IBM

VisualAge Pacbase 2.5

Pacbench Client/Server

User's Guide – Volume I

Concepts - Architectures - Environments

DDOVC000255A

Note

Before using this document, read the general information under "Notices" on the next page.

According to your license agreement, you may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:
http://www.ibm.com/software/ad/vapacbase/support.htm

Consult the Catalog section in the Documentation home page to make sure you have the most recent edition of this document.

5th Edition (December 1999)

This edition applies to the following licensed program:

- **VisualAge Pacbase Version 2.5**

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at:
http://www.ibm.com/software/ad/vapacbase/support.htm
or to the following postal address:
IBM Paris Laboratory
VisualAge Pacbase Support
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

NOTICES

TRADEMARKS

Table of Contents

Foreword

The Pacbench C/S function of VisualAge Pacbase allows you to develop the components of a Client/Server application :

- **The Server component** – with a wide range of target environments.

  ☞ The term Server component is a global term used to designate the various software elements[1] at work in a VisualAge Pacbase Server component.

- **The Client component** – *graphic* (for both VisualAge for Java and Smalltalk environments, and COM[2] standard environments) and *TUI* (Textual User Interface, also referred to as character-mode or dumb client) on a wide range of target environments.

The Middleware component – which ensures the communication between Client and Server, uses standard communication protocols[3] totally encapsulated.

Pacbench C/S Documentation

- The ***User's Guide***, in three volumes :
  - The present volume (*Volume I)* describes the main principles of the VisualAge Pacbase Pacbench Client/Server Function and the software architecture of the generated applications. It also presents the development and execution environments available as well as the physical architectures according to the different types of environments.

    Whatever your responsibility in a project with Pacbench C/S, it is recommended to read this volume [REF : DDOVC_].
  - *Volume II : Business Logic* is dedicated to the development of the Server component [REF : DDOAU_].
  - V*olume III : Graphic clients* is dedicated to the development of the graphic Client component [REF : DDOVA_].

- The ***Business Logic – TUI Clients Reference Manual*** [REF : DDOA_].

- The ***Graphic Clients – Public Interface of Generated Components Reference Manual*** [REF : DDOVI_].

Typographical conventions in use

The following symbols are used :

☞	points out a note, remark.
👓	refers to further information in another part of the documentation ; the titles of the manuals and chapters are *in italics*.
💡	hint or useful piece of information
⚠	recommends to proceed with caution to perform operations properly (limits for use, risky or irreversible action).

☞ Simple-mode development is not documented in this edition any more.

[1] These elements are presented in section 3.1.1,

[2] Component Object Model

[3] See subchapter 5.1.

1. Introduction

VisualAge Pacbase is the IBM Application Development tool. It covers the whole application life cycle: design, development, maintenance and legacy applications re-engineering. This tool is made up of a series of products, all integrated in a **unique Repository**.

The global consistency of Clients and Servers is entirely ensured by the VisualAge Pacbase **Information Model**.

1.1. Pacbench C/S Functionalities

Pacbench C/S features four functionalities:

- The **Business Logic** functionality

  It is dedicated to the development of the Server component in the *VisualAge Pacbase* environment.

  👓 Documented in the *Pacbench C/S User's Guide, Vol. II : Business Logic*.

- The **GUI** functionality

  It is dedicated to the development of the graphic Client component in the *VisualAge for Java*[4] or for *Smalltalk workstation*, or in an environment adapted to the *COM* standard.

  ☞ VisualAge for Java or Smalltalk allow to develop standard graphic Client components and also applications accessed from the Web (Intranet or Internet).

  There are two parts in this development :

  - Use of an object called **Proxy** directly generated from the specifications of the Server component.
    The Proxy object remotely commands – from the Client component – the services provided by the Server component.

    👓 The principles of Proxy objects are documented in detail in the *Pacbench C/S User's Guide, Vol. III : Graphic Clients* and in *the Graphic Clients : Public Interface of Generated Components* Reference Manual.

  - Development of the graphical interface and specific local processing regarding the graphic Client.

    👓 Refer to the documentation of VisualAge for Java or Smalltalk or to the documentation of development tools adapted to the COM standard.

- The **TUI Client** functionality

  It is dedicated to the development of the TUI Client component in the *VisualAge Pacbase environment*.

  👓 Refer to the *Business Logic* - *TUI Clients* Reference Manual.

[4] You can also use the Java Development Kit, version 1.1 or higher, or other tools of the same functional level.

- **VisualAge for Java or Smalltalk <> VisualAge Pacbase bridge** functionality

  This functionality enables you to use a unique reference Database, the VisualAge Pacbase Repository, in order to store – via a *backup* – the references of objects proceeding from VisualAge for Java or Smalltalk. These objects can thus benefit from the management, cross-referencing and security functions of the Repository.

  ☞ The VisualAge for Smalltalk bridge also allows you to save and retrieve source code.

  Refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

1.2. Independence between Server and Client

Servers can be accessed by TUI Clients as well as by graphic Clients.

This independence of Server from Client components allows for a smooth evolution towards graphical interfaces and does not jeopardize the global architecture of the system in use.

In this context, the Pacbench C/S function offers the possibility of developing applications where graphic Clients (standard GUI or WEB, COM) and TUI Clients (IBM 3270, BULL Questar...) can coexist.

Servers are then developed independently of Clients, knowing that some specific rules apply when developing for a graphic Client.

These rules are clearly stated, under the generic mention 'Graphic Applications' in the *Pacbench C/S User's Guide, Vol. II: Business logic.*

2. Principles of the Pacbench C/S Function

2.1. The Logical View Concept

The **Logical View** is a specific entity of the VisualAge Pacbase Information Model.

It is used to define an information concept in your company's data system. It is reusable as it can be used by several applications.

A Logical View is made up of a collection of elementary data. Its description is independent of the storage medium chosen to store its instances.

To be used in a Client/Server application, a Logical View must be called in a **Business Component**. This Business Component makes the link between the structure of the storage entities (**Segments**) and the structure of the Logical View.

A Logical View has an iterative capacity. This capacity is the maximum number of instances that can be processed by a Business Component in one execution.

The specifics of a Logical View are documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic,* Chapter *Logical View.*

2.2. Services Associated with a Business Component

When a **Business Component** calls a Logical View, several data access services are automatically generated according to the information defined in the call of Segments description.

Specific processing can also be defined for each Logical View.

2.2.1. Check and Update

A *check* processes all the intrinsic and correspondence checks indicated in the VisualAge Pacbase Repository. In case of error, it sends an error message to the Client.

An *update* allows to create, modify or delete a Logical View instance in the Database. The action code associated with the update can be implicit or explicit. In case of error, an error message is sent to the Client.

These services are documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Components*, Subchapter *Check / Update Services*.

2.2.2. Selection

This service allows to retrieve on the Client workstation a collection of Logical View instances. This retrieval is based on:

- Standard selection criteria:

  The selection service returns the number of instances requested. If the number of instances returned is lower than the number of instances of the selected collection, it returns the identifier of the first instance to be displayed in the next selection page.

  This service also allows to select one Logical View instance only.

  This service is documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Components*, Subchapter *Selection Service*.

- Customized selection criteria:

  You can add / replace specific clauses in a standard generated SQL access or create a new physical access.

  Also, using the customized selection criteria, implemented by extraction methods, you can retrieve a collection of instances of a Logical View.

  For more information, refer to the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Components*, Subchapter *Inserting Specific code*, section *Insertion Relative to the 'Physical Accesses' Level.*

2.2.3. User Service

A User Service is a consistent processing unit executed upon the Client's request. Each User Service must therefore be identified by a code which will necessarily belong to the Client public interface[5]. For each Logical View called in a Business Component, the developer can define his/her own processing in an insert point reserved for that purpose. Its coding follows the standard rules of the VisualAge Pacbase structured code.

This type of service is documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Components*, Subchapter *Other Services*.

2.2.4. Logical Lock (graphic applications)

A logical lock allows – upon the Client's request – to reserve for one user the update of a coherent set of data.

This type of service is documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Component*, Subchapter *Other Services*.

[5] Notion presented in section **2.4.2.**

2.2.5. Call of a Business Component by another Business Component

This service allows a Business Component to delegate the execution of a service to another Business Component. This allows Business Components to be specialized and reused as much as possible.

👓 This type of service is documented in the *Pacbench C/S User's Guide, Vol. II: Business Logic*, Chapter *Business Component*, Subchapter *Other Services*, section *Call of a Business Component by another Business Component*.

2.2.6. Description of a C/S query

Generally, a logical message between a Client component and a Business Component corresponds to a query sent by the client (or to the response of the server).

A C/S query thus corresponds to a message exchanged between a Client component and a Business Component.

A query includes 1 to n service requests.

A service request triggers the execution of either a series of standard services or a specific procedure.

The possible combinations of standard services included in a message are the following:

- Check (only TUI),
- Read (only TUI),
- Check and update,
- Check, update and selection,
- Check and selection,
- Selection,
- Logical Lock,
- User Service.
  ☞ A message can convey one and only one User Service request.

A Business Component can send a query to another Business Component to execute one of the services mentioned above.

2.2.6.1. User Buffer

A message can also include contextual data related to the execution of the called service. The definition of this data structure and the processing of its contents are the developer's responsibility. This data must be defined in a structure called a *user buffer*.

The corresponding data structure is included in each message exchanged between all the Dialogue's Business Components and the Client components.

👓 For more details, see the *Pacbench C/S User's Guide, Vol. II: Business Logic.*

2.2.6.2. Server Buffer

It is sometimes necessary and sufficient to share data exclusively between the Business Components, whether they are called by the same Services Manager or called between themselves via the call of second-level Business Component.

For example:

- *Computed data (logical identifiers)*
- *Common data such as the authorization levels related to a user's profile, list of users' profiles, data related to security*
- *Indicators (error indicators more precise than the generated indicators, indicator of the real update, transactions counter…)*
- *List of root node instances to be sent to all the servers which process dependent instances*
- *etc.*

With the User Buffer, you can define a memory area shared between the Business Components which participate in the execution of a request.

2.2.7. Initialization / Termination Business Component

An *Initialization/Termination Business Component* (**I/T** *Component*) is used to implement specific procedures before and after the execution of a request associated with a Folder.

☞ This functionality is then only available for graphic applications using the development of a Folder.

An **I/T** Component is called by the Services Manager before the first call of the Business Component associated with the request to be processed and after the last executed Business Component.

An **I/T** Component is available either for an Initialization or for a Termination process.

For an initialization process, the only data that can be processed in input is that sent by the Client component via the User Buffer.

An **I/T** Component meet various application needs:

- The specific procedures are associated with more than one Business Components managed by the Folder,
  For example, the application must modify data in a node following the update of data in another node. This is the case when command lines are added on an existing command; the total amount of the command must be modified (termination process).
- The specific procedures are inserted directly in the **I/T** Component, by an algorithm on data managed via the Server or User Buffer.
  For example, the application must check the global validity of data – at the Folder level – regarding the cumulated controls carried out by all the Business Components (termination process) of the Folder.
- The specific procedures are implemented via the call of a Business Component – external to the Folder – managing a Logical View, also external to the Folder.
  You can thus ask for external memory (initialization process) and ask for the deallocation of this memory (termination process). The ***I/T*** *Component also allows you to manage the authorization controls on the Folder (initialization process).*

An **I/T** Component is used to execute services associated with all the Business Components of a Folder.

The implementation of an **I/T** Component is documented in the *Pacbench C/S User's Guide, Vol. II : Business Logic*.

2.3. Data Representation (graphic applications)

2.3.1. Data Automatic Navigation

2.3.1.1. Functional and Technical Contexts

- **Functional Context**

As a general rule, business activity is modeled by a set of information concepts undergoing transformations.

In VisualAge Pacbase, an information concept is represented by a Logical View which is managed by a Business Component. A Logical View constitutes an homogenous aggregate of elementary data.

These transformations are distributed among the company's different functional processes.

Each transformation, or business act, is made up of a set of processing actions which modify the Information System's contents from one stable state to another.

Also, different business acts can modify one consistent set of data.

In this case, it is interesting that functional relations between the different information concepts in this data set be automatically managed via their grouping in a logical envelop.

☞ If the possible transformations affect only one information concept, it is not necessary (even though it is possible) to use a logical envelop because there is no interaction to manage. In that particular case, this type of development is referred to as single-view development. For more details, see paragraph **2.3.2.**

However, both options can be managed in one same application.

- **Technical Context**

In order to handle the processing load resulting from the business acts simultaneously submitted by a large number of end users, computer manufacturers developed the concept of transactional monitors. These systems are designed to handle a maximum number of processing requests in a minimum amount of time.

In most cases, the transactional resources used by different users are identical and the only way to attain an efficient transactional flow is to monopolize the processing resources required for the shortest possible length of time.

Transactional monitors use a standard resource sharing unit which corresponds to the time taken between the reception of a message, its processing and the sending of the response.

Constraints inherent to this sharing principle require that data needed to reach another stable state, be transmitted altogether to the transactional monitor.

2.3.1.2. Folder-Based Development

According to the logic of high-performance Client/Server architectures, consistent information concepts must therefore be defined, so that they can be easily manipulated through their different graphic representations. Also, a business act must be implemented in order to modify the instances of related information concepts and to send the server the queries for a transition to a stable state in only one message. Within VisualAge Pacbase, this action is modeled by the **Folder** concept.

From a methodological point of view, one may consider that a Folder is derived from a LDM (Logical Data Model) type of entity. The transition consists first in associating a Folder with a set of Segments, linked by a network of relations in a LDM. Then each one of these Segments is associated with a server (a **Business Component)**, which implements the actions allowed on this Segment. As a result, the number of root-Segments defined in a LDM is identical to the number of Folders derived from this LDM.

In this context, requisite information – described by the Folder – is easily made available to the Client workstation where the business act can be dealt with as a whole. Once the business act is completed on the Client workstation, the resulting Folder image is sent back to the Server component.

In addition to reconciling functional description constraints and performance concerns, it can be seen that this type of ‘download/upload’ mechanism improves overall system performance by reducing the number of server calls, which in turn increases the availability of the transactional monitor.

A folder–based application benefits from the following advantages:

- The VisualAge Pacbase Repository represents and manages the functional integrity of a set of Logical Views for each business act.
- Selection and update of a set of instances associated with several Logical Views are automatically performed in just one exchange.

2.3.1.3. Representation in the Metamodel

2.3.1.3.1. The Folder Entity

The **Folder** entity allows to describe a set of elementary data aggregates, i.e. Logical Views, and the functional relationships between them, thus defining a complex information concept – equipped with access and processing services. Dependencies between the different Logical Views and data integrity are managed in standard via the Folder.

For example, if in a company, an order makes sense only if it is linked to a client, the dependence between both concepts of information is defined in a Folder; each concept constitutes a consistent set of data which allows for their manipulation through their different graphic representations.

Functional relationships define the behavior of instances contained in two linked data aggregates. They are of two types:

- The hierarchical relationships, where an instance of a child aggregate depends on one and only one instance of the parent aggregate.
- Referencing relationships, where an aggregate instance references one and only one instance of the referenced aggregate or none.

Each aggregate in a Folder is called a node. There are three types of nodes:

- The root node, only one per Folder, is the parent of all the depending nodes. This aggregate does not depend on any other aggregate.
- The depending node, which is linked to a root node or a depending node by a hierarchical relationship.
- The reference node, which is linked to a root node or a depending node by a referencing relationship.

