HOPEX System Oriented IT Architecture

User Guide

HOPEX V2

INTRODUCTION

HOPEX System Oriented IT Architecture is a tool published by **MEGA International** enabling enterprises and other organizations to represent and document their IT architectures according to a service-oriented architecture.

The purpose of this guide is therefore to present how to make best use of these functionalities for the successful evolution of your information system.

CONNECTING TO HOPEX SYSTEM ORIENTED IT ARCHITECTURE

To connect to **HOPEX System Oriented IT Architecture**, see **HOPEX Common Features**, "HOPEX Web Front-End Desktop".

ABOUT THIS GUIDE

This guide presents how to make best use of **HOPEX System Oriented IT Architecture** to describe the IT architecture of your enterprise by adopting a service oriented approach.

> ☛ *Differences between **MEGA Windows Front-End and MEGA Web Front-End are indicated where appropriate for each functionality.***

Guide Structure

The **HOPEX System Oriented IT Architecture** guide comprises the following chapters:

- "Concepts Overview", page 5 presents a summary of concepts used by **HOPEX System Oriented IT Architecture**.
- "Modeling with HOPEX System Oriented IT Architecture", page 25 explains how to represent the architecture of your enterprise.
- "Modeling with Service Design", page 51 presents how to specify exchange contracts that manage exchanges between components of your IT architecture.
- "Using HOPEX System Oriented IT Architecture Standard Reports", page 63, shows how **HOPEX System Oriented IT Architecture** enables analysis of relationships between components of your enterprise.

Additional Resources

This guide is supplemented by:

- **HOPEX Common Features** guide describes the basic functions common to **HOPEX** products and solutions.

  > ☛ *It can be useful to consult this guide for a general presentation of the interface.*
- the **HOPEX Power Supervisor** administration guide.
- More advanced technical functions are described in the **HOPEX Power Studio** guide.

CONCEPTS OVERVIEW

HOPEX System Oriented IT Architecture is a service-oriented architecture tool designed to facilitate the analysis of communications within your IT architectures. Architecture description is based on specific concepts that enable a more generic use of the tool.

EXAMPLES OF USE

The functionalities proposed by **HOPEX System Oriented IT Architecture** for modeling service oriented architectures are used to represent:

- equipment and organizational components required for IT system operation
- interactions between components
- services offered and used within the modeled architecture

The components described can follow a modeling methodology:

- ascending, from detailed to conceptual
- descending, from conceptual to detailed

Presentation of these functionalities is based on the example of an application system of purchasing processing, presented using a descending approach, from diagrams of:

- application system environment,
- application system structure,
- application structure,
- logical application system environment,
- logical application system structure.

Application System Structure Example

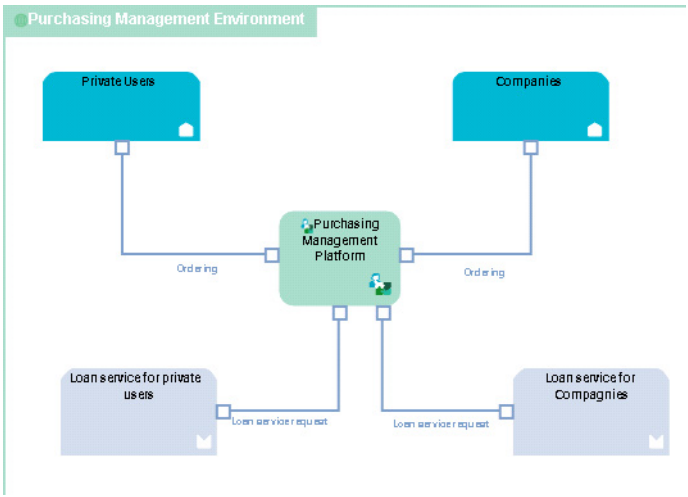
An application system is an assembly of other application systems, applications and end users interacting with application components to implement one or several functions.

Application system environment diagram

An application system environment present an application system use context. It describes the interactions, between the application system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.

The components of an *application system environment* are presented in an application system environment diagram that describes the interactions between the application described, its users and its partner application systems.

The following diagram describes the application system environment corresponding to purchasing request processing.



Environment diagram for the "Purchasing Requests Processing" application system.

```
Purchase requests are formulated by private users or by
companies in different contractual conditions.

The "Purchasing Requests Processing" application system
offers a loan service to its clients within the context of
payment management.
```

The diagram includes:

- An *application system use* element that represents the application system internal to the environment.

  > *An application system use describes the role of an application system in a composition relationship with a parent application system or in an application system environment.*

- Two *partner application systems* that represent the application systems used within the context of the described application system environment.

  > *A partner application system is an application system external to the environment of the described application service. The partner application system can be a service supplier or a service consumer with respect to application system users.*

- Two *system user* components that represent the user category of services provided by the environment.

  > *A system user represents an organizational unit interacting with the boundaries of an application system environment.*

- *Interactions* between the components representing requests for services.

  > *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

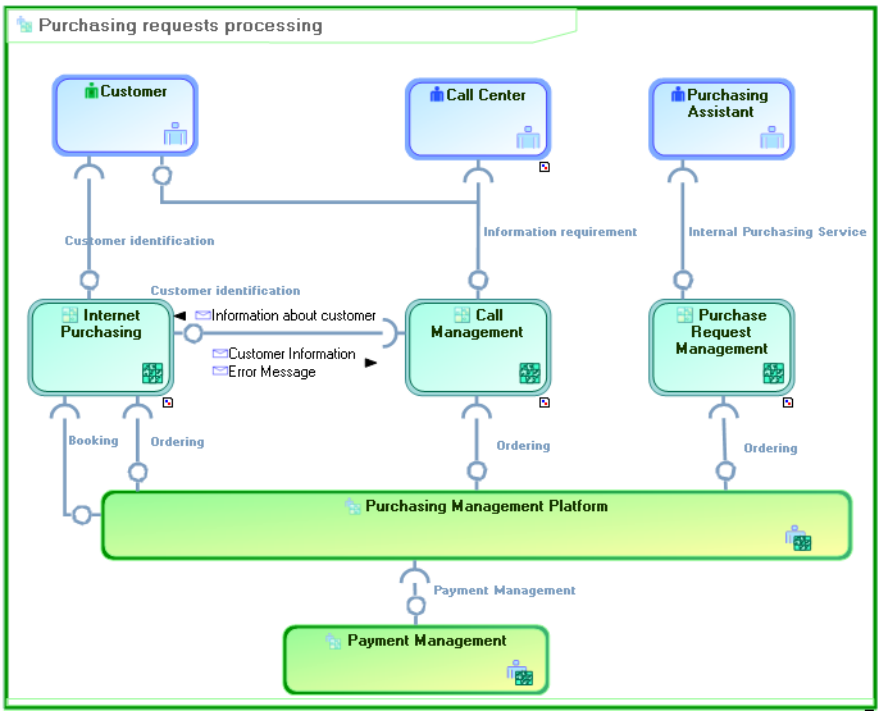
Application system structure diagram

An application system is an assembly of other application systems, applications and end users interacting with application components to implement one or several functions.

Components of the *application system* are described in an application system structure diagram, which describes the internal structure of the application system:

- services offered or required
- processes handled
- components and their interactions
- end users interacting with the application components
- data stores

The following diagram describes the application system corresponding to purchasing requests processing.



The following diagram describes the application system corresponding to purchasing requests processing.

Purchasing requests can be formulated by customers directly via an Internet purchasing application, or indirectly via a call center. Internal purchasing requests are processed by a Purchasing Assistant.

For example, the purchasing requests processing architecture uses the application system services of "Purchasing Management Platform" and "Payment Management".

The diagram includes:

- Two elements of *application system use* type representing the two application systems used.

  > 🕮 *An application system use describes the role of an application system in a composition relationship with a parent application system or in an application system environment.*

- Three *application components* representing the applications used in the context of the described application system.

  > 🕮 *An application service component represents the fact that the application system used plays a role in the described application.*

- Three components of *end user* type representing the categories of participants in the described application system.

  > 🕮 *The end user represents an organizational unit interacting at the boundaries of an application system or a logical application system.*

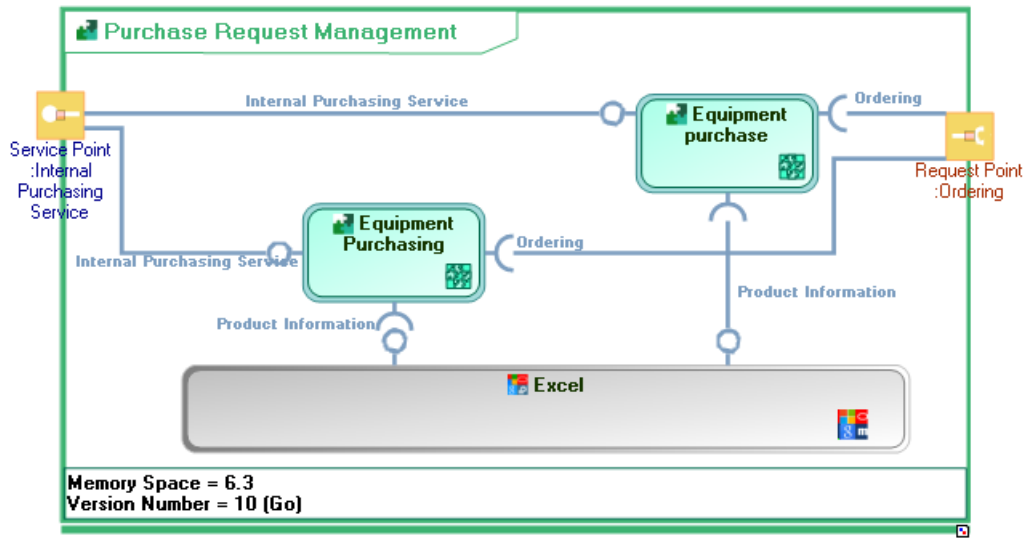
- *Interactions* between the components representing requests for services.

  > 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Application structure diagram

From an application you can access the application structure diagram describing the main elements assuring operation of the application.

An application structure diagram graphically shows first level components of an application, the access points (service point and request point) and the connections between components.



"Purchase Request Management" application structure diagram

```
The purchase request management application, which is used
only for internal purchases, is based on two specialized
applications: one for office supplies and the other for
equipment. Both applications use Microsoft Excel.
```

☛ *The version number that appears at the bottom of the frame is a property of the application. For more details, see "Described element properties", page 20.*

This diagram includes the following elements:

- Two *application components* representing the applications used

  An application service component represents the fact that the application system used plays a role in the described application.

- A *platform application component* associated with the Excel application in the example

  A platform application component represents the fact that the technology used plays a role of platform in the described application. For example, Tomcat is a platform component for SageCRM.

- *Interactions* between the components representing requests for services. Note that service point "Internal purchasing service" is activated. "Order Access" activates the ordering service when purchasing requests have been validated.

