Enterprise Architecture Management

Alfabet Reference Manual

Documentation Version Alfabet 10.11.0

CONVENTIONS USED IN THE DOCUMENTATION

Convention	Meaning
Bold	Used for all elements displayed in the Alfabet interface including, for example, menu items, tabs, buttons, dialog boxes, page view names, and commands. Example: Click **Finish** when setup is completed.
Italics	Used for emphasis, titles of chapters and manuals. this Example: see the *Administration* reference manual.
Initial Capitals	Used for attribute or property values. Example: The object state Active describes...
All Capitals	Keyboard keys Example: CTRL+SHIFT
File > Open	Used for menu actions that are to be performed by the user. Example: To exit an application, select File > Exit
< >	Variable user input Example: Create a new user and enter <User Name>. (Replace < > with variable data.)
	This is a note providing additional information.
	This is a note providing procedural information.
	This is a note providing an example.
	This is a note providing warning information.

TABLE OF CONTENTS

Chapter 1: Introduction to Enterprise Architecture Management

The Enterprise Architecture Management sales package allows you to describe complex IT systems in terms of their business, application, information and technical layers. The capabilities available in the Enterprise Architecture Management sales package enable you to capture your IT inventory, develop standards for change in the enterprise and support enterprise architects to align the IT landscape with the business in order to guide competitive transformation.

The Enterprise Architecture Management sales package can be used by enterprise architects to:

- capture the application architecture including lifecycle and versioning information
- document the information exchanged between applications and the data that is transferred
- assign applications, components, and platforms supporting the application architecture to ICT objects in order to plan and control costs in the IT
- define the enterprise's physical infrastructure and how it is used to support and deploy applications
- specify the enterprise’s IT standards including templates describing the tiers and layers constituting platforms, and standard platforms prescribing the components to use in application platforms
- specify a comprehensive library of business objects and their business data
- describe the enterprise's business processes in order to understand how these are supported by the enterprise's applications
- document the organizational hierarchy in the enterprise

The following information is available:

Chapter 2: Application Architecture Definition

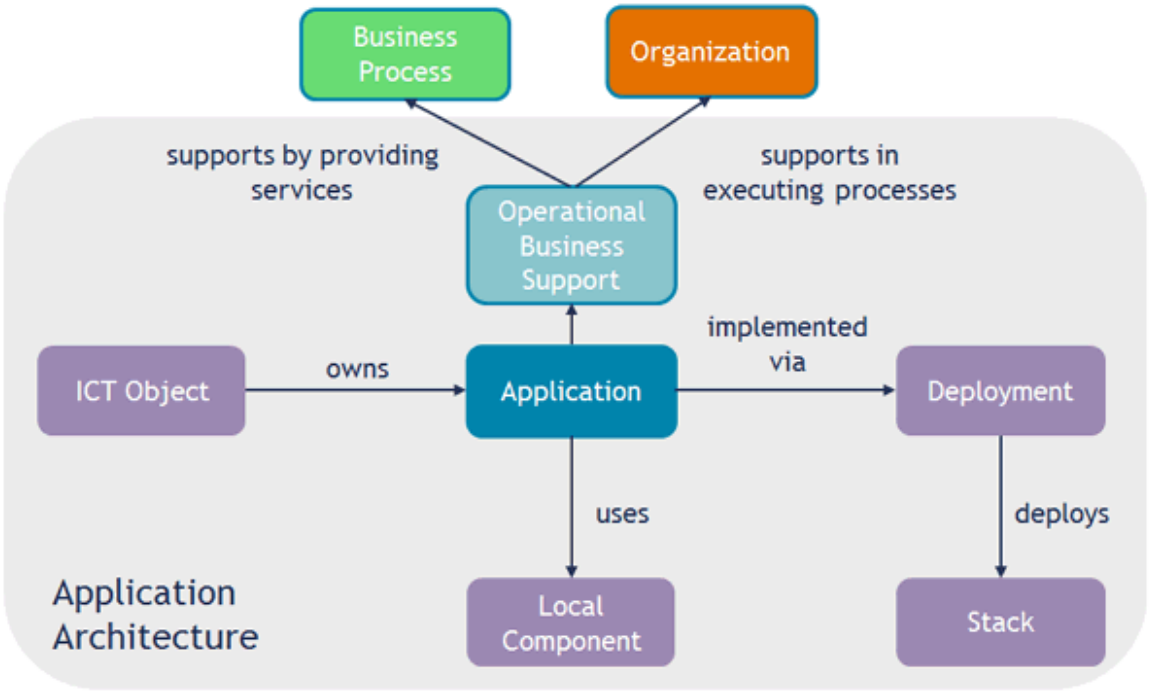
The Application Architecture Definition capability allows you to capture applications and specify the application architecture. The following information is available regarding the Application Architecture Definition capability:

- Methodology: Understanding the Application Architecture
- Understanding Governance and Responsibility for Applications
- Capturing the Applications in Your Enterprise
- Defining Application Lifecycles
- Versioning Applications
- Specifying Local Components for Applications
- Capturing Operational Business Supports
- Capturing ICT Objects to Understand IT Costs
- Capturing Relevant Peripherals

Please note that a context-sensitive Help is available for each view available in the Application Architecture Definition capability. You should refer to the Help if you require an explanation about the functionalities and information available in a specific view.

Methodology: Understanding the Application Architecture

The application architecture is a detailed description of a specific application. An application is a fully functional integrated IT product that provides functionality to the end user and supports the business to accomplish its mission.



The application architecture includes the application itself as well as any local components that are specifically required to implement the application. For example, the trading application Trade*Net might require the components Oracle Database, Apache Web Server, and IBM Power Server.

An application differs from a component in that the application directly supports the enterprise's business processes and organizations via operational business supports, has its own budget, is worked with and is known by name by the end user. Standard and local components do not typically provide functionality to end users but rather provide technical functionality to support an application. An operating system, database management system, or application server are examples of local components that an application typically requires. Any component supporting a specific application is defined as a local component and is not reusable by other applications or components.

A first step to qualify the value of IT with minor effort is to understand an application's operational business supports. An operational business support describes which business processes are supported by an application.

An application may have several physical deployments. Each deployment may have one or more stacks, which define the technically-installed infrastructure required for the operational deployment. Basic information about simple deployments may be captured in the context of defining the application architecture or they may be defined in technical detail in the context of defining the technical architecture as described in the section Defining the Operational Deployment of an Application, Component, or Standard Platform.

In addition to capturing an application's local components and deployment infrastructure, you can also specify the ICT object that owns the application. An ICT object (ICT = Information and Communication Technology) is an abstract object that represents applications regardless of their versioning and is a means to plan and control costs related to the application and its infrastructure. An ICT object owns the application and all of its application versions. The same ICT object can also own components, devices, solution building blocks, vendor products, and standard platforms, thus allowing the applications technological infrastructure to be budgeted along with the application. An ICT object is owned by an organization that is typically responsible for the ICT object's budget.

Finally, in the context of the Application Architecture Definition capability, you can also capture peripherals. A peripheral typically represents an application that is managed by business partners, such as an EDI

gateway or a B2B marketplace. It is outside of the core scope of the Alfabet solution but may be a source or target of an application's information flow.

Understanding Governance and Responsibility for Applications

A number of governance concepts are implemented in the Application Architecture Definition capability:

- **Authorized User:** Each application has an authorized user. An authorized user has primary responsibility for the application and thus has Read/Write access permissions to it. Users may also be assigned to authorized user groups. All users assigned to an authorized user group that has been defined for an application will have Read/Write access permissions to the application.
- **Roles**: A role defines the functional relationship or responsibility that a user or organization has to an application (for example, the Risk Manager or Architect of an application). Roles describe responsibilities but they do not authorize access permissions to the application in Alfabet.
- **Object Class Stereotypes**: Object class stereotypes may be configured by your solution designer for the object class Application. This allows for a different governance approach between different kinds of applications such as, for example, Business Applications and Technical Applications. If object class stereotypes are configured for the object class Application, each stereotype may capture a specified set of attributes, reference data, and class configurations as well as implement a different governance approach.
- **Mandates**: Applications may be managed in a federated architecture. Mandates are typically configured if object class stereotypes have been configured. By means of a federated architecture, it is possible to specify the visibility of individual applications in the Alfabet interface for specific users.
- **Application Groups**: Applications may be structured in one or more application groups. Each application group has an authorized user and may have authorized user groups. The authorized users of an application group will have access permissions to all applications in the application group.

To capture application groups, you must have access to the Application Portfolio Governance capability which is part of the Portfolio Management Basic sales package.

Objects in Alfabet are managed by various access permission concepts. For an overview of the access permission and governance concepts in Alfabet, see the section *Understanding Access Permissions in Alfabet* in the reference manual *Getting Started with Alfabet*.

Capturing the Applications in Your Enterprise

Applications are captured in the *Document Application Functionality* and *Capture Applications Functionality* functionalities. These views allow applications to be documented and defined in a quick and efficient manner. This capability is particularly useful for users with data entry responsibilities.

If object class stereotypes have been configured by your solution designer for the class Application, you will first be asked to select the stereotype that the application is based on. The application is then created and defined by means of the **Application** editor.

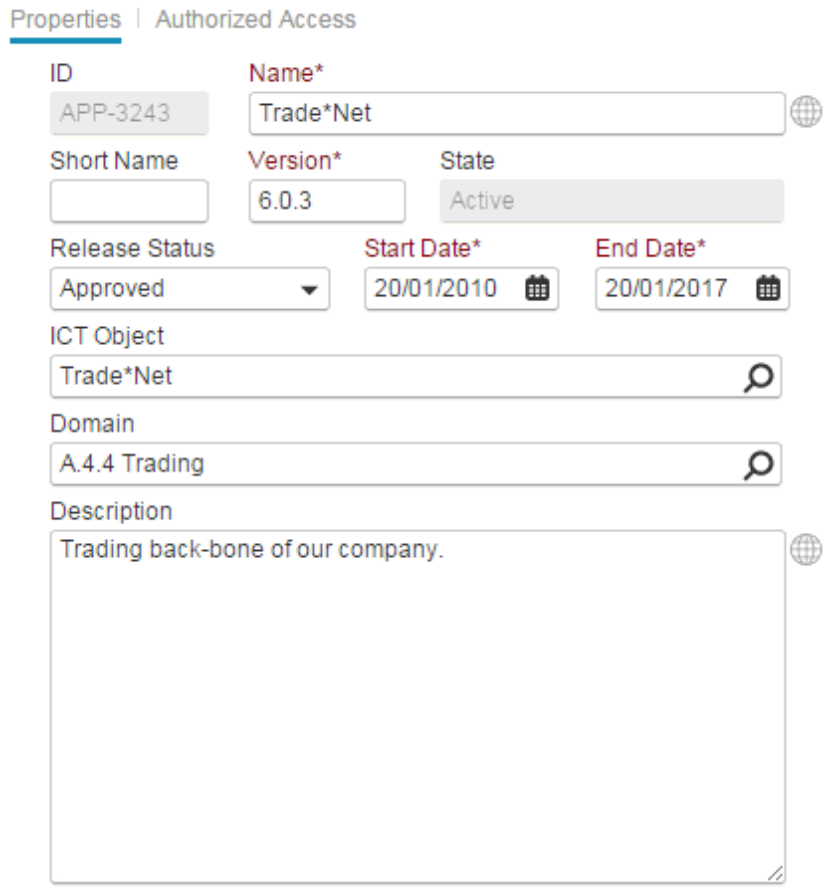


*FIGURE: Application editor used to create the trading application Trade*Net*

The following data is mandatory and must be defined when an application is created:

- Each application requires a unique name and version number. Applications can be versioned as needed. This is described in more detail in the section Versioning Applications.
- Each application requires planned start and end dates specifying when the application will be in production. The lifecycle phases of the application can then be defined later. This is described in more detail in the section Defining Application Lifecycles.
- Each application requires an object state. In the example above, a default object state has been configured that is automatically entered when the application is created. This can be changed however via the **Change State** button available in the application's object profile/object cockpit.
- Each application requires a release status, which typically expresses agreement to the state of the documented information. In the example above, a default release status has been configured that is automatically entered when the application is created. This can be changed in the editor, if needed.

The following information is optional:

- You should define a short abbreviated name to display on the application in diagrams, matrix reports, and business graphics.
- You may specify which ICT object owns the application. For more information about capturing ICT objects, see the section Capturing ICT Objects to Understand IT Costs.

- You may assign the application to a functional domain. From a functional point of view, the domain is the owner of the application.

  To capture domains, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

- You should provide a description of the application so that other users understand the purpose of the application.
- As the creator of the application, you are automatically defined as the authorized user per default. The authorized user of the application can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the application in the **Authorized Access** tab.

Operational costs can be captured for applications as an alternative to managing IT budgets via cost centers or ICT object budgets. In order to capture cost information, you must have access to the Opex Optimization capability which is part of the Portfolio Management Advanced sales package. The various alternatives available to capture and plan operational costs is described in detail in the section *OPEX Optimization* in the reference manual *Portfolio Management Advanced*.

Defining Application Lifecycles

Application lifecycle management includes the process of identifying and managing conflicts in the lifecycles of an application and its application versions and variants in order to ensure the availability and reliability of applications in the enterprise. Each application must have a start and end date and an object state defined.

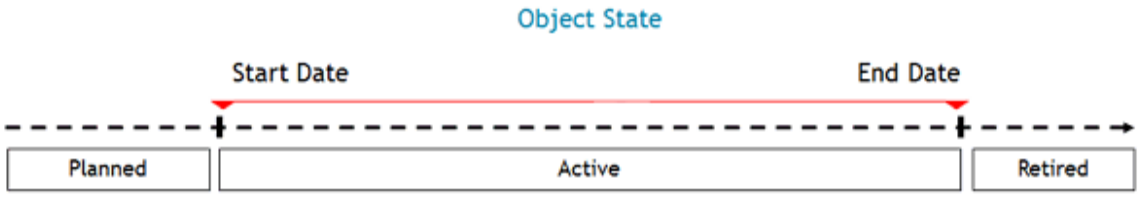


FIGURE: Application object state

The object state specifies the operational status of the application in the enterprise. The object state Active corresponds to the start and end dates of the application.

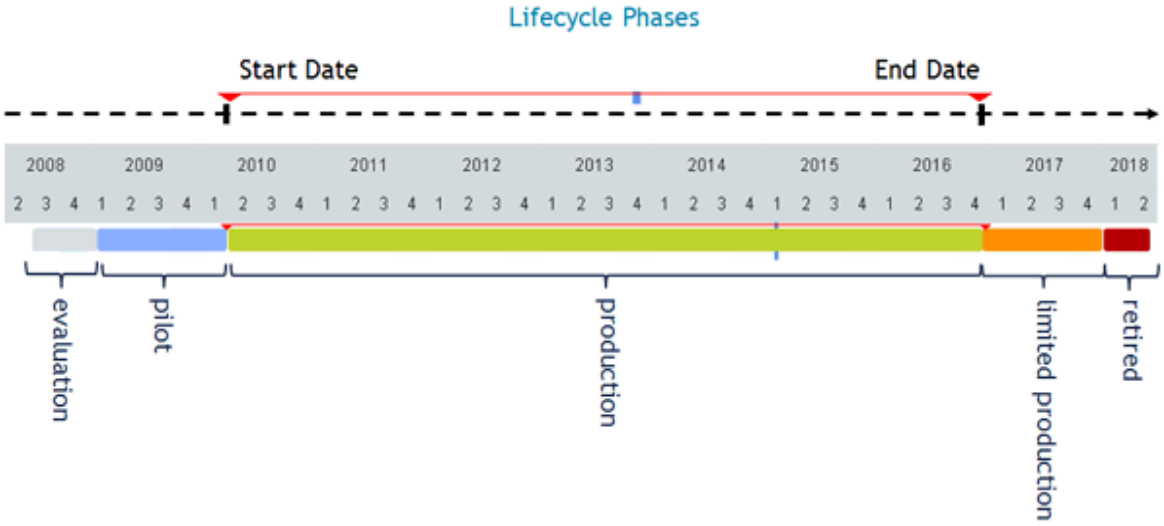


FIGURE: Application lifecycle

It is also possible to define a lifecycle made up of lifecycle phases, although this is optional. The application lifecycle describes the succession of stages that it goes through. The lifecycle is comprised of lifecycle phases that describe the application's status of activity or production. In the figure above, the trading application Trade*Net has the lifecycle phases Pilot, Production, Limited Production, and Retired. One or more lifecycle phases may represent the application's active period. In the example above, the lifecycle phase Production represents Trade*Net's active period.

You can define the application's lifecycle in the *Lifecycle Page View*. The *Lifecycle Page View* page view displays the application's lifecycles well as the lifecycles of its application versions and the ICT object that owns the application. In this way, you can identify and manage any conflicts in these lifecycles.

In order to specify lifecycle phases and manage an application's lifecycle, you must have access to the IT Planning Basic sales package. The methodology and requirements to document and analyze application lifecycles is described in detail in the section *Lifecycle Management* in the reference manual *IT Planning Basic*.

Versioning Applications

Versioning applications describes the transition of one version of an application to the next from the enterprise architecture point of view. Each application that you define is actually an application version with its own defined lifecycle. The application may have predecessor and successor versions, thus providing information about the migration plans for the application as well as the evolution of a specific type of business support or business service.