A node corresponds to a Logical View managed by a Business Component.

Implementing a folder-based development is documented in the Pac*bench C/S User's Guide, Vol. II : Business Logic*, Chapter *Folder and Folder Views*, Subchapter *Folder*.

The generation of a Folder occurrence produces a **Services Manager** capable of interpreting and formatting all of the services associated with the Folder before calling the relevant Business Component.

For more details on the Services Manager, refer to section **3.1.4**.

If the description domain of a Folder is limited enough to be used in full in a given application, an additional generation at the Folder level allows to obtain directly the whole set of classes used to manage the Folder in a graphic application.

2.3.1.3.2.The Folder View Entity

Considering the fact that an application's functionality does not necessarily cover the wide scope of a Folder definition, each business act must be able to work on a partial view of a Folder which guarantees the integrity of the transformation of its instances.

This view is represented by the **Folder View** entity.

- A Folder view represents all or part of the Folder and must then contain the Folder root.

  The consistency of a Folder View description in relation to that of its related Folder is automatically controlled by the VisualAge Pacbase WorkStation.
- A Folder can contain several Folder Views. Each view corresponds to a consistent representation of some of the Folder nodes, according to the data required to implement a business act.
- An application can use several Folder Views proceeding from different Folders.

For complete information on how to specify Folder Views, refer to the *Pacbench C/S User's Guide, Vol. II : Business Logic*, Chapter *Folder and Folder Views*, Subchapter *Folder Views*.

The generation of a Folder View creates in the Client component an object called **Folder View Proxy** which allows a VisualAge Pacbase GUI application to read, modify and transform the Folder's instances.

The Folder View Proxy is documented in paragraph **2.4.3**.

Specifying Folder Views is not a requisite. Indeed, the description domain of a Folder may be limited enough to be used in full in a given application. In this case, an additional generation at the Folder level allows to obtain directly the whole set of classes used to manage the Folder in a graphic application.

2.3.2. Particular Case : Single-View Development

The single-view development is particularly interesting when an elementary data concept – a Logical View – can be independently managed by the Client component.

In this case, data representation is modeled via the Logical View only managed in a Business Component, without specifying a Folder or Folder Views.

The generation of this Business Component produces two sources:

- a **Services Manager**
- a **Folder View Proxy**

For more details on the Services Manager, see section **3.1.4**. The [single-view] Folder View Proxy is presented in paragraph **2.4.3.1**.

2.4. Proxy Objects Generated for the Graphic Client

2.4.1. Introduction

Some objects can be proposed by the development tool, such as window, push-button, entry box, etc. However, to satisfy specific functions such as the taking into account of a Communication protocol or the encapsulation of specific management functions, you must develop the corresponding objects yourself.

One of the goals of Pacbench C/S is to carry out a large part of this development in an automatic and transparent way.

Since the Business Logic generator produces Business Components with the same generic behavior, it is possible to transpose this generic behavior into classes whose objects – called **Proxy** – are used to execute the services of associated Business Components.

In the Client development workstation, a generator controlled by a user interface takes as input the output file resulting from the extraction of the Folder View or the Business Component specified in VisualAge Pacbase.

The generated objects are the following :

- Folder View Proxy
- Elementary Proxies

  Elementary Proxies do not exist autonomously and are necessarily associated with the Folder View Proxy.

Proxy objects generation is documented in detail in the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*.

These objects define - in a given environment - an object which is representative of another object in a remote environment. The object of the remote environment corresponds with the Business Component and with the Logical View it manages.

2.4.2. Visual Programming and Public Interface

In general, the principle for building a GUI application is based on calling different objects (visual or non visual) and on establishing connections between the characteristics of these objects. These characteristics constitute the Public Interface of these objects. The generated Proxy objects, like all the objects proposed by the Client development tool, come therefore with a public interface.

This public interface is made of :

- a set of *properties* (*attributes* for Smalltalk and COM) which are used to transfer the execution parameters of a service and to represent the Logical View instances in the form of a list or a detail,
- a set of *methods* (*actions* for Smalltalk and COM) which correspond to the different services that the Business Component associated with the Proxy can execute,
- a set of *events* (plus errors and exceptions for Java) which characterize the result of the requested execution of a server or local service.

The public interface of the generated components is documented in thematic order in the *Graphic Clients: Public Interface of Generated Components Reference Manual.*

In the VisualAge workstation, the Proxy object corresponds to a non-visual *bean* or *part*. When building your application, you place this object on the *Free Form Surface*.

2.4.2.1. Implementation of Generic Classes

All variable classes inherit from an abstract class, reserved for the developer, which itself inherits from another abstract class containing all the common and invariant methods of variable classes.

Such classes, called generic classes, are installed with the product.

Generic classes are used to avoid the multiplication of classes at each new generation of Proxy objects. They improve applications' memory management and enable developers to add generic behavior in these objects.

This functionality is not available for COM targets.

2.4.2.2. Parameterization of the Generated Classes Names (VisualAge for Smalltalk)

In order to conform the coding of the generated classes to the naming standards chosen by a company, a system of class codes parameterization is implemented in the VisualAge Pacbase Repository.

2.4.3. Proxy Objects

A **Folder View Proxy** is the result of a Folder View generation. A Folder View being described by nodes (Business Component/Logical View pairs), the generation of the Folder View produces a set of classes corresponding to all the nodes which make up the Folder View.

The objects of these classes are **Elementary Proxy objects.** There are three types of Elementary Proxy objects:

- The Root Proxy, corresponding with the root node
- The Depending Proxy, corresponding with the depending node
- The Reference Proxy, corresponding with the reference node

Once incorporated into your client application, the Folder View Proxy is used to handle the Elementary Proxy objects it contains. It also manages automatically their functional interactions.

The Root Proxy is an Elementary Proxy which has its own specific characteristics; only the Root Proxy can execute server update actions and ensure error handling.

Each Elementary Proxy object is associated with one Business Component, for one Logical View.

The public interface of an Elementary Proxy object depends on the characteristics of the Logical View associated with the node, on the Business Component which manages it and on the type of node.

Elementary Proxy objects are thus used to execute the services of the Business Component associated with them ; the Business Component enables the graphic Client to send queries which are service requests to the corresponding remote server. These requests are automatically generated.

These services include:

- The structuring of data
- The encapsulation of the implementation of the Business Component services
- Temporary storage of data to be sent to the Business Component
- Error handling
- Event-driven translation of the Business Component return codes
- Communication
- Local data integrity check in relation to the check defined in the Repository
- Automatic control of version consistency between Client and Server Components
- For VisualAge for Smalltalk : the management of data representation with the Quick Form function.

2.4.3.1. Particular Case : Single-View Development

In the case of a single-view development, a Folder View Proxy is also generated, from the Business component. It is composed of one Elementary Proxy : the Root Proxy.

☞ For the *VisualAge for Java* or *Smalltalk* <> *VisualAge Pacbase* bridge functionality, this particular Folder View Proxy is considered as a Logical View Proxy.

2.5. Graphic Client Development Facilities

All the subjects introduced in this subchapter are documented in detail in the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

2.5.1. Local Cache

The **local cache** automatically manages :

- the consistency and synchronization of a Folder node instances :
  - loading of parent identifiers of depending instances according to the navigation defined between the Folder nodes.
  - loading of the referencing Data Elements of a root or depending node on the transfer request of a reference node instance.
  - checking of the existence of parent depending instances when creating a depending instance.
  - checking of the cardinalities of each link used by a sub-schema before sending a server update.
  - management of the different status associated with a Folder instance (read, read only, modified, server error) and of the induced rules.
  - refreshing of the local image (VisualAge for Smalltalk) with or without local updates.
  - management of each node's status and memorization of the information required to execute paging operations.
- the impact of local updates on the presentation by list attribute:

  An automatic loading of the list of instances of a root or depending node in a Folder (represented by the `rows` or `userRows` attribute) will be carried out as soon as a local update of this node is performed (stored in the attribute `detail` or `userDetail` attribute).

  Each instance created locally in a Folder is placed in the `rows` attribute at the level which respects the ascending sort of instances on their identifiers.

  The same rules apply on the corresponding attributes associated with User Services.
- calculation of the effective update transactions to be transmitted to the Server component:

  The management of effective transactions consists in calculating the resulting update of various local updates made on the same instance of the Folder's node.

  It checks the creation of duplicate instances.

2.5.2. The Management of Collections

When executed, a selection or read action sends a collection of instances to the local cache.

This collection of instances is managed in the local cache according to two different modes:

- in automatic mode (default), the current collection of the local cache is replaced by the new collection sent following a new execution of a selection or read action,
- in manual mode, the current collection of the local cache is completed by the new instances sent following the execution of a series of selection or read actions[6]. So this collection cumulates new instances after each new execution, and then constitutes an heterogeneous collection. To determine the beginning of a new collection, you must then explicitly clear[7] the cache's current collection.

You can select the appropriate management mode for each Elementary Proxy of the Folder View Proxy.

☞ Toggling from one management mode to another does not modify the cache's current collection.

If you select the manual mode:

- the root and reference nodes parameterized to operate in non-extend paging operate in extend paging,
- the paging actions are synchronized with the last selection action sent by the node,
- the 'collection change' event is sent following the execution of the action which clears the current collection.

2.5.3. The Exchange Manager

The **Exchange Manager** *automatically* optimizes the volume of the information flow exchanged between a Client component and a Server component.

According to the maximum size of the communication buffer specified for the Folder, it transforms a query object into one or more message objects carried out via the middleware. Reciprocally, it transforms one or more message objects associated with a server response into a response object.

The calculation of the number of messages required for the transfer of a query or a response is automatic.

The protocol used is adapted to the transactional-type exchanges. A work file[8] used by the Services Manager[9] allows to store temporarily the information required for the constitution of a query or its response when it uses several messages.

[6] If the Business Component sends an instance which is already present in the current collection, this instance is refreshed if the instance of the current collection has not been modified locally.

[7] The action which clears the current collection is also available in the automatic mode to reset a list.

[8] Documented in section 3.1.3.

[9] Documented in section 3.1.4.

2.5.4. The VisualAge for Smalltalk Quick Form

- Automatic creation of containers with one column per Data Element of the Logical View via a *Quick-Form* on the **`rows`** attribute of a Proxy. The title of the column corresponds to the Data Element’s short label in VisualAge Pacbase.

- Graphic representation of the Data Elements can be defined in the VisualAge Pacbase Repository. In this case, it is automatically managed by the *Quick-Form* function.

- For a Data Element, when a graphic representation has been associated in the VisualAge Pacbase Repository with its value list and then when the presentation of this Data Element is managed by the *Quick-Form* tool, this list of values is automatically managed.

2.5.5. VisualAge for Smalltalk Error Manager

A class dedicated to error management is generated independently of the Folder View Proxy. This class is supplied with an action which, from a row in the required error messages, sets the Folder’s node to restore the Logical View instance which caused the error, and then to send the corresponding event (**`errorContextRestored`**).

An error is defined by three characteristics :

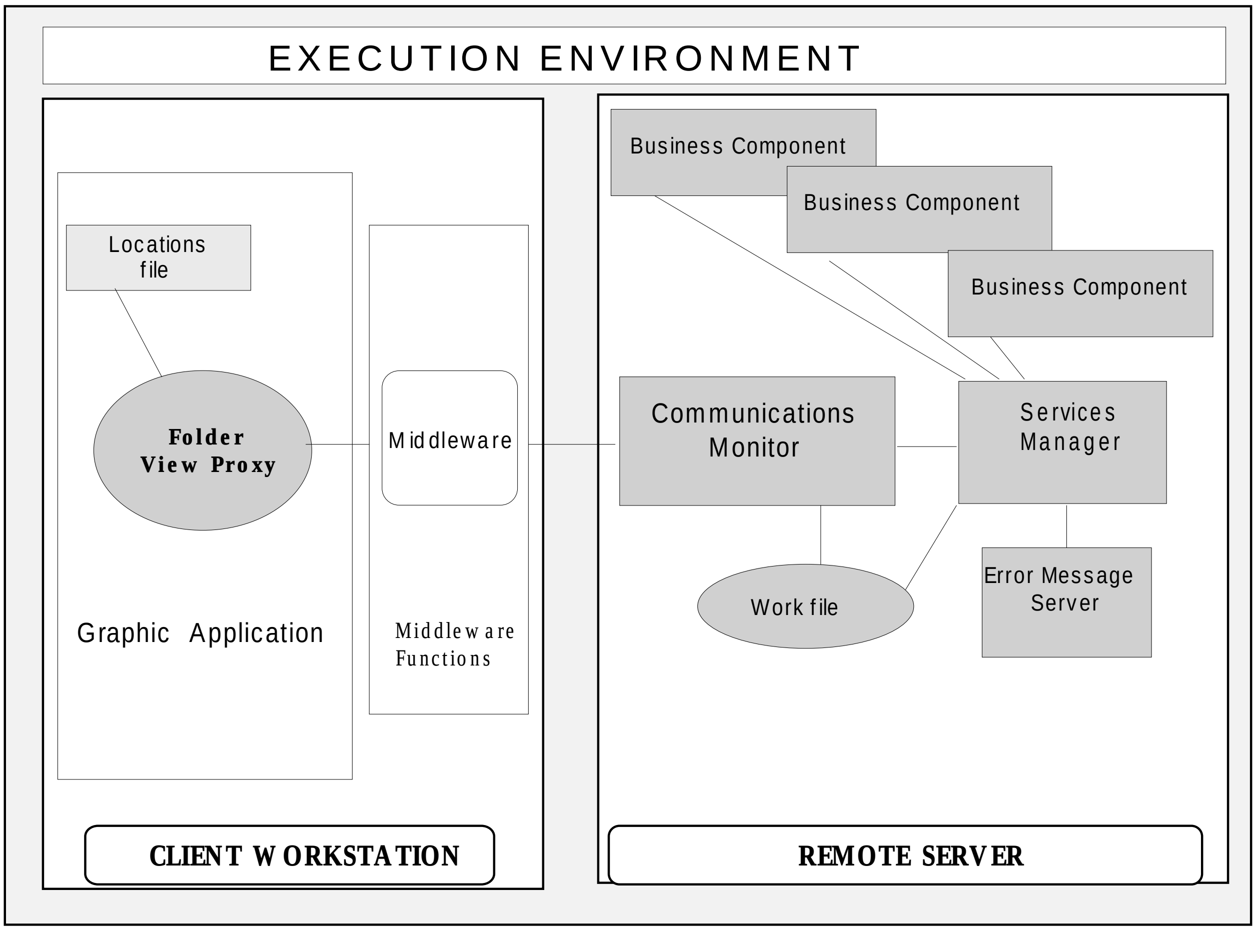
- its key,
- a gravity code,
- the message itself.