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an

enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

Logical Application System Example

📖 *A logical application system is an assembly of other logical application systems, logical applications and end users interacting with application components to implement one or several functions.*

📖 *A logical application is a component of an application that is available to the end user of this application in the context of his/her work. Logical applications represent the split of an application into its basic functional units. A logical application is a consistent, indivisible unit of processing that coordinates a set of messages and events in order to perform a task in the information system. In the application system, the logical application is the most basic functional unit.*

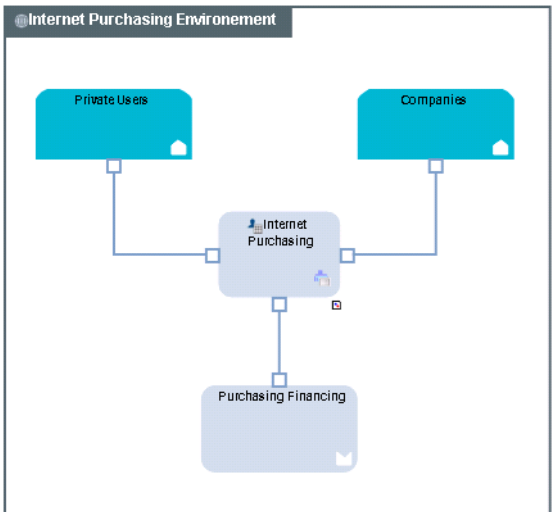
Logical application system environment diagram

📖 *A logical application system environment presents a logical application system use context. It describes the interactions between the logical application system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.*

The components of a *logical application system environment* are presented in an application system environment diagram that describes the interactions between the internal logical application systems, the users and the partner logical application systems.

The following diagram describes the logical application system environment corresponding to Internet purchases.

s



Environment diagram if the logical application system "Internet Purchasing"

```
Purchases are offered to private users and companies.

The "Internet Purchasing" application system uses a
"Purchasing Financing" logical application system that is
external to the described environment.
```

The diagram includes:

- An *application system component* element that represents the application system internal to the environment.

  > 🕮 *A logical application service component represents the fact that the logical application system used plays a role in the described environment.*

- A *partner logical system* that represents a logical application system used in the context of the described environment.

  > 🕮 *A partner logical system is a logical application system external to the environment of the described logical application system. The partner logical system can be a service supplier or a service consumer with respect to components of the logical application system.*

- Two *system user* components that represent the user category of services provided by the environment.

  > 🕮 *A system user represents an organizational unit interacting with the boundaries of an application system environment or a logical application system environment.*

- *Interactions* between the components representing requests for services.

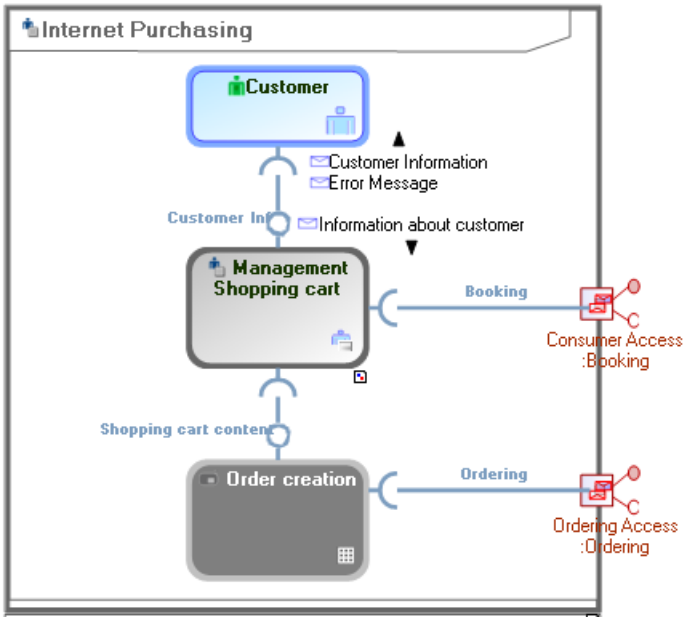
  > 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Logical application system structure diagram

The components of the *logical application system* are described in an application system structure diagram, which presents:

- services offered or required
- processes handled, components and interactions
- end users interacting with the application components

The following diagram describes the structure of the Internet purchasing logical application system proposed to customers.

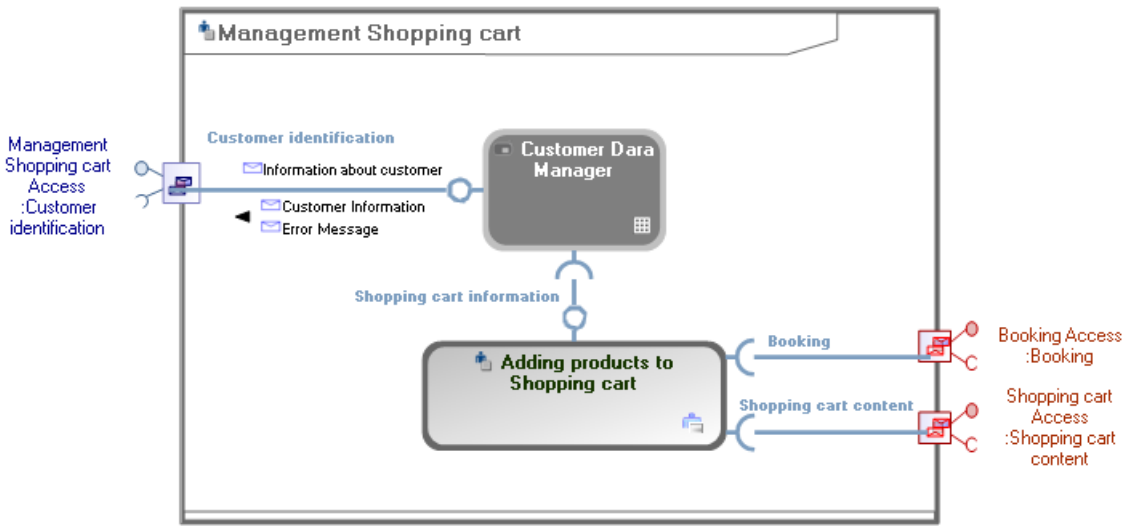


"Internet Purchasing" logical application system structure diagram

Requests made by customers are processed by a "Management Shopping Cart" logical application system. The "Order

Creation" logical application is then used in the context of this architecture.

The logical application system structure diagram, responsible for customer shopping cart management, presents two request points for "Booking" or "Ordering".



"Management Shopping Cart" logical application system structure diagram

These diagrams include:

- A component of *end user* type in this case representing customers.

  The end user represents an organizational unit interacting at the boundaries of an application system or a logical application system.

- Two *logical application system components* that represent the logical application systems used.

  The use of a logical application system describes the role of a logical application system in a composition relationship with a parent application system or in a logical application system environment.

- Tee *logical application components* representing the logical applications used in the context of the described logical application system.

  A logical application component describes the role of a logical application in a composition relationship with a parent logical application or a logical application system.

- *Interactions* between the components representing requests for services.

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

The data store concept

A data store provides a mechanism to update or consult data that will persist beyond the scope of the current process. It enables storage of input message flows, and their retransmission via one or several output message flows.

Introduction to the data store concept

Depending on the context being described, **HOPEX System Oriented IT Architecture** offers the use of RE or RDB data stores.

Relational Database data store represents a data store in the form of tables or table views.

A Relationship Entity data store describes a database in the form of entities or classes.

A data store references an **data domain**.

A physical RDB domain is held by a database. An RDB information storage structure is held by a package.

If you describe an application system, only physical data stores can be used.

A physical data store represents the implementation of a logical data store.

For more details, see "Describing physical data stores", page 41.

If you describe a logical application system, only physical data stores can be used.

A logical data store represents the use of data via application systems without considering how their access will be concretely implemented.

For more details, see "Describing a logical data store", page 42.

Last but not least, **HOPEX System Oriented IT Architecture** is used to distinguish data stores local to a system from external data stores that are positioned on the border of diagrams.

A local data store represents a data store used only inside the system described.

An external data store represents a data store used inside and outside of the system described.

Usage contexts

Data stores can be described in the following diagrams:

- application system structure diagrams
- application structure diagrams
- Logical application system structure diagrams

For more details, see "Using Data Stores", page 41.

Exchange Contract and Interaction Operation Example

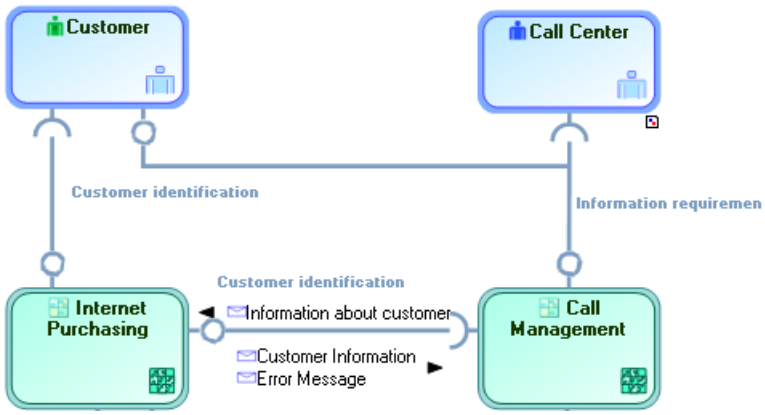
An *Interaction* represents the exchange of information between architecture components.

> 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Content of an interaction is described by an *exchange contract*.

> 🕮 *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*

In the "Purchasing Requests Processing" application system structure diagram, two interactions are used.



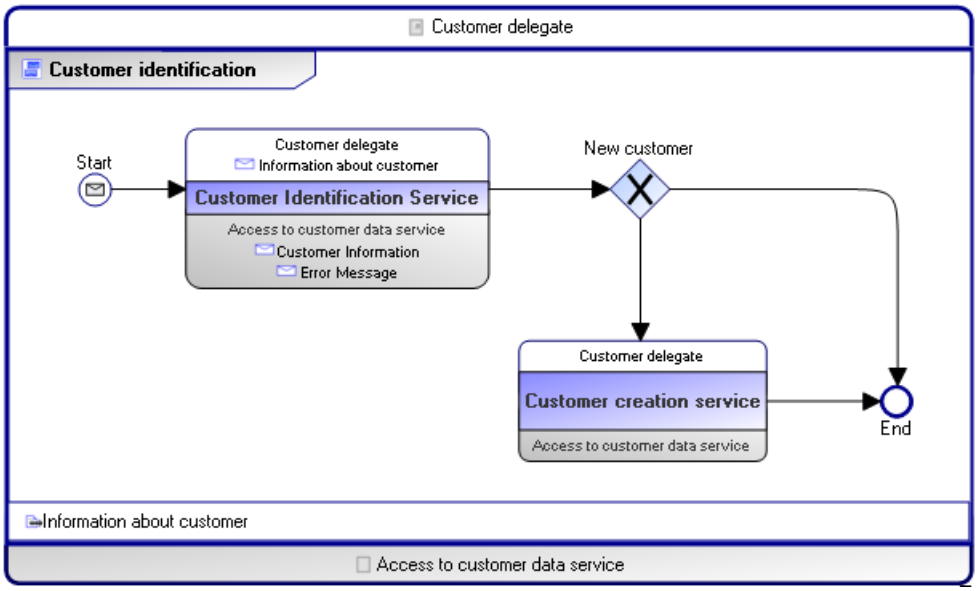
Interactions in the "Purchasing Requests Processing" application system structure diagram

Customers can place orders through an Internet application or a call center, which itself uses the Internet application to obtain the customer identification service.