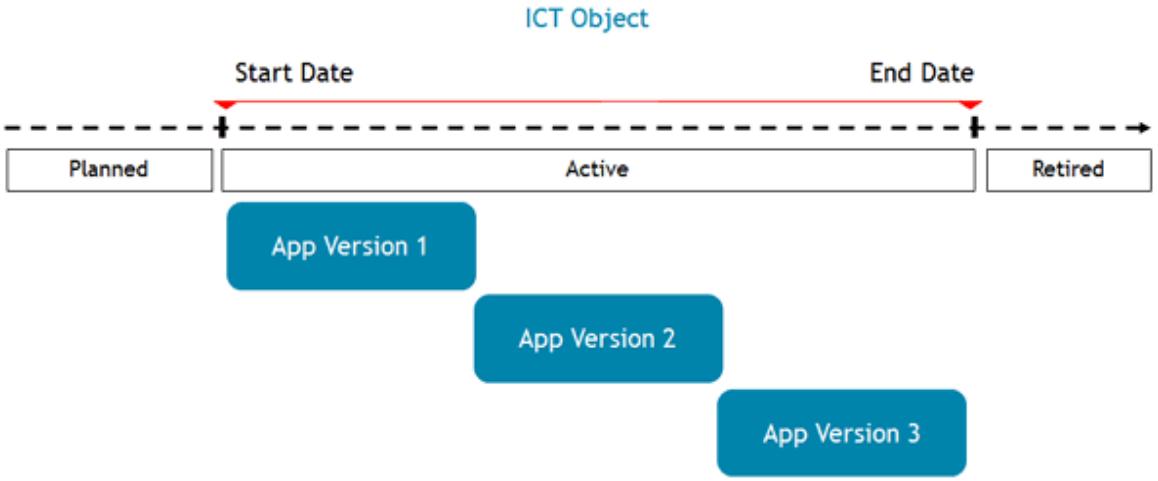


FIGURE: Versioned applications assigned to ICT object

The application version will become the successor version when the selected application reaches its end date. The new application version's start date will be automatically defined to begin one day after the base application's end date and the new application version's end date will be defined to begin 5 years after its start date. These dates may be edited as needed. The application version will automatically be assigned to the same ICT object as the application that it is versioned from. For example, the applications Trade*Net v. 6.0.3 v.1, Trade*Net v. 6.0.3 v.2, and Trade*Net v. 6.0.3 v.3 could be assigned to the ICT object Trade*Net.

The application version will inherit the lifecycle of the base application. You can create a version of a specific application in the *Document Application Functionality* and *Capture Applications Functionality* functionalities.

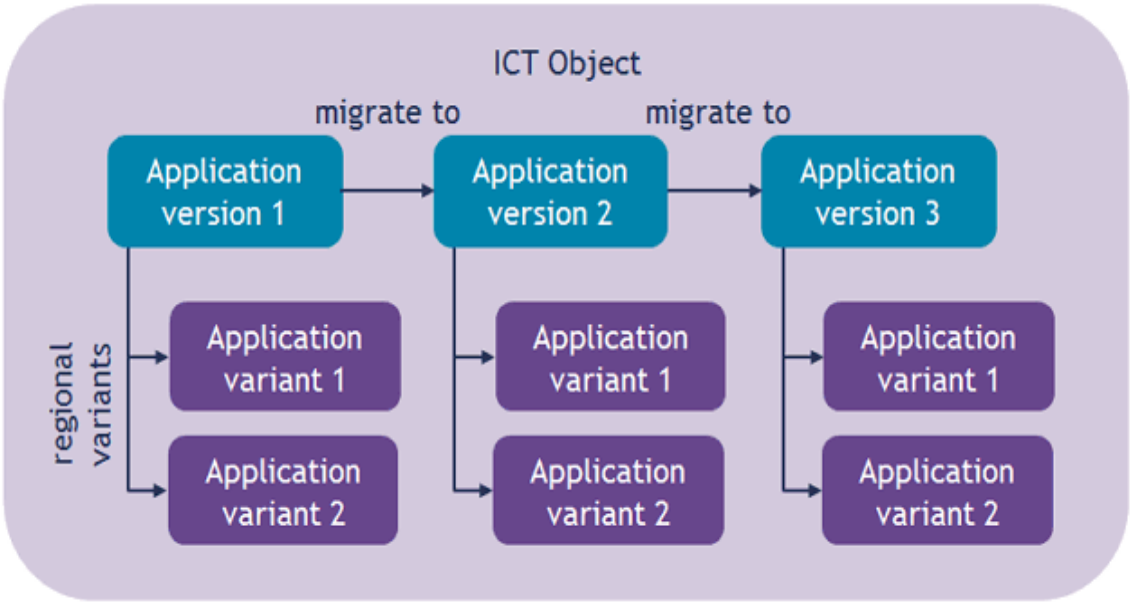


FIGURE: Application versions with variants

Applications may also possess application variants to account for the need to localize specific aspects of the application, such as the information flows needed to integrate the application in a local architecture environment. Variants are "copies" of an application that model local, functional, or technical differences of

an application. Application variants cannot be versioned. For example, the application Trade*Net 6.0.3, might have regional variants such as Trade*Net 6.0.3 Var. DACH and Trade*Net 6.0.3 Var. EMEA.

An application variant can be updated whenever the application that it is based on is modified. In this case, changes made to the base application's information flows, business support, business services, local components, and business data can be updated to the application variant. If necessary, an application variant can be promoted to an application in its own right that is independent of the base application. A new variant cannot be defined based on an existing application variant.

A variant of an application can be defined in the object profile of the relevant application or in the *Document Application Functionality* functionality. The application variant is considered a descendant of its base application and is nested below it in the explorer displayed in the *Document Application Functionality* functionality.

Specifying Local Components for Applications

An application may consist of or need components so that it can function. Typically, components do not provide functionality to end users but rather provide technical functionality to support the application. For example, a local component could be a batch procedure, interface component, or an application server.

In contrast to standard components that may be implemented in multiple platforms, for example, a component supporting an application is defined as a local component and is not reusable by other applications or components. Local components are either application-specific - that is, they are explicitly defined for the application - or they are created based on a standard component that is used for other applications, components, or standard platforms. For more information about standard components, see the section Documenting and Defining the Components in Your Enterprise.

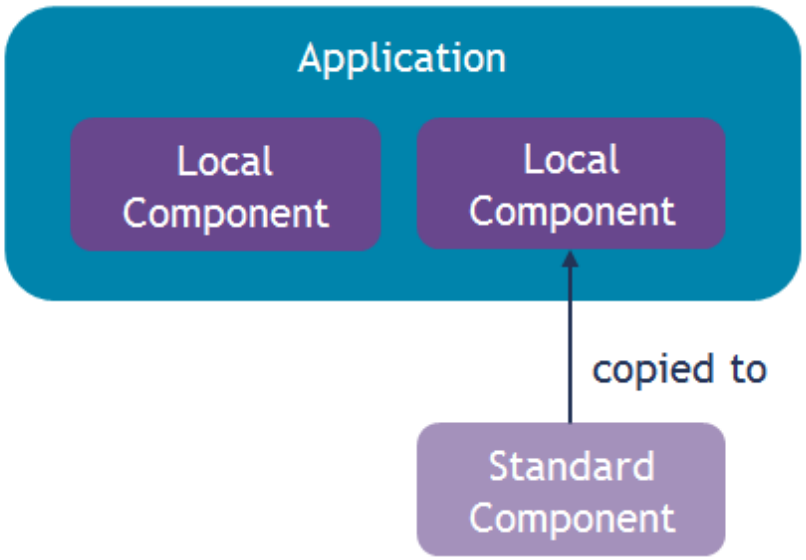


FIGURE: Local component based on a standard component

You can create a local component for the application in the *Components Page View*, where the local component could be explicitly created from scratch or created as a copy of a standard component. When you create a local component that is derived from a standard component, the standard component's name, version, short name, description, and component type is copied to the local component.

Please note that local components are an independent object class and do not inherit the object class stereotype of the component that they are based on. Object class stereotypes are not supported for the object class Local Component.

A local component will be displayed below its application in the application explorers in the Alfabet interface. Because some local components that you define will only be used to logically structure the selected application, you can hide them in application explorers via the **Action** > **Toggle Visibility** option in the *Components Page View*.

Technical services can also be defined for a local component. Technical services allow the technical needs required to support the business services provided by an application or local component to be documented. Technical services can only be created for local components for which the **Component Type** attribute is set to **Service**. For more information about defining technical services, see the section Defining the Technical Services Delivered by the Component.

Capturing Operational Business Supports

Business supports help you to understand which IT system is used, where and when it is used, and what it is used for. A first step to qualify the value of IT with minor effort is to understand an application's operational business supports. An operational business support describes which business processes are supported by an application. A simple assignment of applications to the supported business processes reveals first high-level information about redundancies and gaps. Later, simple impact analysis can be executed in order to ultimately plan a strategic migration from the as-is architecture to the to-be architecture.

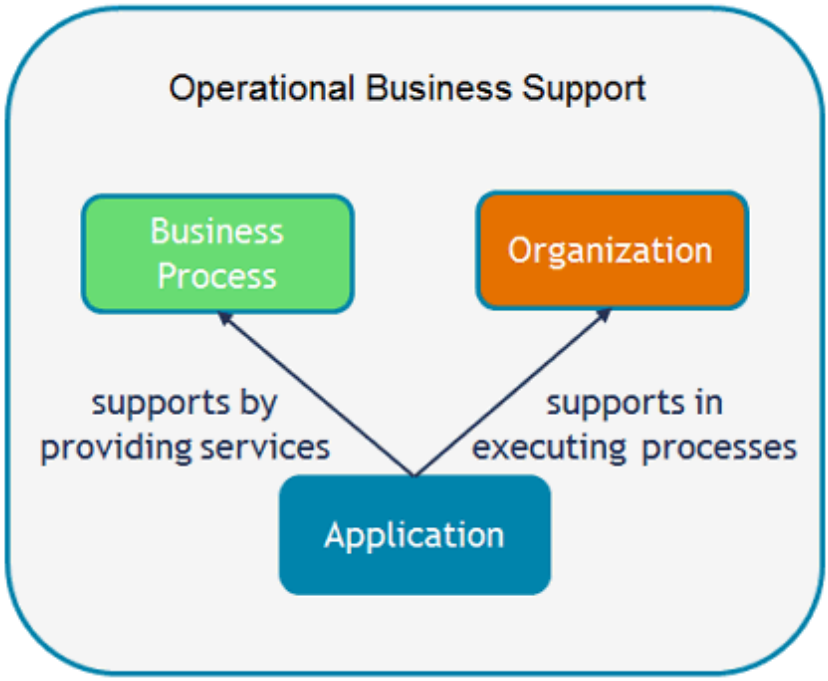


FIGURE: Business support is the relationship between an application, business process, and organization

An operational business support is the active or planned support that is currently provided as a result of ongoing development or roll-out activity. The business support consists of three dimensions:

- The application that is providing business support. In addition to applications, organizations may also be defined as providers of business support in Alfabet.
- The business process that the application supports by providing business services that the business process requires to fulfill a business function. Some companies refer to domains in their domain model instead of business processes.
- The organization that is supported by the application. The application typically supports an organization in its activities to carry out and execute business processes. Some companies refer to market products instead of organizations.

The operational business support has an object state and typically inherits the lifecycle of the providing application. Because operational business supports have a lifecycle definition that is independent of the application's lifecycle definition, the costs of business support can be planned and analyzed separately from the operational costs of the application.

The operational business support that is or will be provided by an application is defined in the *Provided Business Support Page View* for the application.

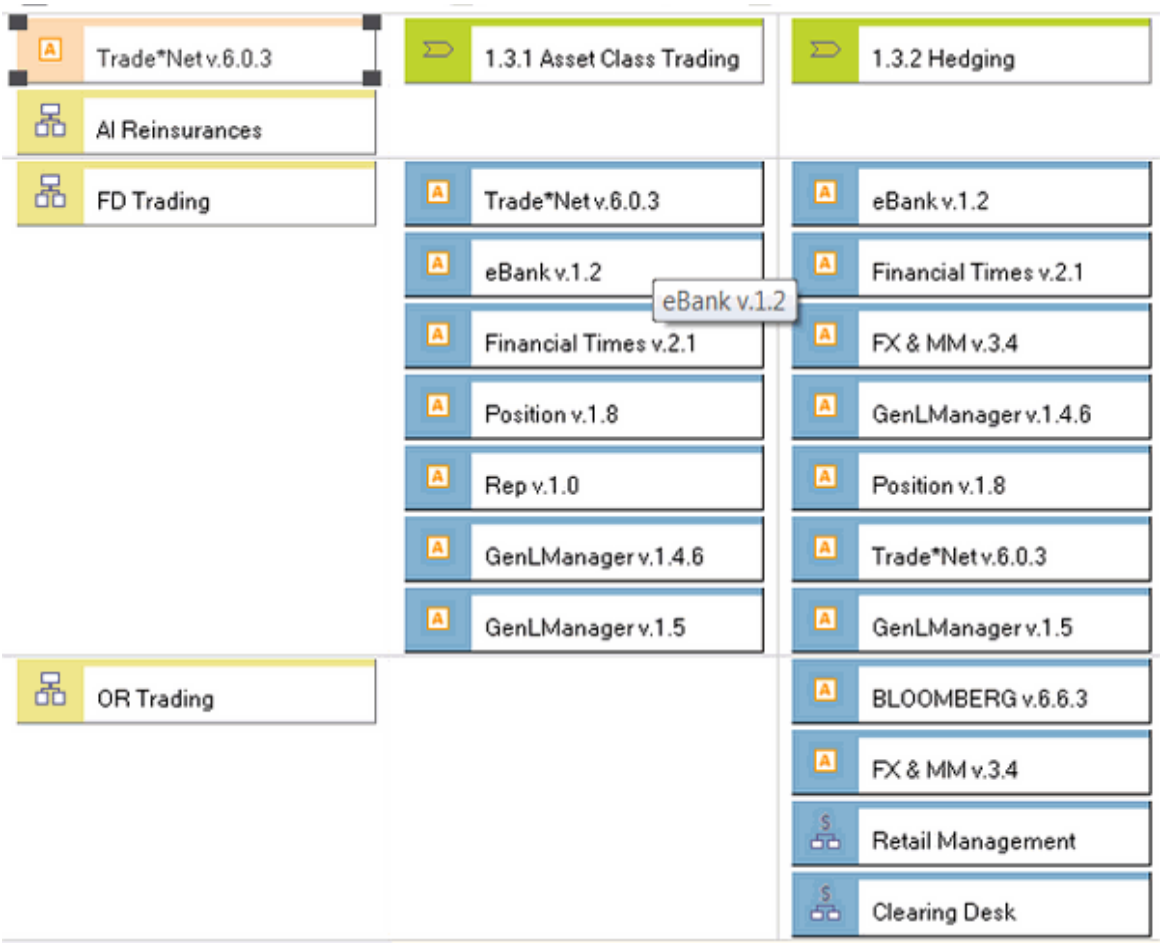


*FIGURE: Business Support Map Report for the application Trade*Net*

Once operational business supports have been defined, you can review the support provided by the application. In the example above, the *Business Support Map Report Page View* provides an overview of the business processes and organizations supported by the application Trade*Net (orange cell). The reports displays all operational business support provided by all applications for the set of business processes and organizations that the application Trade*Net supports. The business support map is an operational IT map that allows for bottom-up planning of existing approved and budgeted business support. In this case,

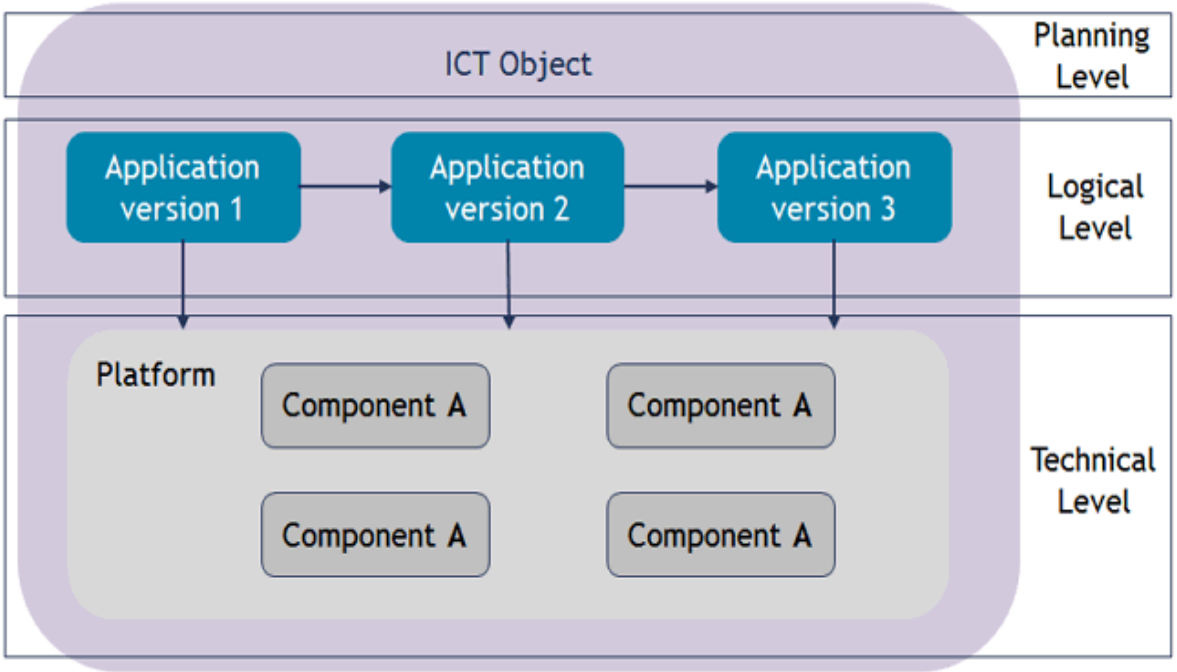
planning is done on the level of application versions for a specified number of years. In the context of the business support map, a matrix cell with multiple business supports could indicate a potential redundancy in the architecture whereas a matrix cell without any business supports could indicate a gap in the support of a business process.

Before you can document the business services that are provided by the business support, you must first create the business service. To capture business services, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

Once you have captured the business support that is provided by applications, you can analyze the current operational support and plan your enterprise's medium-term and long-term IT. In the context of an IT strategy, you can roadmap your company's long-term strategic business support by means of the Business IT Synchronization capability. In the context of a master plan, you can roadmap the medium-term tactical business supports by means of the Target Architecture Definition capability. Both capabilities are part of the IT Planning Basic sales package.

Capturing ICT Objects to Understand IT Costs

ICT objects represent a controlling and planning view of the IT architecture. An ICT object (ICT = Information and Communication Technology) is an abstract object that represents applications regardless of their versioning and is a means to plan and control costs related to the application and its infrastructure. The use of ICT objects is advantageous in that the planner must not initially commit him/herself to a certain version of the application. By means of the ICT object, portfolio managers can understand the operating costs of the application. Later, at the stage of detailed planning, the ICT object can be replaced by a specific concrete version.