3. Software Architecture of Generated Applications

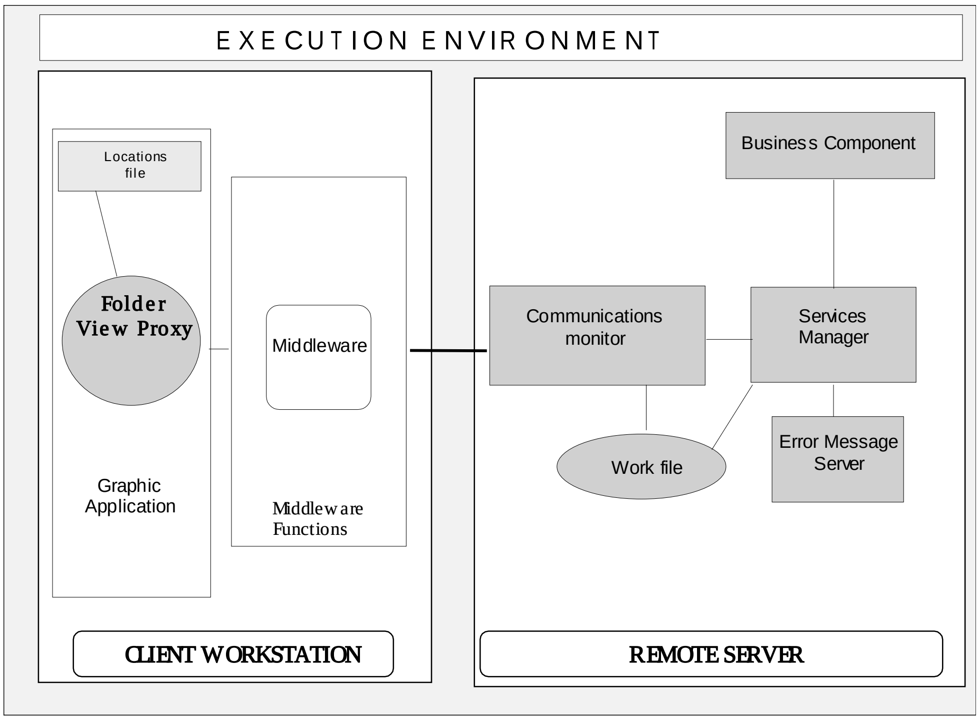
The various elements present in the following charts are detailed hereafter.

3.1. Graphic Applications

3.1.1. General Schema



3.1.2. Single-View Development



3.1.3. The Communications Monitor

The **Communications Monitor** handles the following functions:

- the definition of communication functions (message sending and receiving) according to the target environment.
  - *Multi-environments management:*
    A Pacbench C/S application can be executed on different environments. In this case, a monitor being specific to an execution environment (generation variant and communication type) there will be as many Communications Monitors as environments.
    Moreover, several communication protocols can be used for the same environment (example: CICS ECI and CICS CPIC).
- Checking each message received.
- Controlling the sending/receiving of the query.

  ☞ According to the size allowed for the physical message, several physical messages may need to be issued in order to obtain the complete logical message. A **work file** must therefore be available for a temporary storage of the current query.
- Once the query is completely received, sequential processing of services which compose it by successive calls to the corresponding Services Manager.
- Transactional management (**`COMMIT`**/**`ROLLBACK`**)

  The Communications Monitor uses the **`COMMIT`** and **`ROLLBACK`** services of a Database or a transactional monitor depending on its generation variant.

  The transactional management is always of the **LUW Server** type (Logical Unit of Work). Accordingly, the Server component takes in charge the Database integrity.

  A query is processed as a whole.
  The server part executes – before return to the client – a **`COMMIT`** or a **`ROLLBACK`** according to the error context (protocol error or application error) established at the end of the query processing. In case of error, query processing is ended and an error message is sent back.

  Thus, all the resources monopolized by a query processing are made available once the response has been sent.

👓 For more information on the specifics of a Communications Monitor, refer to the *Pacbench C/S User's Guide, Vol. II : Business Logic,* Chapter *Communications Monitor (graphic application).*

3.1.4. The Services Manager

For each query sent via the Communications Monitor, the Services Manager processes the service requests and sends back the response to the Communications Monitor.

☞ A query targets one and only one Services Manager and includes 1 to n service requests.

The Service Manager processes a query :

- by interpreting the contents of the service into 1 or n elementary requests addressed to the corresponding Business Components.

  ☞ A service is processed by a Business Component which can, in turn, call another Business Component.

- by transmitting whenever relevant the corresponding user buffer to the called Business Component and by updating its contents after the call from the Business Component if the user buffer has been sent by the Client.

There are as many Services Managers as there are Folders specified for the application (or as Business Components in case of single-view development).

The Services Manager is a purely technical component, in which it is not possible to insert specific code.

3.1.5. The Error Message Server

The error message server generates the messages associated with the errors detected by the Business Components. These messages are stored in a dedicated and generated file.

👓 The management of these errors is documented in the *Pacbench C/S User's Guide – Vol. II : Business Logic*, Chapter *Error handling*.

👓 Error handling on the Client side is documented in the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

3.1.6. Communication Achitecture

3.1.6.1. Middleware and Communication Protocols

The middleware functions used by a Proxy object manage the exchanges between the Client and Server components of an application by using leading communication protocols on the market place[10].

The implementation of the selected communication protocol is totally encapsulated. Moreover, the middleware functions are independent of the protocol. Therefore, a change in the communication protocol, for the same type of LUW, does not imply regenerating the Client components.

At the Client component level, the selection of the communication protocol consists in associating a physical location (environment where the Server component is executed) with a logical one. This association is performed via the parameterization of the locations file (**`VAPLOCAT.INI`**) or via a graphical utility supplied for the COM target.

👓 For a complete information, see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

3.1.6.2. Customized Middleware

The use of a customized middleware consists in replacing the Pacbench C/S midddleware by a specific one.

This operation is only possible in a VisualAge for Java or Smalltalk development context.

Usually, this operation consists in using inheritance mechanisms to modify the basic actions associated with the calls of remote servers.

The classes that manage server calls are generic. They are installed with the product and are independent of the variable classes associated with each Proxy object.

These generic classes enable server call actions to be modified once only so that all the Proxy objects can use the new middleware.

These modifications are permanent; they are not overwritten when importing new proxy objects or when re-installing generic classes.

👓 The customization of the middleware is documented in the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

3.1.6.2.1. Local User Buffer (VisualAge for Smalltalk)

This buffer is used to manage – via the Proxy – data from the VisualAge for Smalltalk client. The data can be used by a customized middleware.

For a given Business Component, only one local user buffer can be defined ; all the generated proxies managed by a Business Component thus contain the same instance of the local user buffer associated with the same Smalltalk class.

If data in this buffer is independent of Proxy objects, you can reuse it for all the Business Components of an application, even for all the applications of the site.

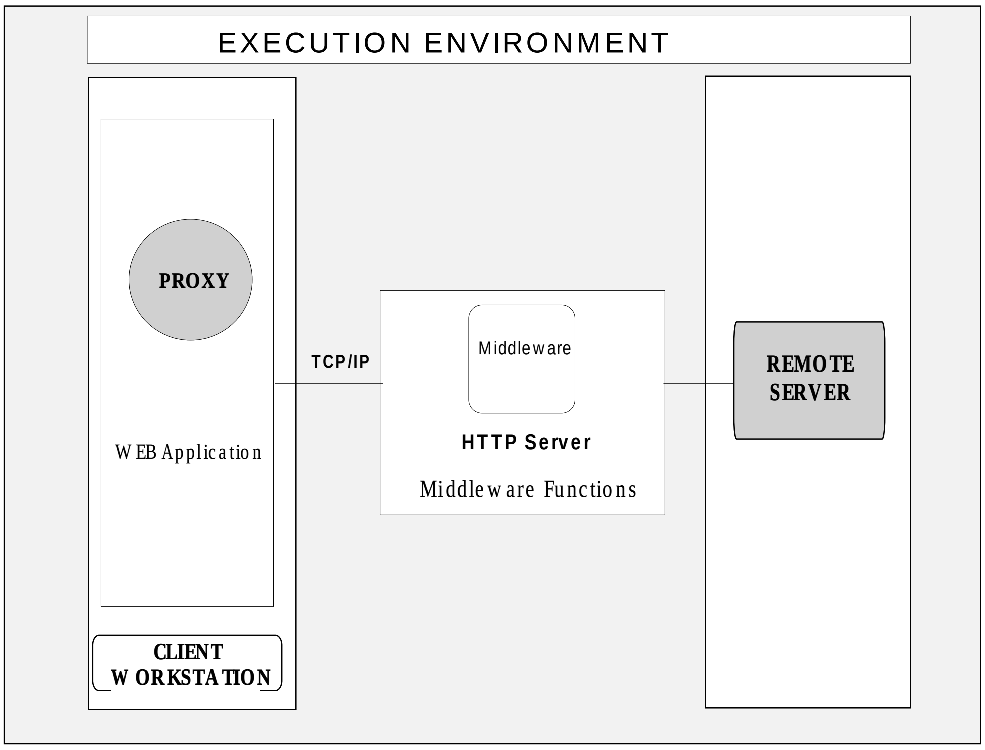
👓 For more information on how to use a local user buffer, refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

10 Refer to subchapter **5.1**.

3.1.6.3. Particular Case : WEB Applications

In case of WEB applications, the middleware is not installed on the client workstation.

3.1.6.3.1. Smalltalk Applications



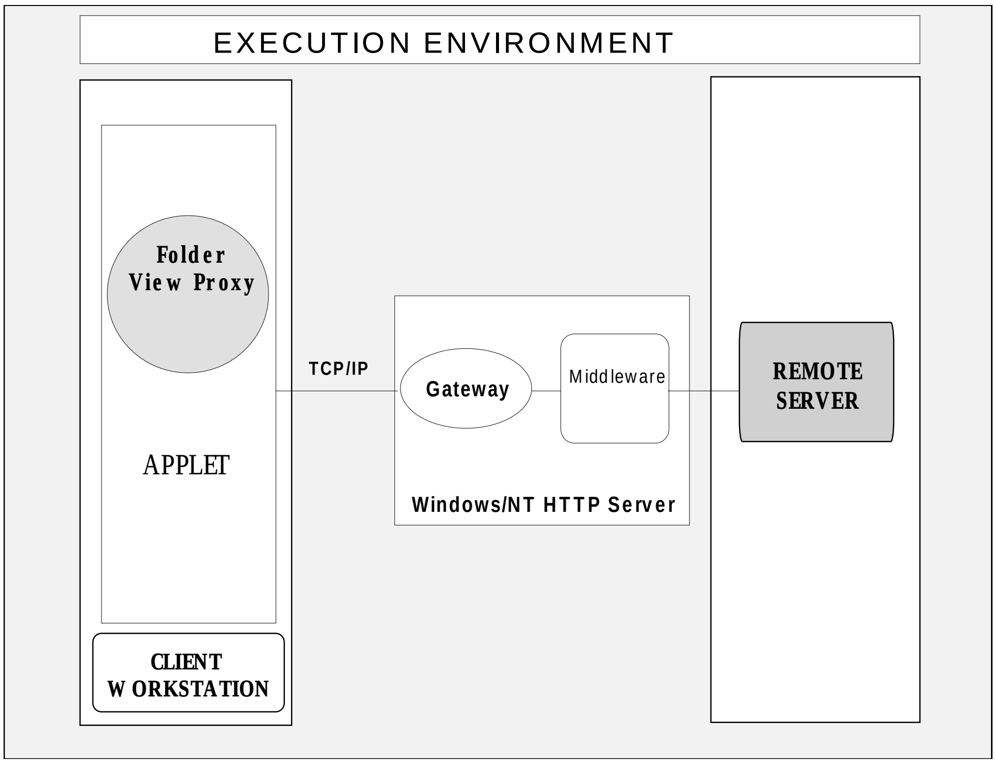
☞ The architecture of the WEB Application and Server components depends on the selected development mode (standard, single-view), refer to the above schemas).

For more details, refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

3.1.6.3.2. Java Applets (Intranet / Internet)

The implementation to be carried out calls on a Gateway (**Intranet**) or on both a Relay and a Gateway (**Internet**).

- ***Access to the Server component via a Gateway***

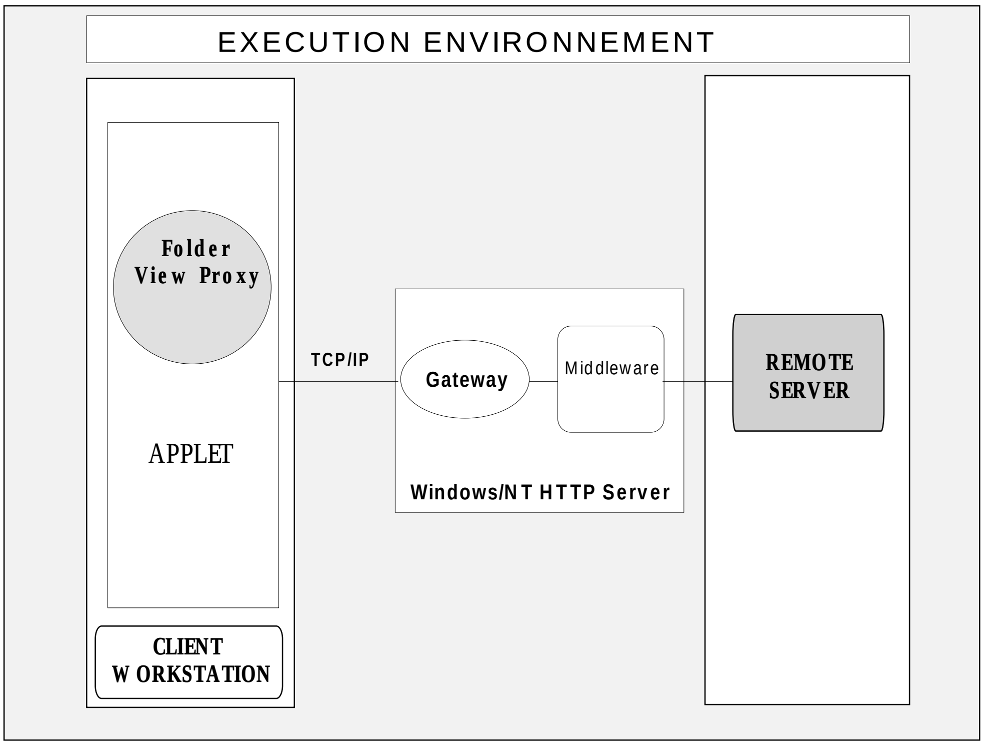


The Gateway is a C++ executable program supplied for Windows/NT. It is shared by all the applets downloaded from the same HTTP server, and must then be run permanently.

☞ The Applet component remains unchanged compared to the graphic application component of the general schema. The architecture of the Server component depends on the selected development mode (standard or single-view, see the above schemas).

For more details, refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*.

- ***Access to the Server component via a relay and a Gateway (for Internet)***



This solution – required for the Internet applications – enables to lighten the HTTP server's load by undertaking the access to the middleware, the management of the servers and contexts of each connected client.

The relay is an executable program (`relay.exe`) compiled for Windows/NT and which enables to relay the queries reaching the HTTP server to another computer hosting the Gateway and the middleware.

☞ The Applet component remains unchanged compared to the graphic application component of the general schema. The architecture of the Server component depends on the selected development mode (standard or single-view).

For more details, refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients.*

3.2. TUI Applications

This subchapter introduces the main principles of a TUI application. However, a more specific presentation of the TUI client can be found in the *Pacbench C/S User’s Guide, Vol. II : Business Logic*, Chapter *TUI Clients Development*.

Complete documentation on the development of TUI Client is found in the *Pacbench C/S : Business Logic & TUI Clients Reference Manual.*

3.2.1. Monitors

3.2.1.1. Introduction

It can be useful to group common information and processing (communications management, compacting, trace, **COMMIT/ROLLBACK**, site-specific features) in technical components, i.e. the Monitors.

For some platforms, e.g. Microfocus and Tuxedo, it is a requirement.

