VisualAge Pacbase 2.5

Pacbench Client/Serveur

Guide de l'Utilisateur – Volume I

Concepts - Architectures - Environnements

DDOVC000255F

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section "Remarques" de la page suivante.

En application de votre contrat de licence, vous pouvez consulter ou télécharger la documentation de VisualAge Pacbase, régulièrement mise à jour, à partir du site Web du Support Technique :

http://www.ibm.com/software/ad/vapacbase/support.htm

La section Catalogue dans la page d'accueil de la Documentation vous permet d'identifier la dernière édition disponible du présent document.

5ème Edition (décembre 1999)

La présente édition s'applique à :

- VisualAge Pacbase Version 2.5

Vous pouvez nous adresser tout commentaire sur ce document (en indiquant sa référence) via le site Web de notre Support Technique à l'adresse suivante :
http://www.ibm.com/software/ad/vapacbase/support.htm

ou en nous adressant un courrier à :
IBM Paris Laboratory
Support VisualAge Pacbase
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

IBM pourra disposer comme elle l'entendra des informations contenues dans vos commentaires, sans aucune obligation de sa part.

REMARQUES

Ce document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM. Cela ne signifie pas qu'IBM ait l'intention de les annoncer dans tous les pays où la compagnie est présente.

Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM.

Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

Intellectual Property and Licensing

International Business Machines Corporation

North Castle Drive, Armonk, New-York 10504-1785

USA

Les détenteurs de licences du présent produit souhaitant obtenir des informations sur celui-ci à des fins : (i) d'échange d'informations entre des programmes développés indépendamment et d'autres programmes (y compris celui-ci) et (ii) d'utilisation mutuelle des informations ainsi échangées doivent s'adresser à :

IBM Paris Laboratory
Département SMC
30, rue du Château des Rentiers
75640 PARIS Cedex 13
FRANCE

De telles informations peuvent être mises à la disposition du Client et seront soumises aux termes et conditions appropriés, y compris dans certains cas au paiement d'une redevance.

IBM peut modifier ce document, le produit qu'il décrit ou les deux.

MARQUES

IBM est une marque d'International Business Machines Corporation, Inc.
AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection et VisualAge sont des marques d'International Business Machines Corporation, Inc. dans certains pays.

Java et toutes les marques et logos incluant Java sont des marques de Sun Microsystems, Inc. dans certains pays.
Microsoft, Windows, Windows NT et le logo Windows sont des marques de Microsoft Corporation dans certains pays.
UNIX est une marque enregistrée aux Etats-Unis et/ou dans d'autres pays et utilisée avec l'autorisation exclusive de la société X/Open Company Limited.

D'autres sociétés peuvent être propriétaires des autres marques, noms de produits ou logos qui pourraient apparaître dans ce document.

Table des Matières

Préambule

Le Module Pacbench Client/Serveur de VisualAge Pacbase permet de développer les composants d'une application Client/Serveur :

- Le **composant Serveur** – avec un grand choix de plates-formes cibles.

  ☞ Le terme composant Serveur est un terme générique utilisé pour désigner les différents éléments logiciels[1] présents dans un composant Serveur VisualAge Pacbase.

- **Le composant Client** – de type *graphique* (pour les environnements VisualAge Java et Smalltalk, pour les environnements au standard COM[2]) et de type *TUI* (Textual User Interface, aussi dénommé mode caractère ou passif) sur un grand choix de plates-formes cibles.

Le composant Middleware – qui assure la communication entre Client et Serveur, utilise les protocoles de communication du marché[3] *en encapsulant entièrement leur mise en œuvre*.

Documentation du Module Pacbench Client/Serveur

- Le ***Guide de l'Utilisateur***, en trois volumes :
  - Le présent volume (*Volume I*) décrit les grands *principes* du Module Pacbench C/S de VisualAge Pacbase et l'*architecture logicielle* des applications générées. Ce volume présente également tous les *environnements de développement et d'exécution* disponibles, ainsi que les *architectures physiques* selon les différents types d'environnement.

    Quelle que soit votre fonction au sein d'un projet avec Pacbench C/S, la lecture de ce Volume est fortement recommandée [REF : DDOVC_]
  - Le *Volume II : Services Applicatifs* est consacré au développement du *composant Serveur* [REF : DDOAU_].
  - Le V*olume III : Clients Graphiques* est consacré au développement du composant *Client graphique* [REF : DDOVA_].

- Le ***Manuel de Référence Services Applicatifs – Clients TUI*** [REF : DDOA_].

- Le ***Manuel de Référence Clients graphiques : Interface publique des composants générés*** [REF : DDOVI_].

Conventions typographiques

Les symboles suivants sont utilisés :

☞ Note, remarque, ou précision.

Renvoi à un autre emplacement dans la documentation ; les titres de manuel ainsi que les titres de chapitre sont en *italiques*.

Truc ou astuce, précision utile.

Information particulièrement importante.

☞ La documentation concernant le mode de développement en mode simple n'est plus disponible dans cette édition.

---

1 Ces éléments sont présentés dans la section 3.1.1.

2 Component Object Model.

3 Voir le sous-chapitre 5.1.

1. Introduction

VisualAge Pacbase est la solution Génie Logiciel de IBM. Elle couvre la totalité du cycle de vie des applications : conception, réalisation d'applications, reprise et maintenance de l'existant. Cette solution est constituée d'un ensemble de produits intégrés autour d'un **Référentiel unique**.

La cohérence globale de l'ensemble des Clients et Serveurs est entièrement assurée par le **Modèle d'information**.

1.1. Les fonctionnalités du Module Pacbench C/S

Le Module Pacbench C/S présente les quatre fonctionnalités suivantes :

- La fonctionnalité **Services Applicatifs**

  Elle permet le développement du composant Serveur dans l'environnement *VisualAge Pacbase*.

  ☞ Cette fonctionnalité est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

- La fonctionnalité **Clients Graphiques**

  Elle permet le développement du composant Client graphique dans la Station *VisualAge Java*[4] ou *Smalltalk*, ou dans un environnement adapté au standard *COM*.

  ☞ VisualAge Java et Smalltalk permettent de développer à la fois des clients exécutables sur micro-ordinateur ou accessibles depuis le Web (Intranet ou Internet).

  Ce développement recouvre deux aspects :

  - Exploitation d'un objet appelé **Proxy**, généré à partir de spécifications du composant Serveur.
    L'objet proxy permet au composant Client de commander à distance les services offerts par le composant Serveur.

    ☞ Cette fonctionnalité est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques* et dans le *Manuel de Référence Clients graphiques : Interface publique des composants générés.*

  - Développement de la partie purement graphique et des traitements spécifiques locaux appartenant au Client graphique.

    ☞ Consultez la documentation [VisualAge] Java ou VisualAge Smalltalk ou la documentation d'outils de développement au standard COM.

- La fonctionnalité **Clients TUI**

  Elle permet le développement du composant Client TUI dans l'environnement *VisualAge Pacbase*.

  ☞ Voir le *Manuel de Référence Services Applicatifs - Clients TUI*.

[4] Vous pouvez également utiliser Java Development Kit, version 1.1 ou plus, ou d'autres outils du marché de même niveau fonctionnel.

- La fonctionnalité **Pont VisualAge Java ou Smalltalk <> VisualAge Pacbase**

  Cette fonctionnalité vous permet de disposer d'une base de référence unique, le Référentiel de VisualAge Pacbase, pour y stocker – via une *sauvegarde* – les références des objets provenant de VisualAge Java ou Smalltalk. Ces objets bénéficient ainsi des fonctions de gestion, de références croisées et de sécurité du Référentiel.

  ☞ Le Pont VisualAge Smalltalk permet également de stocker les sources, et par conséquent de procéder à une *restauration*.

  Voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

1.2. Indépendance des Serveurs vis-à-vis de la partie cliente

Les Serveurs peuvent être appelés par des Clients aussi bien TUI que graphiques.

Cette indépendance des Serveurs vis-à-vis de la partie Cliente permet notamment d'évoluer naturellement vers les interfaces graphiques sans remettre en cause l'architecture globale du système.

Dans ce contexte, le Module Pacbench C/S offre la possibilité de développer des applications où cohabitent des Clients graphiques (GUI standard ou WEB, COM) et des Clients TUI (IBM 3270, BULL Questar, ...).

Les Serveurs sont donc développés de façon totalement indépendante de la partie cliente, quelques règles de construction – liées au développement de clients graphiques – doivent cependant être respectées.

Ces règles sont signalées par la mention « Applications graphiques » dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

2. Les Principes du Module Pacbench C/S

2.1. Notion de Vue Logique

La **Vue Logique** est une entité spécifique du Modèle d'Information VisualAge Pacbase.

Elle permet de définir un concept d'information dans le système de données d'une entreprise. Elle est réutilisable ; elle peut être exploitée par plusieurs applications.

Une Vue Logique est constituée d'une collection de données élémentaires. Sa description est indépendante du système de stockage choisi pour mémoriser ses différentes instances.

Pour être manipulée dans une application Client/Serveur, une Vue Logique doit être appelée dans un **Composant Applicatif**. C'est ce Composant Applicatif qui fait le lien entre la structure des entités de stockage (**Segments**) et la structure de la Vue Logique.

Une Vue Logique possède une capacité itérative. Cette propriété définit le nombre maximum d'instances pouvant être traitées par un Composant Applicatif en une exécution.

☞ La spécification d'une Vue Logique est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs,* chapitre *Vue Logique*.

2.2. Services proposés par un Composant Applicatif

Lorsqu'un **Composant Applicatif** appelle une Vue Logique, plusieurs services d'accès aux données sont automatiquement générés en fonction des informations définies dans la description des appels de Segments.

Des traitements personnalisés peuvent également être définis pour chaque Vue Logique.

2.2.1. Contrôle et mise à jour

Un service de *contrôle* exécute tous les contrôles intrinsèques et de correspondances spécifiés dans le Référentiel VisualAge Pacbase. En cas d'anomalie, il renvoie un message d'erreur au Client.

Un service de *mise à jour* permet de créer, de modifier ou d'annuler une instance de Vue Logique dans la base de données. Le code action associé à la mise à jour peut être implicite ou explicite. En cas d'anomalie, un message d'erreur est renvoyé au Client.

☞ Ces types de service sont documentés dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs,* chapitre *Composant Applicatif*, sous-chapitre *Service de contrôle / mise à jour*.

2.2.2. Sélection

Ce service permet de transférer sur le poste Client une collection d'instances de Vue Logique à partir de :

- Critères de sélection standard :

  Le service de sélection retourne le nombre d'instances demandé. Dans le cas où le nombre d'instances demandé est inférieur au nombre total d'instances répondant au critère de sélection, il retourne l'identifiant de la première instance de la prochaine page de la sélection.

  Ce service permet également de sélectionner une seule instance de Vue Logique.

  ☞ Ce type de service est documenté dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Composant Applicatif*, sous-chapitre *Service de sélection*.

- Critères de sélection personnalisés :

  Vous pouvez ajouter / remplacer des clauses particulières dans un accès SQL généré en standard ou créer un nouvel accès physique.

  Aussi, les critères de sélection implémentés via des méthodes d'extraction permettent de lire une collection d'instances d'une Vue Logique.

  ☞ Pour plus d'informations, consultez le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Composant Applicatif*, sous-chapitre *Insertion de traitements spécifiques*, section *Insertion par rapport au niveau "Accès Physiques"*.

2.2.3. Service Utilisateur

Un Service Utilisateur constitue un ensemble cohérent de traitements dont l'exécution est demandée par le composant Client. Chaque Service Utilisateur doit donc être identifié par un code qui fera donc partie de l'interface publique[5] du Client. Pour chaque code de Vue Logique appelée dans un Composant Applicatif, le développeur peut définir son propre traitement dans un point d'insertion reservé à cet effet. Son codage suit les règles standard du Langage Structuré de VisualAge Pacbase.

☞ Voir le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Composant Applicatif*, sous-chapitre *Autres Services*.

2.2.4. Verrouillage logique (applications graphiques)

Le verrouillage logique permet, suite à la demande d'un composant Client, de réserver pour un seul utilisateur la mise à jour d'un ensemble de données du Système d'Information.

☞ Voir le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Ecriture d'un Composant Applicatif*, sous-chapitre *Autres Services*.

---

5 Notion présentée dans la section **2.4.2**.

2.2.5. Appel d'un Composant Applicatif dans un autre

Ce service permet à un Composant Applicatif de déléguer l'exécution d'une demande de service à un autre Composant Applicatif. Ceci permet de spécialiser les Composants Applicatifs et donc d'exploiter au maximum leur réutilisabilité.

☞ Ce type de service est documenté dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs,* chapitre *Composant Applicatif,* sous-chapitre *Autres Services, section Appel d'un Composant Applicatif dans un autre.*

2.2.6. Description d'une requête C/S

De façon générale, un message logique entre un composant Client et un composant Serveur correspond à une requête Client (ou une réponse Serveur).

Une requête C/S correspond donc à un message échangé entre un composant Client et un Composant Applicatif.

Une requête inclut 1 à n demandes de service.

Une demande de service déclenche l'exécution soit d'une série de services standard soit d'une procédure spécifique.

Les différentes combinaisons de services contenus dans une demande sont les suivantes :

- Contrôle (TUI uniquement)
- Lecture (TUI uniquement)
- Contrôle et mise à jour
- Contrôle, mise à jour et sélection
- Contrôle et sélection
- Sélection
- Verrouillage logique
- Service Utilisateur

  ☞ Un message ne peut transporter qu'une et une seule demande de Service Utilisateur.

Un Composant Applicatif peut appeler un autre Composant Applicatif pour exécuter un des services ci-dessus.

2.2.6.1. Buffer Utilisateur

Un message peut également transporter des informations contextuelles liées à l'exécution du service demandé. La définition de la structure de ces informations et la gestion de leur contenu est à la charge du développeur. Elles doivent être définies dans une structure appelée *buffer utilisateur*.

Cette structure de données est passée dans chaque message échangé entre tous les Composants Applicatifs du Dialogue et les composants Client.

☞ Pour plus de détails, voir le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

2.2.6.2. Buffer Serveur

Il est parfois nécessaire et suffisant de pouvoir partager des données exclusivement entre les Composants Applicatifs, qu'ils soient appelés par le même Gestionnaire de Services ou appelés entre eux par le mécanisme d'appel de Composant Applicatif par un autre Composant Applicatif.

Par exemple :

- *données calculées (identifiants logiques)*
- *données communes comme les niveaux d'autorisation liés à un profil utilisateur, liste de profils utilisateurs, données liées à la sécurité*
- *indicateurs divers (indicateurs d'erreurs plus précis que ceux générés, indicateur de mise à jour réelle, compteur de mouvements, …)*
- *liste des instances de nœud racine à passer à tous les serveurs traitant les instances dépendantes*
- *etc.*

Avec le Buffer Serveur, vous pouvez définir une zone de mémoire partagée entre les Composants Applicatifs qui participent à l'exécution d'une requête.