A customer passing via the call center can make other requests in addition to orders.

Exchange contract diagram (BPMN)

The exchange contract diagram associated with the customer identification exchange contract describes, in BPMN formalism, the operations executed.



Exchange Contract Diagram (BPMN) "Customer Identification"

```
Customer identification protocol starts with a customer
identification step. If the customer is found the exchange
contract returns customer information, if not, a customer
creation exchange contract is activated.
```

Progress steps are represented by *exchange uses*.

Exchange diagram

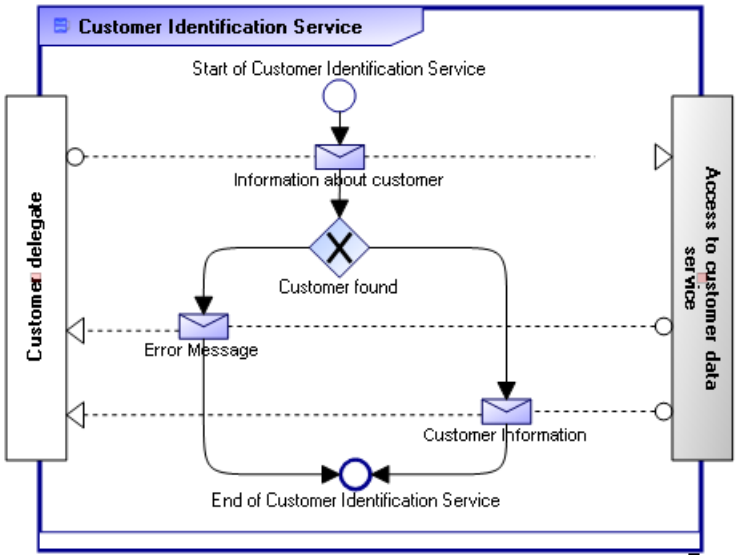
With the **MEGA Service Design** option, an *exchange use* is associated with an *exchange*.

An exchange use represents the usage of an exchange in another exchange contract.

An exchange specifies message flow exchanges between two participants.

An *exchange* is described by an exchange diagram presenting the sequence flow of messages exchanged.

> *An exchange specifies message flow exchanges between two participants.*



"Customer Identification Service" Exchange Diagram

The customer identification service protocol begins by sending information enabling identification of the customer. An error message appears if the customer is not found, otherwise customer information is sent (customer identification, status of orders, etc.).

Advanced communication exchange contract example

With the **MEGA Service Design** option, an exchange contract is described by a sequence flow of steps which are represented:

- by *exchange contract uses*
- by *exchange uses*

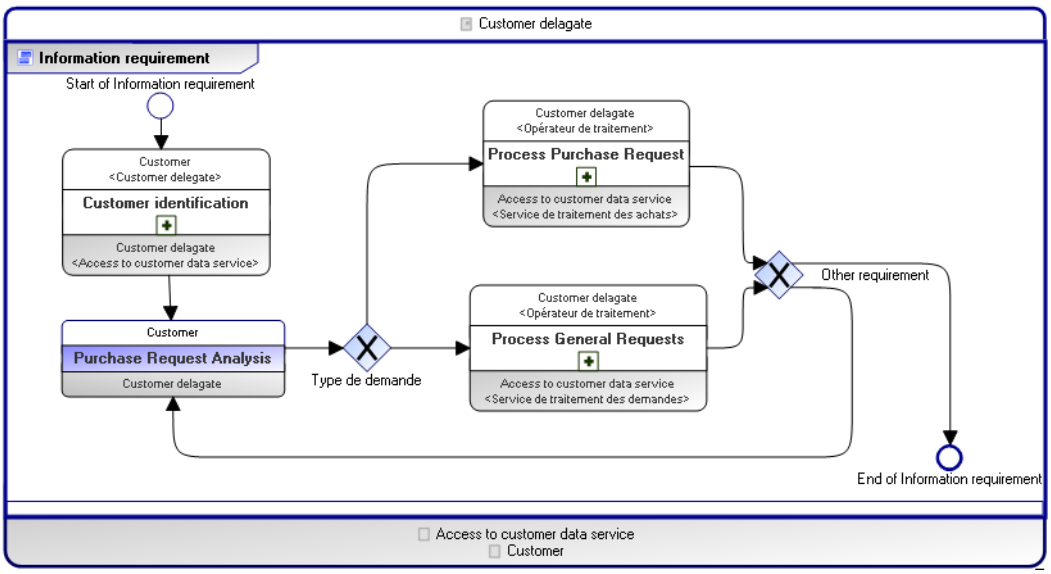
> *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*

> *An exchange use represents the usage of an exchange in another exchange contract.*

The exchange contract roles, presented at the border of the frame, represent participants:

- customer/supplier, or
- sender/recipient

An exchange can be described by involving more than two participants. In this case, one role is the initiator of the exchange contract and the others are contributors.



"Information Requirement" Exchange Contract Diagram (BPMN)

The "Information Request" exchange contract is used by the call center to take account of a customer request online. There are therefore three participants in this exchange contract: the customer, the IT applications and the customer representative who is the effective requester of the service (in this case the call center).

This exchange contract consists of identifying the customer, then analyzing the request. The request is then processed as a purchase request or as another request if it is an information request for example.

ELEMENTS DESCRIBED IN THE DIAGRAMS

Service oriented architectures diagrams are based on two types of concept:

- ***Elements Described*** in the diagram, either equipment or organizational, that are owned by a system, such as: system application use, end user or application component. Objects associated with these elements can be modified without impacting the elements themselves or their environment.
- ***Objects Associated*** to described elements, that represent real objects. These are application systems, logical application systems, service operations, org-units or applications.

  In the example of purchasing requests processing, "Purchasing Management Platform" and "Payment Management" are uses of the application system. These application systems themselves describe different resources: application components and application system use.

Described Objects pop-up menu

In all diagrams presented in this guide, the pop-up menus of the majority of components (for example **Application Component**) present:

- commands specific to the object type used by the component (for example **Application**)
- commands relating to the component itself
- commands relating to the graphics

Described element properties

To access the object that defines the component of a system:

1. Open the properties dialog box of the component (for example an **Application Component**) and select the **Characteristics** tab.
   The object that defines the component (in this case an **Application**) appears in the **Application Used** field.
2. Click the arrow at the right of the **Application Used** field to access the pop-up menu of the object used, or to replace it without changing the current component.

DIAGRAMS AND THEIR OBJECTS

The table below summarizes terms used for the different elements of a service oriented architecture as a function of the context in which the component is implemented.

Diagram	Diagram objects	Associated object	Comment
Application system environment diagram (describes an application system)	Application system use	Application System	Application system internal to the application system environment
	Partner application system.	Application System	Application system external to the application system environment
	System user	Org Unit	
Application system structure diagram (describes an application system)	Application system use	Application System	Use of one application system by another
	Application component	Applications	Use of an application in an application system
	End user	Org Unit	
Application structure diagram (describes an application)	Application component	Applications	
	Platform application component	Applications	Other type of use of an application in an architecture
Logical application system environment diagram (describes logical applications)	Logical application system component	Logical application system	Logical application system internal to the logical application system environment
	Partner logical system	Logical application system	Logical application system external to the logical application system environment
	System user	Org Unit	
Logical application system structure diagram (describes logical applications)	Logical application system use	Logical application system	Use of a logical application system by another
	Logical application component	Logical application	Use of a logical application in a logical application system
	End user	Org Unit	
Exchange contract diagram (BPMN) describing exchange contract	Exchange use	Exchange	
	Exchange contract use	Exchange contract	

Associated Object Properties

You can specify properties of certain objects: for example assign a memory size to applications, a version number or any other particular characteristic.

Such properties can be defined for the following objects:

- application system
- logical application architecture
- org-unit
- an application.

Creating a property

To create a property:

1. Open the properties dialog box of the object, select the **Properties** tab.
2. Click the **New** button.
   The Creation of Property Value dialog box opens.
3. Click the arrow at the right of the **Property Type** box and select **Create Property Type**.
   The Creation of Property Type dialog box appears.
4. Specify the **Name** and click **OK**.
5. Specify the **Value** and the **Unit** on different objects.
6. Click **OK**.
   The property, its value and its unit appear in diagrams describing the object.

Property inheritance

When an object inherits the properties of another, inheritance relates to the value of the property and not of the type.

```
If you have defined a property of "Version Number" type on
an object with value "6.3", objects inheriting this object
will all have a "Version Number" with value "6.3".
```

To modify the value of an inherited property:

1. Create a new property of the same type with a new value (for example, "Version Number of XP Workstations").
2. Replace the inherited property with the new property.

☛ *For more details on variations, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".*

MODELING WITH HOPEX SYSTEM ORIENTED IT ARCHITECTURE

HOPEX System Oriented IT Architecture allows you to describe the application architecture for your enterprise using a service-oriented approach based on two concepts:

- application systems
- logical application systems

The following points are covered here:

APPLICATION SYSTEM ENVIRONMENT DESCRIPTION

An application system environment present an application system use context. It describes the interactions, between the application system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.

Creating an application system environment

To create an *application system environment* from a library:

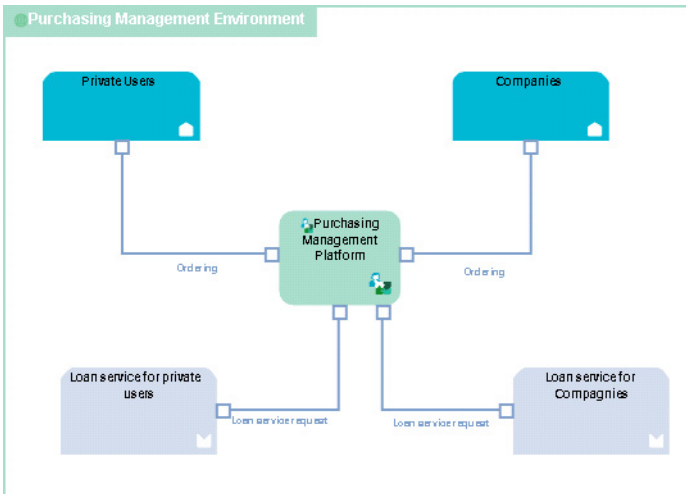
1. Click on the library in question to display its pop-up menu.
2. Click **New > Model building block**.
   The **Choose MetaClasse** properties dialog box opens.
3. Select **Application system environment**.
   The **Creation of Application System Environment** window opens.
4. Enter the **Name** of your application system environment and click **OK**.
   The new application system environment appears in the navigation tree.