3.2.1.2. The Client Monitor Component

This ensures the link between the Clients, and either calls a monitor which manages communications between an application's Business Components, or directly calls the Business Component in question.

Information is exchanged between the Monitor and the Clients through a communication area which includes the following data:

- Technological data specific to the Clients,
- Data required for the call and the Business Component's answer, such as:
  - the name of the service to call,
  - the name of the Server monitor (this name is defined in the Dialog which includes the services),
  - the code of the Logical View processed by the Client,
  - the branching function which returns control to the proper Client function.

3.2.1.3. The Server Monitor Component

It calls the appropriate service and passes control either to the Client monitor or directly to the Client.

The **main processing generated** in the Server monitor is the:

- Reception of the logical message
- Service call
- Formatting of the message before the return to the Client.

This processing mainly depends on the communication mode used.

3.2.2. Options without Monitors

3.2.2.1. Client Processing

Each Client directly calls a Server monitor or a Business Component.

In an architecture without monitor, the call processing is directly coded in each Client component.

Each Client manages the sending of its data to the service, according to the communication mode in use. The return to the Client is performed just after the call function and the processing is performed in sequence.

You can always add specific processing by inserting code before or after the service call.

3.2.2.2. Business Component Processing

Each Business Component is directly called by the Client or by the Client monitor.

The structure of the service does not depend on the presence or absence of a monitor. But you must enter the processing specific to the communications technology or the security system in use in each Business Component.

Processing specific to Database management is coded in each service.

Initialization processing:

Database connection or opening is optionally coded at the beginning of the Business Component processing.

Management of **`COMMIT`** and **`ROLLBACK`**:

An indicator allows you to condition the execution of these commands. This function will be executed via a **`PERFORM`**; you will be able to modify its contents and its call via a user procedure in an **`*R`**-type function. It is standardly executed at the end of the Logical View processing before the return to the Client.

4. Development Environments

The aim of this chapter is to define the technical environment required for the proper operation of Pacbench C/S in terms of both hardware and software.

This chapter does not give information on the middleware. In test phase, it is recommended to use the middleware which will be used during execution. For more details, refer to Chapter *Execution Environments*, Subchapters **5.2** and **5.3**.

4.1. Remote Components

The *Business Logic* and *TUI Clients* functionalities of the Pacbench C/S are operational on a wide range of IBM and non-IBM platforms:

- IBM (MVS-CICS, MVS-IMS, VSE-CICS, AIX, OS/2)
- Bull (Escala, DPX20, GCOS7, GCOS8)
- Digital Equipment UNIX
- Hewlett-Packard HP/UX
- Sun Solaris
- Tandem Integrity-IRIX
- Unisys 2200 Series
- Microsoft Windows NT

Configuration information is supplied in the *VisualAge Pacbase 2.5 Operations Manual, Vol. I – Environment and Installation* specific to each platform.

4.2. Local Components

☞ When software release numbers are specified, the *Graphic Clients* functionality operate on more recent releases only if they strictly ensure upward compatibility.

For more details on the parameterization of external software, refer to the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*.

4.2.1. VisualAge for Smalltalk Workstation

4.2.1.1. Standard Applications

4.2.1.1.1. Windows 95 or NT

• Hardware

Subject to changes, the recommended minimum hardware configuration is :

- Intel Pentium or compatible processor
- 32 Mb. 48 Mb of RAM recommended for simultaneous use of a Database manager or communication services.
- VGA graphic monitor
- CD-ROM drive
- network adapter
- Initial disk space : 50 Mb

• Software

- Windows 95, NT 3.51 or NT 4.0
- Communication software adapted to selected middleware (see the *Pacbench C/S User’s Guide, Vol. III: Graphic Clients*).

4.2.1.1.2.OS/2

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - Intel Pentium or compatible processor
  - 32 Mb, 48 Mb of RAM recommended for simultaneous use of a Database manager or communications services
  - VGA graphics monitor
  - CD-ROM drive
  - Network adapter
  - Initial disk space: 50Mb

- Software
  - OS/2 WARP Version 3.0 or 4.
  - Communication software adapted to selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic clients.*

4.2.1.2. WEB Smalltalk Specifics

In addition to the prerequisite of standard applications, the development of a WEB application requires WEB Connection (a VisualAge feature).

4.2.2. VisualAge for Java Workstation

The environments described below concern the VisualAge for Java development workstation. However, the proxy objects and the Pacbench C/S generator can be used in any Java environment supporting the JDK 1.1.

4.2.2.1. Windows NT and 95

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - Intel Pentium or compatible processor
  - 48 Mb, 64 or 96 Mb of RAM recommended for simultaneous use of a Database manager or services communication + 200 Mb available for VisualAge for Java and Pacbench C/S
  - VGA graphic monitor
  - CD-ROM drive
  - network adapter
  - Initial space disk : 50 Mb

- Software
  - Windows 95 or NT 4.0
  - The communication software adapted to selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*).
  - TCP/IP

4.2.2.2. OS/2

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - Intel Pentium or compatible processor
  - 48 Mb, 64 or 96 Mb of RAM recommended for simultaneous use of a Database manager or services communication.
  - VGA graphic monitor
  - CD-ROM drive
  - Network adapter
  - Initial space disk : 50Mb

- Software
  - OS/2 Warp version 4.
  - The communication software adapted to the selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients).*
  - TCP/IP

4.2.3. COM Environments

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - Intel Pentium or compatible processor
  - 24 Mb, 32 Mb of RAM recommended for simultaneous use of a Database manager or communication services.
  - VGA graphic monitor
  - CD-ROM drive
  - Network adapter
  - Initial space disk : 50 MB

- Software
  - Windows 95 or NT 5.
  - Visual C++ 4.2 or higher
  - The communication software adapted to the selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients).*

5. Execution Environments

The aim of this chapter is to define the technical environment required for the proper operation of generated applications in terms of both hardware and software.

5.1. Remote Components

Vendors	Server Platforms	Communication Types	Databases
IBM	MVS/CICS	ECI CPI-C/APPC LU6.2 MQ SERIES TCP-IP Socket	ADABAS ALLBASE/SQL AS/400 Phys. File DB2 family DDL Tandem DL/1 DM4 TP DMS DMS II IDMS IDS2 INFORMIX INGRES INTEREL/RDBC INTEREL/RFM MS SQLServer NON-STOP SQL ORACLE RDMS SQL/400 SQL/DS SYBASE
	MVS/IMS	CPI-C/APPC LU6.2 MQ SERIES	
	VSE/CICS	ECI CPI-C/APPC LU6.2	
	AIX	TUXEDO XATMI CICS6000/ECI TCP-IP Socket MQ SERIES	
	OS/2	ECI CPI-C/APPC LU6.2 MQ SERIES	
	OS/400		
BULL	GCOS7/TDS	TUXEDO between Client & UNIX TUXEDO/HOST-Connect and CPI-C /XCP2 between UNIX & GCOS7	
	GCOS8/TP8 ou DM4-TP	TUXEDO between Client & UNIX TUXEDO/HOST-Connect and CPI-C /XCP2 between UNIX & GCOS8	
	ESCALA ou DPX20	MQ SERIES TCP-IP Socket TUXEDO XATMI	
	GCOS6		
DIGITAL EQUIPMENT	DIGITAL UNIX	TUXEDO XATMI TCP-IP Socket	
	VMS		
	Open VMS Alpha	TCP-IP Socket	
HEWLETT-PACKARD	HP-UX	MQ SERIES TUXEDO XATMI TCP-IP Socket	
	MPE		
ICL	VME		
MICROSOFT	Windows/NT	ECI TCP-IP Socket TUXEDO XATMI MQ Series	
SUN	Solaris	TUXEDO XATMI TCP-IP Socket	

Vendors	Server Platforms	Communication Types	Databases
TANDEM	Integrity-IRIX	TUXEDO XATMI TCP-IP Socket	
	Guardian	Non Stop TUXEDO Pathway TCP-IP Socket	
UNISYS	2200 Serie	TCIS	
	A Serie		

5.2. Local Components

5.2.1. End User Workstation

5.2.1.1. Windows 95 and NT

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - Intel Pentium or compatible processor
  - 486/33 MHz PC capable of supporting Microsoft Windows 95 or NT
  - 16 Mb of RAM
  - VGA graphic monitor
  - Network adapter

- Software
  - Microsoft Windows 95 or NT 4.0
  - The communication software adapted to the selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*).

5.2.1.2. OS/2

- Hardware

Subject to changes, the recommended minimum hardware configuration is:

  - 486/33 MHz PC capable of supporting OS/2 2.11
  - 16 Mb of RAM
  - VGA graphic monitor
  - Network adapter

- Software
  - OS/2 Version 3 or Version 4
  - The communication software adapted to the selected middleware (see the *Pacbench C/S User's Guide, Vol. III : Graphic Clients*).

5.2.2. COM Environments

The applications based on the use of a COM proxy can be executed on all the environments which can access COM servers.

5.3. Middle Components for the Web

For Web applications, the middleware is not installed on the client workstation.

For more details, see paragraph **3.1.6.3.**

5.3.1. Smalltalk Applications

In addition to the standard applications prerequisites, the execution of a Smalltalk Web application requires an HTTP server supporting CGI[11] 1.1, in a Windows version (3.1, 95 or NT), OS/2 or AIX. This server must host the "**`CGI Link`**" **`abtcgil`** program delivered with the Web Connection feature of Smalltalk.

The end user workstation must be equipped with a browser (Netscape Communicator 4.0x or Internet Explorer 4.0).

5.3.2. Java Applications

In addition to the standard applications prerequisites, the execution of a Smalltalk Web application requires an HTTP server running on a Windows/NT or AIX 4.1 environment.

The end user workstation must be equipped with a browser supporting the JDK version 1.1 (Netscape Communicator 4.0x or Internet Explorer 4.0).

[11] Common Gateway Interface

6. Physical Architecture According to Environment Types

6.1. TUXEDO

6.1.1. Database Management

DBMS automatically supported by TUXEDO Business Components and TUI Client Components are the following :

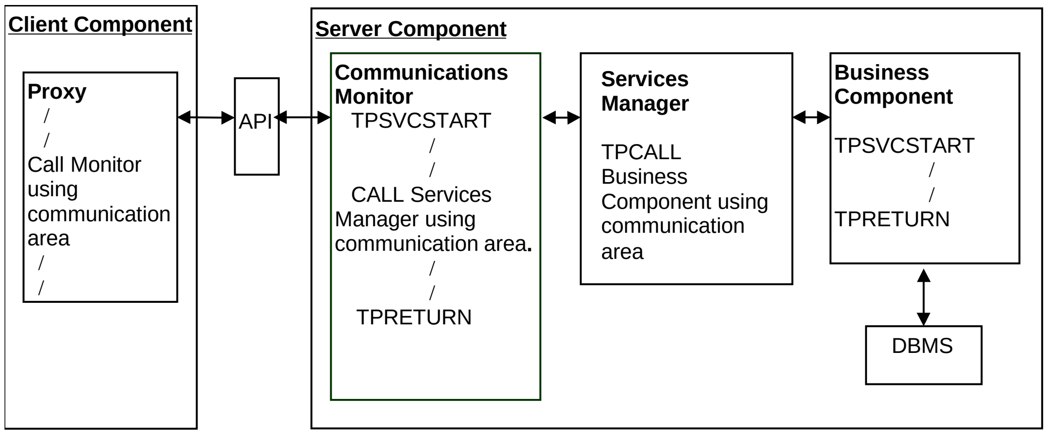
- Microfocus indexed files,
- Oracle V6 and V7,
- Sybase V10.0.1 and higher (ANSI-mode).

6.1.2. Graphic Application

6.1.2.1. Functions of the Communications Monitor

- TUXEDO-XA

The message which identifies the service to be executed is received by a Communications Monitor. This monitor initializes the service and calls the appropriate Business Component. Once the service has been executed, the Communications Monitor ends the service and sends the answer to the Proxy.


Client Component
Proxy / / Call Monitor using communication area / /
API
Server Component
Communications Monitor TPSVCSTART / / CALL Services Manager using communication area. / / TPRETURN
Services Manager TPCALL Business Component using communication area
Business Component TPSVCSTART / / TPRETURN
DBMS

- TUXEDO-XA (TUXEDO 6.2. onwards)

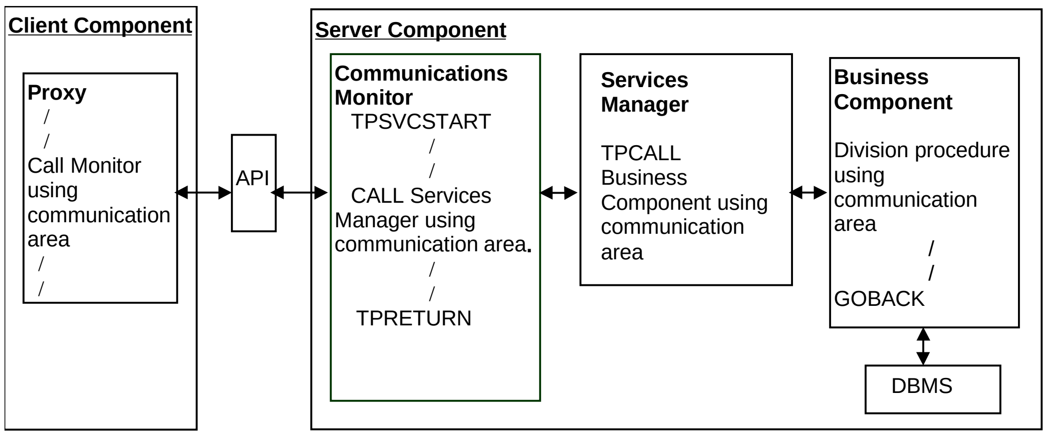
With Tuxedo 6.2, you can use the functions mechanisms. A function is an entry point for a program written in C or in COBOL. Therefore, Business Components are not directly activated.

The use of functions has two advantages :

- there is only one program structure for the Business Components, whatever the type of client calling it or the type of Database.

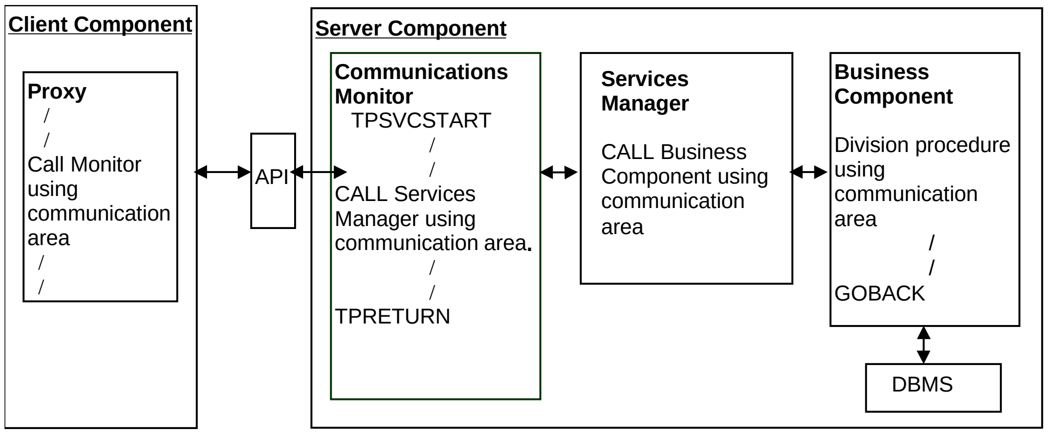
  The buffer used can be unified (**CARRAY**). The structure of these Business Components is that of a sub-program; there is thus no more TUXEDO order of message receiving (**TPSVCSTART**) or sending (**TPRETURN**).
- Business Components can be called either by CALL (required for the Business Components using the non XA Database, because in this mode the notion of global transaction allowing to take into account the updates executed by several different services does not exist) or by TPCALL (in this case, the function is systematically activated and the Business Component is called by CALL).