Pour plus de détails, voir le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

2.2.7. Composant Applicatif d'Initialisation/Terminaison

Un *Composant Applicatif d'Initialisation/Terminaison* (Composant **I/T**) permet d'implémenter des traitements spécifiques avant et après l'exécution d'une requête liée à un Dossier.

Cette fonctionnalité n'est donc disponible que pour les applications graphiques impliquant un développement de Dossier.

Un Composant **I/T** est appelé par le Gestionnaire de Services avant le premier appel de Composant Applicatif concerné par la requête à traiter et après le dernier Composant Applicatif exécuté.

Le Composant **I/T** est disponible indifféremment pour un traitement d'initialisation ou de terminaison.

Pour un traitement d'initialisation, les seules données qui peuvent être traitées en entrée sont celles envoyées par le composant Client via le Buffer Utilisateur.

Un Composant **I/T** peut répondre à plusieurs besoins applicatifs :

- Les traitements spécifiques sont associés à plusieurs Composants Applicatifs gérés par le Dossier,
  Par exemple, l'application doit modifier des données dans un nœud suite à la mise à jour de données dans un autre nœud. C'est le cas lors de la création de lignes de commande supplémentaires sur une commande existante; le montant total de la commande doit être modifié (traitement de terminaison).
- Les traitements spécifiques sont intégrés directement au niveau du Composant **I/T**, par algorithme sur les données gérées via le Buffer Serveur ou Utilisateur.
  Par exemple, l'application doit contrôler la validité globale des données – au niveau du Dossier – en regard des contrôles cumulés effectués par tous les Composants Applicatifs du Dossier (traitement de terminaison).

- Les traitements spécifiques sont implémentés via l'appel d'un Composant Applicatif – externe au Dossier – gérant une Vue Logique, également externe au Dossier.
  Vous pouvez ainsi demander l'acquisition de mémoire externe (traitement d'initialisation) et demander la désallocation de cette mémoire (traitement de terminaison). De même, le Composant ***I/T*** *vous permet de gérer les contrôles d'habilitations sur le Dossier (traitement d'initialisation).*

Un Composant **I/T** permet d'exécuter les services associés à tous les Composants Applicatifs d'un Dossier.

L'implémentation d'un Composant **I/T** est documentée en détail dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*.

2.3. Représentation applicative des données (applications graphiques)

2.3.1. Navigation automatique dans les données

2.3.1.1. Contextes fonctionnel et technique

- **Contexte fonctionnel**

L'activité d'une entreprise est généralement modélisée par un ensemble de concepts d'information sur lesquels on applique des transformations.

Dans VisualAge Pacbase, un concept d'information est représenté par une Vue Logique gérée par un Composant Applicatif. Une Vue Logique constitue un agrégat homogène de données élémentaires.

Les transformations se répartissent entre les différents processus fonctionnels de l'entreprise.

Chaque transformation appelée acte de gestion est composée d'un ensemble de traitements pouvant faire transiter le système d'information à travers plusieurs états stables.

Plusieurs actes de gestion différents peuvent modifier le même ensemble cohérent de concepts d'information.

Il est donc intéressant dans ce cas de pouvoir gérer en standard les relations fonctionnelles entre différents concepts d'information faisant partie d'un même système d'information, en regroupant ces concepts au sein d'une enveloppe fédératrice.

Si les transformations possibles n'affectent qu'un seul concept d'information, il n'est pas nécessaire (bien que cela soit possible) de passer par une enveloppe fédératrice puisqu'il n'y a aucune interaction à gérer. Nous parlerons dans ce cas de développement de type mono-vue. Pour plus de détails, voir le paragraphe 2.3.2.
Il est bien sûr possible de gérer au sein d'une même application les deux cas de figure.

- **Contexte technique**

Pour absorber en temps réel la charge de traitements induite par les actes de gestion soumis par de nombreux utilisateurs finals, les constructeurs informatiques ont développé le concept de moniteur transactionnel. Le rôle de ces systèmes est de prendre en compte un nombre maximum de demandes de traitement dans un minimum de temps.

Dans la plupart des cas, les ressources transactionnelles consommées par différents utilisateurs sont identiques et le seul moyen d'obtenir un flux transactionnel performant est de monopoliser les ressources nécessaires à un traitement le moins longtemps possible.

Les moniteurs transactionnels proposent une unité standard de partage correspondant au temps consommé entre la réception d'un message, son traitement et l'émission de la réponse.

Les contraintes inhérentes à ce principe de partage imposent que les informations nécessaires à la transition vers un nouvel état stable d'un ensemble de concepts d'information soient transmises en une seule fois au moniteur transactionnel.

2.3.1.2. Développement via un Dossier

Dans la logique des architectures client/serveur performantes, des concepts d'information homogènes doivent être définis afin de pouvoir les manipuler facilement à travers leurs différentes représentations graphiques, des actes de gestion doivent être mis en oeuvre afin de modifier les instances de différents concepts d'informations liés entre eux et envoyer au serveur les demandes de transition d'état stable en une fois. Dans VisualAge Pacbase, une telle démarche est possible lorsqu'elle est modélisée à travers la notion de **Dossier**.

Du point de vue méthodologique, il est à considérer dans la plupart des cas qu'un Dossier est déductible d'une entité de type MLD (Modèle Logique de Données). La transition consiste à associer à une occurrence de Dossier un ensemble de Segments liés par un réseau de relations dans une occurrence de MLD et d'associer à chaque Segment un serveur (**Composant Applicatif**) qui implémente les actions permises sur ce Segment. Par conséquent, le nombre de Segments-racine définis dans un MLD est égal au nombre d'occurrences de Dossier déductibles de ce MLD.

Dans ce contexte, ce système permet d'acquérir les informations nécessaires et suffisantes – le Dossier – pour qu'un acte de gestion puisse être traité de manière autonome sur le composant Client. Lorsque l'acte de gestion est terminé, le système renvoie l'image finale du Dossier sur le composant Serveur.

Au-delà d'une conciliation des contraintes de description fonctionnelles et du souci de performance, ce mécanisme de type « download/upload » favorise les performances globales du système en minimisant le nombre de sollicitations du serveur, donc en augmentant le taux de disponibilité du moniteur transactionnel.

Une application de type dossier vous permet donc de bénéficier des apports suivants :

- Le référentiel VisualAge Pacbase représente et gère l'intégrité fonctionnelle d'un ensemble de Vues Logiques pour chaque acte de gestion.

- La sélection et la mise à jour d'un ensemble d'instances associées à plusieurs Vues Logiques se fait automatiquement en un seul échange.

2.3.1.3. Représentation dans le Métamodèle

2.3.1.3.1. L'entité Dossier

L'entité **Dossier** permet de décrire un ensemble d'agrégats de données élémentaires, c'est-à-dire de Vues Logiques, et les relations fonctionnelles qui les unissent pour constituer un concept d'information complexe doté de services d'accès et de traitements. La dépendance entre ces différentes Vues Logiques et l'intégrité des données sont gérées en standard à travers le Dossier.

Par exemple, si dans une société, une commande n'a de sens que lorsqu'elle est liée à un client, la dépendance entre ces deux concepts d'information est définie dans un Dossier. Chacun des concepts constitue un ensemble homogène de données qui permet sa manipulation à travers ses différentes représentations graphiques.

Les relations fonctionnelles définissent le comportement des instances contenues dans deux agrégats de données liés. Elles se répartissent en deux types :

- Les relations hiérarchiques pour lesquelles une instance de l'agrégat fils dépend d'une et une seule instance de l'agrégat père.
- Les relations de référencement pour lesquelles une instance de l'agrégat référençant référence soit aucune, soit une et une seule instance de l'agrégat référencé.

Dans un Dossier, chaque agrégat est appelé un noeud. Les noeuds se répartissent en trois types :

- Le noeud *racine*, unique dans un Dossier, est père de tous les noeuds dépendants. Cet agrégat ne dépend d'aucun autre agrégat.
- Le noeud *dépendant*, lié par une relation hiérarchique à un noeud racine ou dépendant.
- Le noeud *référence*, lié par une relation de référencement à un noeud racine ou dépendant.

Un noeud de Dossier correspond à une Vue Logique gérée par un Composant Applicatif.

L'implémentation des Dossiers est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Dossier et Vues de Dossier*, sous-chapitre *Dossier*.

La génération d'une occurrence de Dossier produit un **Gestionnaire de Services** capable d'interpréter et de formater tous les services associés au Dossier avant l'appel du Composant Applicatif approprié.

Pour plus de détails sur le Gestionnaire de Services, consultez la section **3.1.4**.

Si le domaine de description du Dossier est suffisamment limité pour être exploitable intégralement dans une application donnée, une génération supplémentaire au niveau du Dossier permet d'obtenir directement l'ensemble des classes permettant de gérer le Dossier dans une application graphique.

2.3.1.3.2. L'entité Vue de Dossier

La fonctionnalité d'une application ne couvre pas nécessairement toute l'étendue d'une définition de Dossier. Chaque acte de gestion doit donc pouvoir travailler sur une vue partielle d'un Dossier qui garantit l'intégrité de la transformation de ses instances.

Cette vue partielle est représentée par l'entité **Vue de Dossier**.

- Une Vue de Dossier représente tout ou partie du Dossier et doit nécessairement contenir la racine du Dossier.

  La Station de Travail VisualAge Pacbase contrôle automatiquement la cohérence de la description d'une Vue de Dossier par rapport à celle du Dossier associé.
- Un Dossier peut être représenté par plusieurs Vues de Dossier. Chacune d'elles correspond à une représentation cohérente de certains des noeuds du Dossier, déduite des informations nécessaires à la mise en oeuvre d'un acte de gestion.
- Une application peut utiliser plusieurs Vues de Dossier issues de Dossiers différents.

Pour plus de détails sur la façon de spécifier une Vue de Dossier, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Dossier et Vues de Dossier*, sous-chapitre *Vues de Dossier*.

La génération d'une Vue de Dossier produit un objet appelé **Proxy Vue de Dossier**, qui permet à une application graphique VisualAge Pacbase de lire, modifier, transformer les instances d'un Dossier.

La Proxy Vue de Dossier est présentée au paragraphe **2.4.3**.

La spécification de Vues de Dossier n'est pas obligatoire. En effet, le domaine de description d'un Dossier peut être suffisamment limité pour être exploitable intégralement dans une application donnée. Dans ce cas, une génération supplémentaire au niveau du Dossier permet d'obtenir directement l'ensemble des classes permettant de gérer le Dossier dans une application graphique.

2.3.2. Cas particulier : Développement mono-vue

Le développement mono-vue est particulièrement intéressant lorsqu'un concept d'information élémentaire – une Vue Logique – peut être traité de façon totalement autonome au niveau du composant Client.

Dans ce cas, la représentation des données est modélisée uniquement à travers la Vue Logique gérée dans un Composant Applicatif, sans recourir à la spécification d'un Dossier et de Vues de Dossier.

La génération de ce Composant Applicatif produit deux sources :

- un **Gestionnaire de Services**
- une **Proxy Vue de Dossier**

Pour plus de détails sur le Gestionnaire de Services, consultez la section **3.1.4**. La Proxy Vue de Dossier [mono-vue] est présentée au paragraphe **2.4.3.1**.

2.4. Les objets Proxy générés pour le Client Graphique

2.4.1. Introduction

Certains objets peuvent être proposés par l'outil de développement Client. C'est le cas d'une fenêtre, d'un bouton-poussoir ou d'une boîte de saisie. En revanche, pour répondre à des besoins tels que la prise en compte d'un protocole de communication ou l'encapsulation de fonctions de gestion, vous devez développer vous-même les objets correspondants.

L'un des apports du Module Pacbench C/S est de prendre en charge, de façon automatique et transparente, une grande partie de ce développement.

Puisque la fonctionnalité *Services Applicatifs* produit des composants Serveur ayant tous le même comportement générique, il est possible de transposer ce comportement générique à travers des classes dont les objets – dénommés **Proxy** – permettent de commander à distance les services des Composants Applicatifs associés.

Dans la station de développement Client, un générateur piloté par interface graphique prend en entrée le fichier résultant de l'extraction de la Vue de Dossier ou du Composant Applicatif spécifié dans VisualAge Pacbase.

Les objets générés sont les suivants :

- une *Proxy Vue de Dossier*
- des *Proxy Elémentaires*

  Les Proxy Elémentaires n'ont pas d'existence autonome. Elles sont nécessairement associées à la Proxy Vue de Dossier.

La génération des objets Proxy est documentée en détail dans le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

Ces objets définissent, dans un environnement donné, un objet délégué d'un objet fonctionnant dans un autre environnement. L'objet de l'environnement distant correspond au Composant Applicatif et à la Vue Logique qu'il gère.

2.4.2. Programmation visuelle et interface publique

De façon générale, le principe de construction d'une application graphique est basé sur l'appel de différents objets (visuels ou non) et sur l'établissement de connexions entre les éléments caractéristiques de ces objets. Ces éléments constituent l'interface publique de ces objets. Les objets Proxy générés, à l'instar des autres objets proposés par l'outil de développement Client, sont dotés d'une interface publique.

Celle-ci est composée :

- d'un ensemble de *propriétés* (*attributs* pour Smalltalk et COM) qui permettent de transmettre des paramètres pour l'exécution d'un service, de représenter les instances de la Vue Logique sous forme de liste ou de fiche,
- d'un ensemble de *méthodes* (*actions* pour Smalltalk et COM) qui correspondent aux différents services que le Composant Applicatif associé à la Proxy peut exécuter,
- d'un ensemble d'*événements* (plus erreurs et exceptions pour Java) qui caractérisent le résultat de l'exécution du service demandé au retour du serveur ou d'un service local.

☞ L'interface publique des composants générés est documentée thématiquement dans le *Manuel de Référence Clients graphiques : Interface publique des composants générés*.

Dans la Station VisualAge, l'objet Proxy correspond à un *bean* ou *part* non-visuel. Vous placez cet objet sur la *Free Form Surface* lors de la construction de votre application cliente.

2.4.2.1. Implémentation des classes génériques

Toute classe variable hérite d'une classe abstraite réservée aux développeurs qui elle même hérite d'une classe abstraite contenant toutes les méthodes communes et invariantes des classes variables.

Ces classes – dites génériques – sont chargées une seule fois à l'installation.

Les classes génériques permettent d'éviter la multiplication des classes à chaque nouvelle génération d'objets Proxy, améliorent la gestion de la mémoire des applications et permettent aux développeurs d'ajouter des comportements génériques au niveau de ces objets.

☞ Cette fonctionnalité n'est pas disponible pour les cibles COM.

2.4.2.2. Paramétrage des noms de classes générées (VisualAge Smalltalk)

Pour permettre d'adapter la codification des classes générées aux normes de nommage choisie par une entreprise, un système de paramétrage des codes des classes est implémenté en amont dans le Référentiel VisualAge Pacbase.