In addition to applications, ICT objects may also own components, devices, solution building blocks, vendor products, and standard platforms that are related to the applications for technical or business reasons, In this way, an application's technological infrastructure can be budgeted along with the application.

The object classes or their object class stereotypes that may be owned by an ICT object stereotypes must be configured by your solution designer in the XML object ***ICTObjectManager***. For more information, see the section *Configuring the ICT Object Hierarchy* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The ICT object is owned by an organization that is usually responsible for the budget of the architecture elements assigned to the ICT object.

For example, the ICT object Trade*Net could own the following:

- The application Trade*Net 6.0.3
- The standard platforms relevant for the application: Trading Application Server v. 1, Trading Client v. 1, and Trading Platform v. 2.3
- The deployments Trade*Net v. 6.0.3 #3 EMEA and Trade*Net v. 6.0.3 #3 ROW

The ICT object can be created in the *Document ICT Objects Functionality* or *Capture ICT Objects Functionality*. If object class stereotypes have been configured by your solution designer for the class ICT Object, you will first be asked to select the stereotype that the ICT object is based on. The ICT object is then created and defined by means of the **ICT Object** editor.

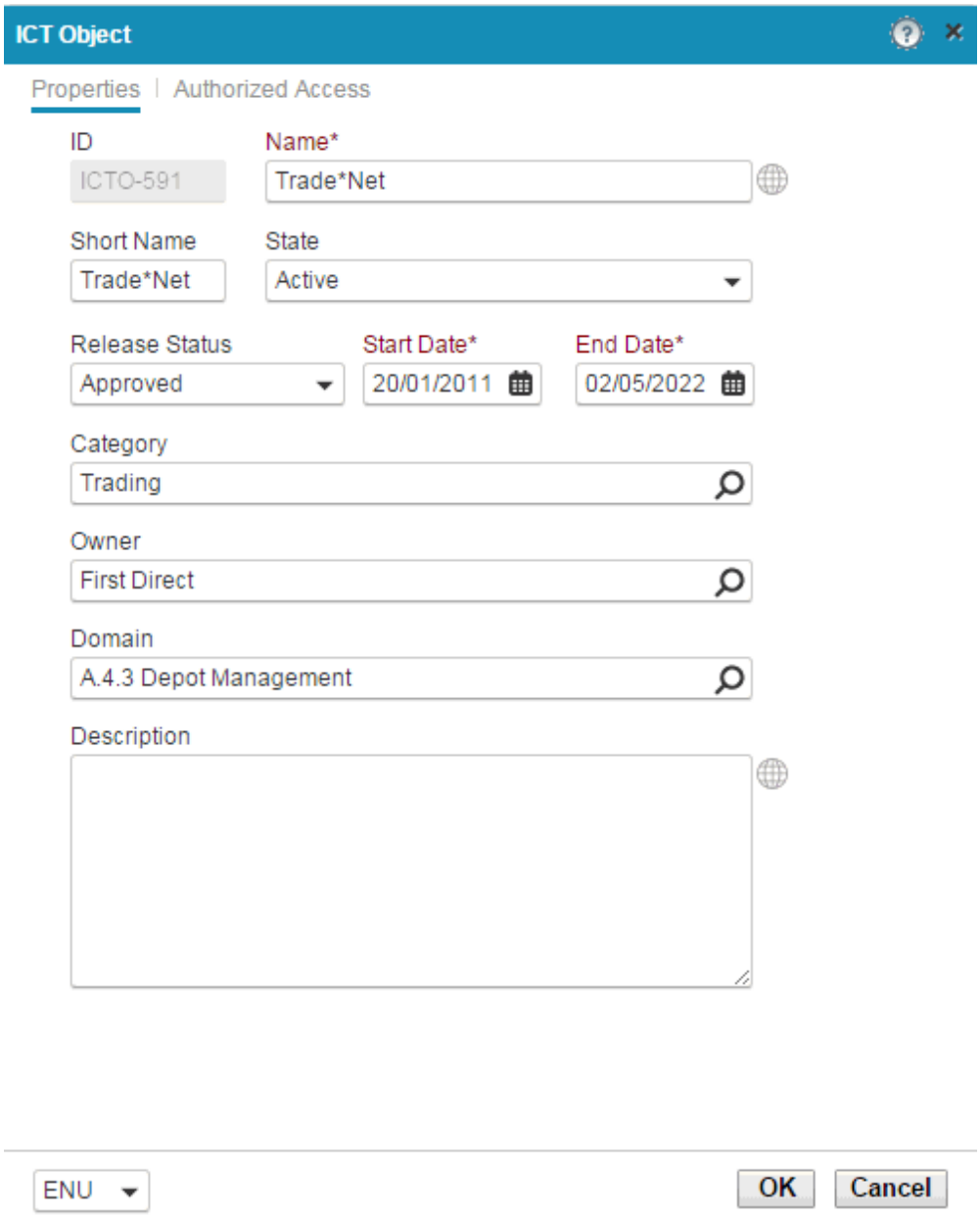


FIGURE: ICT Object editor used to create an ICT object

The following data is mandatory and must be defined when an ICT object is created:

- Each ICT object requires a unique name.
- Each ICT object requires planned start and end dates. These dates should be defined so that they begin before and end after the start/end dates of the applications that they own. The lifecycle phases of the ICT object can be defined later in the *Lifecycle Page View*. The lifecycle of the ICT object as well as all applications, components, standard platforms, etc. that the ICT object owns will be displayed in the *Lifecycle Page View*, allowing you to ensure the alignment of the ICT object and application lifecycles.
- Each ICT object requires an object state.
- Each ICT object requires a release status, which typically expresses agreement to the state of the documented information.

The following information is optional:

- You should define a short abbreviated name to display on the ICT object in diagrams, matrix reports, and business graphics.
- You may specify which ICT object category bundles the ICT object. Assigning the ICT object to an ICT object category allows the ICT objects to be organized and analyzed for costs and architectural complexity. ICT object categories can be created in the *ICT Objects by Category Explorer*. ICT objects can be assigned to the relevant ICT object category in the *ICT Objects Page View* of the ICT object category.
- You may assign the organization that owns the ICT object and is responsible for its budget.
- You may assign the ICT object to a functional domain. From a functional point of view, the domain is the owner of the ICT object.

  To capture domains, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

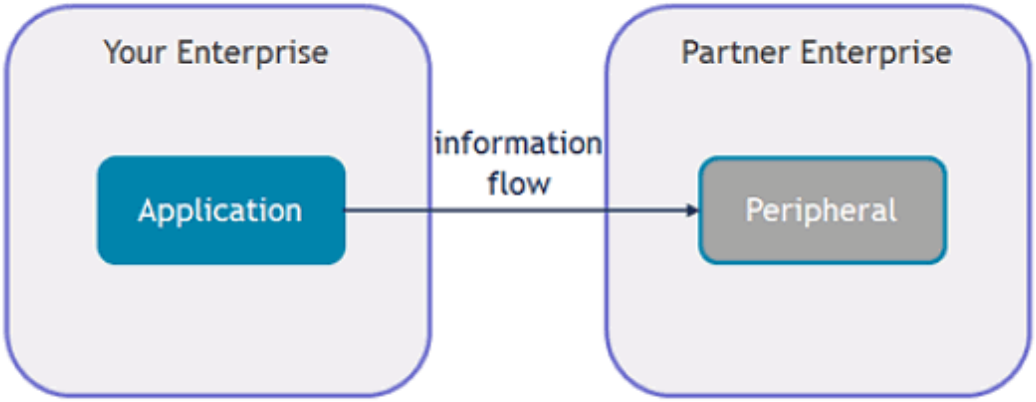
- You should provide a description of the ICT object so that other users understand the purpose of the ICT object.
- As the creator of the ICT object, you are automatically defined as the authorized user per default. The authorized user of the ICT object can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the ICT object in the **Authorized Access** tab.

Once the ICT object is created, you can assign the applications, components, and standard platforms that it owns in the *Applications Page View*, *Components Page View*, and *Standard Platform Page View*.

Operational costs can be captured for ICT objects as an alternative to managing IT budgets via cost centers or application budgets. In order to capture cost information, you must have access to the Opex Optimization capability which is part of the Portfolio Management Advanced sales package. The various alternatives available to capture and plan operational costs is described in detail in the section *Methodology: Understanding OPEX Optimization* in the reference manual *Portfolio Management Advanced*.

Capturing Relevant Peripherals

A peripheral is an element in the IT landscape that is outside of the core scope of the Alfabet solution. A peripheral typically represents an application that is managed by business partners, such as an EDI gateway or a B2B marketplace. Peripherals are connected to applications in the IT landscape by means of information flows. In contrast to an application, a peripheral's local components or business services as well as the details of its technical platform are of no relevance in the Alfabet solution.



A peripheral that is relevant for an application in the Alfabet inventory can be created in the *Document Peripherals Functionality* or *Capture Peripherals Functionality* views. The following data is mandatory and must be defined when a peripheral is created:

- Each peripheral requires a unique name and version.
- Each peripheral requires planned start and end dates.
- Each peripheral requires an object state.

The following information is optional:

- You should define a short abbreviated name to display on the peripheral in diagrams and business graphics.
- You should provide a description of the peripheral so that other users understand the purpose of the peripheral.
- As the creator of the peripheral, you are automatically defined as the authorized user per default. The authorized user of the peripheral can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the peripheral in the **Authorized Access** tab.

Once the peripheral has been created, it can be selected as an incoming or outgoing source of an information flow as described in the section Defining Information Flows Between Applications, Local Components, and Peripherals.

Chapter 3: Information Architecture Definition

Please note that a context-sensitive Help is available for each view available in the Information Architecture capability. You should refer to the Help if you require an explanation about the functionalities and information available in a specific view.

The following information is available:

- Methodology: Understanding the Information Architecture
- Prerequisite: Specifying the Connection Information for Information Flows
- Defining Information Flows Between Applications, Local Components, and Peripherals
- Defining the Business Data That Is Exchanged
- Defining Local Components as Interfaces
- Defining an Interface System

Methodology: Understanding the Information Architecture

The interface logic that is required for the exchange of information is an integral part of the information architecture. The information architecture describes the information flows that exchange business data between source and target applications and/or peripherals (application > application, application > peripheral, peripheral > peripheral). While information flows may also be defined between components (component > component) or devices (device > device), the descriptions below will focus on the information flows in the application architecture. For each information flow between applications, one application is considered the source of the information flow and the other application is considered the target of the information flow. The number of information flows that an application is associated with helps the application architect determine the complexity of the application, potential redundancies among applications, and the effort involved in migrating or replacing an application.

The information architecture can be documented in a number of ways. The method you choose will depend on the methodology used in your enterprise as well as the level of maturity of your IT architecture.

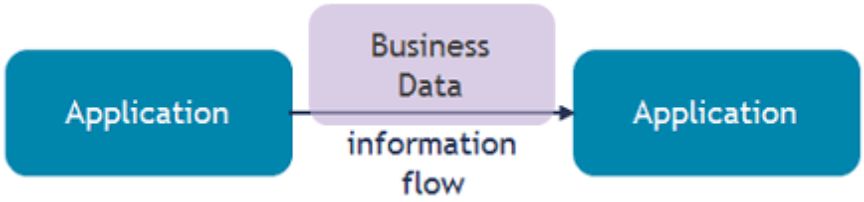


FIGURE: Information flow between a source application and target application

The most basic and simple means to document the information flows in the IT architecture is to capture the transfer of business data exchanged between applications. As shown in the image above, one application is the source of the information and the other application is the target of the information. The transferred information is documented as business data. Each applications may be both a source and target of information flows. In this case, two information flows exist between the applications, whereby App1 is the

source and App2 is the destination for InformationFlow1 and App2 is the source and App1 is the destination for InformationFlow2.

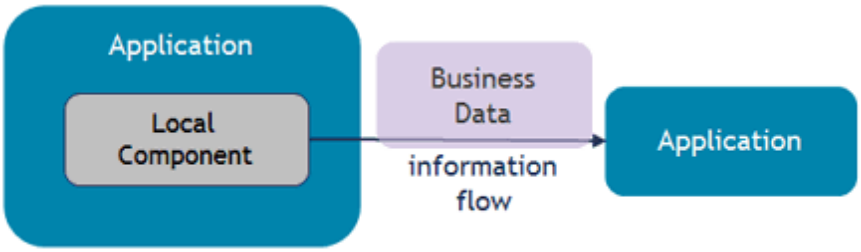


FIGURE: Information flow between a source local component and target application

Instead of the application, the local component might be the source or target of the business data exchange. This might be the case, for example, for an application App2 that communicates with an application server (local component) of App1. It is also possible that the business data is exchanged between a local component assigned to App1 and a local component assigned to App2.

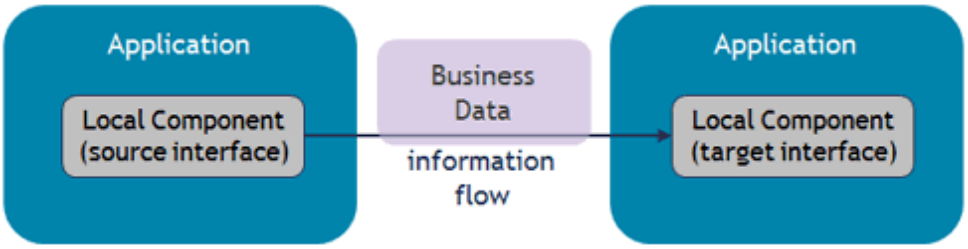


FIGURE: Information flow between interfaces

A more complex situation is if an information flow exchanges business data between the interfaces of two applications, as shown in the figure above. An interface is the technical component associated with an information flow that realizes the technical compilation necessary for the data exchange. In this case, local components are specified as the source and target interfaces that enable data transfer. Please note that the source and target interfaces must be defined as components or local components in the platform of the respective application.

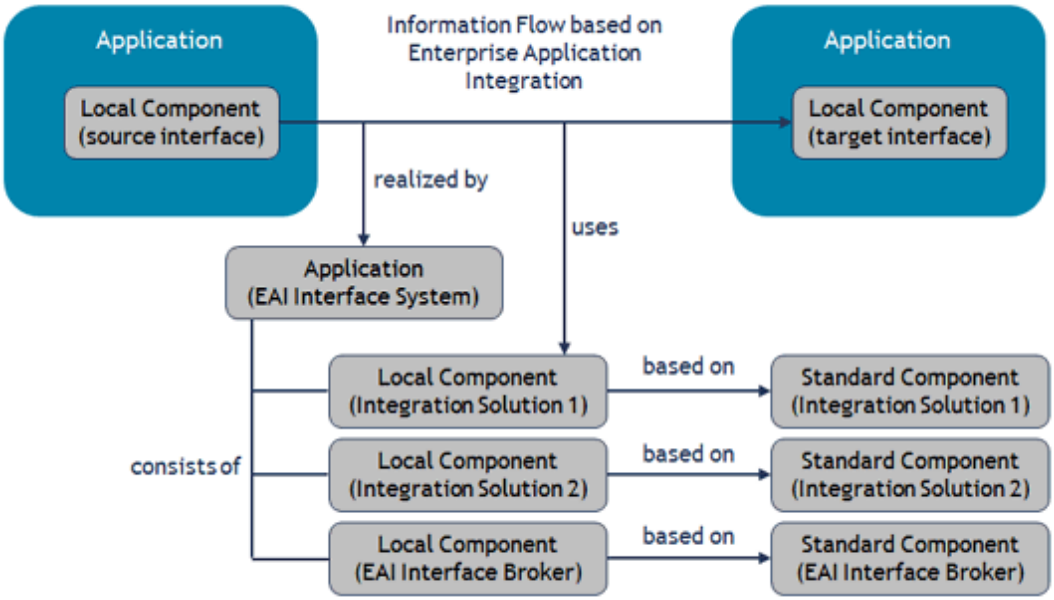


FIGURE: Information flow via an interface system

In some cases, the exchange of data is accomplished using a dedicated communication system, such as an EAI bus, which can be modeled by an information flow. The system is referred to as an interface system and constitutes the architecture elements that transport and transform business data associated with information flows. In this case, the information flows between applications are realized through concrete interface systems which are typically embedded in the application's platform. Enterprise Application Integration platforms, message queues, request brokers, bus systems, or service orchestrators are typical examples of interface systems.

Prerequisite: Specifying the Connection Information for Information Flows

One aspect that can be documented about an information flow is the connection information. The following data about the information flow's connection can be captured in the **Information Flow** editor.

- The connection type describes the mode of transfer used by a specific information flow to transfer business data between the two associated applications or their respective components. Examples include batch and online.
- The connection method describes the method of transfer used by a specific information flow to transfer business data between the two associated applications or their respective components. Examples include TCP/IP, file transfer, message queue.
- The connection frequency describes how often a specific information flow is used to transfer business data between the two associated applications or their respective components. Examples include daily, monthly.
- The connection data format describes the data format used for the transfer of business data via a specific information flow. Examples include ASCII, XML.

The values that fill the **Connection Data Format**, **Connection Data Format**, **Connection Method**, **Connection Type** fields in the **Information Flow** editor must first be defined in the **Reference Data** functionality before you or other users can define the connection information. This is described in detail in the chapter *Configuring the Connection Data for Information Flows* in the reference manual *Configuring Evaluation and Reference Data in Alfabet*.

Defining Information Flows Between Applications, Local Components, and Peripherals

Information flows are created in the *Information Flows Page View* of the application, peripheral, or local component that is either the source or target of the information flow.