In this architecture, the Services Manager and the Communications Monitor are found on the same TUXEDO server and constitute one same service (Services Manager is a sub-program of the Communications Monitor). However, Business Components must be found on one or more different servers.


Client Component
Proxy
/
/
Call Monitor using communication area
/
/
API
Server Component
Communications Monitor
TPSVCSTART
/
/
CALL Services Manager using communication area.
/
/
TPRETURN
Services Manager
TPCALL Business Component using communication area
Business Component
Division procedure using communication area
/
/
GOBACK
DBMS

- TUXEDO NON XA

In this architecture, only one TUXEDO service can exist. Indeed, since a global transaction is not possible, Business Components cannot be distinct services. The only thing to do to validate the updates made by several Business Components is to group them on the same TUXEDO server and service as the Communications Monitor and Services Manager.


Client Component
Proxy
/
/
Call Monitor using communication area
/
/
API
Server Component
Communications Monitor
TPSVCSTART
/
/
CALL Services Manager using communication area.
/
/
TPRETURN
Services Manager
CALL Business Component using communication area
Business Component
Division procedure using communication area
/
/
GOBACK
DBMS

6.1.2.2. Management of COMMIT/ROLLBACK

The Server component manages the Database integrity. The Database **COMMIT** or **ROLLBACK** command is executed by the Communications Monitor according to the contents of the **TECH-COMMIT** field (**C** or **R**). This field belongs to the communication area which passes information between the components of the Server part. It is initialized by the Business Component, depending on the progress of the called service.

After the execution of a **ROLLBACK**, the Communications Monitor sends back a message containing the reasons why the Proxy failed to complete the service successfully.

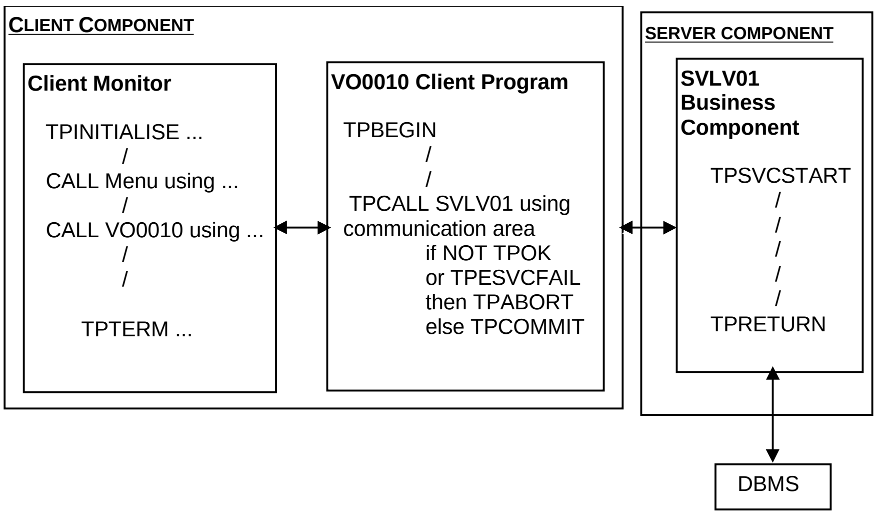
6.1.3. TUI Application

For more information on the **Working Storage Section** areas and the complete structure of the Client component, refer to the *Dialogue Microfocus Reference Manual* [REF : DD OPC 000 021 A].

6.1.3.1. Architecture

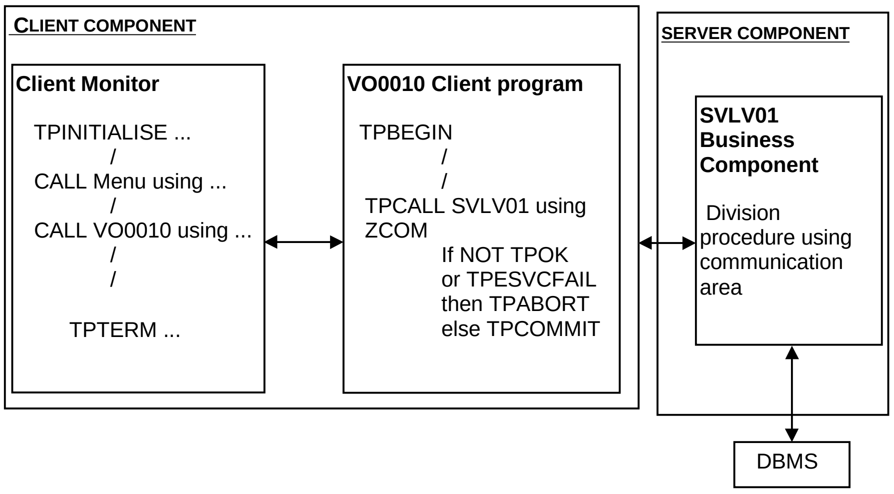
- TUXEDO-XA

In the TUXEDO environment, the Client component is constituted of a Client monitor and Client functions such as menu, display, list, detail, etc. The notion of Communications Monitor does not exist and the Business Component is directly called par the Client monitor which initializes and ends the TUXEDO service.

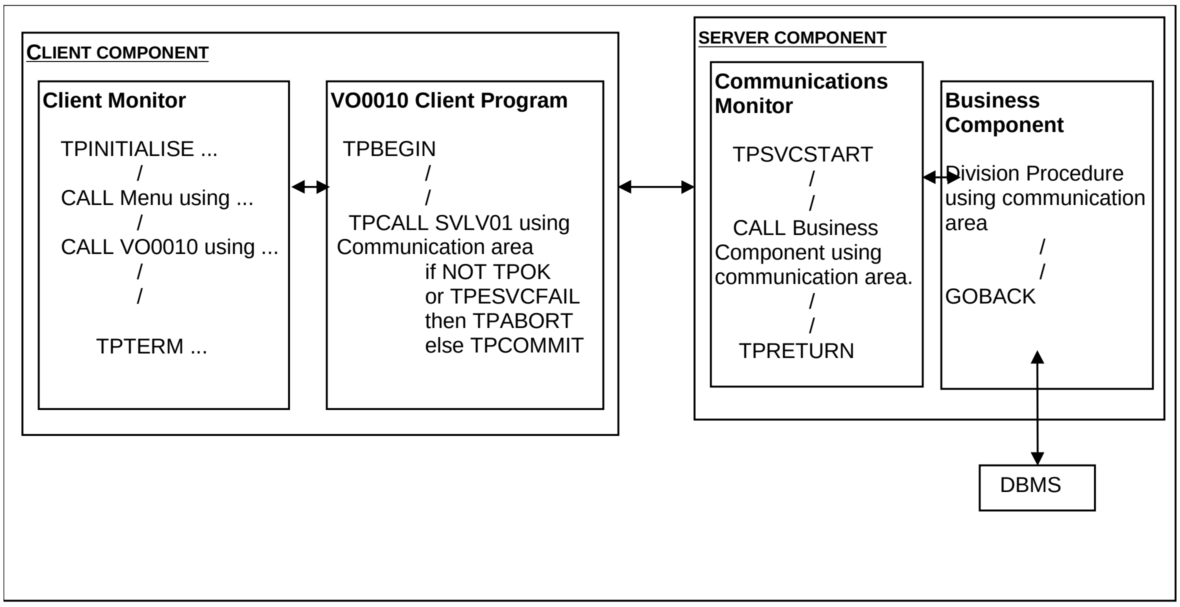


- TUXEDO XA (TUXEDO version 6.2. onwards)

In this architecture, the Communications Monitor and the Business Component necessarily belong to different TUXEDO servers (it is not possible to call a service in a same server).



- TUXEDO NON XA



6.1.3.2. Management of `COMMIT/ROLLBACK`

In case of blocking error (exception on a Database access or structure incompatibility between the Client and the Server components), the Business Component sets the **`TPESVCFAIL`** TUXEDO variable which is interpreted by the client monitor to execute a **`ROLLBACK`** on the Database. Otherwise, the client monitor executes a **`COMMIT`** on the Database.

6.2. CICS

6.2.1. Database Management

DBMS automatically supported by CICS Business Components and TUI Client Components are the following :

- DB2,
- Oracle V6 and V7,
- SQL/DS.

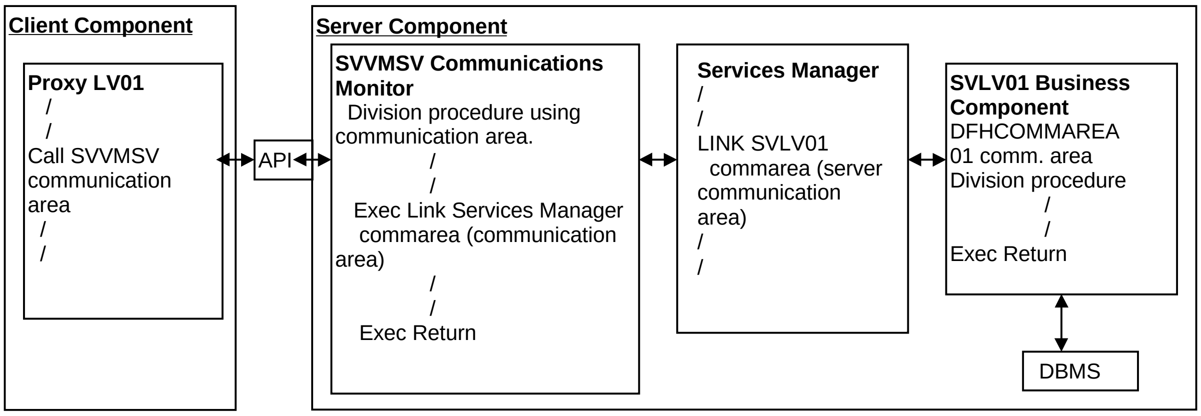
6.2.2. Graphic Application

6.2.2.1. CICS/ECI

6.2.2.1.1. Functions of the CICS/ECI Communications Monitor

In ECI mode, a Server call by the Client is considered as a **`LINK`**-type call.

In this case, the role of the Communications Monitor is to receive the message and to branch to the Services Manager which, in turn, calls the Business Component to be executed.


Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division procedure using communication area.
/
/
Exec Link Services Manager commarea (communication area)
/
/
Exec Return
Services Manager
/
/
LINK SVLV01 commarea (server communication area)
/
/
SVLV01 Business Component
DFHCOMMAREA
01 comm. area
Division procedure
/
/
Exec Return
DBMS

6.2.2.1.2. Management of `COMMIT/ROLLBACK`

In Non Extend mode, each error detected by the Business Component sets the **`TECH-COMMIT`** field to **`R`**. This value is then interpreted by the Communications Monitor to execute a **`ROLLBACK`** on the Database.

Otherwise, an implicit **`COMMIT`** is executed on the Database when the monitor passes control to CICS on the **`RETURN`** instruction.

In all cases, the communication area is sent back to the Proxy.

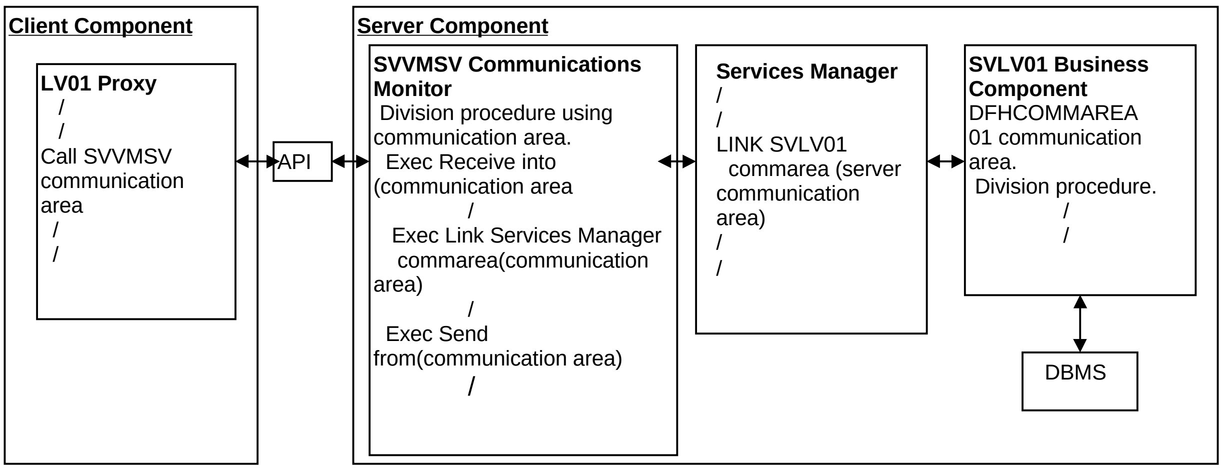
6.2.2.2. CICS/CPI-C

6.2.2.2.1. Functions of the CICS/CPI-C Communications Monitor

With the **CPI-C LU62** communication protocol, the exchange of information between the Client and the Server components is based on the message mode.

The reception and sending of messages executed in the Communications Monitor are based on the implicit **CPI-C** mode which consists in using the standard communication verbs **RECEIVE** and **SEND**.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** then transferred to the Business Component via a **LINK** (or **CALL**) type call.


Client Component
LV01 Proxy
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division procedure using communication area.
Exec Receive into (communication area
/
Exec Link Services Manager commarea(communication area)
/
Exec Send from(communication area)
/
Services Manager
/
/
LINK SVLV01 commarea (server communication area)
/
/
SVLV01 Business Component
DFHCOMMAREA
01 communication area.
Division procedure.
/
/
DBMS

6.2.2.2.2. Management of COMMIT/ROLLBACK

With the **LU62 CPI-C** communication protocol**,** the management of the **COMMIT**/**ROLLBACK** operates according to the same principles as those of the **CICS ECI** protocol.

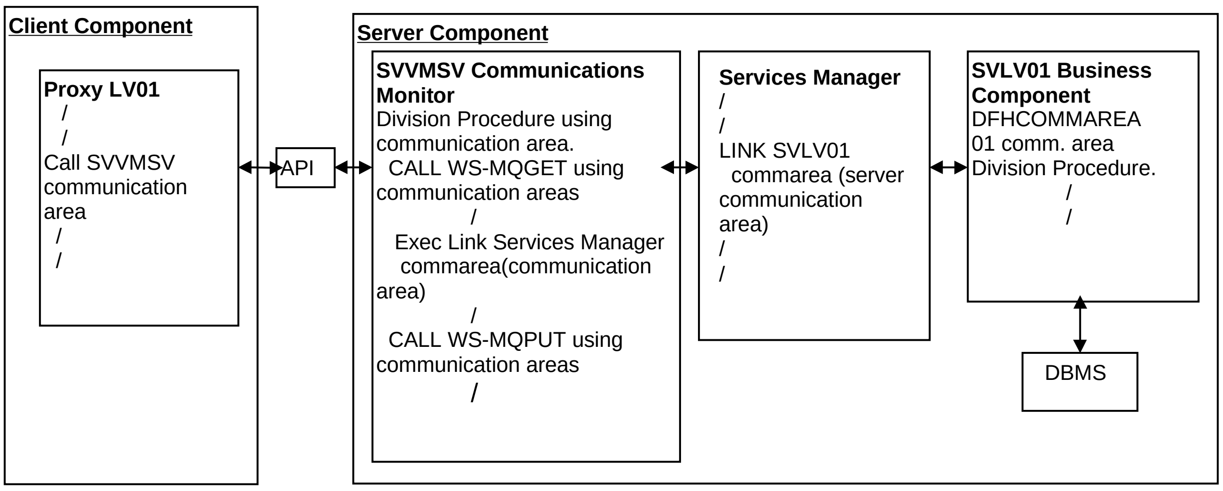
6.2.2.3. CICS/MQSERIES

6.2.2.3.1. Functions of the CICS/MQSERIES Communications Monitor

With the **MQSERIES** protocol, the exchange of information between the Client and Server components is based on the message mode.