2.4.3. Les objets Proxy

La **Proxy Vue de Dossier** est issue de la génération d'une Vue de Dossier. Une Vue de Dossier étant décrite par des noeuds (couples Composant Applicatif/Vue Logique), la génération de la Vue de Dossier produit un ensemble de classes correspondant à tous les noeuds qui composent la Vue de Dossier.

Les objets de ces classes sont des **Proxy Elémentaires** qui se répartissent en trois types :

- la Proxy *Racine*, correspondant au nœud racine,
- la Proxy *Dépendante*, correspondant au nœud dépendant,
- la Proxy *Référence*, correspondant au nœud référence.

La Proxy Vue de Dossier, une fois intégrée à votre application cliente, permet de manipuler les Proxy Elémentaires qui la composent et gère automatiquement leurs interactions fonctionnelles.

La Proxy Racine est une Proxy Elémentaire qui possède des caractéristiques qui lui sont spécifiques. Elle seule permet d'exécuter des actions de mise à jour serveur et d'assurer la gestion des erreurs.

Chaque Proxy Elémentaire est associée à un seul Composant Applicatif, pour une seule Vue Logique.

L’interface publique d’une Proxy Elémentaire dépend des caractéristiques de la Vue Logique associée au noeud, du Composant Applicatif qui la gère et du type de noeud.

Les Proxy Elémentaires permettent donc d’exécuter les services du Composant Applicatif qui leur est associé : il permet au Client graphique d’émettre des requêtes qui sont des demandes de service à un serveur distant, ces demandes étant automatiquement générées.

Ces services incluent :

- la structuration des données,
- l'encapsulation de la mise en oeuvre des services du Composant Applicatif,
- le stockage temporaire des informations à transmettre au Composant Applicatif,
- la gestion des erreurs,
- la traduction événementielle des statuts de retour du Composant Applicatif,
- la prise en compte de la communication,
- le contrôle local de l'intégrité des données par rapport au contrôle défini dans le Référentiel,
- le contrôle de cohérence des versions entre la partie cliente et la partie serveur,
- pour VisualAge Smalltalk : la gestion de la présentation des données avec la fonction Quick-Form.

2.4.3.1. Cas particulier du développement mono-vue

Dans le cas d'un développement mono-vue, une Proxy Vue de Dossier est également générée, à partir du Composant Applicatif, et elle n'est composée que d'une seule Proxy élémentaire : la Proxy racine.

☞ Pour la fonctionnalité *Pont VisualAge Java* ou *Smalltalk <> VisualAge Pacbase*, cette Proxy Vue de Dossier particulière est prise en compte en tant que Proxy Vue Logique.

2.5. Facilités de développement du Client graphique

☍ Tous les sujets abordés dans ce sous-chapitre sont documentés en détail dans le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques.*

2.5.1. Cache local

Le **cache local** gère *automatiquement* :

- la cohérence et la synchronisation des instances des différents noeuds d'un Dossier :
  - alimentation des identifiants hiérarchiquement supérieurs d'une instance dépendante en fonction de la navigation dans les noeuds du Dossier.
  - alimentation des Rubriques référençantes d'un noeud racine ou dépendant sur une demande de transfert d'une instance d'un noeud référence.
  - contrôle de l'existence des instances hiérarchiquement supérieures à la création d'une instance dépendante.
  - contrôle du respect des cardinalités des liens exploités par un sous-schéma avant d'émettre un service de mise à jour serveur.
  - gestion des différents états associés à l'instance d'un Dossier (lecture, lecture uniquement, modifié, en erreur serveur), avec les règles induites.
  - calcul de l'image (VisualAge Smalltalk) avec et sans mises à jour locales.
  - gestion de l'état de chaque noeud et mémorisation des informations nécessaires pour effectuer des paginations.
- la répercussion des mises à jour locales sur l'attribut de présentation par liste :

  Une alimentation automatique de la liste des instances d'un noeud racine ou dépendant dans un Dossier (représentée par l'attribut `rows` ou `userRows`) sera effectuée dès lors qu'une mise à jour locale de ce noeud est effectuée (stockée dans l'attribut `detail` ou `userDetail`).

  Toute instance créée localement dans un Dossier est placée dans l'attribut `rows` au rang qui respecte le tri ascendant des instances sur leur identifiant.

  Les mêmes règles sont appliquées sur les attributs correspondants associés aux services Utilisateur.

- le calcul des mises à jour utiles à transmettre au composant Serveur :

  La gestion des mises à jour utiles consiste à calculer la mise à jour résultante de plusieurs mises à jour locales effectuées sur la même instance de noeud du Dossier.

  Elle contrôle la création d'instances en double.

2.5.2. Gestion des collections

L'exécution d'une action de sélection ou de lecture déclenche l'envoi d'une collection d'instances dans le cache local.

Cette collection d'instances est gérée dans le cache local selon deux modes :

- en mode automatique (par défaut), la collection courante du cache local est remplacée par la nouvelle collection envoyée suite à une nouvelle exécution d'une action de sélection ou de lecture,
- en mode manuel, la collection courante du cache local est complétée par les instances envoyées suite à l'exécution d'une série d'actions de sélection ou de lecture[6]. Cette collection s'enrichit donc de nouvelles instances après chaque nouvelle exécution, formant ainsi une collection hétérogène. Pour déterminer le début d'une nouvelle collection, vous devez donc éliminer explicitement la collection courante du cache[7].

Vous pouvez choisir le mode de gestion adéquat pour chaque Proxy Elémentaire composant votre Proxy Vue de Dossier.

☞ Le passage d'un mode de gestion à un autre n'induit pas de modification de la collection courante du cache.

Si vous choisissez le mode manuel :

- les nœuds racine et référence d'un Dossier paramétrés pour fonctionner en pagination non-extend fonctionnent en fait en pagination extend,
- les actions de pagination sont synchronisées avec la dernière action de sélection émise par le nœud,
- l'événement de changement d'une collection est émis suite à l'exécution de l'action d'élimination de la collection courante.

2.5.3. Gestionnaire d'échanges

Le **gestionnaire d'échanges** optimise *automatiquement* le volume du flux d'informations échangé entre un composant Client et un composant Serveur.

En fonction de la taille maximum du buffer de communication spécifiée au niveau du Dossier, il transforme un objet requête en un ou plusieurs objets messages véhiculés par le middleware. Inversement, il transforme un ou plusieurs objets messages associés à une réponse du serveur en un objet réponse.

Le calcul du nombre de messages nécessaires au transfert d'une requête ou d'une réponse est automatique.

---

6 Si le Composant Applicatif envoie une instance déjà présente dans la collection courante, cette instance est rafraîchie si l'instance de la collection courante n'a pas été modifiée localement.

7 L'action d'élimination de la collection courante est aussi disponible en mode automatique pour ré-initialiser une liste.

Le protocole utilisé est adapté aux échanges de type transactionnel. Un fichier de travail[8] utilisé par le Gestionnaire de Services[9] permet de mémoriser les informations nécessaires à la constitution d'une requête ou de sa réponse lorsqu'elle utilise plusieurs messages.

2.5.4. Apports dans le Quick Form VisualAge Smalltalk

- Création automatique de *containers* avec une colonne par Rubrique de la Vue Logique sur un *Quick-Form* de l'attribut `rows` d'une proxy. Le titre de la colonne correspond au libellé court de la Rubrique dans VisualAge Pacbase.

- La représentation graphique des Rubriques peut être définie en amont dans le Référentiel VisualAge Pacbase. Dans ce cas, elle est exploitée automatiquement via la commande *Quick-Form*.

- Pour une Rubrique, lorsqu'une représentation graphique a été associée dans le Référentiel VisualAge Pacbase à sa liste de valeurs et qu'ensuite la présentation de cette Rubrique est gérée par l'outil de *Quick-Form*, cette liste des valeurs permises est automatiquement gérée.

2.5.5. Gestionnaire d'erreurs VisualAge Smalltalk

Une classe dédiée à la gestion des erreurs est générée indépendamment de la Proxy Vue de Dossier. Cette classe est dotée d'une action qui, à partir d'un rang dans les messages d'erreur requis, impose au nœud du Dossier de restaurer l'instance de Vue Logique ayant provoqué l'erreur, puis d'émettre l'événement ad-hoc (`errorContextRestored`).

Une erreur est définie par trois caractéristiques :

- Sa clé,
- Sa gravité,
- Son libellé.

[8] Documenté dans la section 3.1.3.

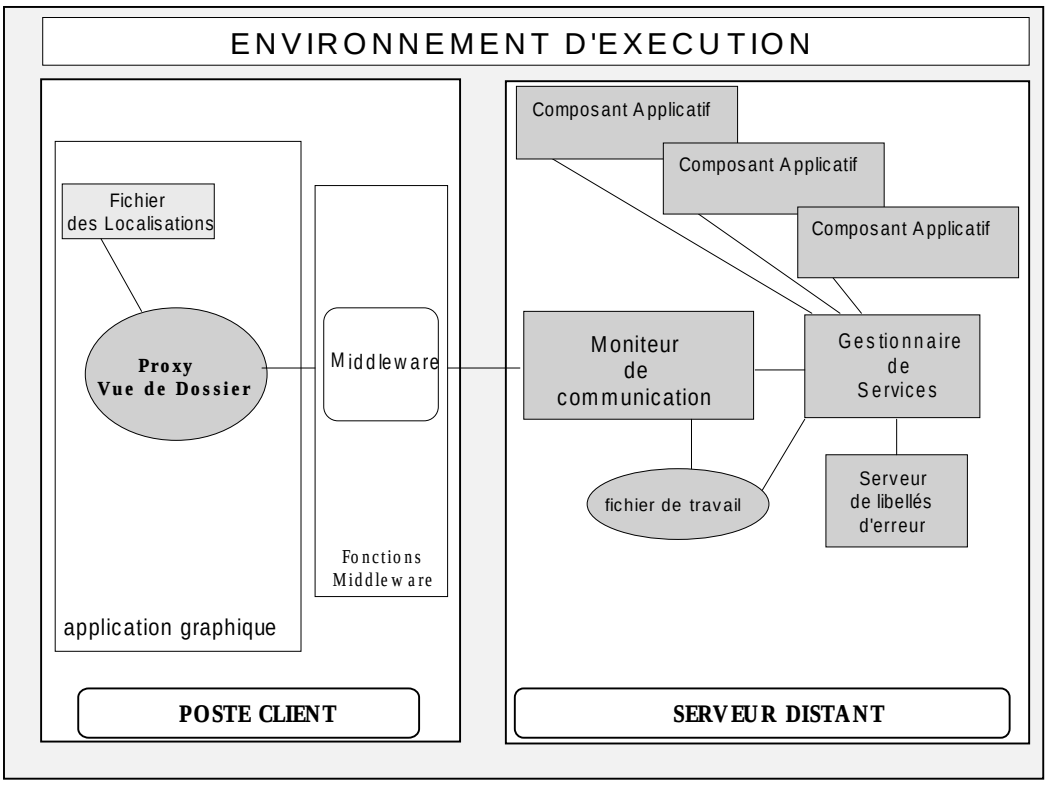
[9] Documenté dans la section 3.1.4.

3. Architecture logicielle des applications générées

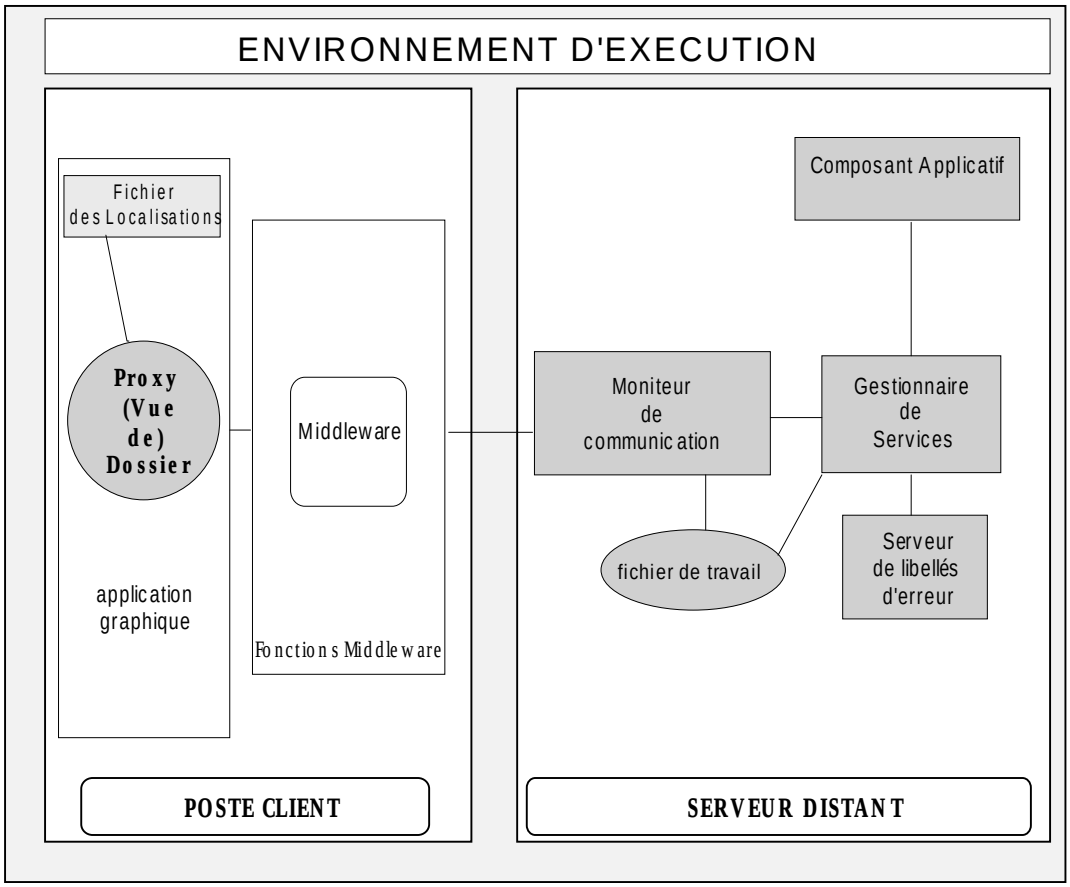
3.1. Applications graphiques

Les différents éléments des schémas suivants sont commentés ci-après.

3.1.1. Schéma général



3.1.2. Cas particulier du développement mono-vue



3.1.3. Rôle du Moniteur de Communication

Le **Moniteur de Communication** gère les fonctions suivantes :

- la définition des fonctions de communication (envoi et réception de messages) liées aux différents types de plates-formes concernées.
  - *Gestion multi-environnements* :
    Une application Pacbench C/S peut s'exécuter sur des environnements différents. Dans ce cas, il y aura autant de Moniteurs de Communications que d'environnements puisqu'un Moniteur est spécifique à un environnement d'exécution (variante de génération et type de communication).
    De plus, plusieurs protocoles de communication peuvent être utilisés pour un même environnement (exemple : CICS ECI et CICS CPIC).
- la vérification de chaque message reçu.
- le pilotage de l'émission/réception de la requête.

