Modeling Framework, 2nd Edition By Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks Published by Addison-Wesley Professional.

efficient Java implementation of a core subset of the MOF API. However, to avoid any confusion, the MOF-like core meta model in EMF is called Ecore. EMF can transparently read and write serializations of EMOF.

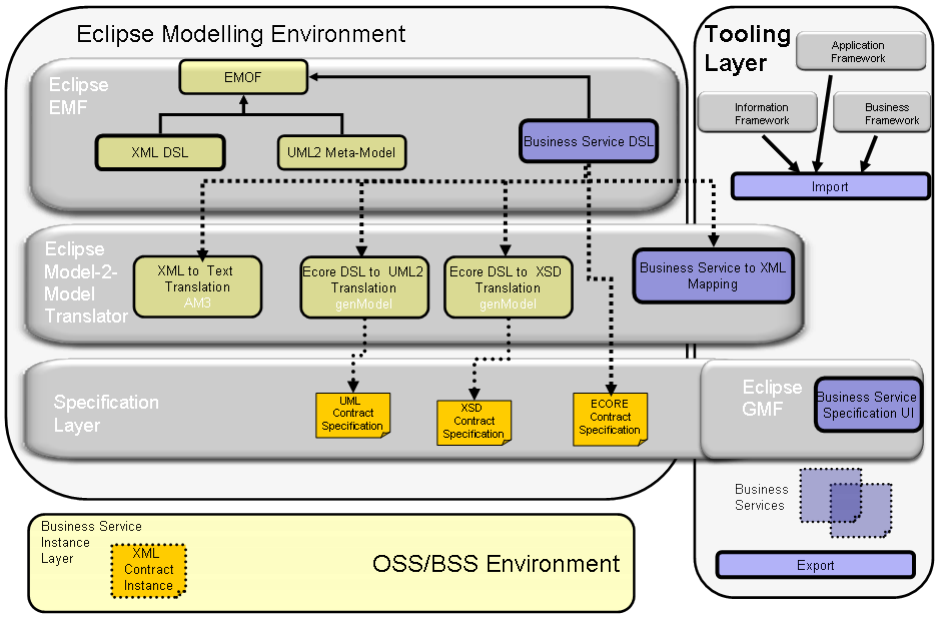


Figure 3.10-2 Business Service reference architecture

The above diagram is decomposed into three areas, 1) the eclipse modeling environment – which is in turn sub-divided into three parts-, 2) the tooling layer and the OSS/BSS environment. Note that the components in Blue are being developed and specified as part of the Architecture Harmonization team. These five components are 1) Business Service DSL or meta model, this is the normative part of the business service or Business Service specification 2) import component which is use to import other part of the TM Forum frameworks to be included in the business 3) the Business service to XML Mapping 4) Business Service Specification UI which is an user interfaces for the specification of Business Services, and 5) export component which is used to export Business Service. Green components are items use from the EMF where as orange components are that are outputs from the various process.

3.10.3 Business Service DSL overview

The Business service DSL is the formal specification for Business Service and on which the reference tooling is constructed. This component is sub-divided into 6 parts or modules, this allows for isolation of various aspects of Business Services. The six packages are defined as follows:

1. ***Business Service packages***: contains all the concepts and relationship for the core of the DSL for business service specification. {BusinessService, BusinessServiceHistory, BusinessServiceCondition,,BusinessServiceOrder, BusinessServiceRecord,TaskExecution,TaskExecutionHistory,BusinessPuopose,

TaskOrder,TaskOutput,Postcondition,Task,Precondition,LegacyTask,ManualTask,AbstractTask}

2. ***Information Model:*** package: this provides the binding to other Frameworx information model such as Information Framework and MTMN.
3. ***Frameworx package:*** core concepts relating to core concepts for the Frameworx.
4. ***Policy package***:, a DSL presentation of policy aspect associated with Business Services/
5. ***SLA package:***, this package contains the concepts and principles for the specification of service level agreement contained within the **Business Services** aka Contracts
6. ***Application Framework package***, this package model the core concepts need to express **Business Services** between Application Framework applications.