The Communications Monitor receives and sends the messages by calling **MQSERIES** routines.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** and then transferred to the Business Component via the Services Manager, via a **LINK** (or **CALL**) type call.


Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division Procedure using communication area.
CALL WS-MQGET using communication areas
/
Exec Link Services Manager commarea(communication area)
/
CALL WS-MQPUT using communication areas
/
Services Manager
/
/
LINK SVLV01 commarea (server communication area)
/
/
SVLV01 Business Component
DFHCOMMAREA
01 comm. area
Division Procedure.
/
/
DBMS

6.2.2.3.2. Management of COMMIT/ROLLBACK

With the **MQSERIES** communication protocol, the **COMMIT**/**ROLLBACK** management operates according to the same principles as those of the **CICS ECI** protocol.

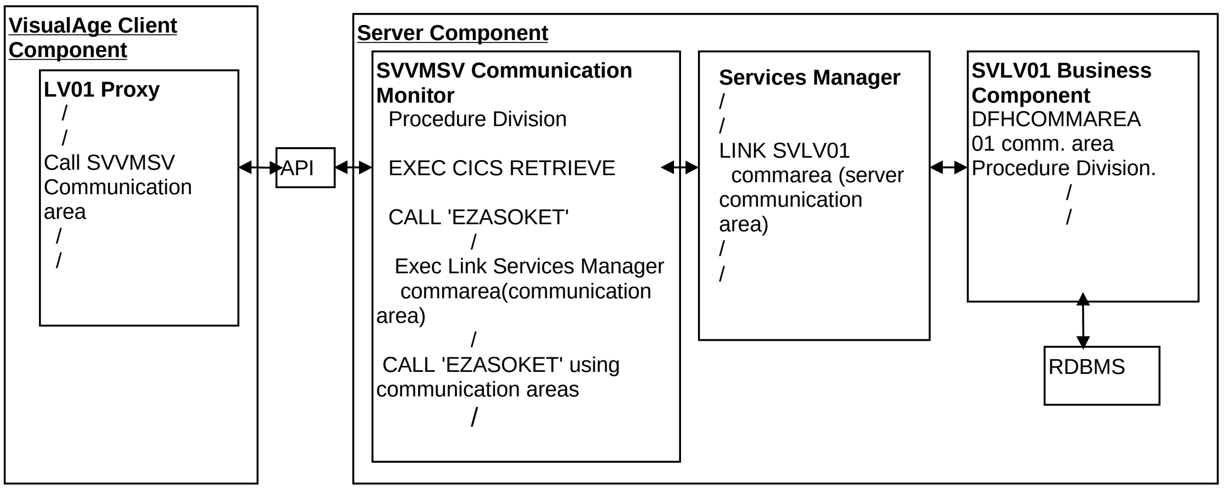
6.2.2.4. CICS/SOCKET

6.2.2.4.1. Functions of the CICS/ SOCKET Communications Monitor

With the **SOCKET** communication protocol, the exchange of information between the Client and the Server components is based on the message mode.

The Communications Monitor receives and sends the messages by calling **SOCKET** routines.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** then transferred to the Business Component via a **LINK** (or **CALL**) type call.


VisualAge Client Component
LV01 Proxy
/
/
Call SVVMSV Communication area
/
/
API
Server Component
SVVMSV Communication Monitor
Procedure Division
EXEC CICS RETRIEVE
CALL 'EZASOKET'
/
Exec Link Services Manager commarea(communication area)
/
CALL 'EZASOKET' using communication areas
/
Services Manager
/
/
LINK SVLV01 commarea (server communication area)
/
/
SVLV01 Business Component
DFHCOMMAREA
01 comm. area
Procedure Division.
/
/
RDBMS

6.2.2.4.2. Management of **COMMIT/ROLLBACK**

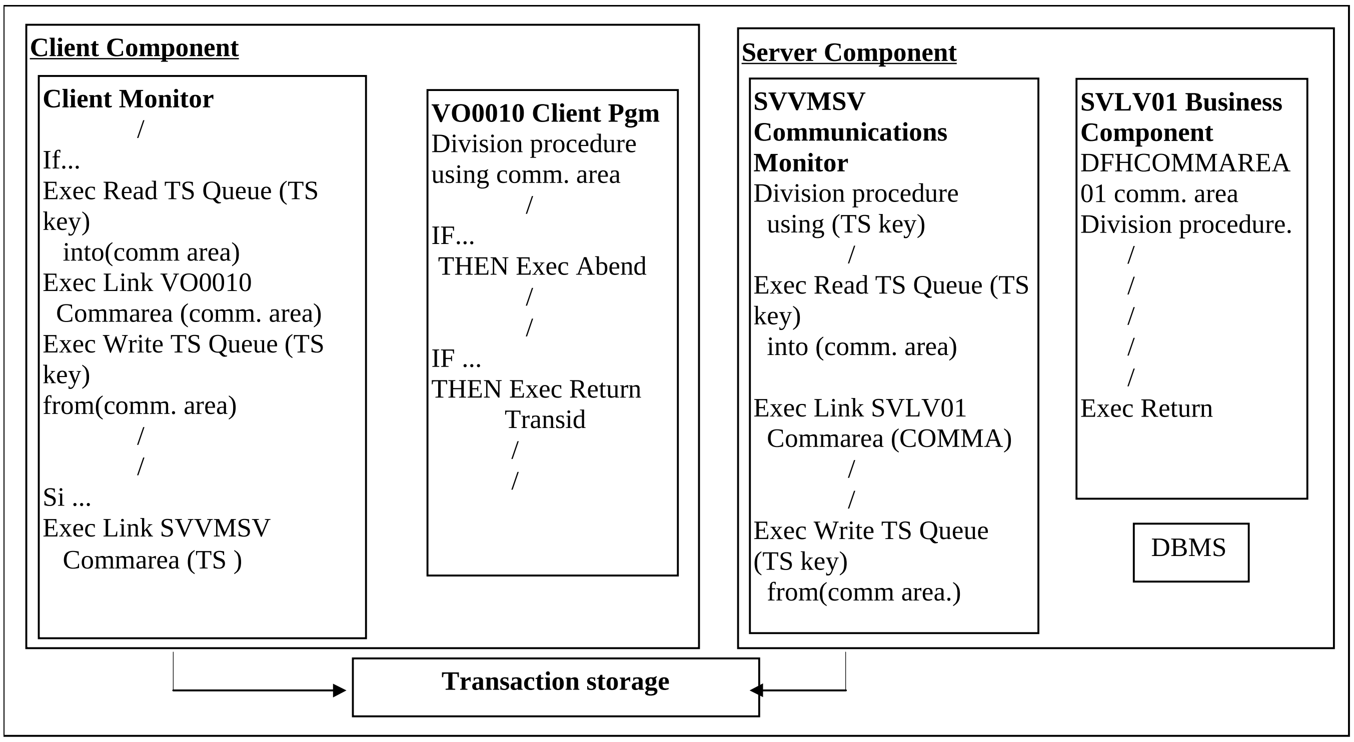
With the **SOCKET** communication protocol, the **COMMIT**/**ROLLBACK** management operates according to the same principles as those of the **CICS ECI** protocol.

6.2.3. TUI Application

For more information on the **Working Storage Section** areas and the complete structure of the Client component, refer to the *CICS OLSD Reference Manual* [REF : DD OCI 000 151 A].

6.2.3.1. Architecture

The different components of a TUI application are executed in the same CICS. The Client component includes a monitor and several programs which manage the display on dumb terminals. The server part includes a Communications Monitor and Business Components. The program calls have a **LINK** (or **CALL**) type.



6.2.3.2. Management of COMMIT/ROLLBACK

The application runs in implicit **COMMIT**. There is no particular synchronization level. If the Business Component detects a blocking error, the execution of an **EXEC CICS ABEND** order is triggered by the server (invalidates the execution of the **COMMIT**). If no error is detected, the client ends with a **RETURN TRANSID** order (validates the execution of the **COMMIT**).

6.3. GCOS7

6.3.1. Database Management

DBMS automatically supported by GCOS7 Business Components are the following :

- UFAS indexed files,
- Oracle V6 and V7.

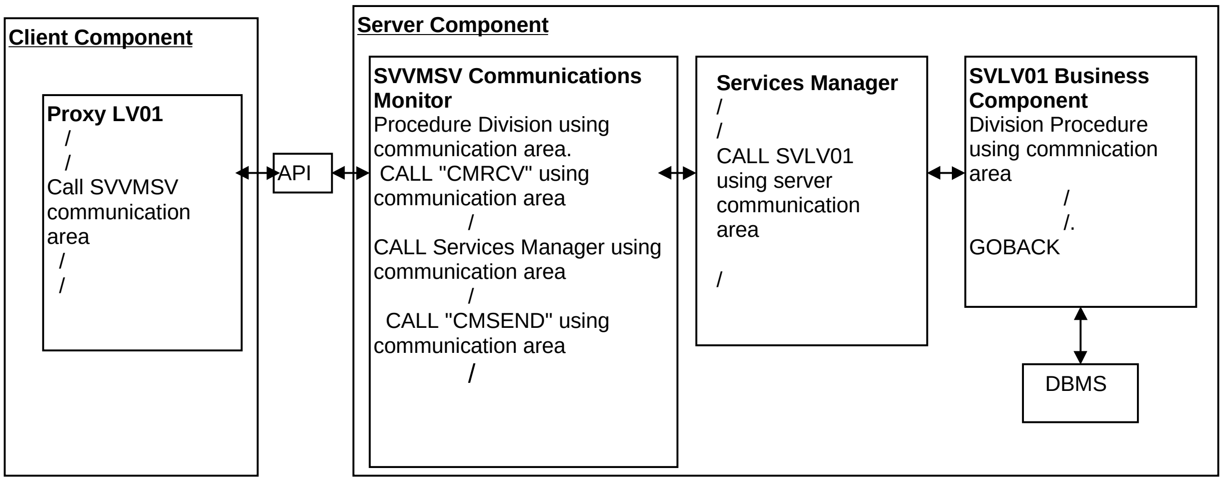
6.3.2. Graphic Application

6.3.2.1. Functions of the XCP2/CPI-C Communications Monitor

With the **`LU62 CPI-C`** communication protocol the exchange of information between the Client and Server Components is based on the message mode.

The receiving and sending of messages executed in the Communications Monitor are based on the **`CPI-C`** mode which consists in using the "**`CMRCV`**" module to receive the message and the "**`CMSEND`**" module to send it.

When the message is received, the communication area is initialized in the **`WORKING-STORAGE SECTION`** and then transferred to the Business Component via the Services Manager, via a **`CALL`** type call.


Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Procedure Division using communication area.
CALL "CMRCV" using communication area
/
CALL Services Manager using communication area
/
CALL "CMSEND" using communication area
/
Services Manager
/
/
CALL SVLV01 using server communication area
/
SVLV01 Business Component
Division Procedure using commnication area
/
/.
GOBACK
DBMS

6.3.2.2. Management of `COMMIT/ROLLBACK`

Each error detected by the Business Component sets the **`TECH-COMMIT`** variable to **`R`**. This value is then interpreted by the Communications Monitor to execute a **`ROLLBACK`** on the Database.

The **`COMMIT`** and **`ROLLBACK`** orders are executed by SQL commands which depend on the type of DBMS accessed by the Business Components.

6.4. GCOS8

6.4.1. Database Management

DBMS automatically supported by GCOS8 Business Components are the following :

- UFAS indexed files,
- SQL INTEREL Databases.

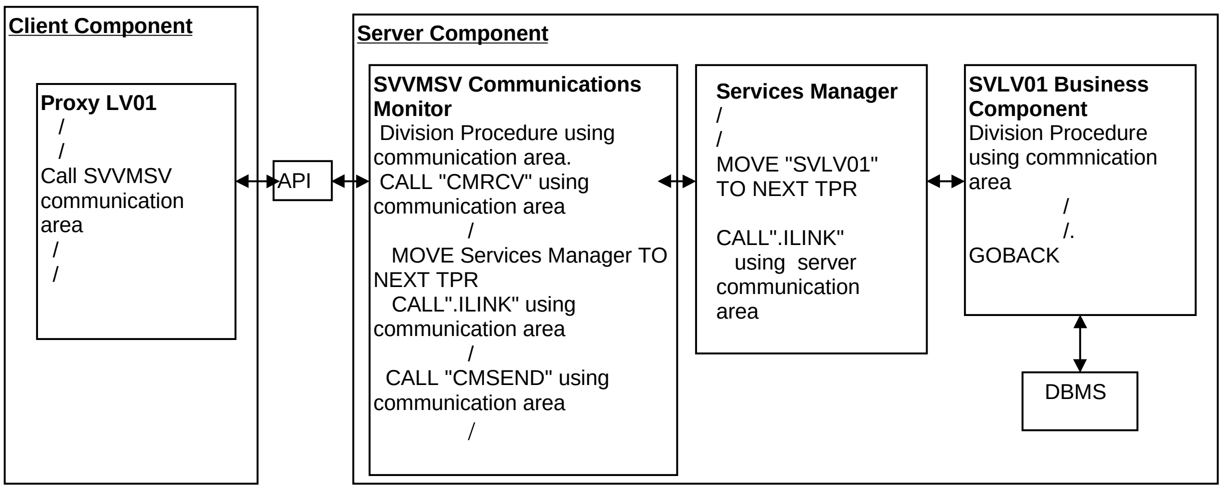
6.4.2. Graphic Application

6.4.2.1. Functions of the XCP2/CPI-C Communications Monitor

With the **LU62 CPI-C** communication protocol, the exchange of information between the Client and Server Components is based on the message mode.

The reception and sending of messages executed in the Communications Monitor are based on the **CPI-C** mode which consists in using the "**CMRCV**" module to receive a message and the "**CMSEND**" module to send it.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** and then transferred to the Business Component via the Service Manager via a **CALL".ILINK"** type call.


Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division Procedure using communication area.
CALL "CMRCV" using communication area
/
MOVE Services Manager TO NEXT TPR
CALL".ILINK" using communication area
/
CALL "CMSEND" using communication area
/
Services Manager
/
/
MOVE "SVLV01" TO NEXT TPR
CALL".ILINK" using server communication area
SVLV01 Business Component
Division Procedure using commnication area
/
/.
GOBACK
DBMS

6.4.2.2. Management of COMMIT/ROLLBACK

Each error detected by the Business Component sets the **TECH-COMMIT** variable to **R**. This value is then interpreted by the Communications Monitor to execute a **ROLLBACK** on the Database.