  ☞ En fonction de la taille autorisée pour le message physique, plusieurs messages physiques peuvent avoir besoin d'être émis afin d'obtenir le message logique en entier. Un **fichier de travail** doit donc être disponible pour un enregistrement temporaire de la requête en cours.
- à réception complète de la requête, le traitement séquentiel des services qui la composent par appels successifs au Gestionnaire de Services correspondant.
- la gestion transactionnelle (**`COMMIT`**/**`ROLLBACK`**)

  Le Moniteur de Communication utilise les services de **`COMMIT`** et **`ROLLBACK`** d'une base de données ou d'un moniteur transactionnel en fonction de la variante de génération qui lui est affectée.

  La gestion transactionnelle est toujours du type **LUW Serveur** (Logical Unit of Work). Le composant Serveur a donc la charge de l'intégrité de la base de données.

  Une requête est traitée dans sa globalité :
  La partie Serveur exécute – avant retour au client – un **`COMMIT`** ou un **`ROLLBACK`** en fonction du contexte d'erreur (erreur de protocole ou erreur applicative) déterminé à la fin du traitement de la requête. En cas d'erreur, le traitement de la requête est terminé et un message d'erreur est envoyé.

  Ainsi, toutes les ressources monopolisées par le traitement d'une requête sont libérées à l'émission de la réponse.

☞ Vous trouverez les informations pratiques concernant la spécification d'un Moniteur de Communication dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Moniteur de Communication (applications graphiques).*

3.1.4. Rôle du Gestionnaire de Services

Pour chaque requête envoyée via le Moniteur de Communication, le Gestionnaire de Services traite les demandes de service et fournit ensuite la réponse au Moniteur de Communication.

☞ Une requête vise un et un seul Gestionnaire de Services et inclut 1 à n demandes de service.

Le Gestionnaire de Services traite une requête :

- en interprétant son contenu en 1 ou n demandes élémentaires pour les adresser aux Composants Applicatifs correspondants.
  - ☞ Un service n'est traité que par un Composant Applicatif qui peut à son tour appeler un autre Composant Applicatif.
- en transmettant, le cas échéant, au Composant Applicatif appelé, le buffer utilisateur qui lui correspond et remettre à jour son contenu après appel de ce Composant Applicatif si ce buffer utilisateur a été transmis par le client.

Il y aura autant de Gestionnaires de Services que de Dossiers spécifiés pour l'application (ou que de Composants Applicatifs si vous développez en mode mono-vue).

Le Gestionnaire de Services est un composant purement technique, dans lequel il est impossible d'insérer des traitements spécifiques.

3.1.5. Rôle du Serveur de libellés d'erreur

Le Serveur de libellés d'erreur a pour fonction de gérer l'envoi des libellés associés aux erreurs détectées par les Composants Applicatifs. Ces libellés sont stockés dans un fichier dédié et généré.

☍ La gestion de ces erreurs est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. II : Services Applicatifs*, chapitre *Gestion des Erreurs*.

☍ La gestion des erreurs côté client est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

3.1.6. Architecture des communications

3.1.6.1. Middleware et protocoles de communication

Les fonctions middleware utilisées par un objet Proxy permettent de gérer les échanges entre les composants Client et Serveur d'une application en utilisant les principaux protocoles de communication du marché[10].

La mise en œuvre du protocole de communication choisi est entièrement encapsulée. De plus, les fonctions middleware sont indépendantes du protocole. Un changement de protocole de communication (pour un même type de LUW) n'implique donc pas de regénération des composants Client.

Au niveau du composant Client, le choix du protocole de communication consiste à associer une localisation physique (environnement où s'exécute le composant Serveur) avec la localisation logique. Cette association se fait par paramétrage du fichier des localisations (`VAPLOCAT.INI`) ou via un utilitaire graphique fourni pour la cible COM.

Pour une documentation complète, consultez le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques.*

3.1.6.2. Personnalisation du middleware

L'utilisation d'un middleware personnalisé consiste à remplacer le middleware du Module Pacbench C/S par un middleware spécifique.

Cette opération est possible uniquement dans un contexte de développement VisualAge Java ou Smalltalk.

De manière générale, cette opération consiste à utiliser les mécanismes d'héritage pour modifier les actions de base associées aux appels de serveurs distants.

Les classes qui gèrent les appels de serveurs sont génériques. Elles sont chargées une seule fois à l'installation du produit et sont indépendantes des classes variables associées à chaque objet Proxy.

Cette généricité permet de modifier les actions d'appel serveur une seule fois pour que tous les objets Proxy utilisent le nouveau middleware.

Ces modifications sont permanentes, elles ne seront pas remplacées lors de l'importation de nouveaux objets Proxy ou par le rechargement des classes génériques.

La personnalisation du middleware est documentée dans le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

3.1.6.2.1. Buffer utilisateur local (VisualAge Smalltalk)

Ce buffer permet de passer – via la Proxy – des informations du Client VisualAge Smalltalk qui peuvent être utilisées, entre autres, par un middleware personnalisé.

Pour un Composant Applicatif donné, un seul buffer utilisateur local peut être défini; toutes les Proxy gérées par un Composant Applicatif contiennent donc la même instance de buffer utilisateur local associé à la même classe Smalltalk.

---

10 Voir le sous-chapitre **5.1**.

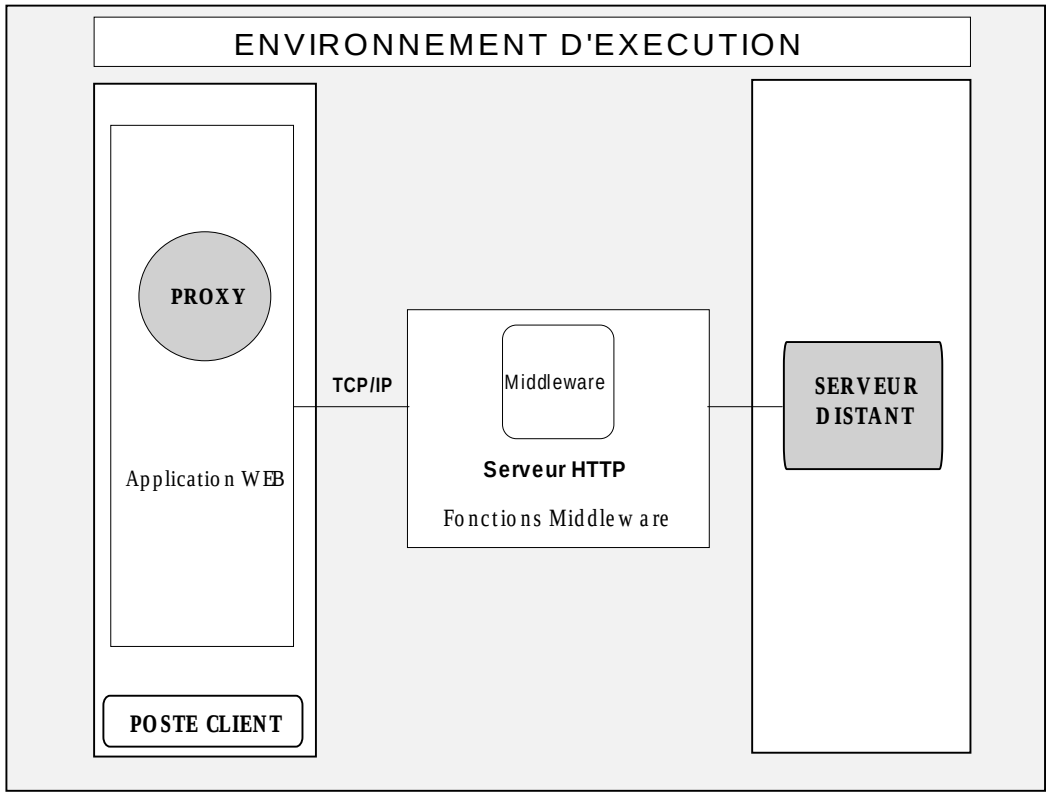
Si les informations de ce buffer sont indépendantes des Proxy, vous pouvez le réutiliser pour tous les Composants Applicatifs d'une application, voire pour toutes les applications du site.

☞ Pour savoir comment exploiter un buffer utilisateur local, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

3.1.6.3. Cas particulier des applications WEB

Dans le cas de développement d'applications WEB, le middleware n'est pas installé sur le poste client.

3.1.6.3.1. Applications Smalltalk



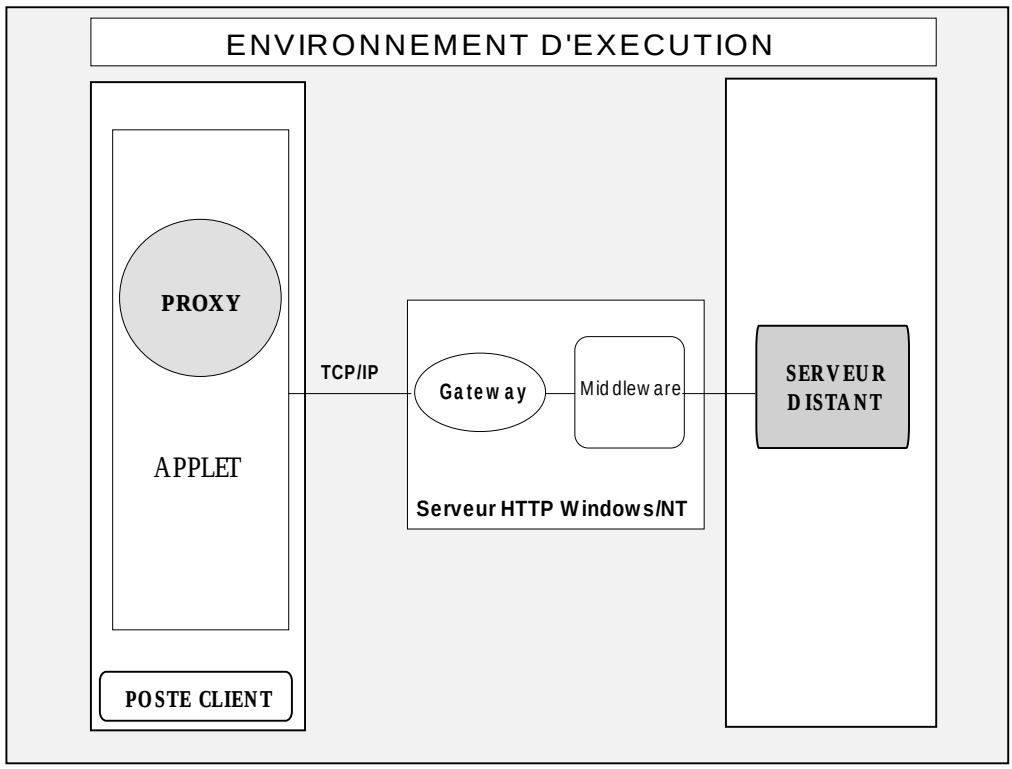
☞ L'architecture des composants Application Web et Serveur dépend du mode de développement choisi (standard, mono-vue), voir les schémas plus haut.

☞ Pour plus de détails, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

3.1.6.3.2. Applets Java (Intranet / Internet)

On doit mettre en place une implémentation faisant appel à une Gateway (**Intranet**) ou à un Relais accompagné d'une Gateway (**Internet**).

- ***Accès au composant Serveur via une Gateway***

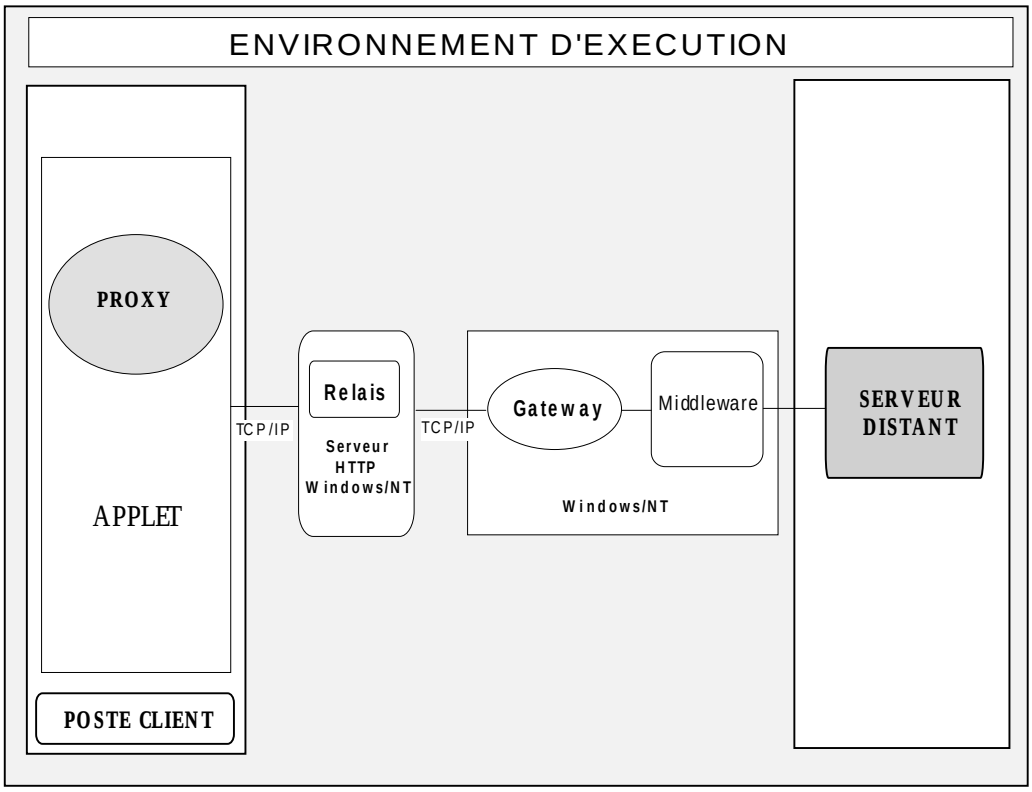


La Gateway est un exécutable C++ fourni pour Windows/NT. Elle est partagée par toutes les applets téléchargées du même serveur HTTP, et doit donc être lancée en permanence.

☞ Le composant Applet reste inchangé par rapport au composant application graphique du schéma général. L'architecture du composant Serveur dépend du mode de développement choisi (standard ou mono-vue, voir les schémas plus haut).

☞ Pour plus de détails, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

- ***Accès au composant Serveur via un relais et une Gateway (pour l'Internet)***



Cette solution – obligatoire pour les applications sur Internet – permet de décharger le serveur HTTP de la tâche d'accès au middleware et de la gestion des serveurs et des contextes de chaque client connecté.