The following data can be defined when an information flow is created:

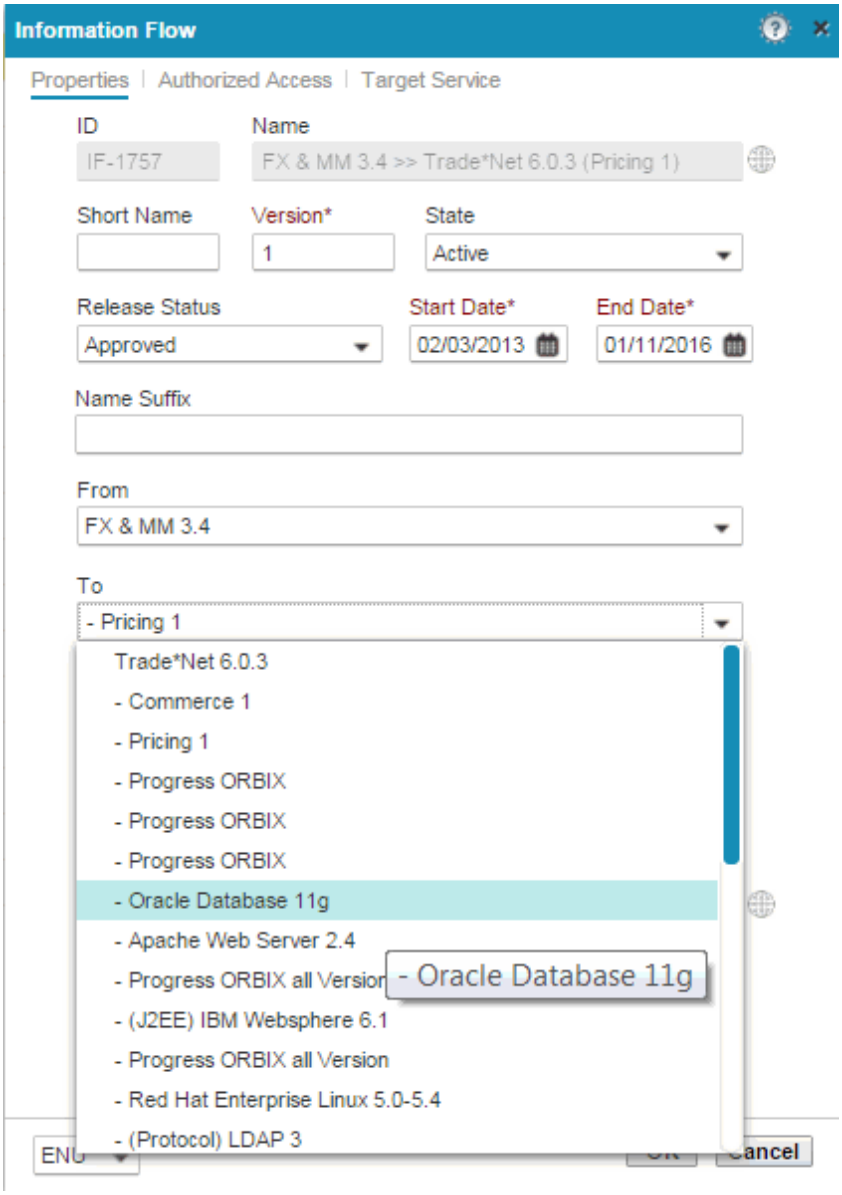
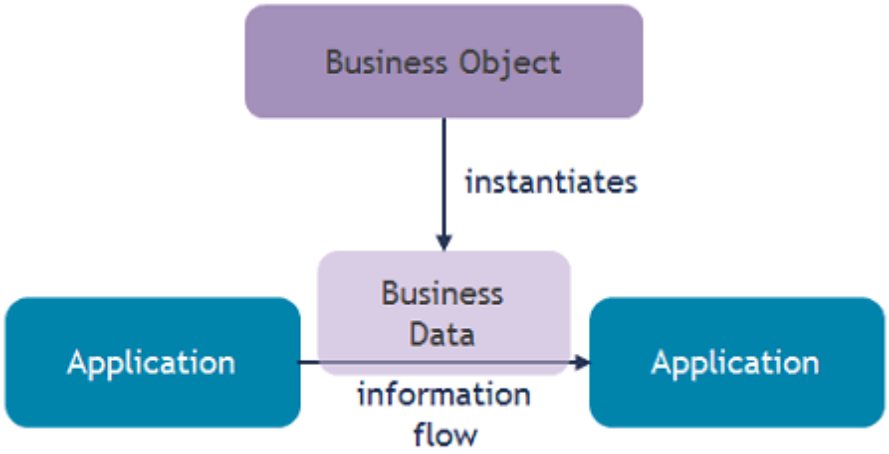


FIGURE: Creating an information flow targeting a local component

- The application that you are currently defining will be automatically entered in the **From** or **To** field, depending on whether you have selected the option to create an incoming or outgoing information flow. The syntax for the information flow will be displayed in the *Information Flows Page View* as <SourceApplication> >> <TargetApplication> or if the case may be <SourceApplication> >> <TargetApplication (LocalComponent)>.
- You should define a short abbreviated name to display on the information flow in diagrams and business graphics.
- You must define the planned start and end dates specifying when the information flow will be in production. When you create a new information flow, the information flow's start date is, by default, the earliest date defined for either the source or target component and the end date is, by default, the latest date defined for either the source or target component. You may edit the information flow's start and end dates, but the dates must fall within the range of the lifecycle dates for the source and/or target components. If the dates are not aligned with the source or destination applications, the date cell will be colored red in the view.
- You may be able to define the object state of the information flow. Please note the following:
  - The information flow's object state may only be defined as active if its source and target applications also have an active object state.
  - The information flow must have a retired object state if either the source or target application has a retired state.
- You should define a release status, which typically expresses agreement to the state of the documented information.
- You may specify the connection information for the information flow if values have been configured for the **Connection Data Format**, **Connection Data Format**, **Connection Method**, **Connection Type** fields in the **Reference Data** functionality. This is described in detail in the chapter *Configuring the Connection Data for Information Flows* in the reference manual *Configuring Evaluation and Reference Data in Alfabet*.
- You should provide a description of the information flow so that other users understand the purpose of the information flow.
- As the creator of the information flow, you are automatically defined as the authorized user per default. The authorized user of the information flow can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the information flow in the **Authorized Access** tab.

Defining the Business Data That Is Exchanged

Business data are exchanged between applications and their technical components by means of information flows. They are processed in the business services that are provided by the applications/components.



A business data is the concrete instantiation of a business object, which represents an abstract entity that is relevant to the enterprise's business domain. A business object is typically not versioned and represents, for example, customer, transaction, target market, statistical data, stock trade, future trade, etc. Business data, on the other hand, represents the concrete instances that are used by an application and typically are versioned. A business object Future Trade, for example, could have the business data Future Trade v. 1, Future Trade v. 2, Future Trade 2016, etc.

You can capture the business data that is transferred by an information flow in the *Business Data Page View* in the information flow's object profile. You can also capture the business data that an application or local component operates on in the *Business Data Page View* in the object profile of the respective application or local component.

When you create the business data, you will first be asked to select the business object that it instantiates. The name of the business data will be the same as the name of the selected business object, but this can be changed. You must also define a version number for the business data.

Before you can document the business data that is transferred by an information flow, you must first create the business object that the business data is associated with or make sure that a business object exists that your new business data can be based on. To capture business objects, you must have access to the Information Portfolio Governance capability which is part of the Portfolio Management Basic sales package.

Once you have captured the business data that is exchanged by applications, you can specify and analyze the business data usage and think about consolidating information objects and developing and establishing information sourcing strategies which is a focus of the Information Portfolio Governance capability.

Defining Local Components as Interfaces

An information flow can exchange business data between the interfaces of 2 applications. An interface is the technical component associated with an information flow that realizes the technical compilation necessary for the data exchange. In this case, local components are specified as the source and target interfaces that enable data transfer. The source and target interfaces must be defined as components or local components in the platform of the respective application.

The interface can only be defined if an application is the source/target of the information flow. It cannot be defined if a local component is the source/ target of the information flow. In other words, the local component is specified as the interface for the information flow but not the source or target of the information flow.

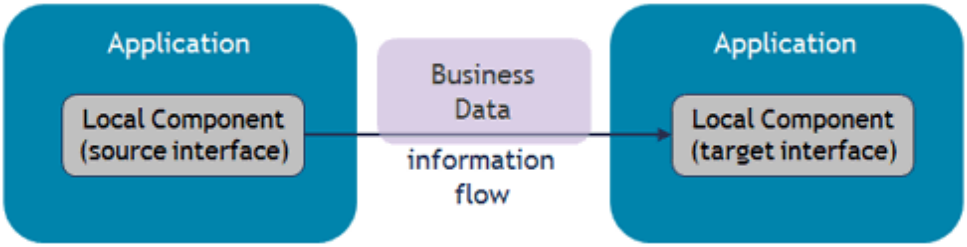
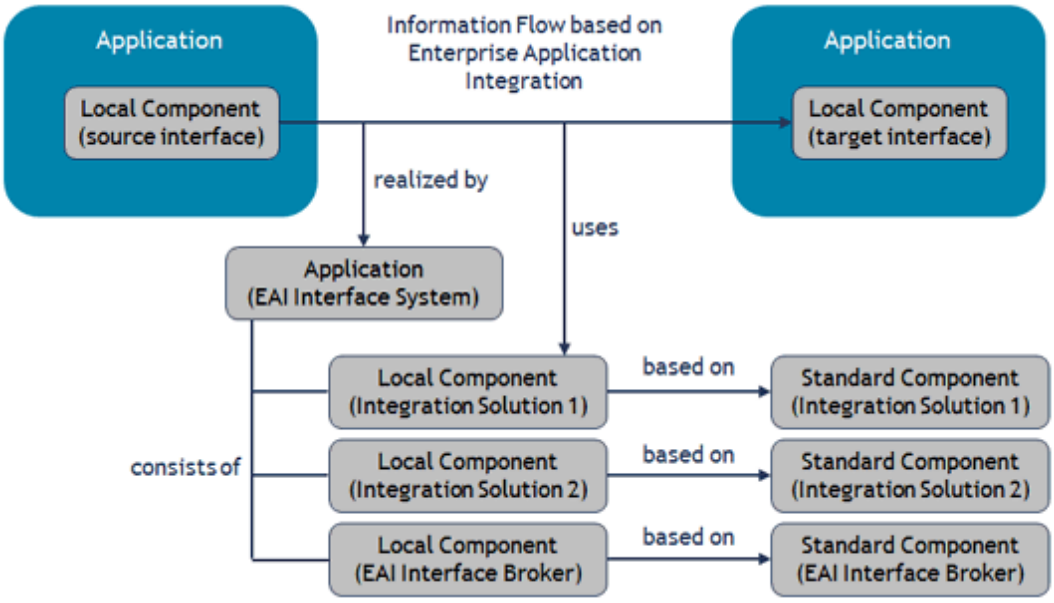


FIGURE: Information flow between interfaces

To define the source and target interfaces of an information flow, you must first ensure that the relevant local component that will be defined as the interface is assigned to the source application and the target application of the information flow in the *Components Page View* of the relevant applications. The information flow's interfaces can then be defined in the *Interfaces Page View* in the information flow's object profile.

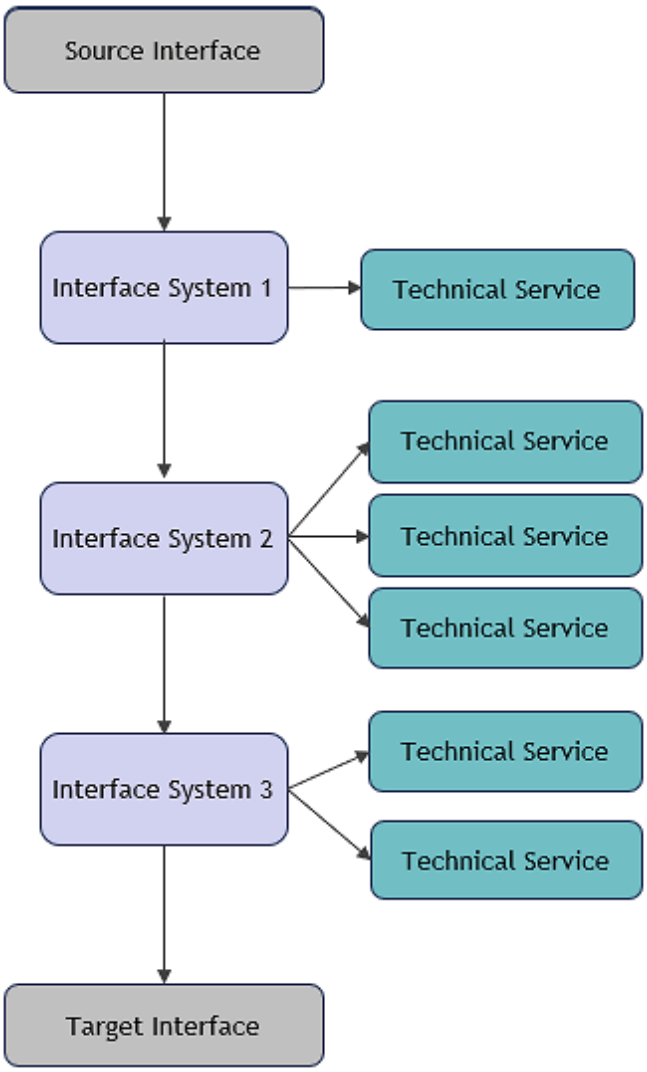
Defining an Interface System

You can model an interface system as middleware for an information flow if, for example, the exchange of data is accomplished using a dedicated communication system such as an Enterprise Application Integration system or bus technology. The interface system is the collection of architecture elements that transport and transform business data associated with information flows. The information flows between applications are realized through concrete interface systems which are typically embedded in the application's platform. Enterprise Application Integration platforms, message queues, request brokers, bus systems, or service orchestrators are typical examples of interface systems.



Both applications and components may model an interface system. An interface system may provide business support in its own right, in which case it should be modeled as an application.

Alternatively, the interface system may provide technical support for the exchange of information, in which case it could be modeled as a component.



An interface system can be derived from a component, application, or local component of the component or application. If an interface system is based on a component or local component of the type **Service**, you can specify the technical services and their technical service operations in order to specify the service calls that are relevant for the interface system. the interface systems can be sequenced in order to capture the order that the services must be called between the source and target interfaces.

The information flow's interface system is defined in the *Interfaces Page View* in the information flow's object profile. The application or component that the interface system is based on must already be defined in the database.

Chapter 4: Technology Architecture Definition

The technology architecture defines the technical view of the enterprise architecture and describes the software and hardware components, technical infrastructure, and the associated standards that are implemented. Components and standard platforms are the core elements of the technical architecture from an application-centric view.

Please note that a context-sensitive Help is available for each view available in the Technology Architecture Definition capability. You should refer to the Help if you require an explanation about the functionalities and information available in a specific view.

The following information is available:

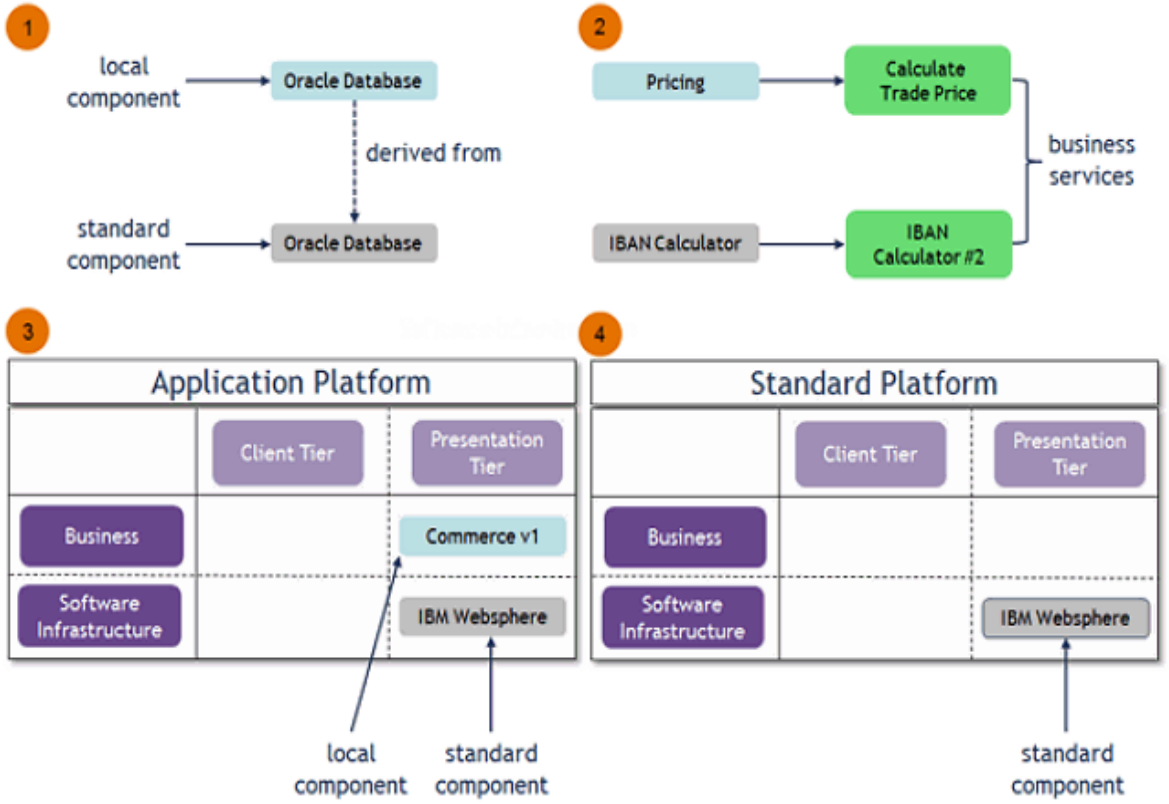
- Methodology: Understanding the Technical Architecture
  - Components in the Technical Architecture
  - Standard Platforms in the Technical Architecture
  - Deployments in the Technical Architecture
- Understanding Governance and Responsibility for Components
- Documenting and Defining the Components in Your Enterprise
- Defining the Component's Lifecycle
- Versioning the Component
- Defining the Technical Services Delivered by the Component
- Defining Platform Templates and Standard Platforms
- Defining the Platform Architecture for an Application or Component
- Defining the Operational Deployment of an Application, Component, or Standard Platform
- Defining and Managing Technical Networks and Deployments

Methodology: Understanding the Technical Architecture

An enterprise technical architecture defines the technical view of the enterprise architecture and describes the enterprise's software and hardware components, its platforms and associated standards, such as platform templates and standard platforms, and the deployment architecture.