The **COMMIT** and **ROLLBACK** orders are executed by SQL commands which depend on the type of DBMS accessed by the Business Components.

6.5. MICROFOCUS UNIX, OS/2 or WINDOWS/NT

6.5.1. Database Management

DBMS automatically supported by Microfocus Business Components and TUI Client Components are the following :

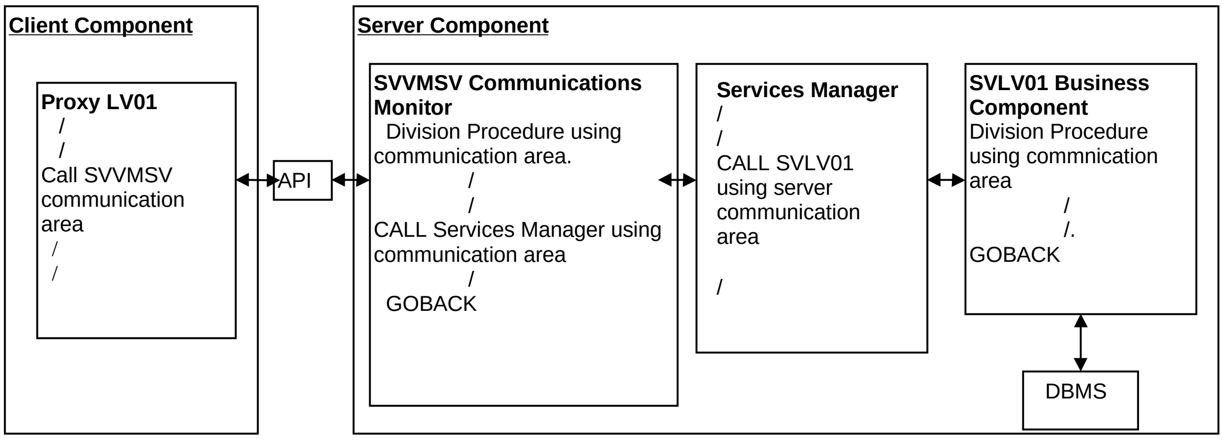
- Microfocus indexed files,
- Oracle V6 and V7,
- Sybase V10.0.1 and higher (ANSI-mode),
- DB2/2,
- SQL Informix (except for OS/2).

6.5.2. Graphic Application

6.5.2.1. Microfocus /Sockets

6.5.2.1.1. Functions of the Microfocus /Sockets Communications Monitor

For the **`TCP-IP Socket`** protocol, the Communications Monitor does not integrate specific communication verbs. The exchange principle is based on the writing of a message in a socket on the host machine. A Server associated with this socket is then executed. This Server translates the message, calls the Communications Monitor, via the Services Manager, and sends the message by reference.


Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division Procedure using communication area.
/
/
CALL Services Manager using communication area
/
GOBACK
Services Manager
/
/
CALL SVLV01 using server communication area
/
SVLV01 Business Component
Division Procedure using commnication area
/
/.
GOBACK
DBMS

6.5.2.1.2. Management of `COMMIT/ROLLBACK`

Each error detected by the Business Component sets the **`TECH-COMMIT`** variable to **`R`**. This value is then interpreted by the Communications Monitor to execute a **`ROLLBACK`** on the Database.

The **`COMMIT`** and **`ROLLBACK`** orders are executed by SQL commands which depend on the type of DBMS accessed by the Business Components.

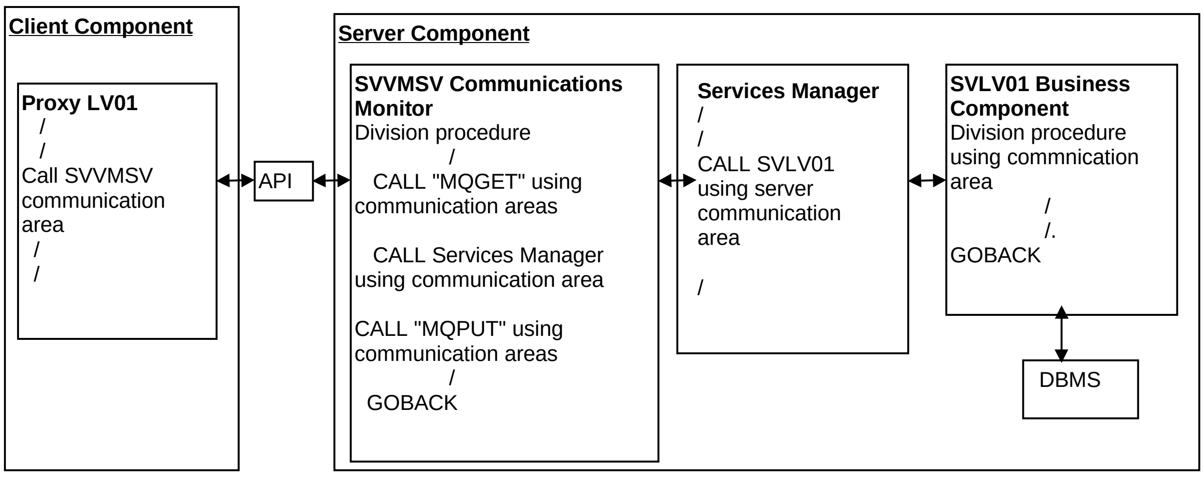
6.5.2.2. Microfocus / MQSERIES

6.5.2.2.1. Functions of the Microfocus / MQSERIES Communications Monitor

With the **MQSERIES** protocol, the exchange of information between the Client and Server components is based on the message mode.

The Communications Monitor receives and sends the messages by calling **MQSERIES** routines.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** and then transferred to the Business Component via the Services Manager, via a **LINK** (or **CALL**) type call.



Client Component
Proxy LV01
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division procedure
/
CALL "MQGET" using communication areas
CALL Services Manager using communication area
CALL "MQPUT" using communication areas
/
GOBACK
Services Manager
/
/
CALL SVLV01 using server communication area
/
SVLV01 Business Component
Division procedure using commnication area
/
/.
GOBACK
DBMS

6.5.2.2.2. Management of COMMIT/ROLLBACK

Each error detected by the Business Component sets the **TECH-COMMIT** variable to **R**. This value is then interpreted by the Communications Monitor to execute a **ROLLBACK** on the Database.

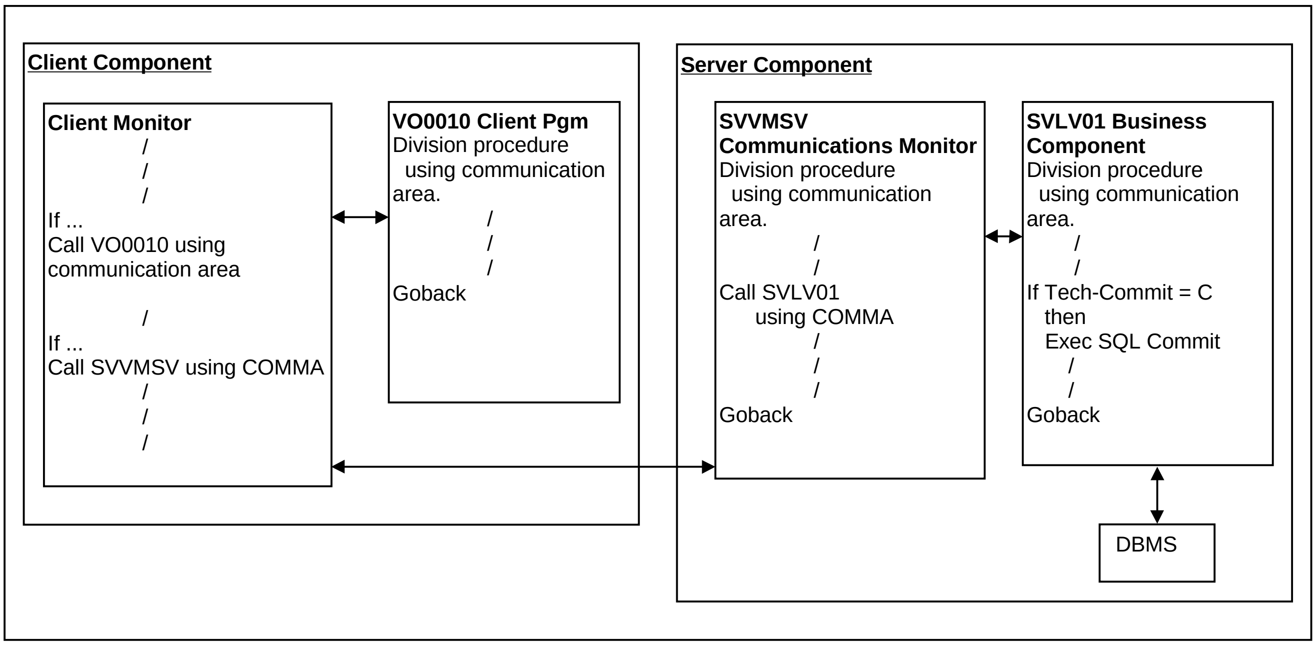
The **COMMIT** and **ROLLBACK** orders are executed by SQL commands which depend on the type of DBMS accessed by the Business Components.

6.5.3. TUI Application

For more information on the **Working Storage Section** areas and the complete structure of the Business Component, refer to the *Microfocus Dialogue Reference Manual* [REF : DD OPC 000 021 A].

6.5.3.1. Architecture

The different components of a TUI application are executed in the same station. The Client part includes a monitor and several programs which manage the dumb terminals display. The Server part includes a Communications Monitor and Business Components. The program calls have a dynamic **CALL** type.


Client Component
Client Monitor
/
/
/
If ...
Call VO0010 using communication area
/
If ...
Call SVVMSV using COMMA
/
/
/
VO0010 Client Pgm
Division procedure using communication area.
/
/
/
Goback
Server Component
SVVMSV Communications Monitor
Division procedure using communication area.
/
/
Call SVLV01 using COMMA
/
/
/
Goback
SVLV01 Business Component
Division procedure using communication area.
/
/
If Tech-Commit = C then Exec SQL Commit
/
/
Goback
DBMS

6.5.3.2. Management of COMMIT/ROLLBACK

The **TECH-COMMIT** variable which conditions the execution of the SQL **COMMIT** is set to **C** by the generator for this particular environment (which supposes the default execution of the **COMMIT** in the Business Component). If the Business Component detects a blocking error (Database access error, incorrect contents of the technical area, ...) the **COMMIT** is not executed.

6.6. IMS

6.6.1. Database Management

DBMS automatically supported by IMS Business Components and TUI Client Components is SQL DB2.

For the Business Components only, the DL/1 Databases are also automatically supported.

6.6.2. Graphic Application

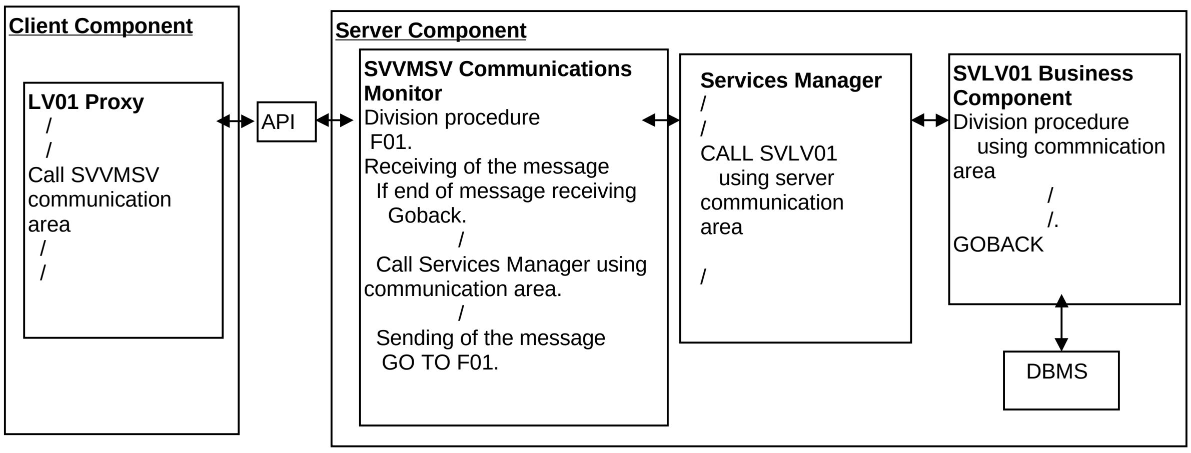
6.6.2.1. IMS CPI-C

6.6.2.1.1. Functions of the IMS CPI-C Communications Monitor

With the **LU62 CPI-C** communication protocol, the exchange of information between the Client and the Server Components is based on the message mode.

The reception and sending of messages executed in the Communications Monitor are based on the implicit **CPI-C** mode which consists in using the **GETUNIT** and **INSERT** standard communication verbs.

When the message is received, the communication area is initialized in the **WORKING-STORAGE SECTION** and then transferred to the Business Component, via the Services Manager, via a **CALL** type call.



6.6.2.1.2. Management of `COMMIT/ROLLBACK`

With the **LU62 CPI-C** communication protocol, the **COMMIT**/**ROLLBACK** management operates according to the same principles as those of the **CICS ECI** protocol.

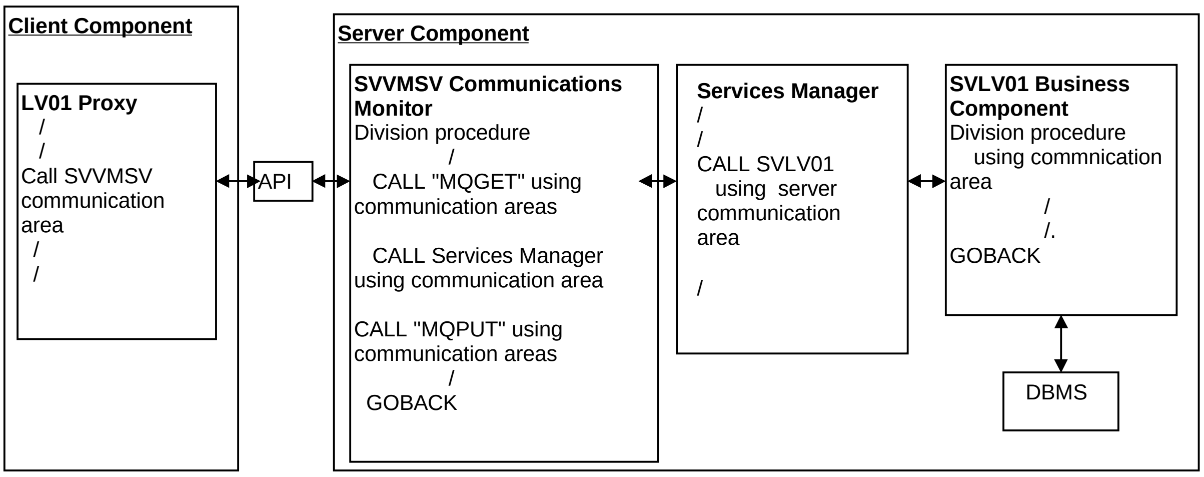
6.6.2.2. IMS / MQSERIES

6.6.2.2.1. Functions of the IMS / MQSERIES Communications Monitor

With the **MQSERIES** communication protocol, the exchange of information between the Client and Server Components is based on the message mode.

The Communications Monitor receives and sends the messages by calling **MQSERIES** routines.

When the message is received, the communication area is initialized in the **LINKAGE SECTION** and then transferred to the Business Component, via the Services Manager, via a **CALL** type call.