Le relais est un exécutable (**`relay.exe`**) compilé pour Windows/NT et qui permet de relayer les requêtes arrivant à un serveur HTTP vers une autre machine abritant la Gateway et le middleware.

☞ Le composant Applet reste inchangé par rapport au composant application graphique du schéma général. L'architecture du composant Serveur dépend du mode de développement choisi (standard ou mono-vue).

Pour plus de détails, reportez-vous au *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*.

3.2. Applications TUI

Ce sous-chapitre ne fait que présenter les grands principes d'une application TUI. Cependant, vous trouverez une présentation plus spécifique sur le Client TUI dans le *Guide de l'Utilisateur Pacbench C/S, Volume II : Services Applicatifs,* chapitre *Développement Client TUI*.

La documentation complète du développement client TUI se trouve dans le *Manuel de Référence Pacbench C/S : Services Applicatifs & Clients TUI.*

3.2.1. Les Moniteurs

3.2.1.1. Introduction

Il peut être utile de regrouper des informations et des traitements communs (gestion des communications, compactage, trace, `COMMIT/ROLLBACK`, particularités du site) dans des composants techniques que sont les Moniteurs.

Pour certaines plates-formes telles que Microfocus et Tuxedo, il s'agit d'une obligation.

3.2.1.2. Le composant Moniteur Client

Le composant Moniteur Client assure la liaison entre les différents Clients et effectue l'appel soit à un moniteur qui gère la communication entre les Composants Applicatifs de l'application, soit il appelle directement le Composant Applicatif concerné.

Le passage des informations entre le moniteur et les Clients se fait par l'intermédiaire d'une zone de communication qui contient les données suivantes :

- informations technologiques propres aux Clients,
- informations nécessaires à l'appel et au retour du Composant Applicatif, telles que :
  - le nom du service à appeler,
  - le nom du moniteur Serveur (ce nom est défini au niveau du Dialogue comprenant les services),
  - le code de la Vue Logique traitée par le Client,
  - la fonction d'aiguillage qui permet le retour en séquence sur les traitements Client.

3.2.1.3. Le Composant Moniteur Serveur

Il assure l'appel du service demandé et rend la main soit au moniteur Client, soit directement au Client.

Les **principaux traitements générés** dans un moniteur Serveur sont :

- Récupération du message logique
- Appel du service
- Mise en forme du message avant retour au Client

Ces traitements dépendent essentiellement du mode de communication.

3.2.2. Options sans Moniteur

3.2.2.1. Traitements Client

Chaque Client appelle directement soit un moniteur Serveur, soit un Composant Applicatif.

Dans une génération sans moniteur, les traitements d'appel sont codés directement dans chaque composant Client.

Suivant le mode de communication, chaque Client gère l'envoi de ses données au service. Le retour est fait juste après la fonction d'appel et les traitements sont effectués en séquence.

Vous avez toujours la possibilité de rajouter vos propres traitements en vous insérant avant ou après l'appel du service.

3.2.2.2. Traitements du Composant Applicatif

Chaque Composant Applicatif est appelé directement par le Client ou par le moniteur Client.

La structure du service ne dépend pas de la présence ou non d'un moniteur. En revanche, vous devez coder dans chaque Composant Applicatif les traitements propres à la technique de communication ou au système de sécurité.

Les traitements propres à la gestion des bases de données sont codés dans chaque service.

Traitements d'initialisation :

Les traitements de connexion ou d'ouverture des bases sont optionnellement codés en début des traitements du Composant Applicatif.

Gestion des **`COMMIT`** et **`ROLLBACK`** :

Un indicateur permet de conditionner l'exécution de ces ordres. Cette fonction sera exécutée par **`PERFORM`**, son contenu ainsi que son appel seront modifiables par traitements utilisateur codés dans une fonction de type **`*R`**. En standard, elle sera exécutée en fin de traitement de la Vue Logique avant le retour au Client.

4. Environnements de Développement

Le but de ce chapitre est de définir l'environnement technique nécessaire au bon fonctionnement de Pacbench Client/Serveur, tant au niveau matériel qu'au niveau logiciel.

Ce chapitre ne présente pas d'information sur le middleware. En phase de test, il est recommandé d'utiliser le middleware qui sera utilisé lors de l'exécution. Pour plus de détails, consultez le chapitre *Environnements d'Exécution*, sous-chapitres **5.2** et **5.3**.

4.1. Composants distants

Les fonctionnalités *Services Applicatifs* et *Clients TUI* du Module Pacbench C/S fonctionnent sur de nombreuses plates-formes IBM et non-IBM :

- 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

Toutes les informations de configuration sont fournies dans le *Manuel d'Exploitation VisualAge Pacbase 2.5 Vol. I – Environnement et Installation* spécifique à chaque plate-forme.

4.2. Composants locaux

☞ Lorsque des identifications de version de logiciel sont indiquées, la fonctionnalité *Clients Graphiques* n'est censée fonctionner sur des versions plus récentes que si ces versions respectent une stricte compatibilité ascendante.

☞ Le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques* contient les informations nécessaires au paramétrage des logiciels externes.

4.2.1. Station VisualAge Smalltalk

4.2.1.1. Applications standard

4.2.1.1.1. Windows 95 ou NT

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

- Intel Pentium ou processeur compatible
- Mémoire 32 Mo. 48 Mo recommandés si utilisation concurrente de gestionnaire de base de données ou de services de communication.
- Moniteur graphique VGA
- Lecteur CD-ROM
- Carte adaptée au réseau du site
- Espace disque initial : 50 Mo

- Logiciels
  - Windows 95, NT 3.51 ou NT 4.0
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).

4.2.1.1.2. OS/2

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

- Intel Pentium ou processeur compatible
- Mémoire 32 Mo. 48 Mo recommandés si utilisation concurrente de gestionnaire de base de données ou de services de communication.
- Moniteur graphique VGA
- Lecteur CD-ROM
- Carte adaptée au réseau du site
- Espace disque initial : 50 Mo

- Logiciels
  - OS/2 WARP Version 3 ou Version 4
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).

4.2.1.2. Spécificité WEB Smalltalk

Outre les prérequis des applications standard, le développement d'une application WEB nécessite WEB Connection (un feature de VisualAge).

4.2.2. Station VisualAge Java

Les environnements cités ici concernent la station de développement VisualAge Java. Cependant, les objets Proxy et le générateur Pacbench C/S peuvent être utilisés dans tout environnement Java supportant le JDK 1.1.

4.2.2.1. Windows NT et 95

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

  - Intel Pentium ou processeur compatible
  - Mémoire 48 Mo. 64 ou 96 Mo recommandés si utilisation concurrente de gestionnaire de base de données ou de services de communication + 200 Mo libres pour VisualAge Java et Pacbench C/S
  - Moniteur graphique VGA
  - Lecteur CD-ROM
  - Carte adaptée au réseau du site
  - Espace disque initial : 50 Mo

- Logiciels
  - Windows 95 ou NT 4.0 avec Service Pack 3.
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).
  - TCP/IP

4.2.2.2. OS/2

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

  - Intel Pentium ou processeur compatible
  - Mémoire 48 Mo. 64 ou 96 Mo recommandés si utilisation concurrente de gestionnaire de base de données ou de services de communication + 200 Mo libres pour VisualAge Java et Pacbench C/S
  - Moniteur graphique VGA
  - Lecteur CD-ROM
  - Carte adaptée au réseau du site
  - Espace disque initial : 50 Mo

- Logiciels
  - OS/2 WARP Version 4
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).
  - TCP/IP

4.2.3. Environnements COM

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

  - Intel Pentium ou processeur compatible
  - Mémoire 24 Mo. 32 Mo recommandés si utilisation concurrente de gestionnaire de base de données ou de services de communication.
  - Moniteur graphique VGA
  - Lecteur CD-ROM
  - Carte adaptée au réseau du site
  - Espace disque initial : 50 Mo

- Logiciels
  - Windows 95 ou NT 4.0
  - Visual C++ à partir de la version 5
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).

5. Environnements d'Exécution

Le but de ce chapitre est de définir l'environnement technique nécessaire au bon fonctionnement des applications générées, tant au niveau matériel qu'au niveau logiciel.

5.1. Composants distants

Constructeurs	Plates-formes serveur	Type de communication	Bases de données
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 entre Client et UNIX TUXEDO/HOST-Connect et CPI-C /XCP2 entre UNIX et GCOS7	
	GCOS8/TP8 ou DM4-TP	TUXEDO entre Client et UNIX TUXEDO/HOST-Connect et CPI-C /XCP2 entre UNIX et 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	

Constructeurs	Plates-formes serveur	Type de communication	Bases de données
TANDEM	Integrity-IRIX	TUXEDO XATMI TCP-IP Socket	
	Guardian	Non Stop TUXEDO Pathway TCP-IP Socket	
UNISYS	2200 Serie	TCIS	
	A Serie		

5.2. Composants locaux

5.2.1. Poste utilisateur final

5.2.1.1. Windows 95 et NT

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

- Intel Pentium ou processeur compatible
- Mémoire 16 Mo
- Moniteur graphique VGA
- Carte adaptée au réseau du site

- Logiciels
  - Microsoft Windows 95 ou NT 4.0
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).

5.2.1.2. OS/2

- Matériel

Sous réserve d'évolution, la configuration matérielle minimale *recommandée* est :

- Intel Pentium ou processeur compatible
- Mémoire 16 Mo
- Moniteur graphique VGA
- Carte adaptée au réseau du site

- Logiciels
  - OS/2 Version 3 ou Version 4
  - Le logiciel de communication adapté au middleware choisi (voir le *Guide de l'Utilisateur Pacbench C/S, Vol. III : Clients Graphiques*).

5.2.2. Environnements COM

Les applications basées sur l'utilisation d'une Proxy COM sont exécutables sur tous les outils capables d'accéder à des serveurs COM.

5.3. Composants intermédiaires pour le Web

Dans le cadre de l'exécution des applications Web, le middleware n’est pas installé sur le poste client.

Pour plus de détails, voir le paragraphe 3.1.6.3, page 32.

5.3.1. Applications Smalltalk

Outre les prérequis des applications standard, l'exécution d'une application Web Smalltalk nécessite un serveur HTTP du marché supportant CGI[11] 1.1, en version Windows (3.1, 95 ou NT), OS/2 ou AIX. Ce serveur doit héberger le programme "**`CGI Link`**" **`abtcgil`** livré avec le feature Web Connection de Smalltalk.

Le poste de l'utilisateur final doit être équipé d'un navigateur (Netscape Communicator 4.0x ou Internet Explorer 4.0).

5.3.2. Applications Java

Outre les prérequis des applications standard, l'exécution d'une application Web Java nécessite un serveur HTTP du marché. Celui-ci doit s'exécuter sous un environnement Windows/NT ou AIX 4.1.

Le poste de l'utilisateur final doit être équipé d'un navigateur supportant la version 1.1 du JDK (Netscape Communicator 4.0x ou Internet Explorer 4.0).

[11] Common Gateway Interface

6. Architecture physique selon les différents types d'environnement

6.1. TUXEDO

6.1.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs et les composants Client (TUI) TUXEDO sont les suivants :

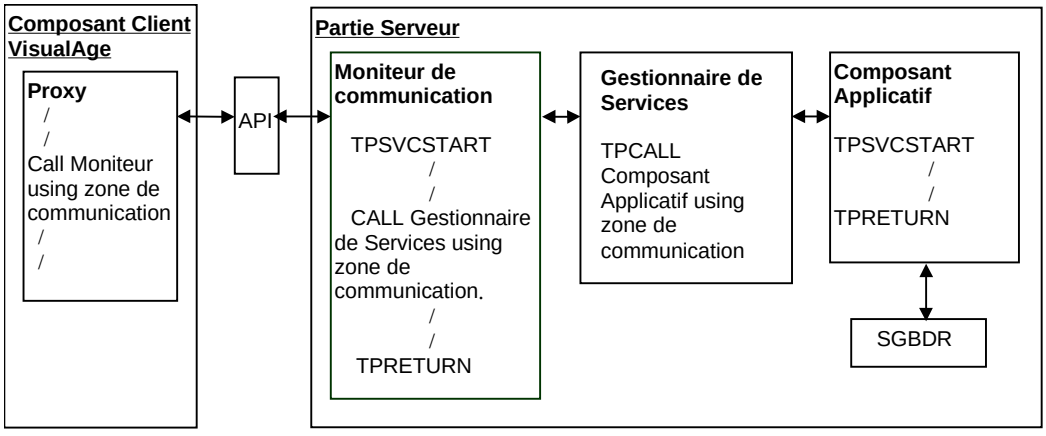
- les fichiers indexés de Microfocus,
- Oracle V6 et V7,
- Sybase V10.0.1 et suivantes (ANSI-mode).

6.1.2. Application graphique

6.1.2.1. Fonctions du Moniteur de Communication

- TUXEDO-XA

Le message identifiant le service à exécuter par un Composant Applicatif est reçu par un Moniteur de Communication qui initialise le service et appelle le Gestionnaire de Services qui appelle à son tour le Composant Applicatif approprié. Une fois le service exécuté, le Moniteur de Communication termine le service et envoie la réponse à la Proxy.



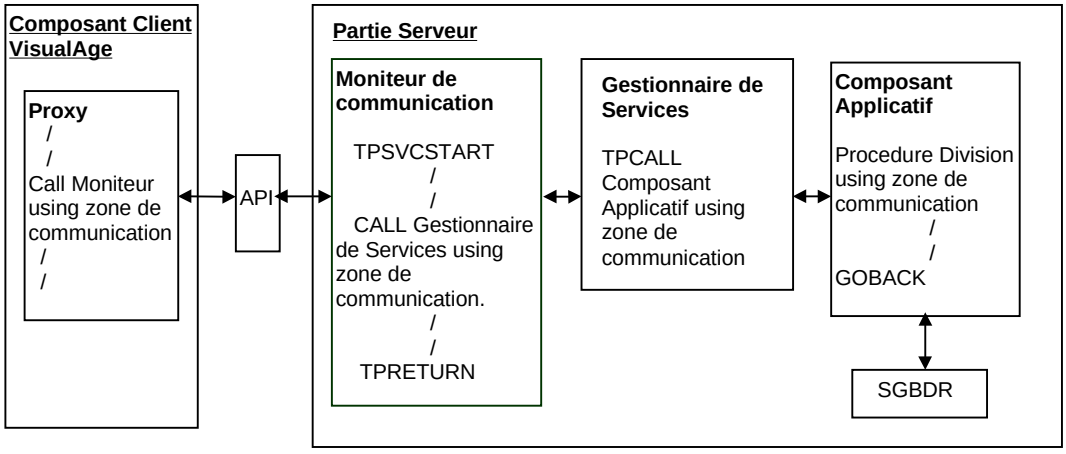
- TUXEDO XA (à partir de TUXEDO version 6.2.)