Components in the Technical Architecture

Unlike a local component which is used by a specific application, a standard component is a reusable block of functionality that is relevant for the enterprise's business operations or infrastructure, or to provide technical services. Standard components may be implemented in Alfabet as follows:



1) A standard component can be used to derive a local component. In this case, the attributes of the standard component are copied to the local component. However, the local component is an independent entity that can be modified in the context of the application that it supports.

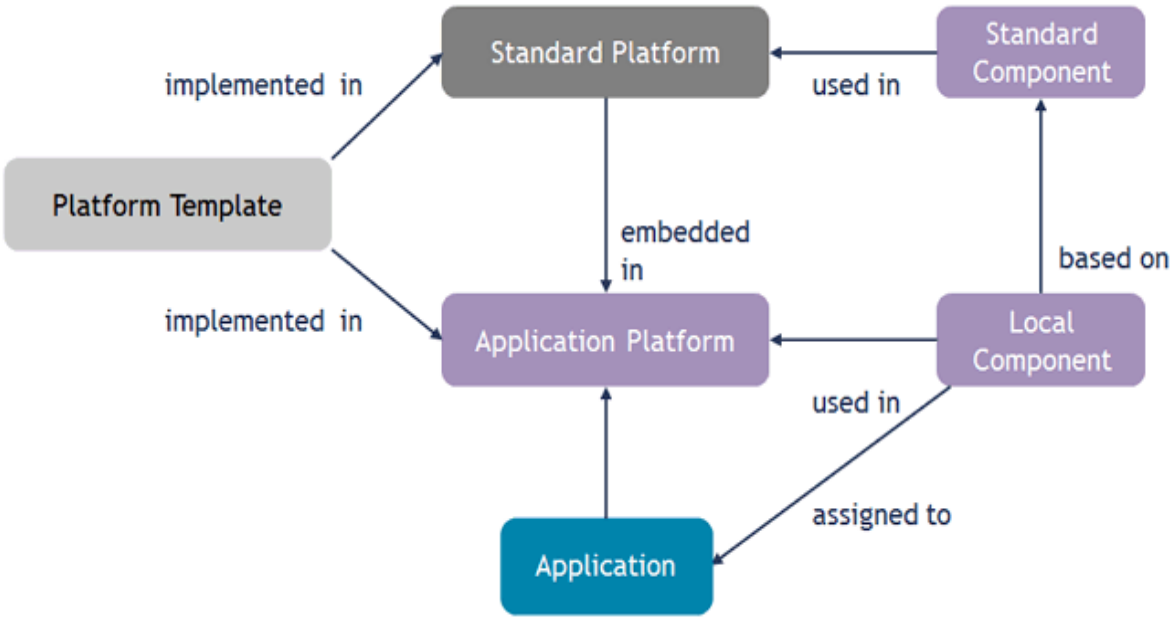
2) A standard component for which the **Component Type** attribute is set to **Service** can deliver technical services that are necessary to support requested business services.

3) A local component or standard component can be assigned to an application's platform or a component's platform. In the context of the application/component platform, the component is considered a platform element. The platform element can have a lifecycle that is different from that of the standard component.

4) A standard component can be assigned to a standard platform. In the context of the standard platform, the component is considered a standard platform element. The standard platform element can have a lifecycle that is different from that of the standard component.

Standard Platforms in the Technical Architecture

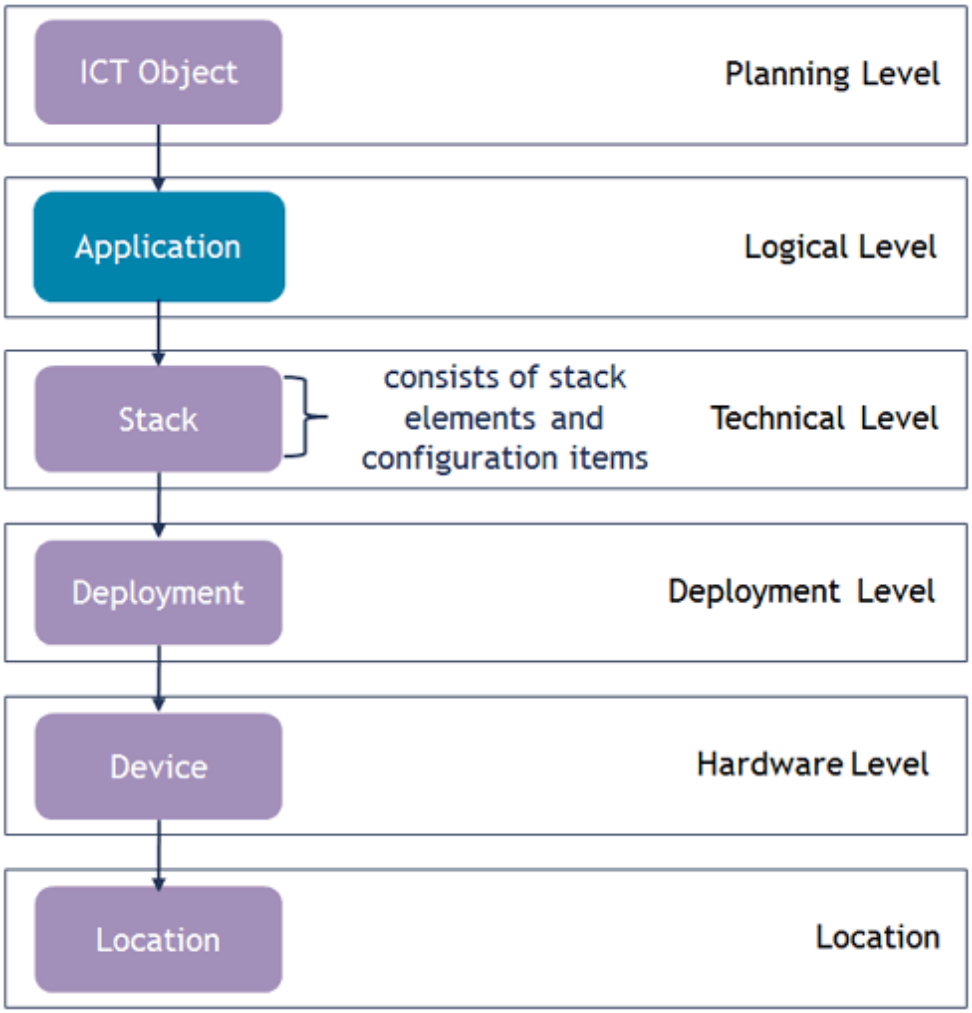
Not only are components implemented in platforms, a component itself, like an application, may also operate on a platform made up of other hardware and software components necessary to run the component. A platform describes the technical architecture in terms of the assembly of standard and local components that a specific application or component runs on. Every platform in Alfabet is based on a platform template organized in a matrix-like structure with platform layers and platform tiers defining the axes. The structure reflects the partitioning of the platform in terms of distribution across multiple hardware and software groups.



To support standardization, a standard platform may be embedded in the platform of an application or component. A standard platform is a configured platform made up of standard platform elements that reference standard components. Standard platforms allow you to bundle technology components in order to improve interoperability, performance, scalability, and reliability. Thus, standard platforms serve as a kind of blueprint architecture or architecture pattern in which platform configurations are used as building blocks to create and structure more complex standard platforms.

Standard platforms have versions and lifecycles and can even consist of other standard platforms. Standard components can be developed and managed independently of the standard platforms and applications that use them and the lifecycle of a standard component can be different from its lifecycle in the context of a standard platform or application platform.

Deployments in the Technical Architecture



In addition to specifying platforms, specification of the technical architecture also includes the infrastructure necessary to deploy applications, components, and standard platforms. A deployment specifies the logical set of installed elements that constitute the instantiation of an application, component, or standard platform. In the context of the deployment, stack elements specify the accepted guideline for the physical elements that are permissible and planned for the installation, and deployment elements are the actual operationally installed elements which usually are imported from a CMDB. The deployment element also identifies the device that the respective architecture element is installed on. The specification of networks and their routers ensures that the transfer of data between deployments is viable.

Operational costs of deployments can be captured in order to understand application costs. In order to capture cost information, you must have access to the Opex Optimization capability which is part of the Portfolio Management Advanced sales package. The various alternatives available to capture and plan operational costs is described in detail in the section *OPEX Optimization* in the reference manual *Portfolio Management Advanced*.

Understanding Governance and Responsibility for Components

A number of governance concepts are implemented in the Technology Architecture Definition capability:

- **Authorized User:** Each component has an authorized user. An authorized user has primary responsibility for the component and thus has Read/Write access permissions to it. Users may also be assigned to authorized user groups. All users assigned to an authorized user group that has been defined for a component will have Read/Write access permissions to the component.
- **Roles**: A role defines the functional relationship or responsibility that a user or organization has to a component. Roles describe responsibilities but they do not authorize access permissions to the component in Alfabet.
- **Object Class Stereotypes**: Object class stereotypes may be configured by your solution designer for the object class Component. This allows for a different governance approach between different kinds of components such as, for example, Software Components and Hardware Components. If object class stereotypes are configured for the object class Component, each stereotype may capture a specified set of attributes, reference data, and class configurations as well as implement a different governance approach.

  Please note that local components are an independent object class and do not inherit the object class stereotype of the component that they are based on.

- **Mandates**: Components may be managed in a federated architecture. Mandates are typically configured if object class stereotypes have been configured. By means of a federated architecture, it is possible to specify the visibility of individual components in the Alfabet interface for specific users.
- **Component Groups**: Components may be structured in one or more component groups. Each component group has an authorized user and may have one or more authorized user groups assigned to it. The authorized users of a component group will have access permissions to all components in the component group. A component may be assigned to multiple component groups.
- **Component Categories**: A component category bundles and classifies content-specific components and allows the components to be hierarchically structured. A component may be assigned to only one component category.
- **Component Catalogs**: Component catalogs allow the level of standardization of components to be assessed. A component catalog typically refers to functional, geographical, or organizational sub-entities. A component may be assigned to multiple component catalogs.

To capture component groups, component categories, and component catalogs, you must have access to the Technology Portfolio Governance capability which is part of the Portfolio Management Basic sales package.

Objects in Alfabet are managed by various access permission concepts. For an overview of the access permission and governance concepts in Alfabet, see the section *Understanding Access Permissions in Alfabet* in the reference manual *Getting Started with Alfabet*.

Documenting and Defining the Components in Your Enterprise

Standard components are captured in the *Document Components Functionality* and *Capture Components Functionality* functionalities. These views allow components to be documented and defined in a quick and efficient manner and are particularly useful for users with data entry responsibilities. All components created in these views are considered "standard" components. Standard components are simply referred to as "components" in the Alfabet user interface as well as the documentation.

Please note that local components constitute an independent object class. Object class stereotypes defined for the object class Component do not apply to local components. Object class stereotypes are not supported for local components.

If object class stereotypes have been configured by your solution designer for the class Component, you will first be asked to select the stereotype that the component is based on. The component is then created and defined by means of the **Component** editor.

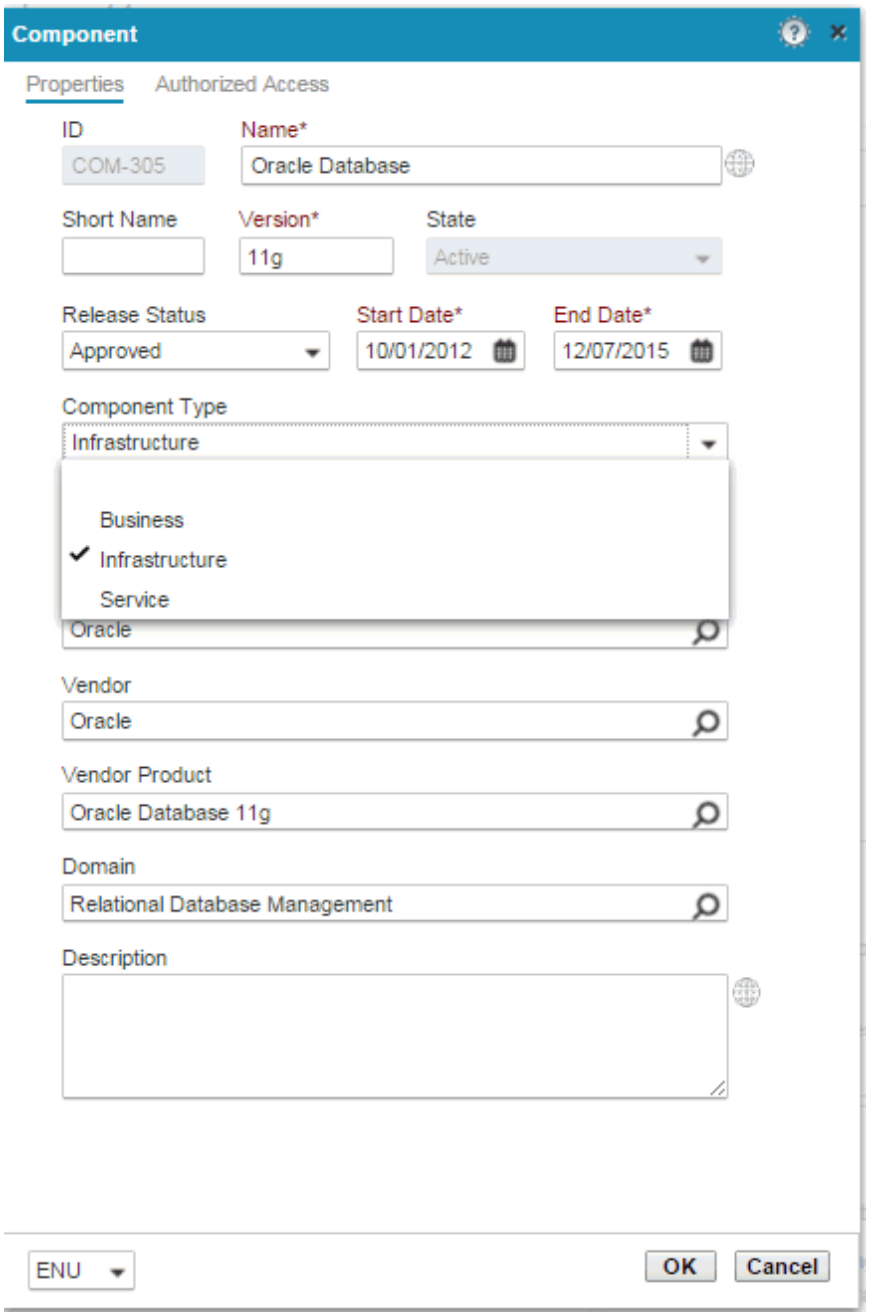


FIGURE: Creating a component in the Component editor

The following data is mandatory and must be defined when a component is created:

- Each component requires a unique name and version number for the component. Components can be versioned as needed. This is described in more detail in the section Versioning the Component.
- Each component requires planned start and end dates specifying when the component will be in production. The lifecycle phases of the component can then be defined later. This is described in more detail in the section Defining the Component's Lifecycle.
- Each component requires an object state. In the example above, a default object state has been configured that is automatically defined. This can be changed however via the **Change State** button available in the component's object profile/object cockpit.

- Each component requires a release status, which typically expresses agreement to the state of the documented information. In the example above, a default release status has been configured that is automatically defined. This can be changed in the editor.

The following information is optional:

- You should define a short abbreviated name to display on the component in diagrams and platform matrices.
- You may define a component type. Please note that if the component provides technical services, you must select **Service** in the **Component Type** field in order to document the technical services. Technical services allow the technical needs required to support the business services provided by an application, component to be documented. Technical services can be created for local components for which the **Component Type** attribute is set to **Service**. For more information about defining technical services, see the section Defining the Technical Services Delivered by the Component.
- You may specify which ICT object owns the component. For more information about capturing ICT objects, see the section Capturing ICT Objects to Understand IT Costs.
- If relevant, you may specify the vendor product that the component is derived from or associated with. For example, the components Oracle 11i Server, Oracle 11i Client, and Oracle 11i OCL may be derived from the vendor product Oracle RDBMS. You can also define the vendor that supplies the vendor product.

  To capture vendor products and vendors, you must have access to the Contract and Vendor Management capability which is part of the Portfolio Management Advanced sales package.

- If relevant, you may assign the component to a functional domain or business capability. From a functional point of view, the domain or capability is the owner of the component.

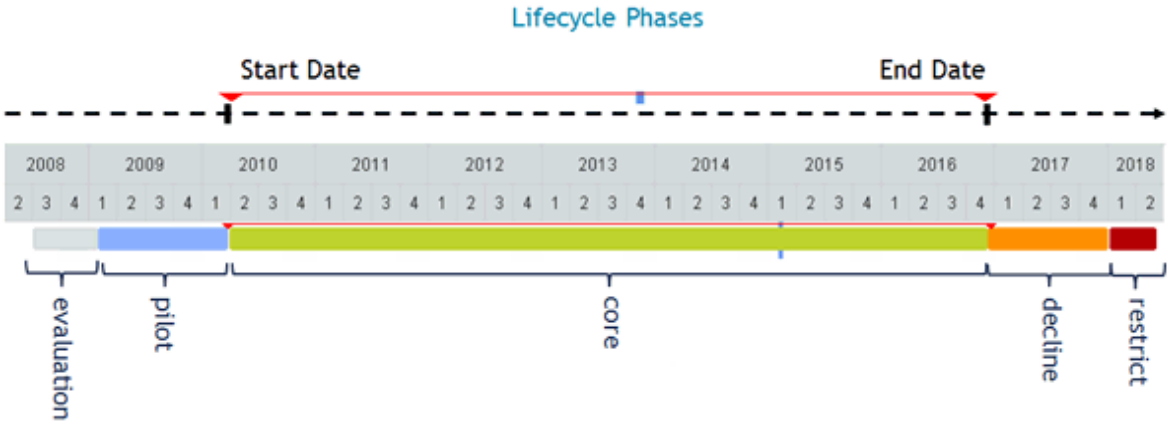
  

  To capture domains and capabilities, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

- You should provide a description of the component so that other users understand the purpose of the component.
- As the creator of the component, you are automatically defined as the authorized user per default. The authorized user of the component can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the component in the **Authorized Access** tab.