4 Administrative Appendix

4.1 Glossary

Item	Description
Eclipse	An Open-Source tool environment/platform
Ecore	Eclipse based tool language for creating meta-models, MOF reference implementation.
GMF	Graphical Modeling Framework – Eclipse tool for generating visual editors from Ecore
MOF	OMG's definition of a meta-meta model.
MTNM	Multi-Technology Network Management
MTOSI	Multi-Technology Operations System Interface
Frameworx	New Generation Operations Systems and Software
SNC	Sub Network Connection
TigerStripe	Open-Source Eclipse tool used by TIP
TIP	TM Forum Interface Program
UML	Unified Modeling Language

4.2 About this document

This is a TM Forum Guidebook. The guidebook format is used when:

The document lays out a 'core' part of TM Forum's approach to automating business processes. Such guidebooks would include the Telecom Operations Map and the Technology Integration Map, but not the detailed specifications that are developed in support of the approach.

Information about TM Forum policy, or goals or programs is provided, such as the Strategic Plan or Operating Plan.

Information about the marketplace is provided, as in the report on the size of the OSS market.

4.3 Document History

4.3.1 Version History

<This section records the changes between this and the previous document version as it is edited by the team concerned. Note: this is an incremental number which does not have to match the release number>

Version Number	Date Modified	Modified by:	Description of changes
0.1	11 Sept 2007	John Reilly	First version that contains Business Service granularity guidelines
0.5	10 Dec 2007	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis	Updates based on team reviews/discussions…intermediate versions included similar changes.
0.7	24 Jan 2008	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis, Alex Zhdankin	Updates based on team reviews/discussions…intermediate versions included similar changes.
0.9	3 Mar 2008	John Reilly, Mike Kelly, Richard Mishra, Nigel Davis, Alex Zhdankin,Steve Orobec	Further Updates based on team reviews/discussions…intermediate versions included similar changes
0.9.1	16th June 2008	Ian Best	Minor formatting changes
1.0	14th July – Aug 18th 2008	Steve Orobec	Final version review comment edits
1.1	20 Aug 2008	Richard Mishra, Steve Orobec	Edits based on Approval Committee and Technical Program Council review comments
1.2	9 Sept 2008	Tina O'Sullivan	Final corrections prior to web posting
2.0	30 Oct 2008	John Reilly	Split into Concepts and Principles and User Guidelines.
2.01	November 2008	John Reilly	Aligned with other document sets and Integration Framework concepts
2.02	December 2008	John Reilly	Updated based on reviews
2.03	18 August 2009	David Cleary	Updates based on review
2.04	28 October 2009	Alicja Kawecki	Minor cosmetic corrections for web posting and ME
2.05	27Jan/2010	Ken Dilbeck	Solution Framework to Frameworx per Marketing
2.06	8 March 2010	Pascale Pecha	Minor edits
2.07	10 March 2010	Tina O'Sullivan	Corrected Release number
2.08	08 June 2010	Alicja Kawecki	Updated to reflect TM Forum Approved status
3.0	19 May 2011	Alicja Kawecki	Upversioned to align with R2.5 for Framework 11.0 – content remains unchanged

4.3.2 Release History

Release Number	Date Modified	Modified by:	Description of changes
1.0	September 2008	Initial release	First release to member for comments
2.0			Split GB942 into separate addenda
2.1	January 2010	Pascale Pecha	Updating for pre-launch of Frameworx.

4.4 Company Contact Details

Company	Team Member Representative
Amdocs	Richard Mishra richard.mishra@amdocs.com +44 1225 327236
BT	Steve Orobec Steve.orobec@bt.com +44 7710 046883
Ericsson	David Cleary David.Cleary@ericsson.com +353906431579
Harris	Alex Zhdankin Alex.zhdankin@hstx.com +1-321-674-4755
Nortel	Nigel Davis nigeld@nortel.com +44-1279-405978
TM Forum	John Reilly jreilly@tmforum.org +1 469 363 4546
TM Forum	Mike Kelly mkelly@tmforum.org +44 1727 863764

4.5 Acknowledgments

[Editor's note: If we have missed anyone please contact one of the editors to have the list corrected and please accept our apologies]

This document was prepared by the members of the TM Forum Architecture Harmonisation team:

Steve Orobec, BT Team Co-Lead
David Cleary, Ericsson Co-Lead
Richard Mishra, Amdocs, Editor
Alex Zhdankin, Harris-Stratex Network, Editor

Nigel Davis, Nortel, Editor
John Reilly, TM Forum, Editor
Mike Kelly, TM Forum, Editor
Steve Fratini
Francesco Caruso
Martin Huddleston
Dave Raymer
Dave Milham
John Petrie
Eric Dillon
Richard Craddock