A partir de la version 6.2. de Tuxedo, vous pouvez utiliser le mécanisme des fonctions. Une fonction est le point d'entrée d'un programme écrit en C ou en COBOL. Les Composants Applicatifs ne sont donc pas activés directement.

L'utilisation de fonctions revêt deux avantages :

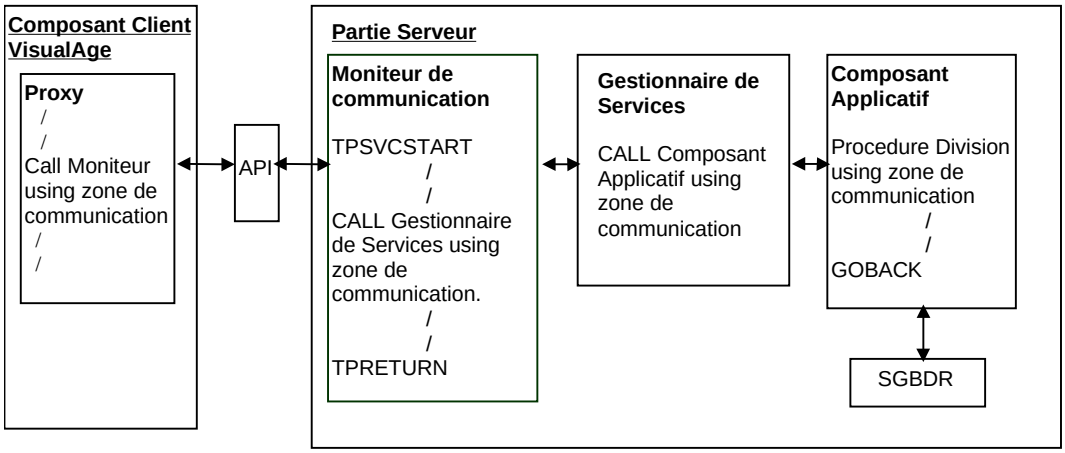
- il n'y a qu'une structure de programme pour les Composants Applicatifs, quels que soient le type de client l'appelant et le type de base de données. Le support de buffer utilisé peut être unifié (**CARRRAY**). La structure de ces Composants Applicatifs est celle d'un sous-programme; il n'y a donc plus d'ordre TUXEDO de réception (**TPSVCSTART**) ou d'émission (**TPRETURN**) de message.
- les Composants Applicatifs peuvent être appelés indifféremment par CALL (nécessaire pour les Composants Applicatifs utilisant des bases de données non XA car dans ce mode, il n'y a pas de notion de transaction globale permettant de prendre en compte les mises à jour effectuées par plusieurs services différents) ou par TPCALL (dans ce cas, la fonction est systématiquement activée et le Composant Applicatif est appelé par CALL).

Dans cette architecture, le Gestionnaire de Services et le Moniteur de Communication se trouvent sur le même serveur TUXEDO et constituent un seul service (le Gestionnaire de Services est un sous-programme du Moniteur de Communication). En revanche, les Composants Applicatifs doivent se trouver sur un ou plusieurs serveurs différents.



- TUXEDO NON XA

Dans cette architecture, il ne peut y avoir qu'un seul service TUXEDO. En effet, comme une transaction globale n'est pas possible, les Composants Applicatifs ne peuvent pas être des services distincts. La seule possibilité pour valider les mises à jour traitées par plusieurs Composants Applicatifs est de les regrouper au sein du même serveur et même service TUXEDO que le Moniteur de Communication et le Gestionnaire de Services.



6.1.2.2. Gestion des COMMIT/ROLLBACK

La responsabilité de l'intégrité de la base de données est à la charge de la partie Serveur. L'ordre de **COMMIT** ou de **ROLLBACK** de la base de données est exécuté par le Moniteur de Communication en fonction du contenu de la rubrique **TECH-COMMIT** (**C** ou **R**). Cette rubrique appartient à la zone de communication qui transite entre les différents composants de la partie serveur. Elle est initialisée par le Composant Applicatif en fonction du déroulement du service demandé.

Après l'exécution d'un **ROLLBACK**, le Moniteur de Communication retourne un message contenant les raisons de l'échec de l'exécution du service à la Proxy.

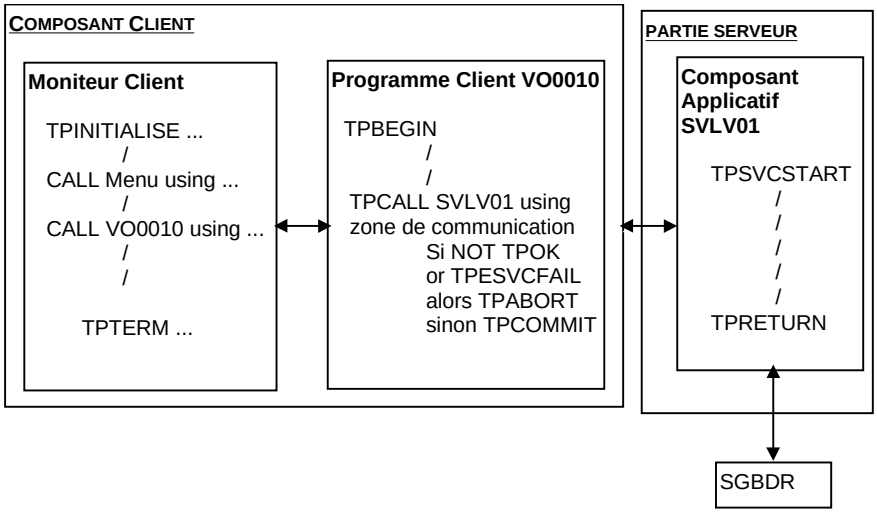
6.1.3. Application TUI

Pour des explications sur les zones de **`Working Storage Section`** et la structure intégrale du composant Client, référez-vous au *Manuel de Référence Dialogue Microfocus* [REF : DD OPC 000 023 F].

6.1.3.1. Architecture

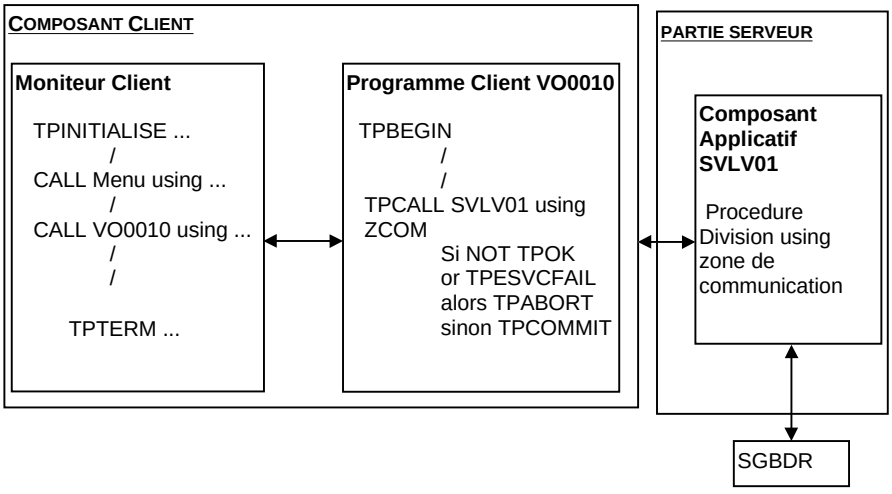
- TUXEDO-XA

Dans l'environnement TUXEDO, le composant Client est constitué d'un moniteur Client et de modules Clients de type menu, affichage liste, fiche, etc. La notion de Moniteur de Communication n'existe pas et le Composant Applicatif est directement appelé par le moniteur client. C'est lui qui initialise et termine le service TUXEDO.

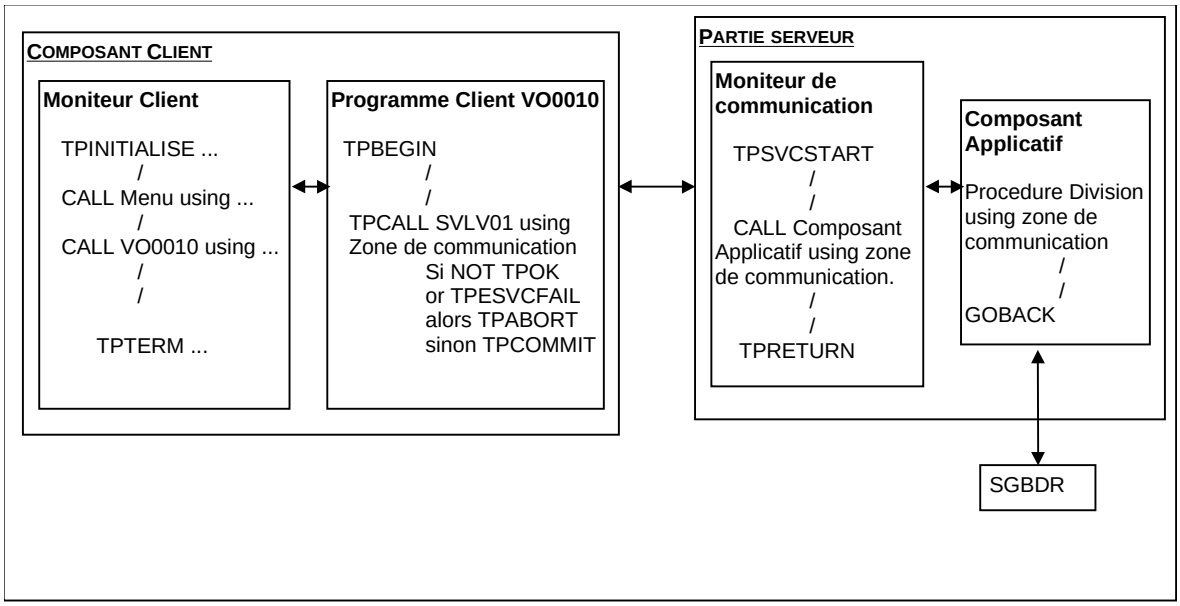


- TUXEDO XA (à partir de TUXEDO version 6.2.)

Dans cette architecture, le Moniteur de Communication et le Composant Applicatif appartiennent obligatoirement à des serveurs TUXEDO différents (il n'est pas possible d'appeler un service au sein du même serveur).



- TUXEDO NON XA



6.1.3.2. Gestion des `COMMIT/ROLLBACK`

En cas d'erreur bloquante (exception sur un accès base ou incompatibilité de structure entre composant Client et composant Serveur), le composant Applicatif positionne la variable TUXEDO **`TPESVCFAIL`** qui est interprétée par le moniteur client pour appliquer un **`ROLLBACK`** sur la base de données. Dans les autres cas, le moniteur client applique un **`COMMIT`** sur la base de données.

6.2. CICS

6.2.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs et les composants Client (TUI) CICS sont les suivants :

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

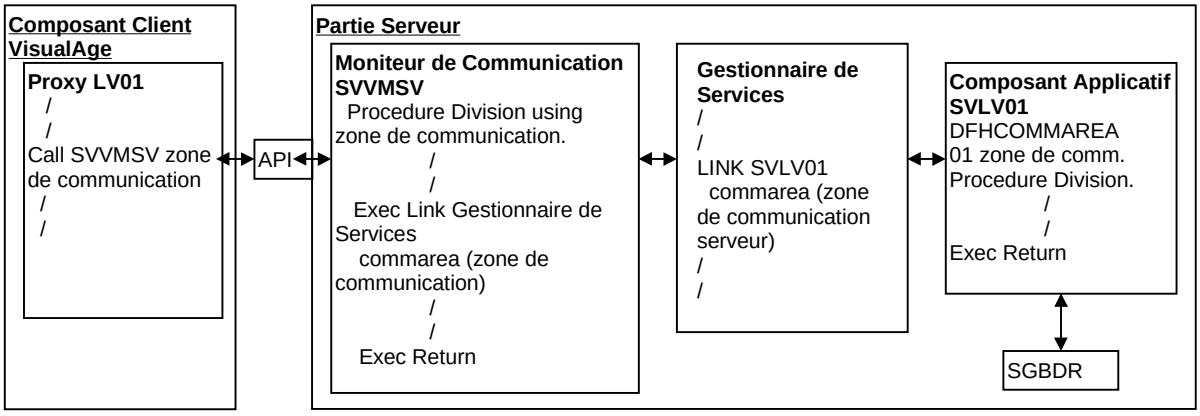
6.2.2. Application graphique

6.2.2.1. CICS/ECI

6.2.2.1.1. Fonctions du Moniteur de Communication CICS/ECI

En mode ECI l'appel d'un Serveur depuis le composant client est considéré comme un appel de programme de type **`LINK`**.

Dans ce cas, le rôle du Moniteur de Communication est de recevoir le message et de se débrancher sur le Gestionnaire de Services qui appelle à son tour le Composant Applicatif à exécuter.



6.2.2.1.2. Gestion des `COMMIT/ROLLBACK`

En mode Non Extend, chaque erreur détectée par le Composant Applicatif positionne la zone **`TECH-COMMIT`** à **`R`**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **`ROLLBACK`** sur la base de données.

Dans les autres cas, un **`COMMIT`** implicite est effectué sur la base de données lorsque le moniteur redonne le contrôle à CICS sur l'instruction **`RETURN`**.

Dans tous les cas, la zone de communication est renvoyée à la Proxy.

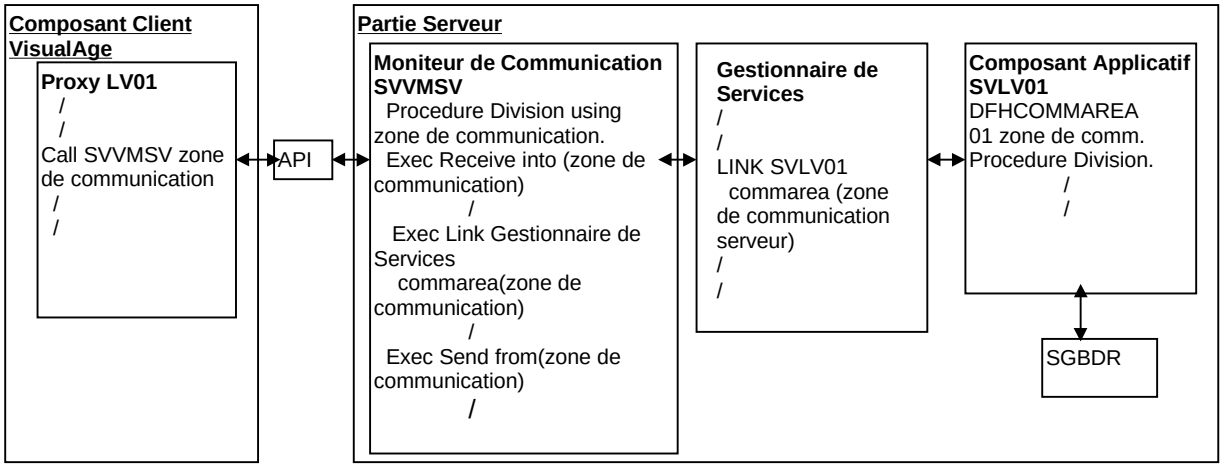
6.2.2.2. CICS/CPI-C

6.2.2.2.1. Fonctions du Moniteur de Communication CICS/CPI-C

Avec le protocole de communication **LU62 CPI-C**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

La réception et l'émission des messages effectuées dans le Moniteur de Communication sont basées sur le mode **CPI-C** implicite qui consiste à utiliser les verbes de communication standard **RECEIVE** et **SEND**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **LINK** (ou **CALL**).