Application system environment description

With **HOPEX System Oriented IT Architecture**, an application system environment is described by an application system environment structure diagram that describes the interactions between the internal application systems, its users and the partner application systems.

An application system environment present an application system use context. It describes the interactions, between the application

system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.



Environment diagram for the "Purchasing Requests Processing" application system.

Purchase requests are formulated by private users or by companies in different contractual conditions.

The "Purchasing Requests Processing" application system offers a loan service to its clients within the context of payment management.

The elements in this diagram are:

- *application system uses* that that represent the application systems internal to the environment.

  In the example, this is the application system that represents the "Purchasing Management Platform".

  An application system use describes the role of an application system in a composition relationship with a parent application system or in an application system environment.

- *system users* that represent the user category of services provided by the environment.

  This concerns two customer categories: individuals and companies

  A system user represents an organizational unit interacting with the boundaries of an application system environment or a logical application system environment.

- *partner application systems* that represent the application systems external to the application system environment described.

  In this example, this concerns two loan services offered to individuals and companies.

  A partner application system is an application system external to the environment of the described application service. The partner

application system can be a service supplier or a service consumer with respect to application system users.

- *interactions* between components

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

Accessing the description of an application system environment

The complete description of an application system environment is accessed in its properties dialog box.

To access the description of an application environment system:

❱ Right-click the application system environment and select **Properties**. Information concerning the application system environment is displayed in different tabs.

To access the objects that are part of the architecture:

❱ Select the **Components** tab.

☛ *For more details, see "Application system environment description", page 26.*

To access the properties that appear in application system diagrams, at the bottom of the described application system frame:

❱ Select the **Properties** tab.

☛ *For more details, see "Associated Object Properties", page 23.*

DESCRIBING AN APPLICATION SYSTEM

An *application system* is described by an application system structure diagram.

> 🕮 *An application system is an assembly of other application systems, applications and end users interacting with application components to implement one or several functions.*

The application system is characterized by:

- services offered: these are represented by service points.
- services used: these are represented by request points.

> ☛ *For more details, see "Describing Service and Request Points", page 45.*

Creating an Application System

To create an *application system* from a library:

1. Select the library that interests you.
2. Right-click the "Application System" folder and select **New > Application System**.
   The **Creation of Application System** dialog box appears.
3. Enter the **Name** of your application system and click **OK**.
   The new application system appears in the navigation tree.

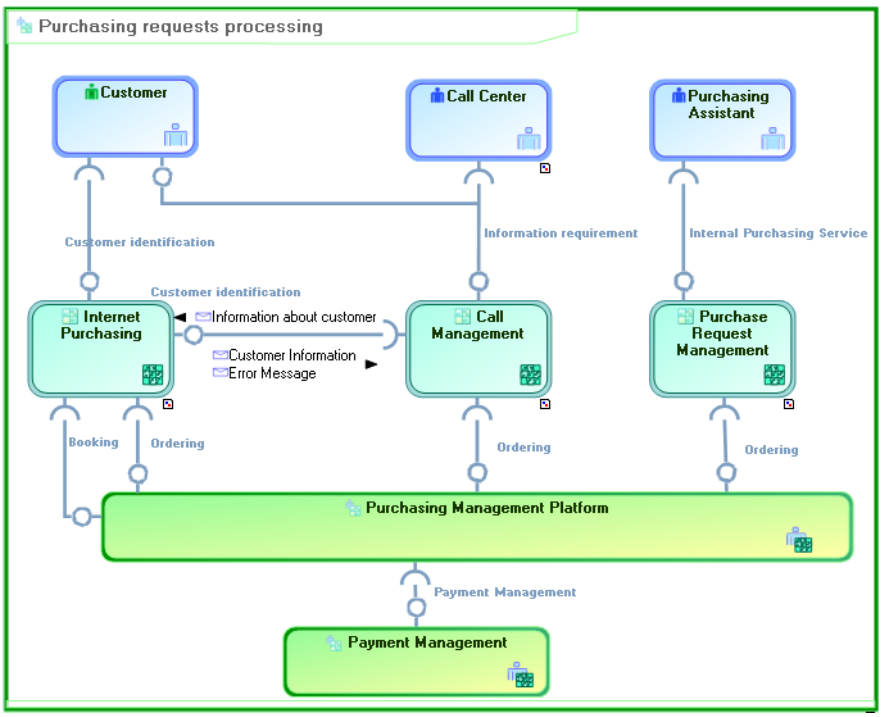
Describing Application System Components

Creating an application system structure diagram

An application system structure diagram describes the internal structure of an application system:

- services offered or required
- processes handled, components and interactions
- end users interacting with the application components

The following diagram describes the application system corresponding to purchasing requests processing.



The following diagram describes the application system corresponding to purchasing requests processing.

To create an application system structure diagram with **MEGA Windows Front-End**:

1. Right-click the application system and select **New > Diagram**.
2. In the window that opens, select "Application System Structure Diagram", confirm that the **Diagram initialization** check box is selected, and click **Create**.
   The diagram window appears. You are now in the **HOPEX** graphic editor.

To create an application system structure diagram with **Web Front-End**:

- Right-click the application system and select **New > Application System Structure Diagram**.

Creating an application system use

To describe that an application system implements another application system, you will:

- create a component of *Application System Use* type
- associate the application system implemented with it.

> *The use of a logical application system describes the role of a logical application system in a composition relationship with a parent application system or in a logical application system environment.*

For example, the purchasing requests processing system uses the "Purchasing Management Platform" and "Payment Management" applications system services.

To create an **application system use**:

1. In the application system structure diagram insert toolbar, click **Application System Use**.
2. Click in the frame of the described application system.
   A creation dialog box asks you to select the **application system use**. This is the application system implemented (for example "Payment Management").
3. Select an existing application system, or create a new application system.
4. Click **Finish**.
   The application system use appears in the diagram.

Creating an application component

To describe that an application system implements an application, you will:

- Creating an *application component*
- associate with this component the *application* implemented.

> *An application service component represents the fact that the application system used plays a role in the described application.*

> *A local data store represents a data store used only inside the system described.*

To create an **Application Component**:

1. In the application system structure diagram insert toolbar, click **Application Component** and click in the frame of the application system described.
   A creation dialog box asks you to select an **Application Used**.
2. Select an existing application or create a new one.
3. Click **OK**.
   The application component appears in the diagram.

Creating an end user

To specify that an application system, such as purchasing request processing, is activated by internal or external org-units, you will:

- create a component of *end user* type
- associate an org-unit with it, for example "Purchasing Assistant".

> 🕮 *The end user represents an organizational unit interacting at the boundaries of an application system or a logical application system.*

To create an end user:

1. In the application system structure diagram insert toolbar, click  **End User** and click in the frame of the diagram.
   A creation dialog box asks you to select the involved organizational resource.
2. Select the org-unit that interests you and click **OK**.

Completing the Application System Description

Accessing application system description

The complete description of an application system is accessed in its properties dialog box.

To access application system description:

- Right-click the application system and select **Properties**.
  Information concerning the application system is displayed in different tabs.

To access the objects that are part of the architecture:

- Select the **Components** tab.

> ☛ *For more details, see "Describing Application System Components", page 30.*

To access the properties that appear in application system diagrams, at the bottom of the described application system frame:

- Select the **Properties** tab.

> ☛ *For more details, see "Associated Object Properties", page 23.*

To access the list of services owned by the system:

- Select the **Service and Request Points** tab.

> ☛ *For more details, see "Describing Service and Request Points", page 45.*

Connecting an application system to an application process

An application system can be created to execute an *application process*.

> 🕮 *A system process is the executable representation of a process. the events of the workflow, the tasks to be carried out during the processing, the algorithmic elements used to specify the way in which*

the tasks follow each other, the information flows exchanged with the participants.

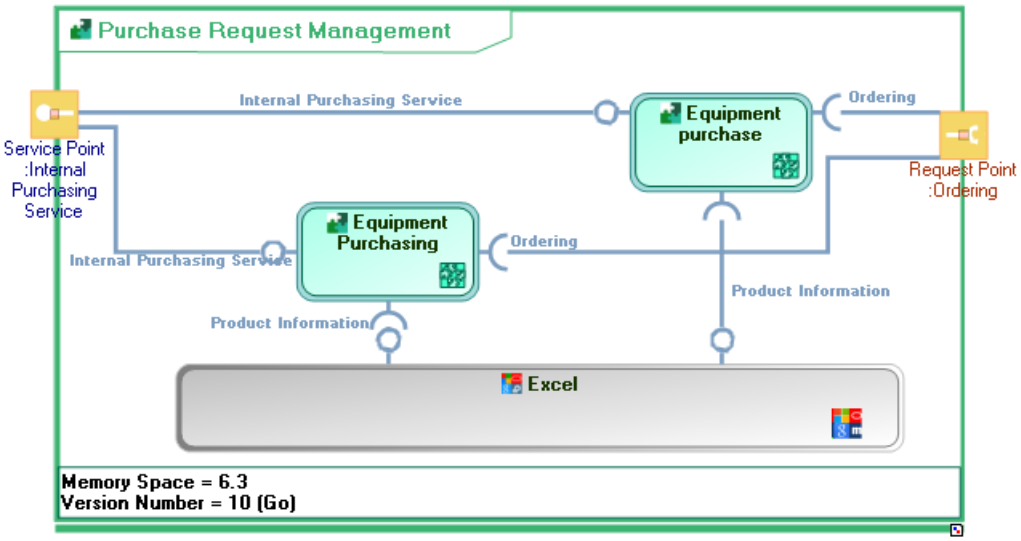
To associate an application system with an application process:

1. Right-click the application system and select **New > Process Performance**.
   The **Creation of Application Process Performance** dialog box appears.
2. From the **Performed Process** field, select the system process that interests you and click **OK**.
   The application process appears in the properties dialog box of the application system, in the **Performed Process** tab.

Describing an Application Structure

With **HOPEX System Oriented IT Architecture**, an application can be described by an application structure diagram.

An application structure diagram graphically shows first level components of an application, the access points (service point and request point) and the connections between components.




```
The purchase request management application, which is used
only for internal purchases, is based on two specialized
applications: one for office supplies and the other for
equipment. Both applications use Microsoft Excel.
```

The version number that appears at the bottom of the frame is a property of the application. For more details, see "Described element properties", page 21.

The diagram includes:

- *application components* representing the applications used

  In the example, this is the office supplies purchasing application and the equipment purchasing application.

  📖 *An application service component represents the fact that the application system used plays a role in the described application.*

- *platform application components* representing the technologies used by the platform

  In the example, this is the Microsoft Excel application.