Client Component
LV01 Proxy
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Division procedure
/
CALL "MQGET" using communication areas
CALL Services Manager using communication area
CALL "MQPUT" using communication areas
/
GOBACK
Services Manager
/
/
CALL SVLV01 using server communication area
/
SVLV01 Business Component
Division procedure
using commnication area
/
/.
GOBACK
DBMS

6.6.2.2.2. Management of `COMMIT/ROLLBACK`

Each error detected by the Business Component set the **TECH-COMMIT** variable to **R**. This value is then interpreted by the Communications Monitor to execute a **ROLLBACK** on the Database.

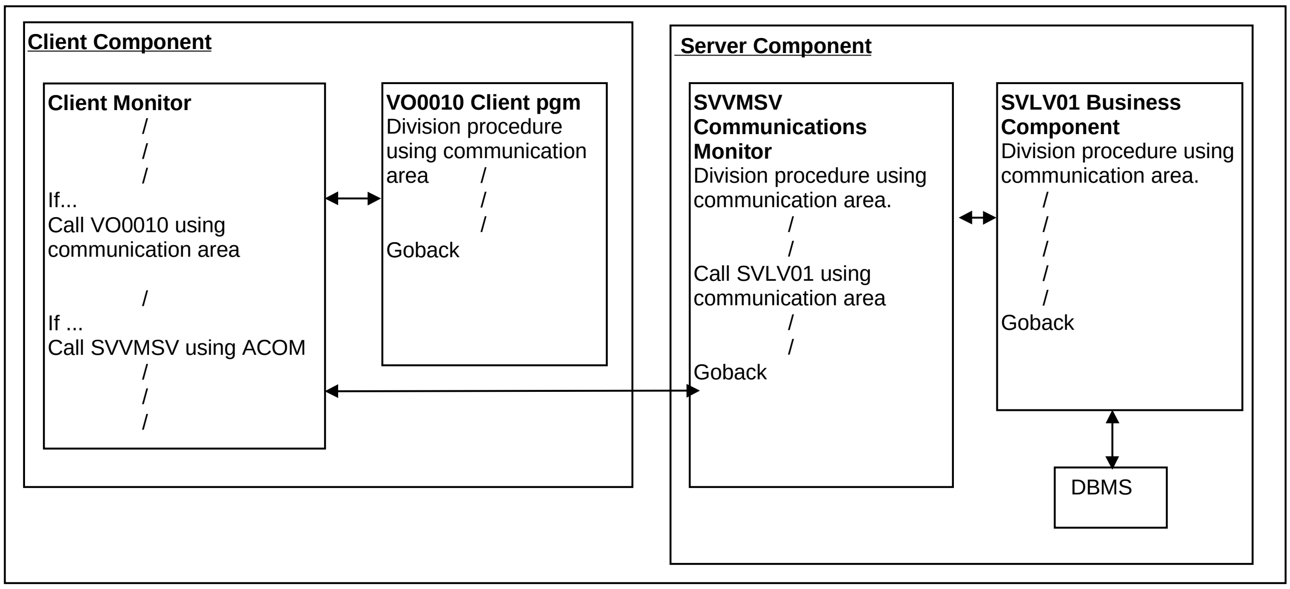
The **COMMIT** and **ROLLBACK** commands are executed by IMS orders.

6.6.3. TUI Application

For more information on the **Working Storage Section** area and the complete structure of the Client component, refer to the *IMS Dialogue Reference Manual* [REF : DD OIM 000 021 A].

6.6.3.1. Architecture

The different components of a TUI application are executed in the same IMS. The Client component includes a monitor and several programs which manage the display on dumb terminals. The Server component includes a Communications Monitor and Business Components. The program calls are of the **CALL** type.


Client Component
Client Monitor
/
/
/
If...
Call VO0010 using communication area
/
If ...
Call SVVMSV using ACOM
/
/
/
VO0010 Client pgm
Division procedure using communication area
/
/
/
Goback
Server Component
SVVMSV Communications Monitor
Division procedure using communication area.
/
/
Call SVLV01 using communication area
/
/
Goback
SVLV01 Business Component
Division procedure using communication area.
/
/
/
/
/
Goback
DBMS

6.6.3.2. Management of COMMIT/ROLLBACK

The application runs in implicit **COMMIT**.

If the Business Component detects a blocking error, no **ROLLBACK** type processing is supplied in standard.

6.7. UNISYS 2200

6.7.1. Database Management

DBMS automatically supported by Unisys 2000 Business Components are the following :

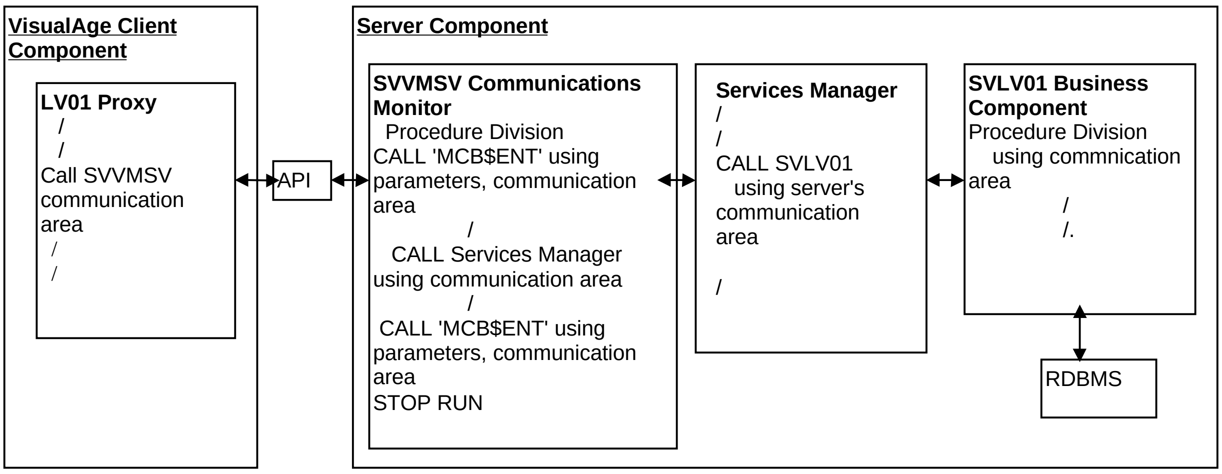
- RDMS/SQL
- SFS indexed files.

6.7.2. Graphical Application

6.7.2.1. Functions of the Unisys 2200 / TCIS Communications Monitor

The receiving and sending of messages executed in the Communications Monitor are based on the **`TCIS`** mode which consists in using the standard **`MCB$ENT`** module for the receiving (**`P-TRINIT`**) and **(`P-SENDD`)**sending functions.

When the message is received, the communication area is initialized in the **`WORKING-STORAGE SECTION`** then transferred to the Business Component via a **`CALL`** type call.


VisualAge Client Component
LV01 Proxy
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Procedure Division
CALL 'MCB$ENT' using parameters, communication area
/
CALL Services Manager using communication area
/
CALL 'MCB$ENT' using parameters, communication area
STOP RUN
Services Manager
/
/
CALL SVLV01 using server's communication area
/
SVLV01 Business Component
Procedure Division
using commnication area
/
/.
RDBMS

6.7.2.2. Management of `COMMIT/ROLLBACK`

Each error detected by the Business Component sets the **`TECH-COMMIT`** variable to **`R`**. This value is then interpreted by the Communications Monitor to execute a **`ROLLBACK`** on the Database.

The **`COMMIT`** and **`ROLLBACK`** orders are executed by SQL commands which depend on the type of DBMS accessed by the Business Components.

6.8. TANDEM

6.8.1. Database Management

DBMS automatically supported by Tandem Business Components are the following :

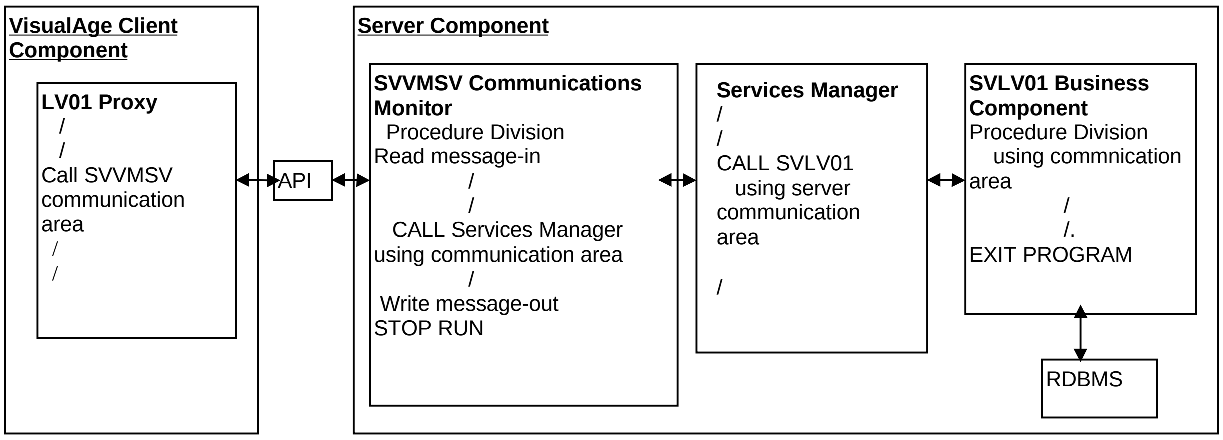
- Non Stop SQL
- Indexed files

6.8.2. Graphical Application

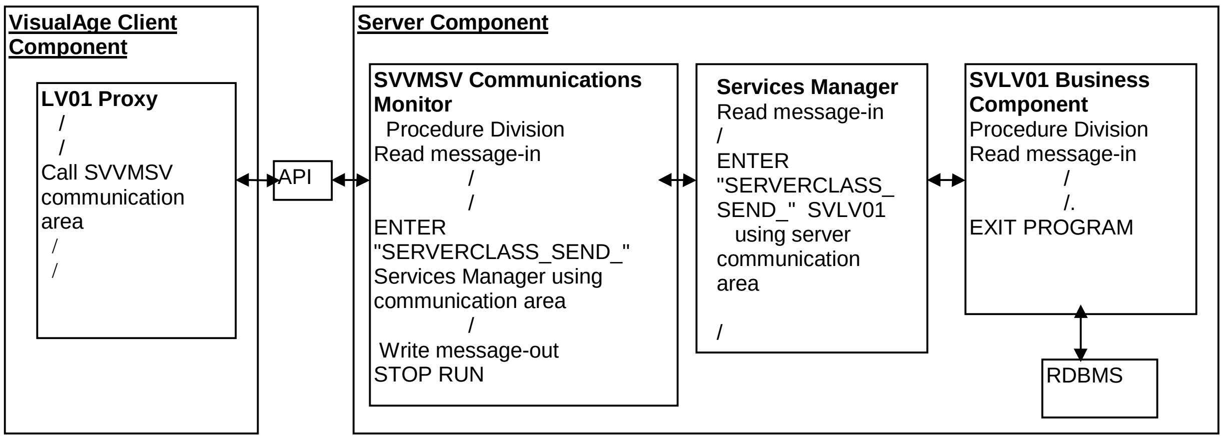
6.8.2.1. TANDEM PATHWAY

6.8.2.1.1.Functions of the Tandem Pathway / Socket Communications Monitor

For the **TCP-IP Socket** protocol, the Communications Monitor does not integrate specific communication verbs. The exchange principle is based on the writing of a message in a "Message" file. A Server associated with this socket is then executed. This Server translates the message, calls the Communications Monitor, via the Services Manager, and sends the message in the "Message" file.


VisualAge Client Component
LV01 Proxy
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Procedure Division
Read message-in
/
/
CALL Services Manager using communication area
/
Write message-out
STOP RUN
Services Manager
/
/
CALL SVLV01 using server communication area
/
SVLV01 Business Component
Procedure Division
using commnication area
/
/.
EXIT PROGRAM
RDBMS

If you specified the option **CALLTYPE=PATHSEND**, the architecture is the following:


VisualAge Client Component
LV01 Proxy
/
/
Call SVVMSV communication area
/
/
API
Server Component
SVVMSV Communications Monitor
Procedure Division
Read message-in
/
/
ENTER "SERVERCLASS_SEND_" Services Manager using communication area
/
Write message-out
STOP RUN
Services Manager
Read message-in
/
ENTER "SERVERCLASS_ SEND_" SVLV01 using server communication area
/
SVLV01 Business Component
Procedure Division
Read message-in
/
/.
EXIT PROGRAM
RDBMS

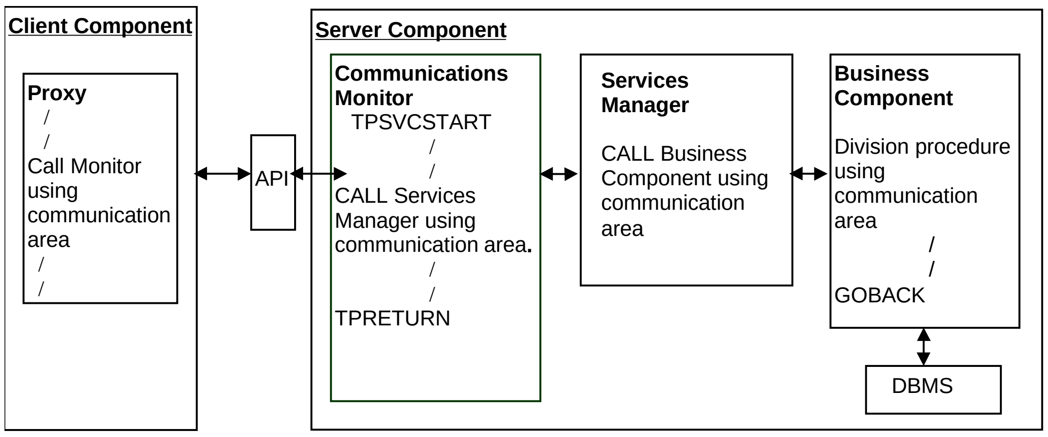
6.8.2.1.2.Management of COMMIT/ROLLBACK

The **COMMIT** and **ROLLBACK** are managed by the Communications Monitor.

6.8.2.2. TANDEM NonStop TUXEDO NON XA

6.8.2.2.1.Functions of the Communications Monitor

In this architecture, only one TUXEDO service can exist. Indeed, since a global transaction is not possible, Business Components cannot be distinct services. The only thing to do to validate the updates made by several Business Components is to group them on the same TUXEDO server and service as the Communications Monitor and Services Manager.


Client Component
Proxy
/
/
Call Monitor using communication area
/
/
API
Server Component
Communications Monitor
TPSVCSTART
/
/
CALL Services Manager using communication area.
/
/
TPRETURN
Services Manager
CALL Business Component using communication area
Business Component
Division procedure using communication area
/
/
GOBACK
DBMS

6.8.2.2.2.Management of `COMMIT/ROLLBACK`

The Server component manages the Database integrity. The Database **`COMMIT`** or **`ROLLBACK`** command is executed by the Communications Monitor according to the contents of the **`TECH-COMMIT`** field (**`C`** or **`R`**). This field belongs to the communication area which passes information between the components of the Server part. It is initialized by the Business Component, depending on the progress of the called service.

After the execution of a **`ROLLBACK`**, the Communications Monitor sends back a message containing the reasons why the Proxy failed to complete the service successfully.