6.2.2.2.2. Gestion des **COMMIT/ROLLBACK**

Avec le protocole de communication **LU62 CPI-C**, la gestion des **COMMIT**/**ROLLBACK** fonctionne selon les mêmes principes qu'avec le protocole **CICS ECI**.

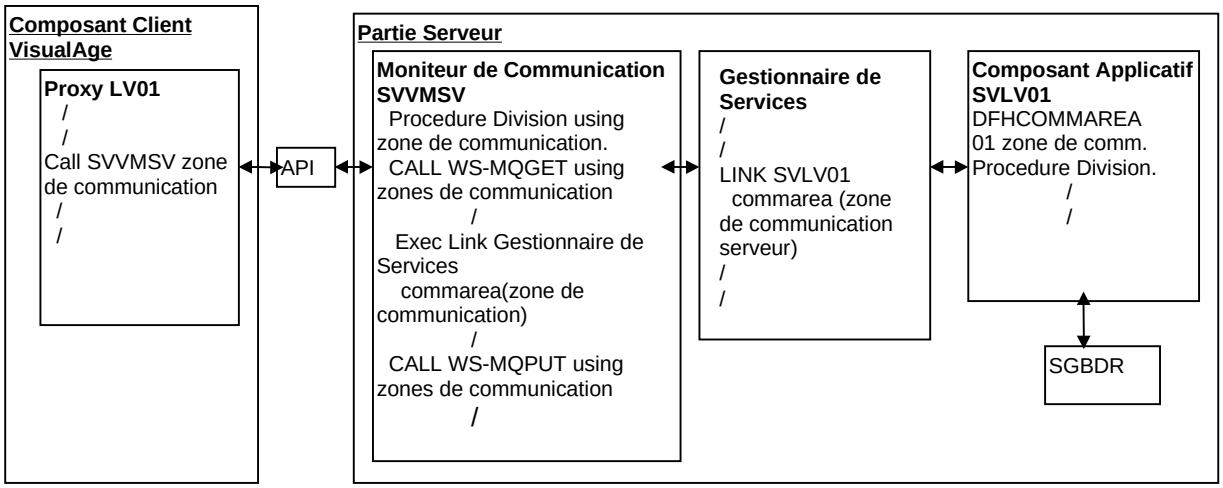
6.2.2.3. CICS/MQSERIES

6.2.2.3.1. Fonctions du Moniteur de Communication CICS/MQSERIES

Avec le protocole de communication **MQSERIES**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

Le Moniteur de Communication reçoit et émet les messages en appelant des routines **MQSERIES**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **LINK** (ou **CALL**).



6.2.2.3.2. Gestion des **COMMIT/ROLLBACK**

Avec le protocole de communication **MQSERIES**, la gestion des **COMMIT**/**ROLLBACK** fonctionne selon les mêmes principes qu'avec le protocole **CICS ECI**.

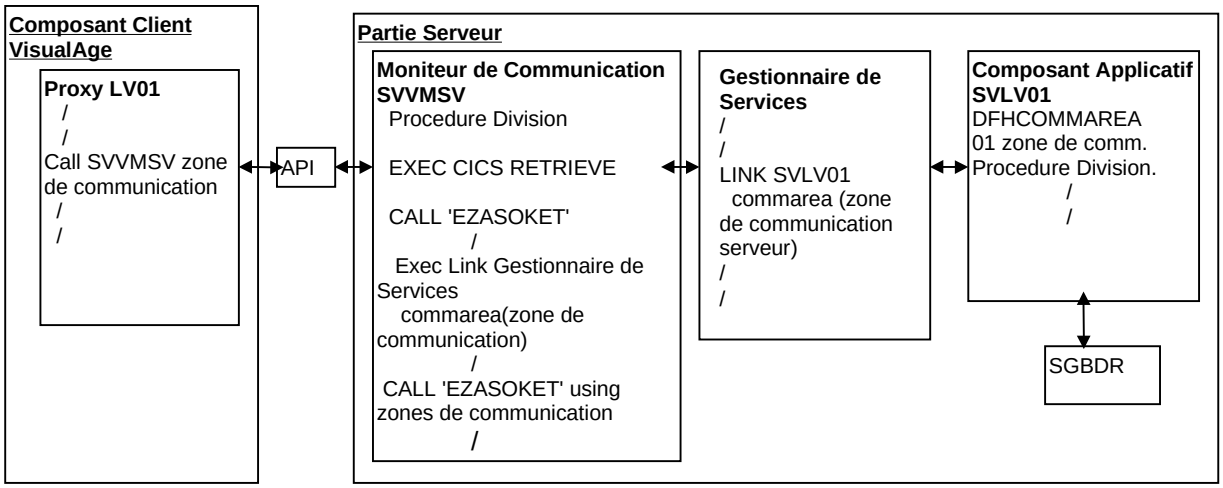
6.2.2.4. CICS/SOCKET

6.2.2.4.1. Fonctions du Moniteur de Communication CICS/ SOCKET

Avec le protocole de communication **SOCKET**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

Le Moniteur de Communication reçoit et émet les messages en appelant des routines **SOCKET**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **LINK** (ou **CALL**).



6.2.2.4.2. Gestion des **COMMIT/ROLLBACK**

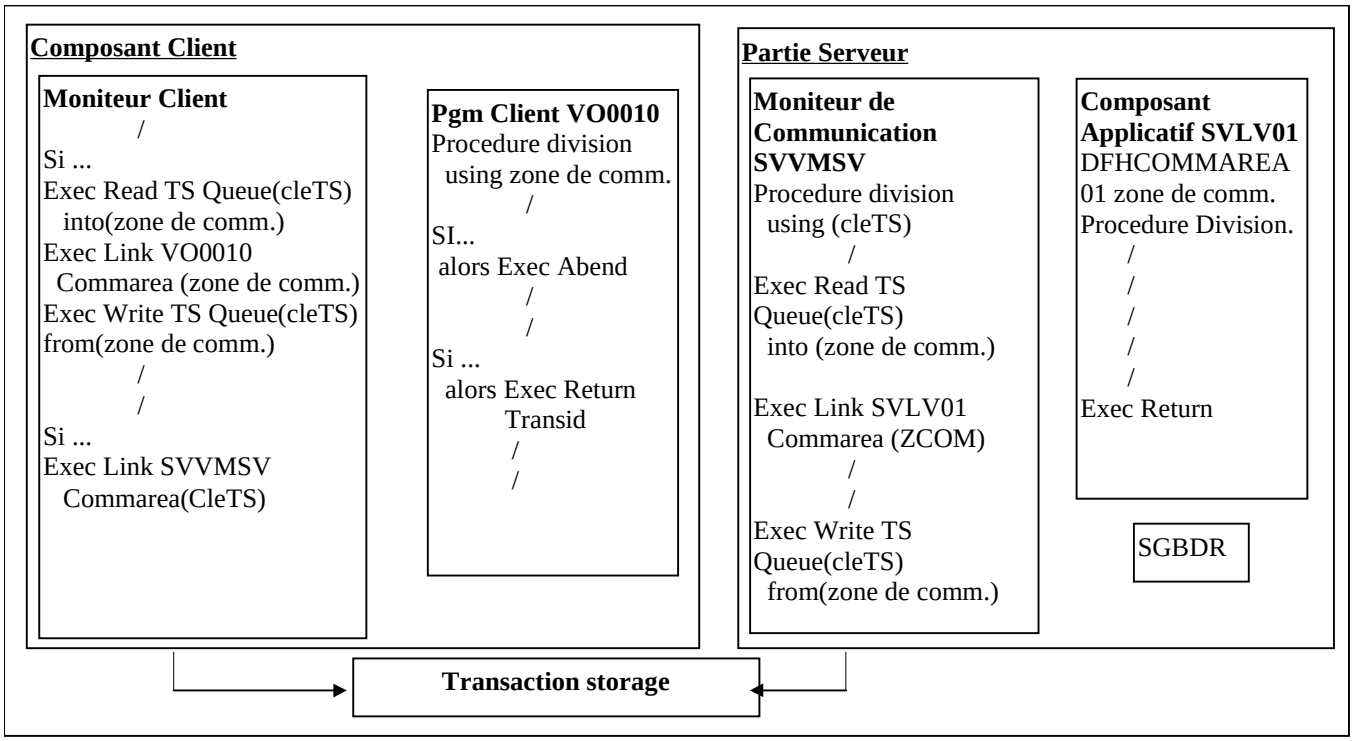
Avec le protocole de communication **SOCKET**, la gestion des **COMMIT**/**ROLLBACK** fonctionne selon les mêmes principes qu'avec le protocole **CICS ECI**.

6.2.3. Application TUI

Pour des explications sur les zones de **`Working Storage Section`** et la structure intégrale du Composant Client, référez-vous au *Manuel de Référence Dialogue CICS* [REF : DD OCI 000 151 F].

6.2.3.1. Architecture

Les différents composants d'une application TUI sont exécutés au sein d'un même CICS. La partie cliente est composée d'un moniteur et d'un ensemble de programmes gérant l'affichage des écrans passifs. La partie Serveur est composée d'un Moniteur de Communication et de Composants Applicatifs. Les appels de programmes sont de type **`LINK`** (ou **`CALL`**).



6.2.3.2. Gestion des `COMMIT/ROLLBACK`

L'application fonctionne en **`COMMIT`** implicite. Il n'y a pas de niveau de synchronisation particulier. La détection d'une erreur bloquante par le Composant Applicatif conditionne l'execution d'un ordre **`EXEC CICS ABEND`** par le Serveur (invalide l'exécution du **`COMMIT`**). Sans détection d'erreur, le Client termine par un ordre **`RETURN TRANSID`** (valide l'exécution du **`COMMIT`**).

6.3. GCOS7

6.3.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs GCOS7 sont les suivants :

- les fichiers indexés UFAS,
- Oracle V6 et V7.

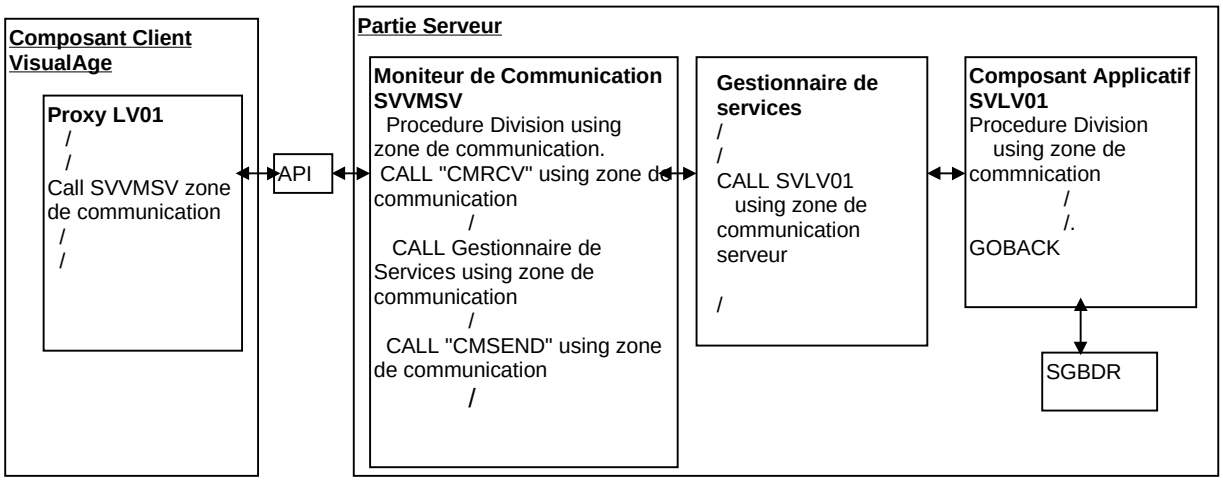
6.3.2. Application graphique

6.3.2.1. Fonctions du Moniteur de Communication XCP2/CPI-C

Avec le protocole de communication **LU62 CPI-C**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

La réception et l'émission des messages effectuées dans le Moniteur de Communication sont basées sur le mode **CPI-C** qui consiste à utiliser les modules "**CMRCV**" pour réceptionner le message et "**CMSEND**" pour l'envoyer.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **CALL**.



6.3.2.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

Les **COMMIT** et **ROLLBACK** sont exécutés par des verbes SQL qui dépendent du type de SGBD appelé dans les Composants Applicatifs.

6.4. GCOS8

6.4.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs GCOS8 sont les suivants :

- les fichiers indexés UFAS,
- Bases de données SQL INTEREL.

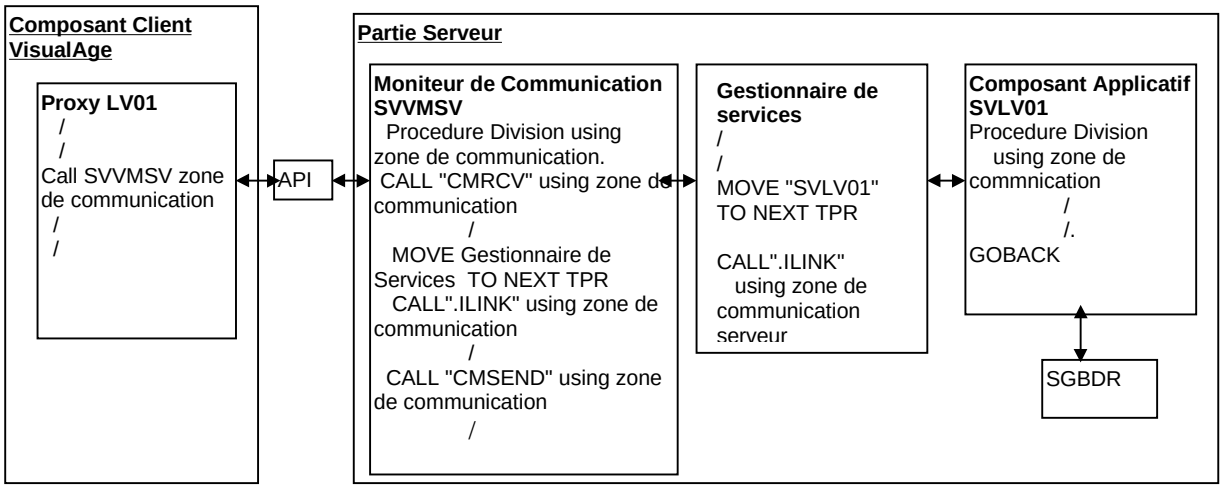
6.4.2. Application graphique

6.4.2.1. Fonctions du Moniteur de Communication XCP2/CPI-C

Avec le protocole de communication **LU62 CPI-C**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

La réception et l'émission des messages effectuées dans le Moniteur de Communication sont basées sur le mode **CPI-C** qui consiste à utiliser les modules "**CMRCV**" pour réceptionner le message et "**CMSEND**" pour l'envoyer.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **CALL".ILINK"**.