  📖 *A platform application component represents the fact that the technology used plays a role of platform in the described application. For example, Tomcat is a platform component for SageCRM.*

  📖 *A technology is a definition or format that has been approved by a standards organization, or is accepted as a standard by the industry.*

- *application service components* representing application services used

  These components are not used in the example.

  📖 *An application service component represents the fact that the application system used plays a role in the described application.*

  📖 *An IT service is a component of an application made available to the end user of the application in the context of his/her work.*

- access, request and service points

  ☛ *For more details, see "Describing Service and Request Points", page 45.*

- *interactions* between components

  📖 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

LOGICAL APPLICATION SYSTEM ENVIRONMENT DESCRIPTION

A logical application system environment presents a logical application system use context. It describes the interactions between the logical application system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.

Creating a logical application system environment

To create a *logical application system environment* from a library:

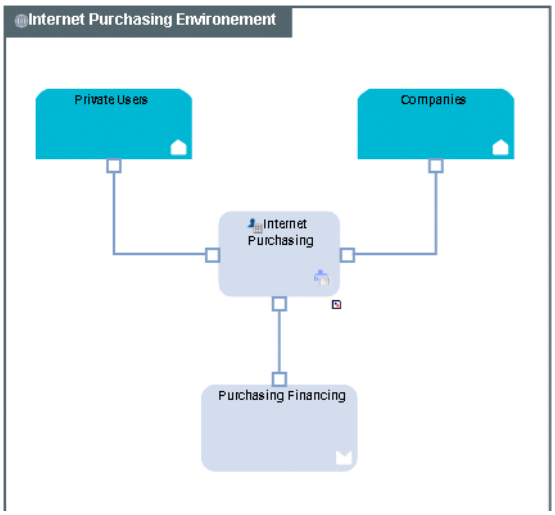
1. Click on the library in question to display its pop-up menu.
2. Click **New > Model building block**.
   The **Choose MetaClasse** properties dialog box opens.
3. Select **Logical application system environment**.
   The **Creation of Logical Application System Environment** window appears.
4. Enter the **Name** of your application system environment and click **OK**.
   The new logical application system environment appears in the navigation tree.

Logical application system environment description

With **HOPEX System Oriented IT Architecture**, a logical application system environment is described by an application system environment structure diagram that describes the interactions between the internal logical application systems, its users and the partner logical systems.

A logical application system environment presents a logical application system use context. It describes the interactions between

the logical application system and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.



Purchases are offered to private users and companies. The "Internet Purchasing" application system uses a "Purchasing Financing" logical application system that is external to the described environment.

The diagram includes:

- *logical application system components* that represent the logical application systems internal to the described environment.

  In the example, this is the logical application system "Internet Purchasing"

  A logical application service component represents the fact that the logical application system used plays a role in the described environment.

- *partner logical systems* that represent the logical application systems external to the described environment.

  In the example, this is the logical application system "Purchasing Financing"

  A partner logical system is a logical application system external to the environment of the described logical application system. The partner logical system can be a service supplier or a service consumer with respect to components of the logical application system.

- *system users* that represent the user category of services provided by the environment.

  A system user represents an organizational unit interacting with the boundaries of an application system environment or a logical application system environment.

- *Interactions* between the components representing requests for services.

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications,

activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

☛ *For more details, see "Describing Service and Request Points", page 45.*

Accessing the description of a logical application environment system

The complete description of a logical application system environment is accessed from its properties dialog box.

To access the description of a logical application environment system:

- Right-click the logical application system environment and select **Properties**.
  Information concerning the logical application system environment is displayed in different tabs.

To access the objects that are part of the architecture:

- Select the **Components** tab.

  ☛ *For more details, see "Logical application system environment description", page 35.*

To access the properties that appear in application system diagrams, at the bottom of the described application system frame:

- Select the **Properties** tab.

  ☛ *For more details, see "Associated Object Properties", page 23.*

DESCRIBING A LOGICAL APPLICATION SYSTEM

A *logical application* is an element of the IT architecture.

> 🕮 *A logical application is a component of an application that is available to the end user of this application in the context of his/her work. Logical applications represent the split of an application into its basic functional units. A logical application is a consistent, indivisible unit of processing that coordinates a set of messages and events in order to perform a task in the information system. In the application system, the logical application is the most basic functional unit.*

A *logical application system* is a component used to describe the organization of *logical applications* according to a service-oriented approach.

> 🕮 *A logical application system is an assembly of other logical application systems, logical applications and end users interacting with application components to implement one or several functions.*
>
> 💣 **In HOPEX System Oriented IT Architecture, there is no direct link between an application system and a logical application system.**

Creating a Logical Application System

To create a *logical application system* :

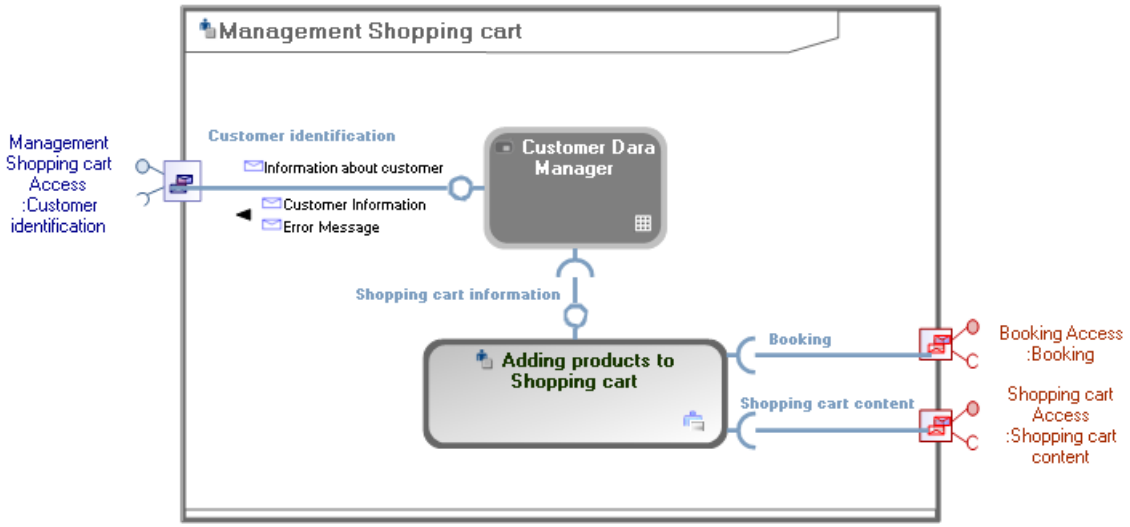
1. Select the library that interests you.
2. Right-click the "Logical Applications" folder and click **New > Logical Application System**.
   The **Creation of a Logical Application System** dialog box appears.
3. Enter the **Name** of your logical application system and click **OK**.
   The new logical application system appears in the navigation tree.

Logical Application System Component Description

A logical application system diagram describes the internal structure of the system:

- services offered or required
- processes handled, components and interactions
- end users interacting with the software components

The following diagram describes the logical application system for customer shopping cart management.



"Management Shopping Cart" logical application system structure diagram

A logical application system diagram is built on the same principle as an application system structure diagram.

☛ *For more details on creating components of this diagram, see "Describing Application System Components", page 30.*

A logical application structure diagram includes:

- d'*end users*

  The end user represents an organizational unit interacting at the boundaries of an application system or a logical application system.

- *application system environment*

  The use of a logical application system describes the role of a logical application system in a composition relationship with a parent application system or in a logical application system environment.

- *logical application components*

  A logical application component describes the role of a logical application in a composition relationship with a parent logical application or a logical application system.

- *interactions* between the components representing requests for services

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an

enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

☛ *For more details on interactions between logical application system components, see "Managing Interactions", page 48.*

- *service points*

  📖 *A service point is a point of exchange by which an agent offers a service to potential customers.*

- *request points*

  📖 *A request point is a point of exchange by which an agent requests a service from potential suppliers.*

  ☛ *For more information on access points, see "Describing Service and Request Points", page 45.*

USING DATA STORES

> 🕮 *A data store provides a mechanism to update or consult data that will persist beyond the scope of the current process. It enables storage of input message flows, and their retransmission via one or several output message flows.*

Data stores can be described in the following diagrams:

- application system structure diagrams
- application structure diagrams
- Logical application system structure diagrams

Describing physical data stores

> 🕮 *A physical data store represents the implementation of a logical data store.*

You can use only the following data stores in an application system structure diagram as well as in an application structure diagram:

- Local or external
- RE (Relationship Entity) or RDB (Relational Database).

> 🕮 *Relational Database data store represents a data store in the form of tables or table views.*

> 🕮 *A Relationship Entity data store describes a database in the form of entities or classes.*

Creating a local RDB physical data store

> 🕮 *A local data store represents a data store used only inside the system described.*

To create, for example, a local RDB physical data store based on an application system structure diagram:

1. Open the diagram that interests you.
2. In the application system structure diagram insert toolbar, click **Local RDB Physical Data Store**.
3. Click in the frame of the described application system.
   A creation dialog box prompts you to select **RDB physical data structure**. This is the RDB physical structure that will concretely support the database.
4. Select an existing **RDB physical domain**, or create a new application system.
5. Click **OK**.
   The local RDB physical data store appears in the diagram with the name of the physical domain.

> ☛ *A physical RDB domain is held by a database.*

Creating an external RE physical data store

> 🕮 *An external data store represents a data store used inside and outside of the system described.*

To create, for example, an external RE physical data store based on an application system structure diagram:

1. Open the diagram that interests you.
2. In the application system structure diagram insert toolbar, click **External ER Physical Data Store**.
3. Click in the frame of the described application system.
   A creation dialog box prompts you to select **RE physical data structure**. This is the RE physical structure that will concretely support the database.
4. Select an existing **RE physical data domain**, or create a new application system.
5. Click **OK**.
   The external RE physical data store appears on the border of the diagram with the name of the external RE physical data domain.

> ☛ *A physical RE domain is held by a package.*

Describing access to a physical data store

The following components can have read and right access to a physical data store:

- *Application system use* type elements representing the two application systems used.

  > 🕮 *An application system use describes the role of an application system in a composition relationship with a parent application system or in an application system environment.*

- *Application components* representing the applications used.

  > 🕮 *An application service component represents the fact that the application system used plays a role in the described application.*

- The *platform application components*.

  > 🕮 *A platform application component represents the fact that the technology used plays a role of platform in the described application. For example, Tomcat is a platform component for SageCRM.*

To create a read access to the data store:

1. In the diagram insert toolbar, click **Link**.
2. Draw a link between the data store and the entity that reads the data (component or application system use).
   A **Read-only access to data storage** is automatically created with the link from the data store to the entity.

> ☛ *To create a link with write access, you must draw a link between this entity and the data store to which it has write access. A **Write access to data storage** is then automatically created.*

Describing a logical data store

> 🕮 *A logical data store represents the use of data via application systems without considering how their access will be concretely implemented.*

In a logical application system structure diagram, you can use only the local or external RE data stores.