Defining the Component's Lifecycle

Component lifecycle management includes the process of identifying and managing conflicts in the lifecycles of a component and its component versions in order to ensure the availability and reliability of components in the enterprise. Each component must have a start and end date and an object state defined. The object state specifies the operational status of the component in the enterprise. The object state Active corresponds to the start and end dates of the component.



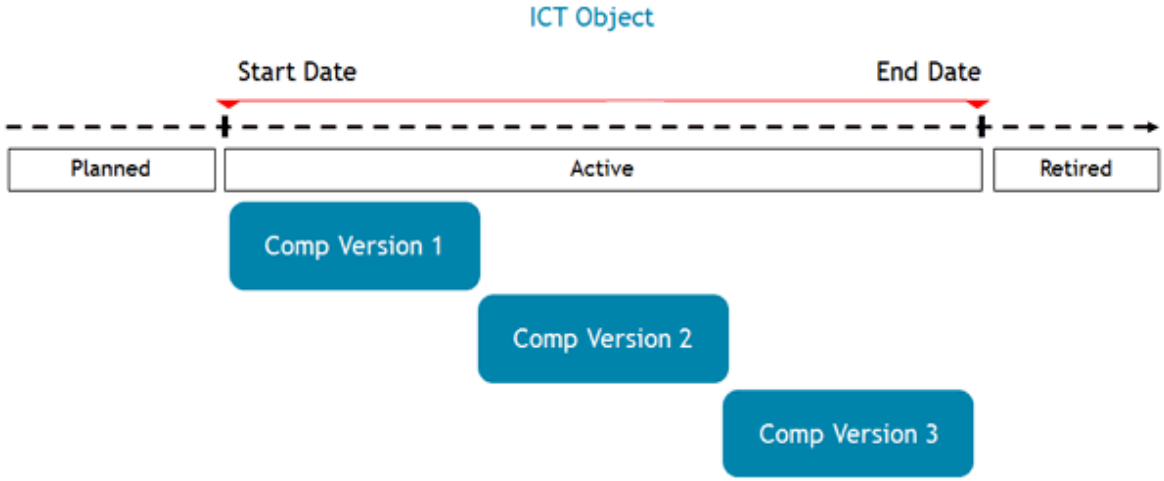
It is also possible to define a lifecycle made up of lifecycle phases, although this is optional. A component's lifecycle describes the succession of stages that it goes through. The lifecycle is comprised of lifecycle phases that describe the component's status of activity or production (such as Evaluation, Pilot, Core, Decline, Restrict). The lifecycle definition also includes the definition of a lifecycle phase that represents the component's active period, which corresponds to its start and end dates.

You can view and define the lifecycle of a component in the *Lifecycle Page View* of the relevant component. The *Lifecycle Page View* displays the lifecycle of the component you are working with as well as the lifecycle of the ICT object that owns the component and any other versions of the component. In this way, you can review the lifecycles of the component's versions and identify and manage any conflicts in these lifecycles.

In order to specify lifecycle phases and manage a component's lifecycle, you must have access to the IT Planning Basic sales package. The methodology and requirements to document and analyze lifecycles or architecture elements is described in detail in the section *Lifecycle Management* in the reference manual *IT Planning Basic*.

Versioning the Component

Versioning components describes the transition of one version of an component to the next from the enterprise architecture point of view. Each component that you define is actually a component version with its own defined lifecycle. The component may have predecessor and successor versions, thus providing information about the migration plans for the component.

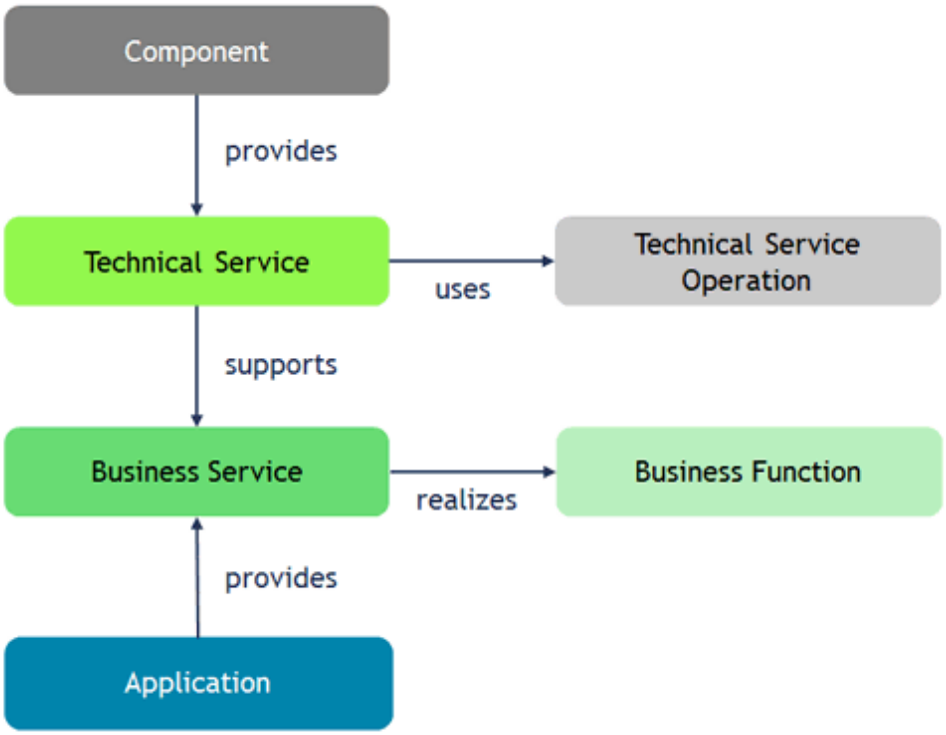


The component version will become the successor version when the selected component reaches its end date. The new component version's start date will be automatically defined to begin one day after the base component's end date and the new component version's end date will be defined to begin 5 years after its start date. These dates may be edited as needed. The component version will automatically be assigned to the same ICT object as the component that it is versioned from. The component version will not inherit the lifecycle of the base component. The lifecycle must be explicitly defined for the new component version. You can create a version of a specific component in the *Document Components Functionality* and *Capture Components Functionality* functionalities.

Defining the Technical Services Delivered by the Component

A technical service is a service provided by a component in order to fulfill technical needs that are necessary to support business services provided by the application or component. The technical service typically defines technical service operations that support the technical realization of the business functions supported by the business services.

Object class stereotypes may be configured for the classes **Technical Service**, **Technical Service Operation**, and **Technical Service Operation**. In this case, only certain technical service operations will be relevant for a particular technical service, and only certain technical service operation methods will be relevant for a particular technical service operation. For more information about the configuration and mapping of the technical service stereotypes, see the section *Configuring Object Class Stereotypes for Technical Services* in the reference manual *Configuring Alfabet with Alfabet Expand*.



Technical services can be defined for components and local components for which the **Component Type** attribute is set to **Service**. When a local component of the type **Service** is embedded in the architecture of an application or another component (of the type **Infrastructure** or **Business**, for example), the technical service operations of the embedded component can be mapped to the business services provided by the application or the local component.

There are a number of means to capture the technical services in your enterprise. You can either manually document them or import them via one of the following mechanisms:

- Manually create technical services for a standard component or local component for which the **Component Type** attribute is set to **Service**. The *Technical Services Page View* is only available in the object profile of the component/local component if the **Component Type** attribute is set to **Service**. When you define technical services for the component, you must define a name, version number and start and end dates of the technical service. By defining technical service operations for the technical service, you can detail how the technical service is to be provided by the component. Technical service operations are defined in the *Technical Service Operations Page View* of the technical service.

  The technical service operations can later be assigned to the business services that are provided by the application that is supported by the local component providing the technical service. In order to associate technical service operations with a business service, business services must first be defined. To capture business services, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

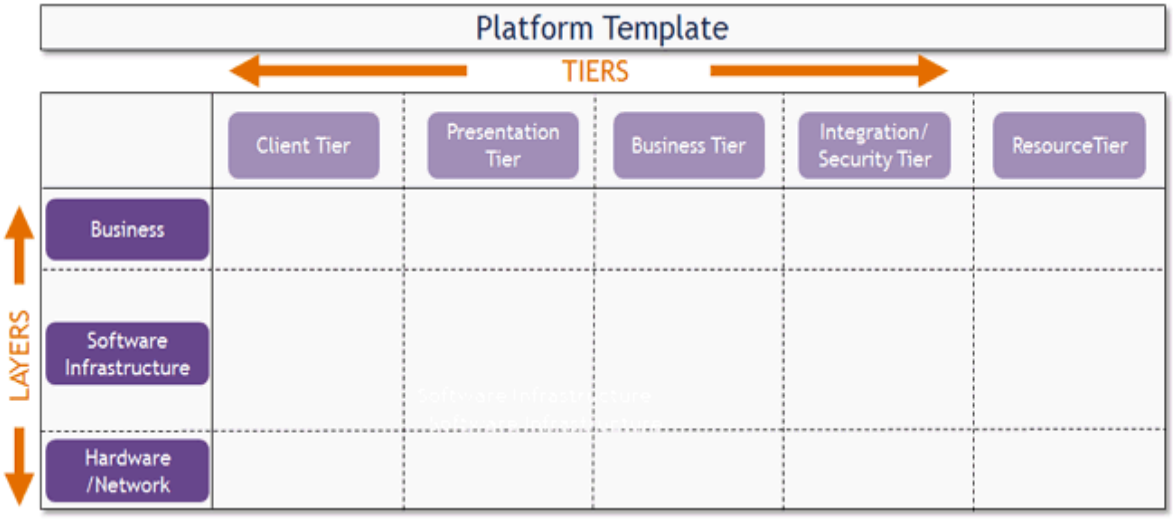
- Import technical services from assets in operational repositories such as CentraSite, webMethods API Portal, and webMethods API Gateway to the Alfabet database. Once the technical services are in the Alfabet database, they can be periodically updated and synchronized with the corresponding assets in the operational repository. Technical services can be created based on assets in a configured operational repository in the *Service Registry Services Page View*. When a technical

service is created based on an asset in an operational repository, an object of the class **Technical Service** will be in the Alfabet database. The name of the technical service created based on an asset in an operational repository will have the same name as the asset in the operational repository. All relevant properties available both in the asset and in Alfabet are automatically mapped. Interoperability with the operational repositories CentraSite, webMethods API Portal, and webMethods API Gateway will only be available if a valid configuration and connection is available. For more information about configuring interoperability with the relevant operational repositories, see the reference manual *API Integration with Third-Party Components*.

- Technical services may be created and managed based on WSDL files as well as OpenAPI Specification Swagger files in the *Technical Services Page View*.

Defining Platform Templates and Standard Platforms

The definition of platform templates and standard platforms allows you to ensure standardization in terms of the specification and documentation of platforms in the enterprise.



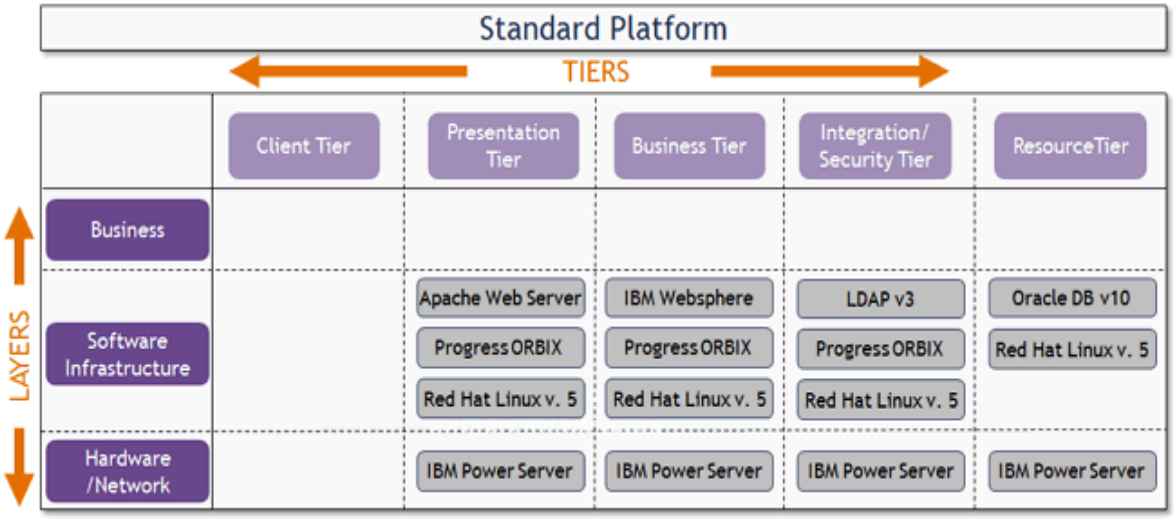
A platform template is organized in a matrix-like structure and defines a standard set of platform layers and platform tiers. It serves as a reference grid for placing components in standard platforms and platforms specified for applications and components. A platform template structures the platform in two dimensions:

- Platform tiers make up the horizontal axis and typically represent groups of components in terms of functionality that can or must be deployed on separate physical machines in order to accomplish better modularization and scalability. Usually, each platform tier communicates only with the tiers next to it. Typically, a four-tier architecture would comprise a database server tier, an application server tier, a Web server tier, and a client server tier.
- Platform layers make up the vertical axis and typically represent groups of components based on the level of abstraction of services that components within a layer provide. Usually, a platform layer depends on the layers below it. A typical layered architecture could comprise a business layer, where business logic resides, a software infrastructure layer, where operating system and infrastructure services reside, and a hardware layer.

A platform template may have as many tiers and layers as required in your enterprise. Multiple platform templates can be defined in your enterprise but only one platform template may be assigned to a standard platform or application/component platform. Platform templates are created and structured with platform

tiers and layers via the *Platform Templates Explorer*. One platform template can be defined as the default template. In this case, it will be automatically assigned to a new application platform. If necessary, the default platform template assigned to an application/component platform can be changed by the user defining the platform.

Once a platform template has been configured, you can establish blueprints for the technologies to be used in the enterprise's platform by creating standard platforms.



A standard platform is a platform that is defined outside of the scope of a concrete application or component. Much like any platform, the standard platform is based upon a platform template with its defined platform tiers and platform layers. Standard platforms can be used to represent technical platforms (for example, "Standard Windows Desktop") as well as business platforms (for example, "Standard Alfabet Platform" or "Standard SAP BW Platform").

Standard components are assigned to the standard platform and represent the technology components that constitute the platform. Each component embedded in the standard platform is considered a standard platform element. The lifecycle of the standard platform element can be different from its lifecycle as a standard component. 2015There are two methods to assemble a complex standard platform:

- Add individual standard platform elements assembled in an existing standard platform and/or individual components, thus structuring the selected standard platform component by component. In this case, the standard platform elements allow for a detailed description of the standard platform architecture. This method should be used to assemble the standard platform if the communication between the components is relevant.
- Add an existing standard platform as an entire package including all of its standard platform elements derived from components to the selected standard platform. This method should be used if all relevant communication between the components takes place within the standard platform. Please note however that only the first hierarchy level of standard platforms embedded in the selected standard platform will be visualized in the *Platform Diagrams Page View*. Any additional standard platforms embedded at lower levels in the platform hierarchy will not be added to the diagram.

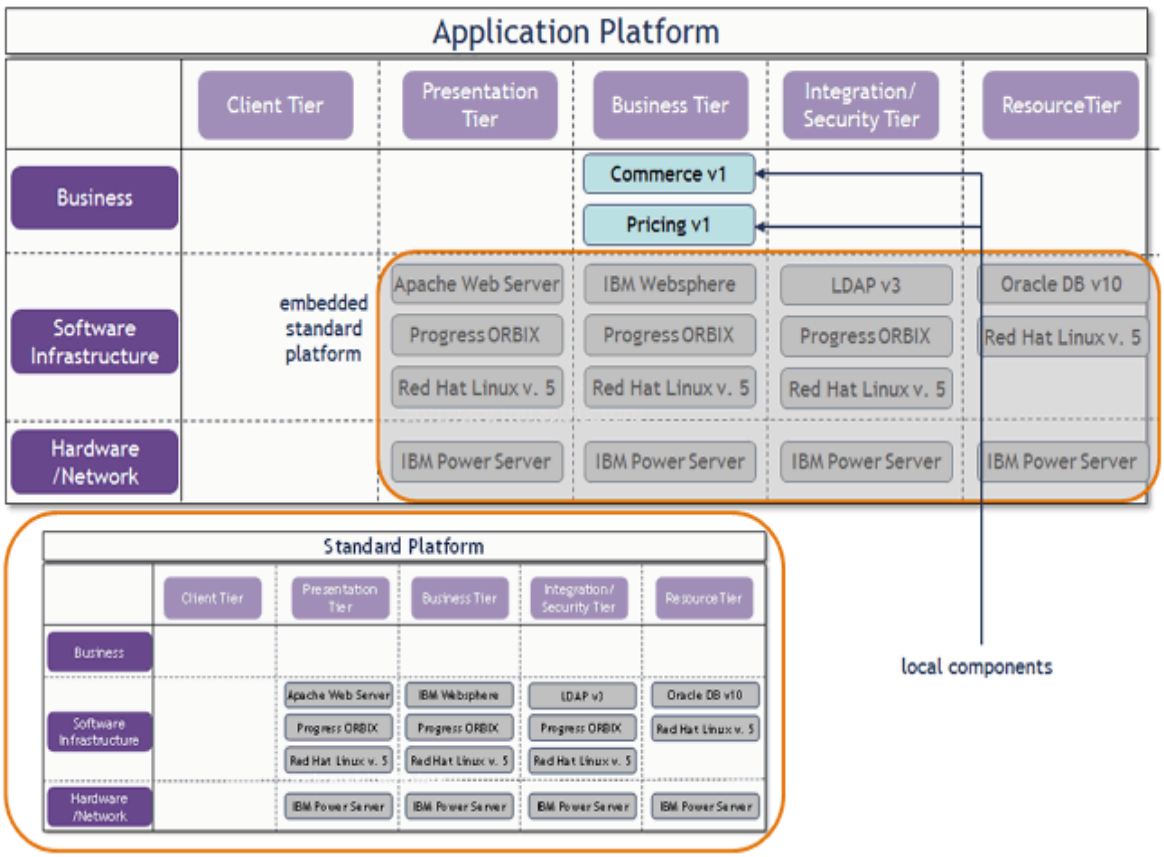
Once a standard platform has been defined, it can be embedded in an application/component platform, thus filling the platform with approved standard components. The standard components that are not relevant may be removed from the application/component platform. By means of standard platforms, you can assess the overlap and degree of standard compliance of application/component platforms with the company's guidelines.