> 🕮 *A logical application system is an assembly of other logical application systems, logical applications and end users interacting with application components to implement one or several functions.*

> 🕮 *A logical application is a component of an application that is available to the end user of this application in the context of his/her work. Logical applications represent the split of an application into its basic functional units. A logical application is a consistent, indivisible unit of processing that coordinates a set of messages and events in order to perform a task in the information system. In the application system, the logical application is the most basic functional unit.*

Creating a local logical data store

> 🕮 *A local data store represents a data store used only inside the system described.*

To create, for example, a local logical data store based on a logical application system structure diagram:

1. Open the diagram that interests you.
2. In the application system structure diagram insert toolbar, click **Local Logical Data Store**.
3. Click in the frame of the described logical application system.
   A creation dialog box prompts you to select the **RE Data Logical Store Structure**. This is the RE physical structure that will concretely support the database.
4. Select an existing **RE logical data domain**, or create a new application system.
5. Click **OK**.
   The local logical data store appears in the diagram with the name of the RE logical data domain.

> ☛ *An RE logical data store structure is held by a package.*

Describing access to a logical data store

The following components can access a logical data store with read and write access:

- *logical application system uses*

  > 🕮 *The use of a logical application system describes the role of a logical application system in a composition relationship with a parent application system or in a logical application system environment.*

- The *logical application components* representing the logical applications used in the context of the described logical application system.

  > 🕮 *A logical application component describes the role of a logical application in a composition relationship with a parent logical application or a logical application system.*

To create read access to the logical data store:

1. In the diagram insert toolbar, click **Link**.

2. Draw a link between the data store and the entity that reads the data (logical application component or logical application system use).
   A **Read-only access to data storage** is automatically created with the link from the data store to the entity.

   ☛ *To create a link with write access, you must draw a link between this entity and the data store to which it has write access. A **Write access to data storage** is then automatically created.*

Realization of a data domain

To describe all the elements of the application structure implemented by a data store, you must define a *Realization* on the data stores in question.

📖 *A realization describes the relationship between a logical entity and a physical entity that implements it. The physical entity gives the list of logical entities that it implements.*

To describe that a data store is implementing an element in the application structure:

1. Open the dialog box of the data store that interests you.
2. Select the **Characteristics** tab and the **Realization** subtab.
3. In the **Component Realization** section, click **New**.
   The window for data store realization opens.
4. In the **Realization** field, specify the logical application or the logical application system in question.
5. In the **Component Realized** field, from among the components available, chose the component implemented.
6. Click **OK.**
   The data store realization appears in the properties page.

DESCRIBING SERVICE AND REQUEST POINTS

In a service oriented architecture, communications are based on:

- *interactions*

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

  For more details on interactions between logical application system components, see "Managing Interactions", page 48.

- access points: *service points* and *request points*.

  A request point is a point of exchange by which an agent requests a service from potential suppliers.

  A service point is a point of exchange by which an agent offers a service to potential customers.

HOPEX System Oriented IT Architectureenables management of these components and consistency checking of specified information.

Concepts Overview

Service points

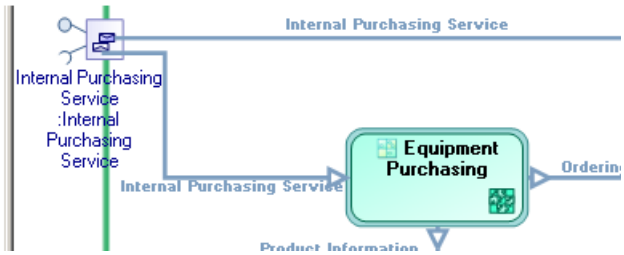
An application system or a logical application system is created to ensure one or more services. These services are represented by *service points*  .

A service point is a point of exchange by which an agent offers a service to potential customers.

The service is requested according to precise terms defined by an *exchange contract* assigned to the service point.

An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.

Components activated to assure a service are linked to the service point by interactions. If it is necessary to activate several components, you have to create several interactions between the service point and the system components.



In the example presented here, the internal purchasing service is linked to two interactions based on the same exchange contract, representing the activation of the equipment purchasing application or the office supplies purchasing application.

☛ *To create a service point, see "Creating a Service Point or a Request Point", page 47.*

Request points

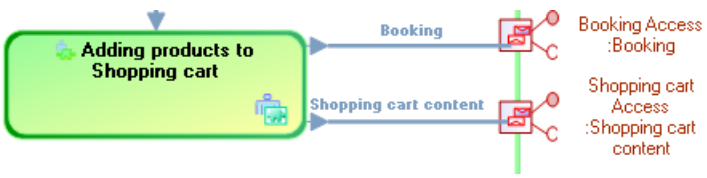
A *request point* enables representation of use of an external service.

A request point is a point of exchange by which an agent requests a service from potential suppliers.

The service is requested according to precise terms defined by an *exchange contract* assigned to the request point.

An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.

Components that issue a request are linked to the request point by an interaction.



In the example, request points represent requests for service executed by the "Adding Products to Shopping Cart" logical application to issue an order or book products.

☛ *To create a request point, see "Creating a Service Point or a Request Point", page 47.*

Creating a Service Point or a Request Point

The process for creating a *service point* or *request point* is identical.

> *A request point is a point of exchange by which an agent requests a service from potential suppliers.*
>
> *A service point is a point of exchange by which an agent offers a service to potential customers.*

To create a service point:

1. In the diagram insert toolbar, click **Service Point**  .
2. Position the object at the edge of the system frame.
   A creation dialog box opens.
3. Click the arrow at the right of the **Exchange Contract** field to define the exchange contract enabling activation of this service point, and select for example **Connect Exchange Contract**.
   A search dialog box opens.
4. Select the exchange contract associated with this service point.
5. Click **Next**.
   A dialog box opens proposing a list of exchange contract roles that can be associated with the service point.

   > ☛ *This second dialog box is not proposed if there is only one candidate role that can be associated with the service point.*
6. Select the role that interests you and click **OK**.
   The service point appears in the diagram.

To change the service point name:

1. Click the name of the service point and press key F2.
2. Enter the new name used at specification of interaction points.

   > ☛ *For more details on interaction points, see "Defining an Element Interaction Point", page 48.*

MANAGING INTERACTIONS

An *Interaction* represents the exchange of information between architecture components.

> 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Content of an interaction is described by an *exchange contract*.

> 🕮 *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*
>
> ☛ *For more details on exchange contract concepts, see "Describing Exchange Contracts", page 52.*

Creating an Interaction

To create an interaction:

1. In the diagram insert toolbar, click the **Interaction** button.
2. Draw a link between the two communication entities.
3. In the add interaction dialog box, specify the exchange contract you wish to use.

   > ☛ *You can also create a new exchange contract.. For more details, see "Creating an Exchange Contract from an Interaction", page 53.*
4. Click **OK**.

Defining an Element Interaction Point

Principle

The interaction point of an element connects an interaction to one of the components in communication. It enables specification of:

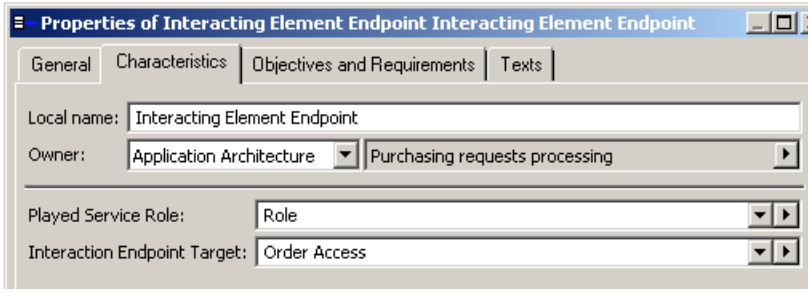
- The service point or query point that intervenes in the communication.
- The role, consumer or supplier, represented by the interaction point in the exchange contract.

Characterizing an element interaction point

To modify properties of an element interaction point:

1. Right-click the interaction beside the communication entity.

2. Open the interaction properties and select the **Characteristics** tab.



3. Select the **Played Service Role**, that is the role of the exchange contract played by the element interaction point.
4. Select the **Interaction Endpoint Target**, that is the service (or request) point concerned by the interaction.
5. Click **OK**.

3

MODELING WITH SERVICE DESIGN

This chapter presents how to describe exchange contracts between IT architecture components.

DESCRIBING EXCHANGE CONTRACTS

An *Interaction* represents the exchange of information between architecture components.

> 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Content of an interaction is described by an *exchange contract*.

> 🕮 *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*

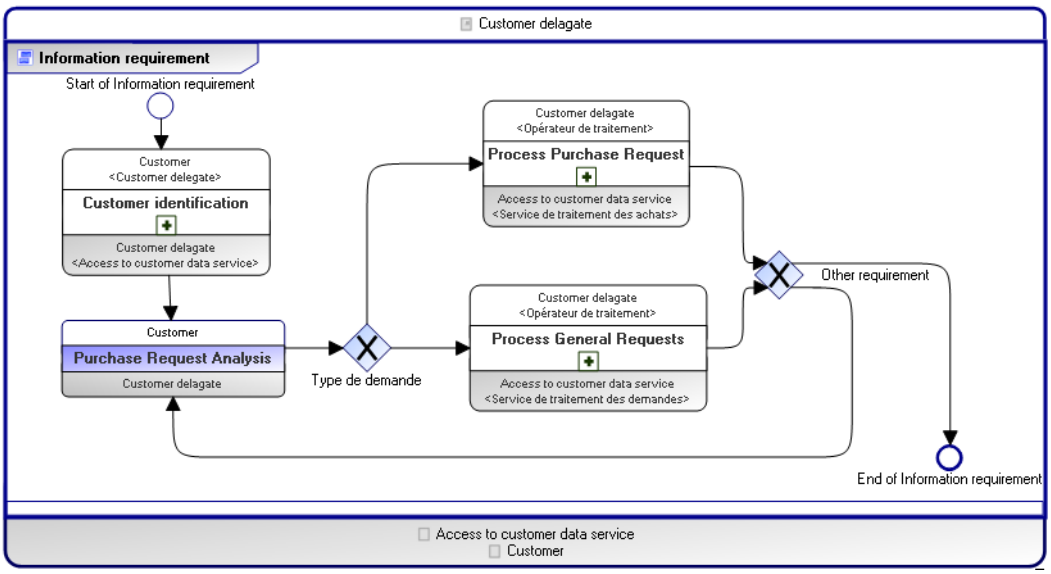
With the **MEGA Service Design** option, an exchange contract is described by a sequence flow of operations which are represented:

- by *exchange contract uses*
- or by *exchange uses*

> 🕮 *An exchange contract use is associated with an exchange contract. It enables representation of complex exchanges.*

> 🕮 *An exchange use represents the usage of an exchange in another exchange contract.*

An exchange contract is described in an Exchange Contract Diagram. The sequence flow of operations is described in BPMN formalism.



Exchange Contract Diagram (BPMN) example

✓ "Creating an Exchange Contract from an Interaction", page 53
✓ "Creating an Exchange Contract Diagram (BPMN)", page 54
✓ "Defining an Exchange or an Exchange Contract Use", page 54
✓ "Creating events", page 55
✓ "Creating Sequence Flows", page 55
✓ "Gateways", page 56

Creating an Exchange Contract from an Interaction

You can also create a new exchange contract:

- from a library
- from an interaction in a diagram

To create an exchange contract, in a diagram, from an interaction:

1. In the diagram insert toolbar, click the **Interaction** button.
2. Draw a link between the two communication entities.
3. In the add interaction dialog box, click the arrow at the right of the **Exchange Contract** box and select **New**.
   The **Creation of Exchange Contract** dialog box opens.
4. Enter the name of the exchange contract in the **Name** box.
5. Click **OK**.
   The interaction and exchange contract are created.

Creating an Exchange Contract Diagram (BPMN)

With the **MEGA Service Design** option, an exchange contract is represented by an Exchange Contract Diagram (BPMN).

To create an Exchange Contract Diagram (BPMN) from an interaction:

1. Right-click the interaction.
2. Select the associated exchange contract and, in its pop-up menu, click **New > Exchange contract diagram (BPMN)**

   ☛ *If you work with **MEGA Windows Front-End**, you should select "Exchange Contract Diagram (BPMN)" from a specific dialog box. Confirm that the **Diagram initialization** check box is selected, and click **Create**.*

   The diagram opens with the exchange contract frame and the two *roles* representing consumer and the supplier.

   🕮 *A role is an integral part of the object that it describes, and is not reusable. It can subsequently be assigned to an org-unit internal or external to the organization or to an IT component. Examples: client, traveler.*

Defining an Exchange or an Exchange Contract Use

In an Exchange Contract Diagram (BPMN), operations are described by:

- *exchange contract uses*
- *exchange uses*

  🕮 *An exchange contract use is associated with an exchange contract. It enables representation of complex exchanges.*

  🕮 *An exchange use represents the usage of an exchange in another exchange contract.*

To create an *exchange contract use* :

1. Click the **Exchange Contract Use** button  and click in the diagram within the exchange contract frame.
   The creation dialog box opens.
2. Click the arrow at the right of the **Specification** box.
3. Select **List** in the drop-down list and select the exchange contract associated with the exchange contract use.
4. In the **From** field, select the described exchange contract role connected to the "Consumer" role of the exchange contract use.
5. In the **To** field, select the described exchange contract role connected to the "Supplier" role of the exchange contract used.
6. Click **Finish**.

Creating events

"Start" and "End" *events* are required in the description of the service assured by the exchange contract.

> *An event represents a fact or an action occurring in the system, such as updating client information. It is managed by a broker. An application indicates that it can produce the event by declaring that it publishes it. If an application is interested in an event, it declares that it subscribes to the event.*

To create the exchange contract "End" event:

1. In the insert toolbar, click the **Event** button.
2. Click in the frame of the described exchange contract.
   The **Creation of Event** dialog box appears.
3. In the **Name** box, enter "End".
4. In the Event Nature frame, select **End**.
5. Click **Finish**.
   The exchange contract "End" event appears in the diagram.

Creating Sequence Flows

A *sequence flow* is a directional link that represents the chronological organization of the different processing steps.

> *A sequence flow is used to show the order in which steps of an exchange contract will be performed. A sequence flow has only one source and only one target.*

Creating Sequence Flows

To create a *sequence flow*:

1. Click the **Sequence Flow** button.
2. Click the first object representing the start step, and, holding the mouse button down, draw a line to the object representing the next step.

Defining a Condition on a Sequence Flow

To define a condition on a sequence flow:

3. Right-click the sequence flow and select **Properties**.
4. In the dialog box that opens, select the **Characteristics** tab.
5. Click the arrow at the right of the **Sequence Flow Type** field.
6. Select "Conditioned" in the drop-down list.
7. Enter the conditioning expression in the **Predicate** text box.
8. Click **OK**.
   The text associated with the condition appears on the link which takes the form .

Gateways

In compliance with the BPMN standard, in the object toolbar, several *gateway* types are available to you.

> 🕮 *Gateways are modeling elements that are used to control how sequence flows interact as they converge and diverge within a process.*

To better understand use case principles, we distinguish between:

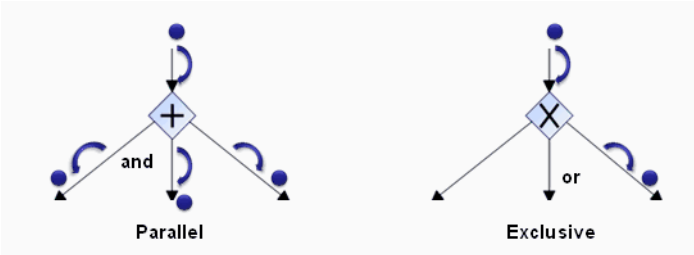
- processing step output gateways
- processing step input gateways

Processing Step Output Gateways

In the case of an **Exclusive** gateway, only one output branch can be selected from those available. The branch can be selected:

- according to **Data** available for the process
- according to **Events** occurring during its execution

In the case of a **Parallel** gateway, all output branches are processed simultaneously.



In the case of an **Inclusive** gateway, one or several output branches can be selected from those available.

A **Complex** gateway represents a combination of those above.

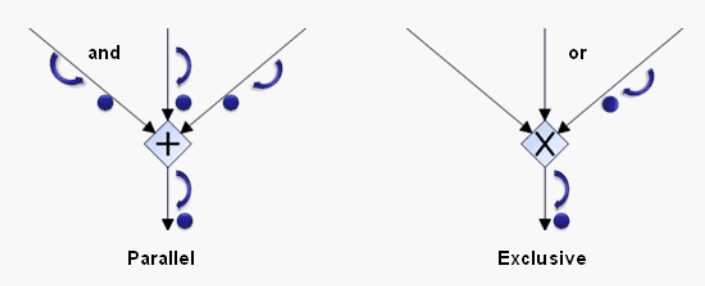
> ☛ *When the gateway has been created, its type can be modified in its properties dialog box.*

Step Input Gateways

At input of a step, a gateway represents a point of convergence of sequence flows.

In the case of an **Exclusive** gateway, the step is triggered when one of these branches is active.

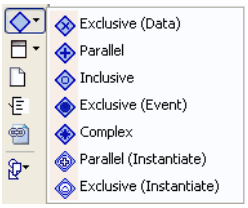
In the case of a **Parallel** gateway, all input branches are processed simultaneously.



Creating gateways

To create a gateway:

1. Click the arrow at the right of the **Gateway** button in the diagram insert toolbar and select the gateway type you wish to create.



2. Click on the diagram.
   The gateway appears in the diagram with the shape appropriate to its type.

Modifying gateways

To modify a gateway:

1. Right-click the gateway and select **Properties**.
   The properties window opens.

2. In the **Characteristics** tab, modify the name or type of the gateway. The different gateway types proposed are:
   - **Complex**: the execution can take a complex combination of paths.
   - **Exclusive (Data)**: the execution can take a single path from several possible paths depending on the value of the data available. This is the default gateway type.
   - **Exclusive (Start)**: the execution is triggered by the first event occurring. The others are ignored.
   - **Exclusive (Event)**: the execution can take a single path from several possible paths depending on the events occurring.
   - **Inclusive**: the execution can take one or several paths simultaneously.
   - **Parallel**: the execution takes several parallel paths simultaneously.
   - **Parallel (Start)**: the execution is triggered by the first event occurring. The other events occurring during progress of the execution are also taken into account.
3. Click **OK**.

DESCRIBING EXCHANGES

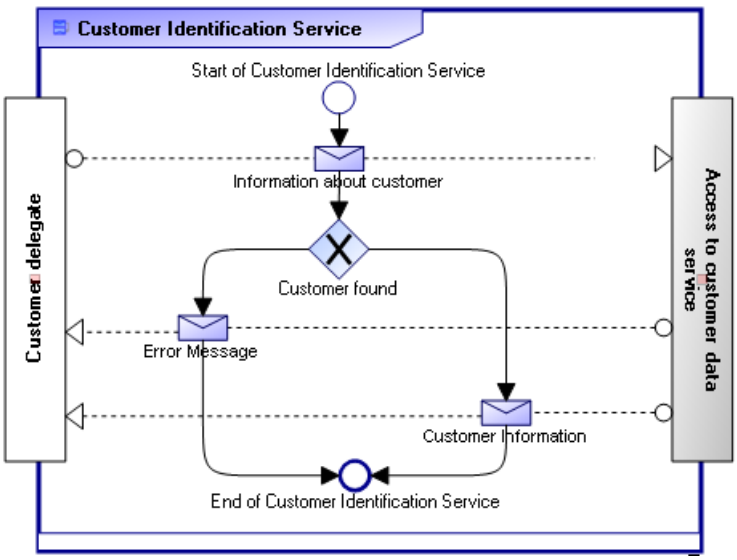
Content of an interaction is described by an *exchange contract*.

> 🕮 *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*

With the **MEGA Service Design** option, an exchange contract is described by a series of exchanges or exchange contract uses. An *exchange use* is associated with an *exchange*.

> 🕮 *An exchange use represents the usage of an exchange in another exchange contract.*
>
> 🕮 *An exchange specifies message flow exchanges between two participants.*

An *exchange* is described by an exchange diagram presenting the sequence flow of messages exchanged.




```
The customer identification service protocol begins by
sending information enabling identification of the
customer. An error message appears if the customer is not
found, otherwise customer information is sent (customer
identification, status of orders, etc.).
```

✓ "Creating an Exchange", page 60
✓ "Describing Exchanges", page 60

Creating an Exchange

You can create an *exchange* from a library or from an exchange contract diagram (BPMN).

> 🕮 *An exchange specifies message flow exchanges between two participants.*

To create an *exchange* from an exchange contract diagram (BPMN).

1. Click the **Exchange Use** button and click in the diagram within the described exchange contract frame.
   The Creation of Exchange Use dialog box opens.
2. Click the arrow at the right of the **Specification** field and select **Create** in the drop-down list.
   The Creation of Exchange dialog box appears.
3. Enter the **Name** of your exchange.
4. Click **OK** to close this dialog box.
   The exchange is automatically created.
5. Continue the exchange use creation procedure.

Describing Exchanges

Creating an exchange diagram (BPMN)

An *exchange* is described by an exchange diagram presenting the sequence flow of messages exchanged.

To create an exchange diagram:

1. Right-click an **Exchange** and select **New** > **Exchange Diagram (BPMN)**.

   > ☛ *If you work with **MEGA Windows Front-End**, you should select "Exchange Diagram (BPMN)" from a specific dialog box. Confirm that the **Diagram initialization** check box is selected, and click **Create**.*

   The diagram opens. The exchange frame is positioned and the two roles (Consumer and Supplier) are created.