The following steps are necessary in order to define a standard platform:

- The standard platform must first be created in the *Standard Platforms Page View* of a standard platform category. Please note that object class stereotypes may be configured for the object class Standard Platform in your enterprise. If standard platform stereotypes are configured for your enterprise, each standard platform stereotype may capture a specified set of attributes, reference data, and class configurations, and implement a different governance approach.
- The standard platform must then be assembled by bundling technology components in the *Standard Platform Elements Page View* of a standard platform.
- The standard platform can then be amended and further specified by adding or removing individual standard platform elements in the *Standard Platform Architecture Page View* of a standard platform.
- The *Standard Platform Usage Page View* provides an overview of the application platforms as well as the standard platforms in which a selected standard platform is embedded.

Defining the Platform Architecture for an Application or Component

A platform describes the technical architecture in terms of the assembly of local components that a specific application or component runs on. This assembly is based on a platform template and ideally has standard components from standard platforms incorporated in it.



A platform is created for an application or component in the *Technical Platform Elements Page*. If a default platform template has been specified in your enterprise, the application platform will be automatically created with the default platform template. Each component embedded in the application/component platform is considered a platform element. The lifecycle of the platform element can be different from its lifecycle as a standard component. Any qualitative aspect of local components as well as information about business services, information flows, or business data is disregarded in the context of a platform.

There are two methods to assemble a complex platform:

- Add individual platform elements assembled in an existing standard platform, thus structuring the selected platform component by component. In this case, the platform elements allow for a detailed description of the platform architecture. This method should be used to assemble the platform if the communication between the components is relevant.
- Add an existing standard platform as an entire package including all of its platform elements derived from components to the selected platform. You should only add standard platforms that are based on the same platform template. This method should be used if all relevant communication between the components takes place within the platform. Please note however that only the first hierarchy level of standard platforms embedded in the selected platform will be visualized in the *Platform Diagrams Page View*. Any additional standard platforms embedded at lower levels in the platform hierarchy will not be added to the diagram.

The standard components assigned to the application/component will be automatically displayed in the relevant platform tier and platform layer in the platform matrix displayed in the *Platform Architecture Page View*. In this view, you can assign additional components to the platform.

Defining the Operational Deployment of an Application, Component, or Standard Platform

Once applications, components, or standard platforms have been defined, you can capture the infrastructure necessary to deploy them. The deployment is the logical set of installed elements that constitute one instantiation of the application, component, or standard platform. It is possible to specify a simple deployment or, optionally, to define a detailed stack. Multiple stacks and deployments can be specified for each application, component, or standard platform.

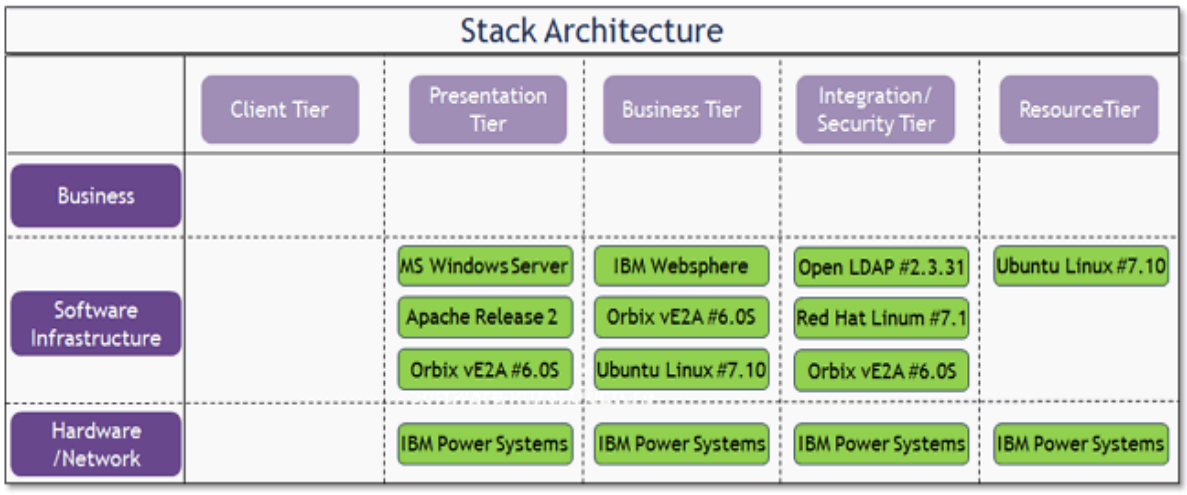
Please note that although stacks and deployments can be defined for applications, components, and standard platforms, for the sake of simplicity the following documentation will refer only to application and stacks and deployments. The process for creating stacks and deployments for components and standard platforms is similar to applications.

An application's stack is an operational refinement of that application and thus formulates the planned guideline that should be installed. As such, it typically describes a refinement of the application platform and the physically installed infrastructure for the application's physical deployment. In the case of components based on vendor products, the stacks could define patch level release versions provided by the vendor.

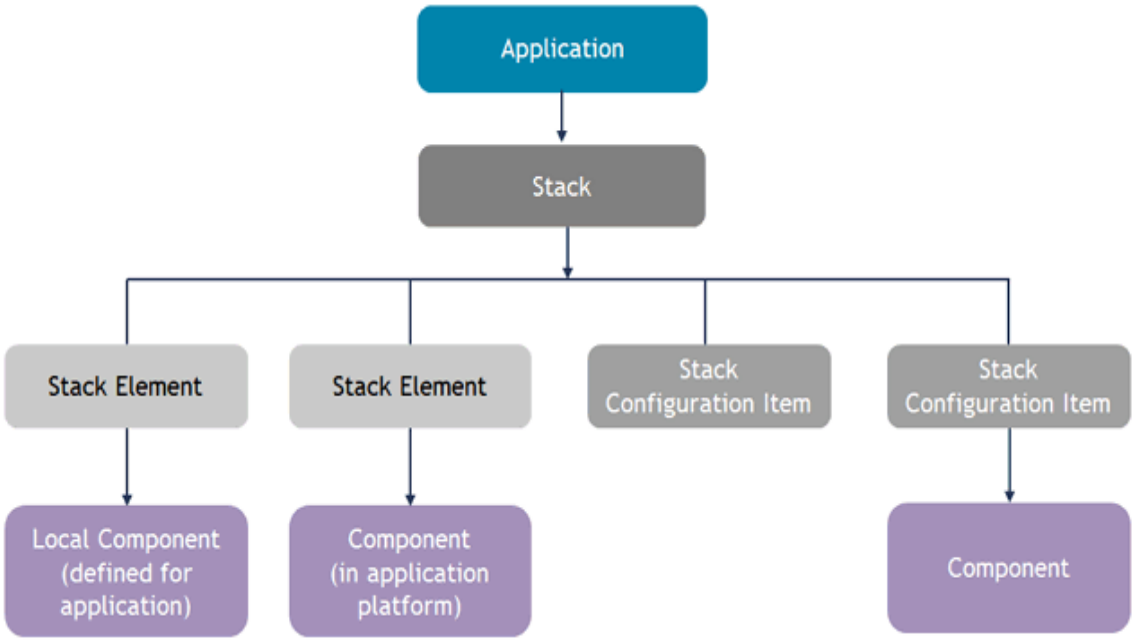
Stacks may be hierarchically organized and contain multiple sub-stacks, thus allowing for various levels of operationalization to be refined and further restricted. The stacks at the higher levels of the stack hierarchy may be more broadly defined and offer a number of options for the proposed deployment. Each level of sub-stack you define allows you to define the stack in more granularity, thus making the stack more concrete for the operational deployment. For example, a first-level stack could address operating system

compatibility between UNIX and Windows in general. The next level of stacks may differentiate between various UNIX dialects or operating system versions. Each stack in the hierarchy may include or exclude elements that have or have not been used in the platform.

A stack is based on the platform architecture of the application's platform. All platform elements in the platform architecture are added to the stack architecture.

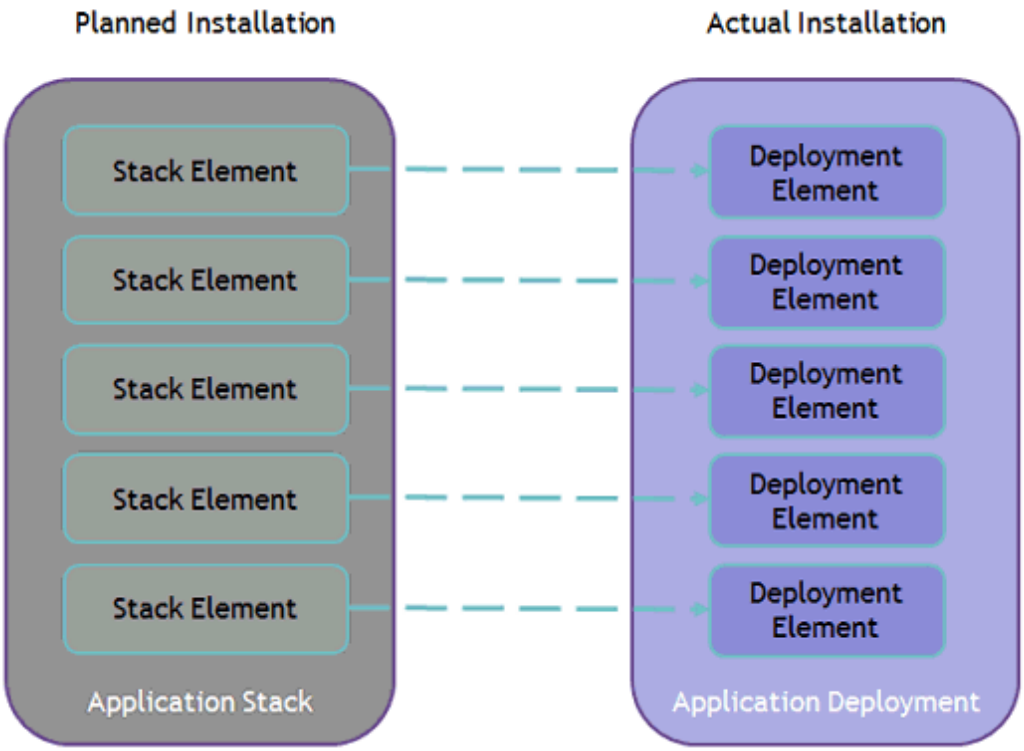


The stack is thus made up of stack elements that either reference the application's local components, the components in the application platform, or stack configuration items, which are typically technical aspects of the application deemed unimportant for strategic planning purposes but relevant for the physical installation. Stack configuration items themselves may reference a component. Typical examples for stack configuration items are install shields, installation scripts, job queues, etc.



A stack is created for the application in the *Stacks Page View*. When you define the stack, you must define the stack's name, release number, the object state of the stack, and the start and end dates. The stack will automatically include all platform tiers and platform layers defined for the application's platform as well as all platform elements. Each platform element has a corresponding stack element in the stack. You can

refine the stack by specifying the inclusion of stack elements and stack configuration items in the *Stack Elements Page View*. The *Stack Architecture Page View* displays the platform tiers and platform layers assigned to the stack as well as all stack elements and any stack configuration items that have been defined for the stack.



A deployment is the logical set of installed elements that constitutes one instantiation of the application. Once a stack has been defined, you can specify the deployment of the stack in the *Stack Deployments Page View* available for the stack or the *Deployments Page View* of the application being deployed.

Whereas stack elements specify the accepted guideline for the physical elements that are permissible and planned for the installation, deployment elements are the actual operationally-installed elements which usually are imported from a CMDB. Ideally, each stack element should be associated with one deployment element. The differentiation between stack elements and deployment elements allows the enterprise to understand whether the actual installation (via deployment elements) corresponds to the planned and approved installation (defined via stack elements). The deployment elements identify all components in the deployment and the devices that they are installed on.

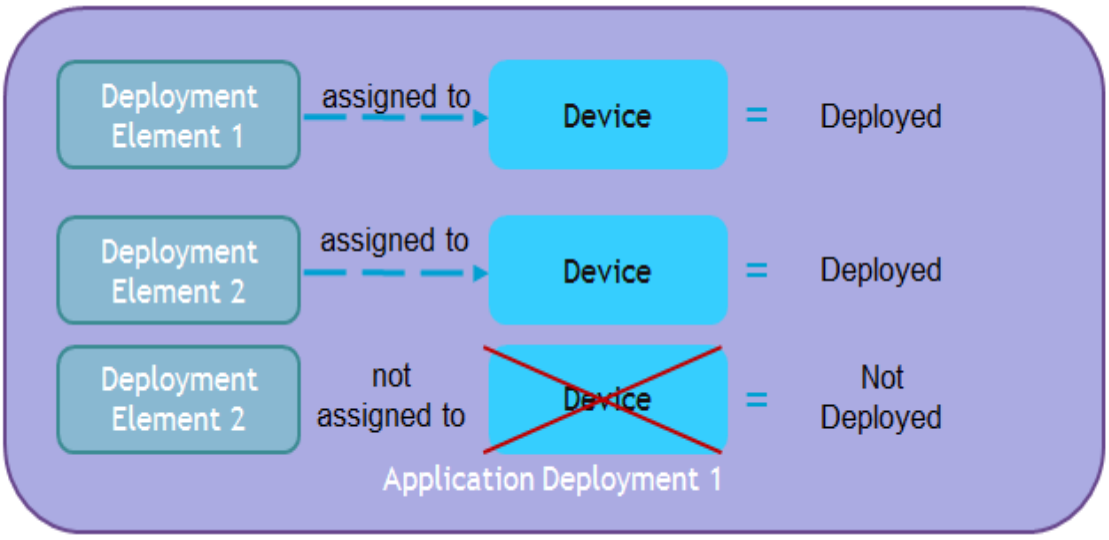
You can specify the details of the deployment in the *Deployment Structure Page View*. For an application deployment, for example, the following deployment elements are included in the view:

- the application being deployed
- the stack elements referencing local components and components assigned to the application platform
- the stack configuration items assigned to the stack

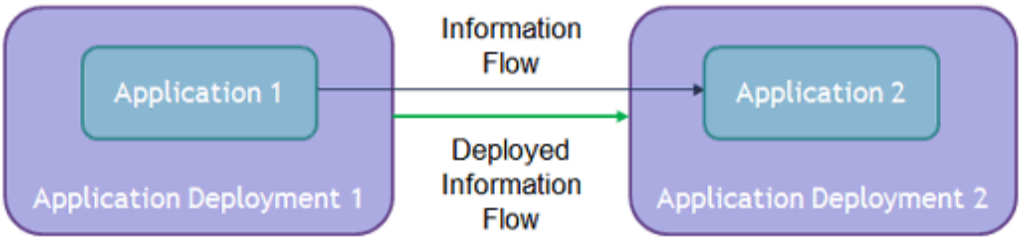
In the *Deployment Structure Page View*, you must explicitly deploy each deployment element that is relevant for the deployment. When you deploy a deployment element, you must specify the physical device

that it runs on. The device represents the piece of hardware that implements the application. Only physical devices can be assigned to deployment elements. Physical devices may be clustered via a logical device.

Devices are created in the *Document Devices Functionality* and *Capture Devices Functionality* functionalities. When you define the device, you must define the device's name, its version number, the object state of the device, and the start and end dates. You can also assign the device to an ICT object for budgetary and planning purposes as well as the location where the device is deployed. The devices are then assigned to specific locations in the enterprise. Locations must first be specified in the *Locations Explorer*.



Next you can capture and visualize information flows between the application deployments in order to better understand the physicality of data flows or where data is destined to be processed. The *Information Flows Page View* of the deployment displays all information flows and platform information flows defined for the deployed application. In order to ensure that the information flows are included in the deployment, you must explicitly deploy the relevant information flows that should be deployed for the application deployment.