Creating a message flow with content

You must specify the *message flows* and their *content* exchanged between the two exchange roles.

> 🕮 *A message flow represents circulation of information within an exchange contract. A message flow transports its content.*

> 🕮 *The content designates the content of a message or an event, independent of its structure. This structure is represented by an XML schema linked to the content. A content may be used by several messages, since it is not associated with a sender and a destination. There can be only one content per message or event, but the same content can be used by several messages or events.*

To create a message flow and its content:

1. In the exchange diagram, click the **Flow With Content** button.
2. Click the role that represents the message flow sender and, holding the mouse button down, draw a link to the message flow recipient.
   The **Creation of Message Flows With Content** dialog box opens.
3. In the **Content** drop-down list, select the content you wish to associate with the flow.
   The message flow is displayed with its content in the diagram.

USING HOPEX SYSTEM ORIENTED IT ARCHITECTURE STANDARD REPORTS

HOPEX System Oriented IT Architecture offers architectures analysis facilities based on reports. These reports enable analysis of costs communications between agents used in the architecture. **HOPEX** Suite uses reports to group sets of repository objects and study their interactions.

☛ *For more details on operation of reports, see the HOPEX Common Features guide, "Generating Reports".*

A list of the different report templates proposed as standard by **HOPEX System Oriented IT Architecture** is displayed:

- ✓ "Agent Exchange Compliance", for more details, see "Agent Exchange Balance", page 66.
- ✓ "Application Exchange Compliance", this report is identical to the "Agent Exchange Compliance" report, but the report subject is restricted to an application.
- ✓ "Agent Exchange Balance", for more details, see "Agent Exchange Compliance", page 64.
- ✓ "Application Exchange Balance", this report is identical to the "Agent Exchange Balance" report, but the report subject is restricted to an application or a logical application.
- ✓ "Service Interaction Matrix", page 68.

AGENT EXCHANGE COMPLIANCE

The Agent Exchange Compliance report summarizes information received by agents configured in Report Subject.

In their different contexts of use, agents are connected to the exterior by interactions, whose associated exchange contracts describe message exchanges.

For each agent, this report lists the interactions received, of which the exchange contract is described by messages.

```
The report below presents messages exchanged via
interactions received by the ""Internet Purchasing" and
"Call Management" applications, and the "Payment
Management" and "Purchasing Management Platform"
application architectures.
```

2. Agent Exchange Compliance

[Add a comment for this chapter]

This table shows for each interaction the compliance between the expected exchange and the information element effectiv

0% of the exchanges are compliant.
12 exchanges are considered.
12 exchanges are missing (100%).

Legend:
OK (Expected information element has effectively been exchanged)
Missing (Expected information element has not been exchanged)
? Not expected (Information element has effectively been exchanged but it was not expected)

Exchange Compliance				
Expected Exchanges	Sources	Targets	Information Elements	E E
Information requirement (Information requirement)	Call Center (Call Center)	Call Management (Call Management)	Information about customer	
			Customer Information	
			Error Message	
Customer identification (Customer identification)	Call Management (Call Management)	Internet Purchasing (Internet Purchasing)	Information about customer	
			Customer Information	
			Error Message	
Customer identification	Customer	Internet Purchasing	Information about	

Report parameters

This report takes only **Agent** as a parameter. This parameter is required. Accepted MetaClasses are:

- Org Unit
- Applications
- Application System
- Logical application architecture
- Logical application
- Business function

AGENT EXCHANGE BALANCE

This report summarizes information received and processed by agents configured for the report.

In their different contexts of use, agents are connected to the exterior by interactions, whose associated exchange contracts describe message exchanges.

An agent is considered as "balanced" if the description of his internal operation handles all messages exchanged with the exterior.

The report below presents a balanced agent, "Manage Customer Shopping Cart", and an unbalanced agent "Manage Customer Shopping Cart" of which internal structure is not described.

2. Agent Exchange Balance

[Add a comment for this chapter]

The following table shows for each agent the balance between around and inside information elements incoming or outgoing the agent.

66% of the exchanges are well balanced.
9 exchanges are considered.
6 exchanges are well balanced (66%).
3 exchanges are not well balanced (33%).

Legend:

- Incomplete: The incoming information element not managed by the item.
- Incomplete: The managed information element is not received by the item.
- OK: The incoming information element is properly managed by the item.
- Incomplete: The outgoing information element is not produced by the item.
- Incomplete: The produced information element is not sent by the item.
- OK: The produced information element is properly sent by the item.

- The around exchange is coming in the balanced item.
- The inside exchange is coming in the balanced item.
- The around exchange is going out the balanced item.
- The inside exchange is going out the balanced item.

Exchange Balance			
Balanced Items	Information Elements	Around Exchanges	Inside Exchanges
Call Management (Call Management)	Information about customer	Information requirement — Call Management / Customer	Information Requirement Access — Customer identification
	Customer Information	Customer identification — Internet Purchasing	
	Error Message	Customer identification — Internet Purchasing	
	Information about customer	Customer identification — Internet Purchasing	
	Customer Information	Information requirement — Call Management / Customer	Information Requirement Access — Customer identification
	Error Message	Information requirement — Call Management / Customer	Information Requirement Access — Customer identification
Management Shopping cart	Information about customer	Customer Inf — Management Shopping cart	Management Shopping cart Access — Customer identification
	Customer	Customer — Management	Management — Customer

Report parameters

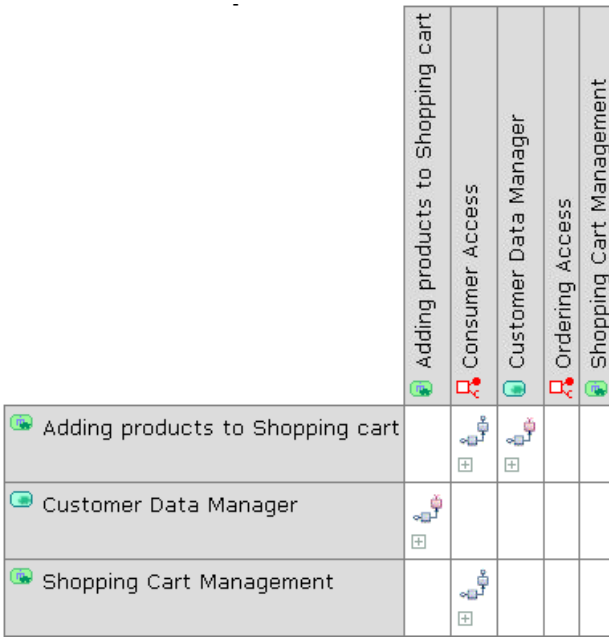
This report takes only **Agent** as a parameter. This parameter is required. Accepted MetaClasses are:

- Org Unit
- Applications
- Application System
- Logical application architecture
- Logical application
- Business function

SERVICE INTERACTION MATRIX

This report lists interactions used in agents configured in the report subject.

Report presentation



The objects in rows play the role of consumer in the interactions ; they ask for services.

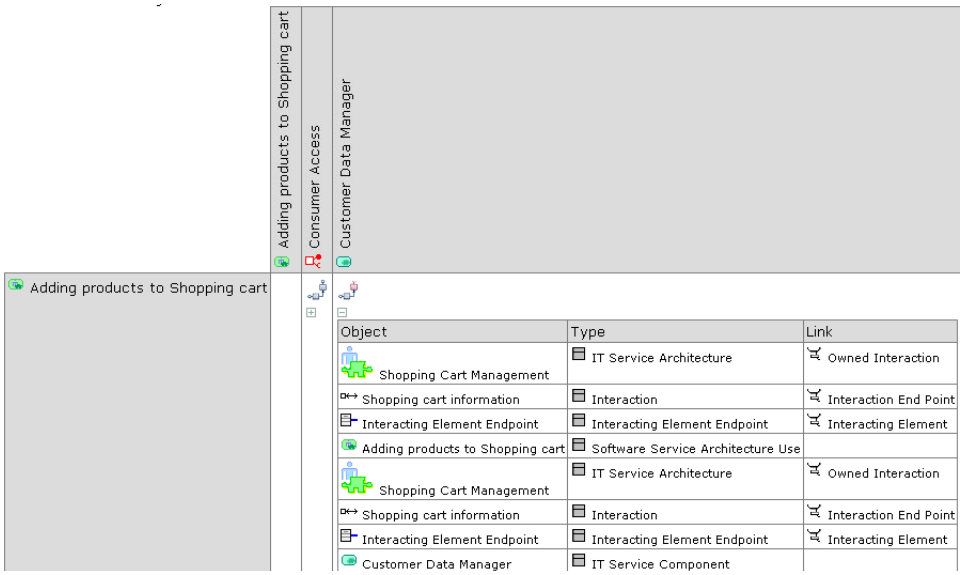
- *The objects in line are directly, or indirectly, linked to agents that are defined in the report parameters.*
- *Only objects (and sub objects) that have a defined interaction point playing the "consumer" role are displayed in the report.*

The objects in columns play the role of supplier in the interactions ; they provide services.

- *The objects in columns are directly, or indirectly, linked to agents that are defined in the report parameters.*
- *Only objects (and sub objects) that have a defined interaction point playing the "supplier" role are displayed in the report.*
- *For more details on interaction points, see "Defining an Element Interaction Point", page 48.*

Exchange details

If you click the button in a cell in the matrix, you obtain the details of the interaction connecting the two objects.



Object	Type	Link
Shopping Cart Management	IT Service Architecture	Owned Interaction
Shopping cart information	Interaction	Interaction End Point
Interacting Element Endpoint	Interacting Element Endpoint	Interacting Element
Adding products to Shopping cart	Software Service Architecture Use	
Shopping Cart Management	IT Service Architecture	Owned Interaction
Shopping cart information	Interaction	Interaction End Point
Interacting Element Endpoint	Interacting Element Endpoint	Interacting Element
Customer Data Manager	IT Service Component	

Information is presented in the following order:

- the interaction connected to the "Consumer" object described successively by:
  - object representing the context: object containing the "consumer" object and the interaction.
  - interaction used by the "consumer" object to send its request.
  - interaction point used by the "consumer" object to send its request.
  - "consumer" object (that is also on line).
- the interaction connected to the "Supplier" object
  - object representing the context: object containing the "consumer" object and the interaction.
  - interaction used by the "supplier" object to receive the request.
  - interaction point used by the "supplier" object to receive the request.
  - "supplier" object (that is also on column).

Report parameters

This consists of defining report input data.

Parameter	Parameter type	Constraints
Agent	Org Unit Applications Application System Logical application architecture Logical application Business function	At least one subject mandatory. Agents are cumulated
Filtering agent part	Org Unit Applications Application System Logical application architecture Logical application Business function	Not mandatory
Filtering agent	Org Unit Applications Application System Logical application architecture Logical application Business function	Not mandatory