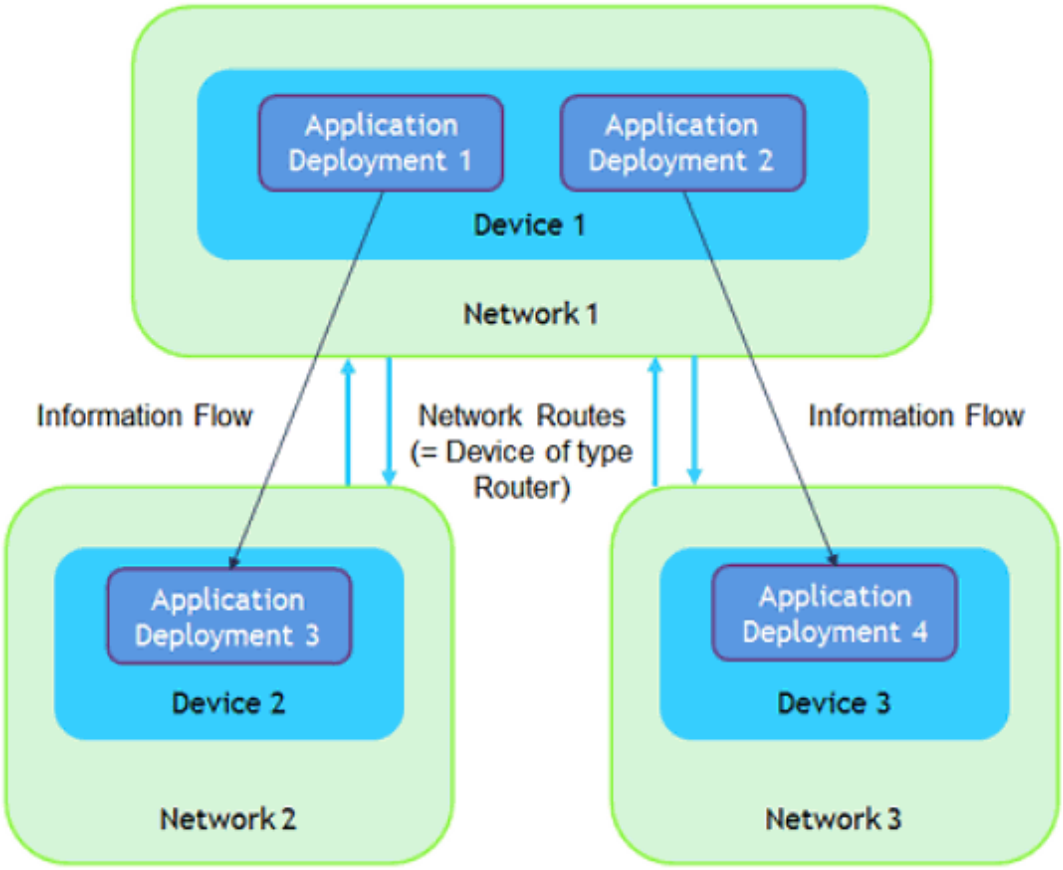


Please note that an application deployment must be defined for both the source and target applications of the information flow in order to deploy the respective information flow. All deployed information flows will be displayed in the *Deployment Diagrams Page View*. If your enterprise captures networks, you will be able to review whether the networks are connected via network routers so that the physical infrastructure is available in order to deploy the information flows. This is described in the section Defining and Managing Technical Networks and Deployments.

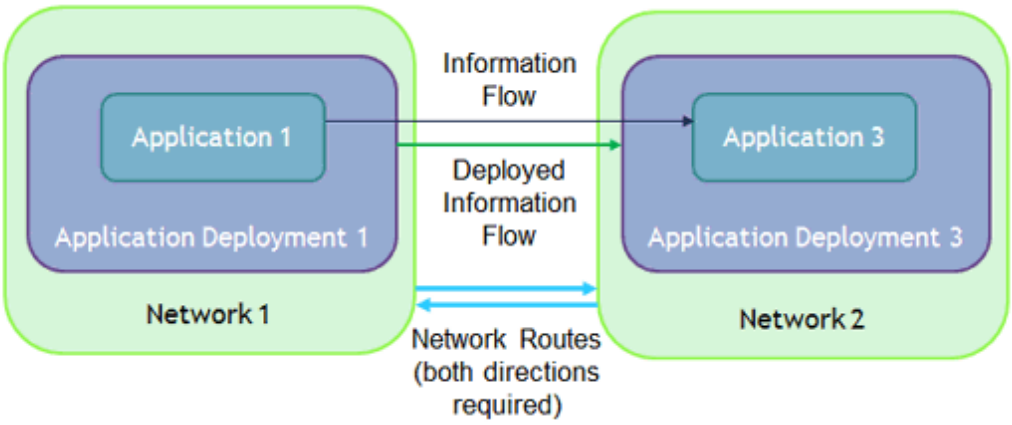
Defining and Managing Technical Networks and Deployments

Documenting and defining the technical networks in your enterprise will help you to achieve architectural efficiencies, strengthen application security, and build a foundation for IoT solution development. When capturing the infrastructure of your deployments, it is advisable to document and manage the technical networks as part of the application deployment strategy. This includes capturing the physical networks relevant for the enterprise and ensuring that the information flows between deployments are physically possible.

A network is the system of devices and other physical networks that are connected. For each network, subordinate networks can be defined as well as the network devices assigned to them. The definition of the network includes an object state, start date and end date, and the spatial scope of the network (such as WLAN, LAN, Internet, etc.). Networks are created in the context of the *Networks Explorer*. All devices that are relevant to the network should be assigned to the network in the *Network Devices Page View*



Network routes represent the communication between networks as well as networks and devices. The network routes are based on physical devices of the type **Router**. The routers must first be assigned to the network in the *Network Devices Page View* and then explicitly specified as the device providing the connection between the source network and target network/device in the *Network Routes Page View*. The allows you to review the overall network infrastructure. In the *Network Diagrams Page View*, red arrows represent the network routes that have been defined between networks as well as between networks and devices. The blue arrows visualize the assignment of devices to a network.



Finally, the *Deployment Diagrams Page View* available for a deployment allows you to check whether the network infrastructure supports the enterprise's deployments. All deployed information flows that are relevant to a deployment should have the same device defined to deploy their source and target objects as the device defined as the router for the network route. You can review whether network routes are available for the deployed information flows by executing the **Check Deployment Connectivity** function in the *Deployment Diagrams Page View*. The information flows between the deployments will be displayed green if network routes are defined and support the information flows. If no router connection is defined and the information flows cannot be deployed, the information flows will be highlighted red.

Chapter 5: Business Process Definition

Business processes represent a business-driven governance structure that supports the alignment of the business and IT as well as a means to analyze, manage, and plan the IT portfolio. Business processes describe how things are done in order to achieve the enterprise's business objectives.

Please note that a context-sensitive Help is available for each view available in the Business Process Definition capability. You should refer to the Help if you require an explanation about the functionalities and information available in a specific view.

The following information is available:

- Methodology: Understanding Business Process Models
- Understanding Governance and Responsibility for Business Process Models
- Defining the Business Process Models in Your Enterprise
- Documenting and Defining the Business Processes
- Specifying Business Process Model Variants

Methodology: Understanding Business Process Models

The business process model describes the hierarchy of business processes in the enterprise. A business process is a set of activities that represent work required to achieve a business objective. Typical business processes include marketing services, selling products, delivering services, distributing products, invoicing for services, and accounting for money received. A business process rarely operates in isolation. Other business processes will depend on it and it will depend on other business processes.

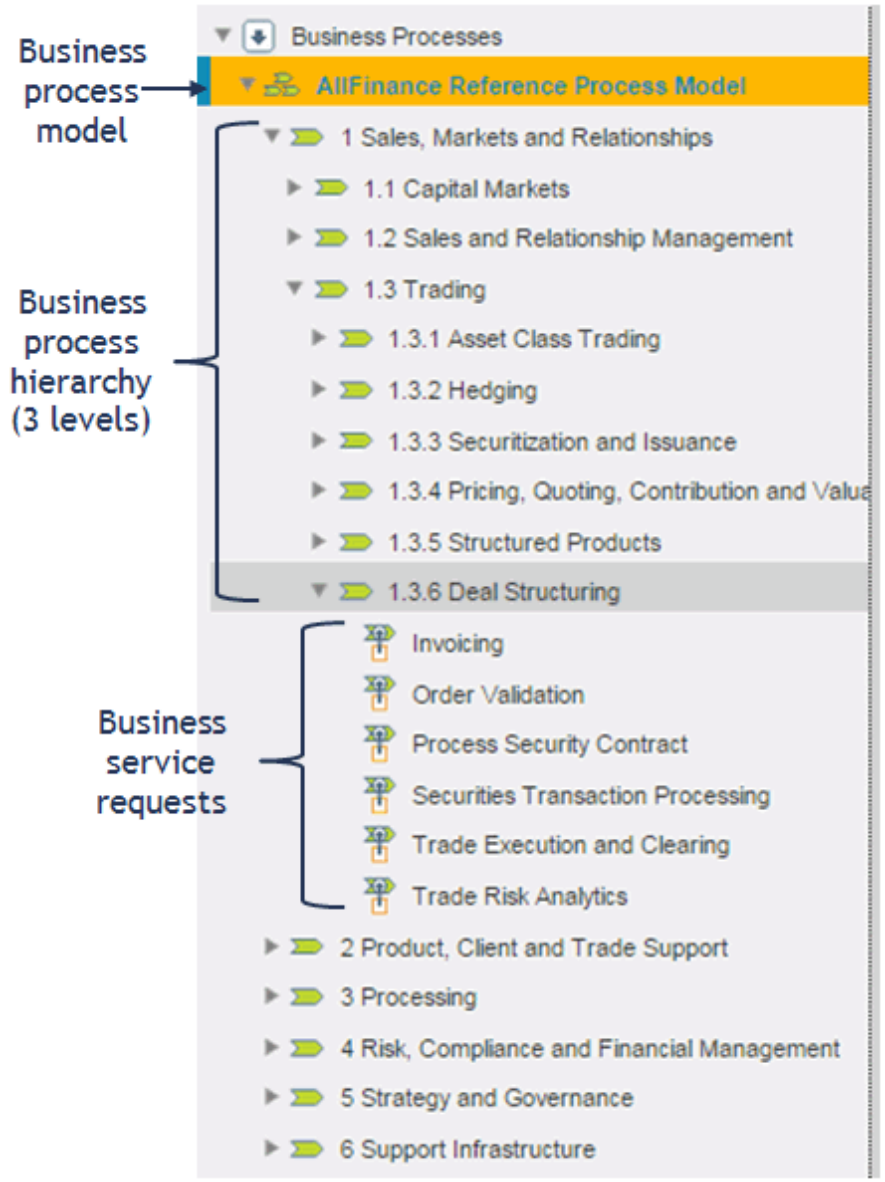


FIGURE: Example of a business process model in Alfabet

A business process may have as many levels of subordinate business processes as needed, although three levels should be sufficient in the business process hierarchy to describe the value chain. Business processes at the top level are primarily abstract and present an overview of the process hierarchy. The business process hierarchy becomes more detailed on the level of the subordinate business processes. At the lowest level, a business process may reference one or more business functions by means of business service requests. The alignment of the business and IT is typically planned at a specific level in the business process hierarchy (for example, at the third level in the business process hierarchy).

Each business process may have multiple business process variants, which allow specific stakeholders to analyze business process execution at a more granular level. For example, the high-level business process Sales Order Management can have business process variants defined for the specific process execution flows for the sectors Automobile and Motorcycle. It is recommended that the business process variants are defined for the leaf-level business processes in the business process hierarchy.

Multiple business process models may be defined to allow for variations over time or over organizational entities.

Please note the following:

- To capture business functions and business service requests, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.
- To plan changes to the business process model, you must work with solution business process models. A solution business process models allow you to plan changes by means of "shadow" objects - solution business processes that are copies of the real business processes. When the amended solution business process model is checked in to the inventory, it will overwrite the existing business process model. To work with solution business models, you must have access to the Scenario Management capability which is part of the IT Planning Complete sales package.

Understanding Governance and Responsibility for Business Process Models

A number of governance concepts are implemented in the Business Process Definition capability.

- **Authorized User:** Each business process model and business process has an authorized user. These may be different users. An authorized user has primary responsibility for the business process and thus has Read/Write access permissions to it. A business process variant inherits the access permissions of the business process that it is based on. Users may also be assigned to authorized user groups. All users assigned to an authorized user group that has been defined for a business process will have Read/Write access permissions to the business process.
- **Mandates**: Business process models and business processes may be managed in a federated architecture. Mandates are typically configured if object class stereotypes have been configured. By means of a federated architecture, it is possible to specify the visibility of individual business processes in the Alfabet interface for specific users.
- **Roles**: A role defines the functional relationship or responsibility that a user or organization has to a business process model or business process. Roles describe responsibilities but they do not authorize access permissions to the business process in Alfabet.

Objects in Alfabet are managed by various access permission concepts. For an overview of the access permission and governance concepts in Alfabet, see the section *Understanding Access Permissions in Alfabet* in the reference manual *Getting Started with Alfabet*.

Defining the Business Process Models in Your Enterprise

The business process model must first be created at the root node of the *Business Processes Explorer* before business processes can be captured. Multiple business process models may be defined to allow for variations over time or over organizational entities.

- You must define a unique name and version number for the business process model.

- You should provide a description of the business process model so that other users understand its purpose.
- As the creator of the business process model, you are automatically defined as the authorized user per default. The authorized user of the business process model can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the business process model in the **Authorized Access** tab.

Documenting and Defining the Business Processes

Business processes at the top level of the business model hierarchy are captured in the *Business Processes Page View* of the business model hierarchy. These business processes should be described broadly. The business process hierarchy becomes more detailed on the level of the subordinate business processes. Each subordinate business process is created in the *Sub-Processes Page View* of its parent business process. The business process is created and defined by means of the **Business Processes** editor.

- Each business process requires a unique name and a short name to display on the business process in diagrams and matrices.
- Each business process requires a number describing the level of the business process in the hierarchy. For an example, see the numbered hierarchy in the section Methodology: Understanding Business Process Models.
- You may assign the business process to a functional domain.

  To capture domains, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

- You should provide a description of the business process so that other users understand its purpose.
- As the creator of the business process, you are automatically defined as the authorized user per default. The authorized user of the business process can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the business process in the **Authorized Access** tab.
- Once business processes have been created you can specify the following:
  - Specify the applications that provide business support to the business process in the *Business Support Page View*.
  - Specify the business services that are requested by the business process to fulfill a business function in the*Business Service Requests Page View*.

    To capture business service requests, you must have access to the Business Capability Management capability which is part of the Portfolio Management Basic sales package.

Specifying Business Process Model Variants

Be specifying variants of business processes provide further analyses at a level more meaningful to specific organizations and stakeholders. For example, the business process Asset Class Trading can have business process variants defined for the specific business process execution flows for the sectors Product Management, FD Trading, and WP Investments. The business process landscape can be modeled each business process variant. The business process can be modeled in Alfabet diagrams or users can transverse from Alfabet to a specific EPC/BPMN diagram in ARIS or choose a diagram in ARIS and link to the business process variant in Alfabet to study the underlying IT. To access ARIS diagrams from Alfabet, your enterprise must have a licence for ARIS - Alfabet Interoperability Interface.

Business process variants can be created for a business process in the *Business Process Variants Page View*. A name, short name, and description as well as business services can be defined for each business process variant. It is recommended that the business process variants are defined for the leaf-level business processes in the business process hierarchy.

In order for a user responsible for a business process to be able to articulate if improvement is needed for the business process, business process gaps can be captured for the business service requests associated with a business process. Such gaps can be effectively introduced into the IT planning and portfolio management process by deriving one or more demands from the gap. To capture business process gaps, you must have access to the Demand Management capability which is part of the IT Planning Advanced sales package.

Chapter 6: Organization Definition

The organization hierarchy represents a business-driven govenernance structure that helps the enterprise to document responsibilities and ownership, report on budgets, and analyze, manage, and plan the IT portfolio.

An organization describes an administrative or functional unit in the enterprise. Organizations form a self-referential hierarchy. An organization may have an unlimited number of subordinate organizations but only one ascendant organization. An organization may have primary responsibility for an object in the same way that an authorized user has or it may by defined as having an important role in relation to an object. Organizations are supported in their business activities through the business support and business services provided by applications.

Object class stereotypes may be configured for the class **Organization**.

Please note that a context-sensitive Help is available for each view available in the Organization Definition capability. You should refer to the Help if you require an explanation about the functionalities and information available in a specific view.

The following information is available:

- Understanding Governance and Responsibility for Organizations
- Capturing Organizations

Understanding Governance and Responsibility for Organizations

A number of governance concepts are implemented for organizations:

- **Authorized User:** Each organization has an authorized user. An authorized user has primary responsibility for the organization and thus has Read/Write access permissions to it. Users may also be assigned to authorized user groups. All users assigned to an authorized user group that has been defined for an organization will have Read/Write access permissions to the organization.
- **Object Class Stereotypes**: Object class stereotypes may be configured by your solution designer for the object class Organization. If organization stereotypes are configured for your enterprise, each organization stereotype may capture a specified set of attributes, reference data, and class configurations, and implement a different governance approach.
- **Mandates**: Organizations may be managed in a federated architecture. Mandates are typically configured if object class stereotypes have been configured. By means of a federated architecture, it is possible to specify the visibility of individual organizations in the Alfabet interface for specific users.
- **Organization Groups**: Organizations may be structured in one or more organization groups. Each organization group has an authorized user. The authorized users of a organization group will have access permissions to all organizations in the component group. An organization may be assigned to multiple organization groups.

To capture organization groups, you must have access to the Project and Release Design capability which is part of the IT Planning Complete sales package.

Please note that an organization can be configured to have a role for an object. Roles define the functional relationship or responsibility that a user or organization has to an object. Roles describe responsibilities but they do not authorize access permissions to the objects in Alfabet.

Objects in Alfabet are managed by various access permission concepts. For an overview of the access permission and governance concepts in Alfabet, see the section *Understanding Access Permissions in Alfabet* in the reference manual *Getting Started with Alfabet*.

Capturing Organizations

An organization may have as many levels of subordinateorganizations as needed, although three levels should be sufficient in the organization hierarchy to provide sufficient information for data analysis and. Each branch of the organization hierarchy should be defined down to the same level of depth to allow for comparisons and analyses of the organizations. You can create an organization at the root level of the **Organizations** explorer. These are root organizations for which you can create an unlimited number of subordinate organizations. Any organization, whether a root or descendant organization, may contain an unlimited number of sub-organizations.

An organization at the top level of the organization hierarchy is created in the *Organizations Explorer*. Each subordinate organization is created in the *Subordinate Organizations Page View* of its parent organization.

- You must define a unique name and a short name to display on the organization in diagrams and matrices.
- You may provide the name of a contact person for the organization.
- You should provide a description of the organization so that other users understand its purpose.
- As the creator of the organization, you are automatically defined as the authorized user per default. The authorized user of the organization can be changed in the **Authorized Access** tab. You can also define any user groups that should have Read/Write access permissions to the organization in the **Authorized Access** tab.

Once the organizations are defined, you can:

- Specify which ICT objects are owned by the organization in the *ICT Objects Page View*.
- Specify the business processes executed by the organization in the *Business Processes Page View*.

Alfabet also allows virtual organizations to be captured. A virtual organization is either a temporary or permanent organization such as a decision board, steering committee, or other bodies that exists outside the formal organizational structure of an enterprise. To capture virtual organizations, you must have access to the Operational Model Planning capability which is part of the IT Planning Complete sales package.