6.4.2.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

Les **COMMIT** et **ROLLBACK** sont exécutés par des verbes SQL qui dépendent du type de SGBD appelé dans les Composants Applicatifs.

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

6.5.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs et les composants Client (TUI) Microfocus sont les suivants :

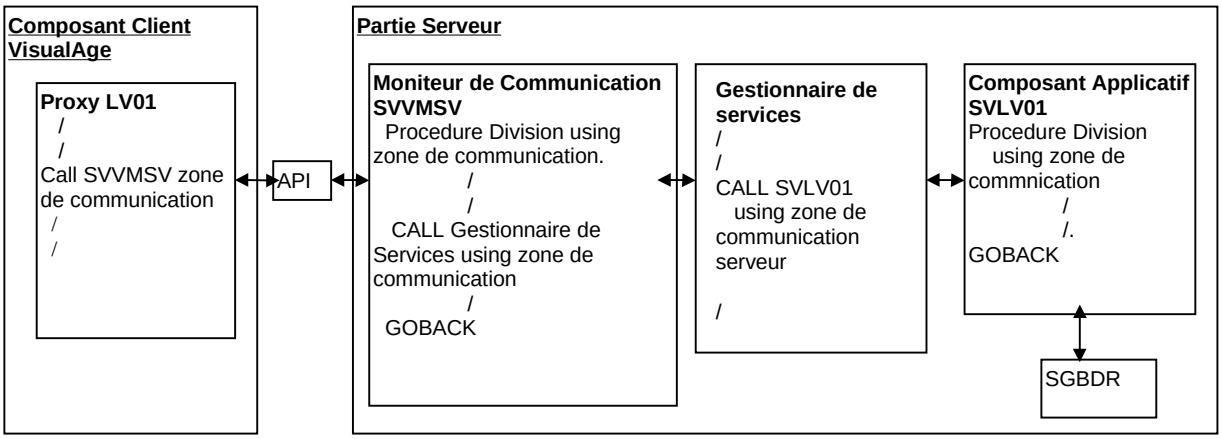
- Les fichiers indexés de Microfocus,
- Oracle V6 et V7,
- Sybase V10.0.1 et suivantes (ANSI-mode),
- DB2/2,
- SQL Informix sauf pour OS/2

6.5.2. Application graphique

6.5.2.1. Microfocus /Socket

6.5.2.1.1. Fonctions du Moniteur de Communication Microfocus /Socket

Pour le protocole **TCP-IP Socket**, le Moniteur de Communication n'intègre pas de verbe de communication particulier. Le principe d'échange est basé sur l'écriture d'un message dans une socket sur la machine Serveur. Un Serveur associé à cette socket est alors exécuté. Celui-ci décode le message et appelle le Moniteur de Communication, via le Gestionnaire de Services, en lui transmettant le message par référence.



6.5.2.1.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

Les **COMMIT** et **ROLLBACK** sont exécutés par des verbes SQL qui dépendent du type de SGBD appelé dans les Composants Applicatifs.

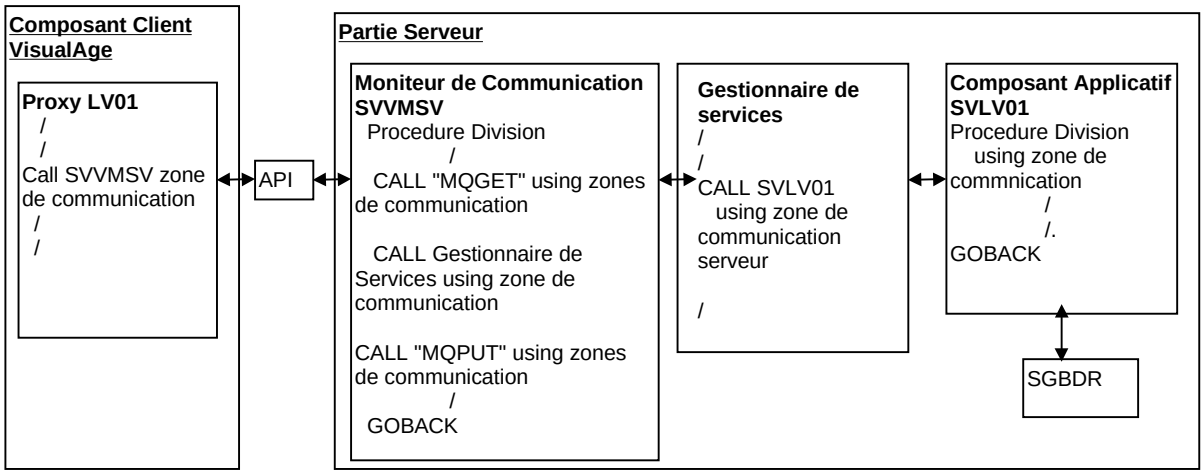
6.5.2.2. Microfocus / MQSERIES

6.5.2.2.1. Fonctions du Moniteur de Communication Microfocus / MQSERIES

Avec le protocole de communication **MQSERIES**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

Le Moniteur de Communication reçoit et émet les messages en appelant des routines **MQSERIES**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **LINK** (ou **CALL**).



6.5.2.2.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

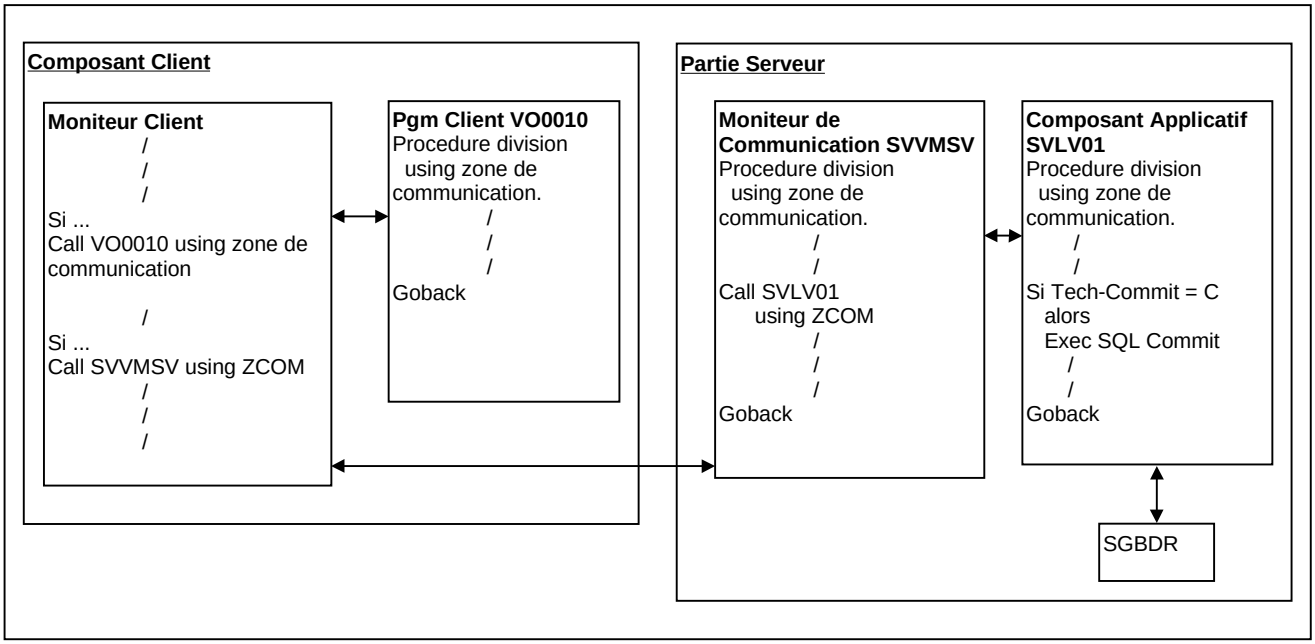
Les **COMMIT** et **ROLLBACK** sont exécutés par des verbes SQL qui dépendent du type de SGBD appelé dans les Composants Applicatifs.

6.5.3. Application TUI

Pour des explications sur les zones de **`Working Storage Section`** et la structure intégrale du Composant Client, référez-vous au *Manuel de Référence Dialogue Microfocus* [REF : DD OPC 000 023 F].

6.5.3.1. Architecture

Les différents composants d'une application TUI sont exécutés au sein du même poste de travail. La partie cliente est composée d'un moniteur et d'un ensemble de programmes gérant l'affichage des écrans passifs. La partie Serveur est composée d'un Moniteur de Communication et de Composants Applicatifs. Les appels de programmes est de type **`CALL`** dynamique.



6.5.3.2. Gestion des `COMMIT/ROLLBACK`

La variable **`TECH-COMMIT`** qui conditionne l'exécution du **`COMMIT`** SQL est positionnée à **`C`** par le générateur pour cet environnement particulier (ce qui suppose l'exécution du **`COMMIT`** par défaut dans le Composant Applicatif). Si une erreur bloquante est détectée par le Composant Applicatif (erreur d'accès à la base, contenu de la zone technique, ...) l'exécution du **`COMMIT`** n'est pas activée.

6.6. IMS

6.6.1. Gestion des données

Le Gestionnaire de données supporté de manière automatique dans les Composants Applicatifs et les composants Client (TUI) IMS est SQL DB2.

Pour les Composants Applicatifs uniquement, les bases de données DL/1 sont supportées aussi de manière automatique.

6.6.2. Application graphique

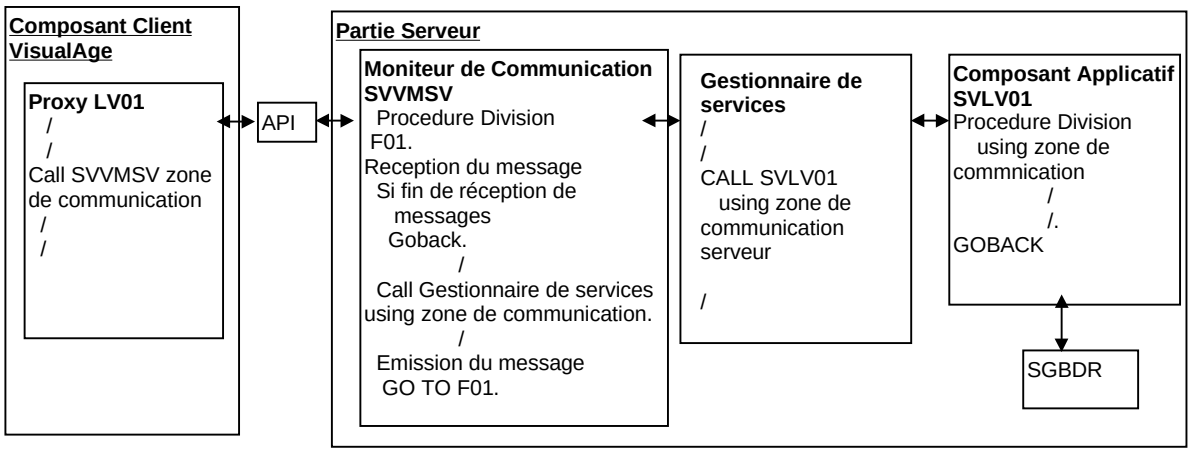
6.6.2.1. IMS CPI-C

6.6.2.1.1. Fonctions du Moniteur de Communication en IMS CPI-C

Avec le protocole de communication **LU62 CPI-C**, l'échange d'informations entre composants client et Serveur est basé sur le mode message.

La réception et l'émission des messages effectuées dans le Moniteur de Communication sont basées sur le mode **CPI-C** implicite qui consiste à utiliser les verbes de communication standard **GETUNIT**, **INSERT**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **CALL**.



6.6.2.1.2. Gestion des **COMMIT/ROLLBACK**

Avec le protocole de communication **LU62 CPI-C**, la gestion des **COMMIT**/**ROLLBACK** fonctionne selon les mêmes principes qu'avec le protocole **CICS ECI**.

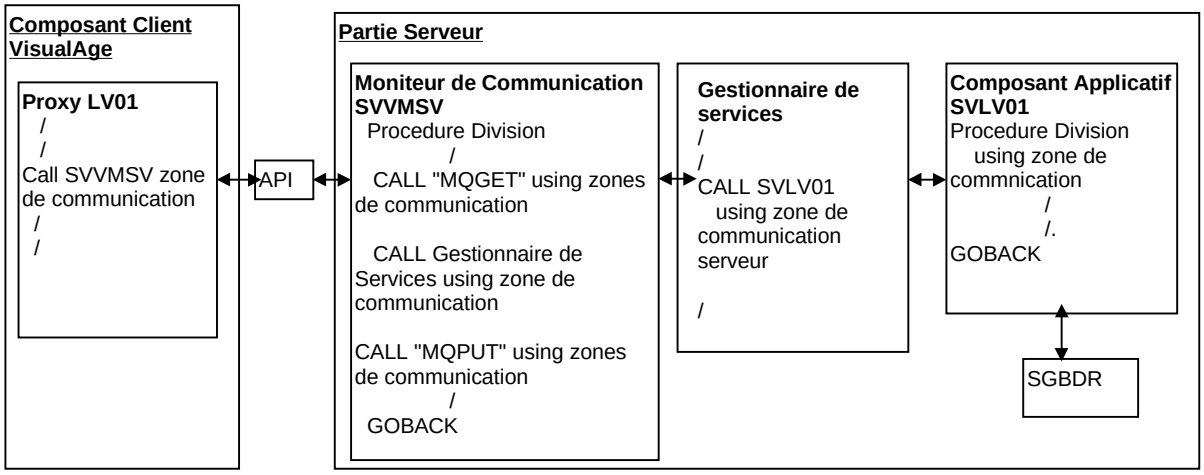
6.6.2.2. IMS / MQSERIES

6.6.2.2.1. Fonctions du Moniteur de Communication IMS / MQSERIES

Avec le protocole de communication **MQSERIES**, l'échange d'informations entre composants Client et Serveur est basé sur le mode message.

Le Moniteur de Communication reçoit et émet les messages en appelant des routines **MQSERIES**.

A la réception du message, la zone de communication est initialisée en **LINKAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **CALL**.



6.6.2.2.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

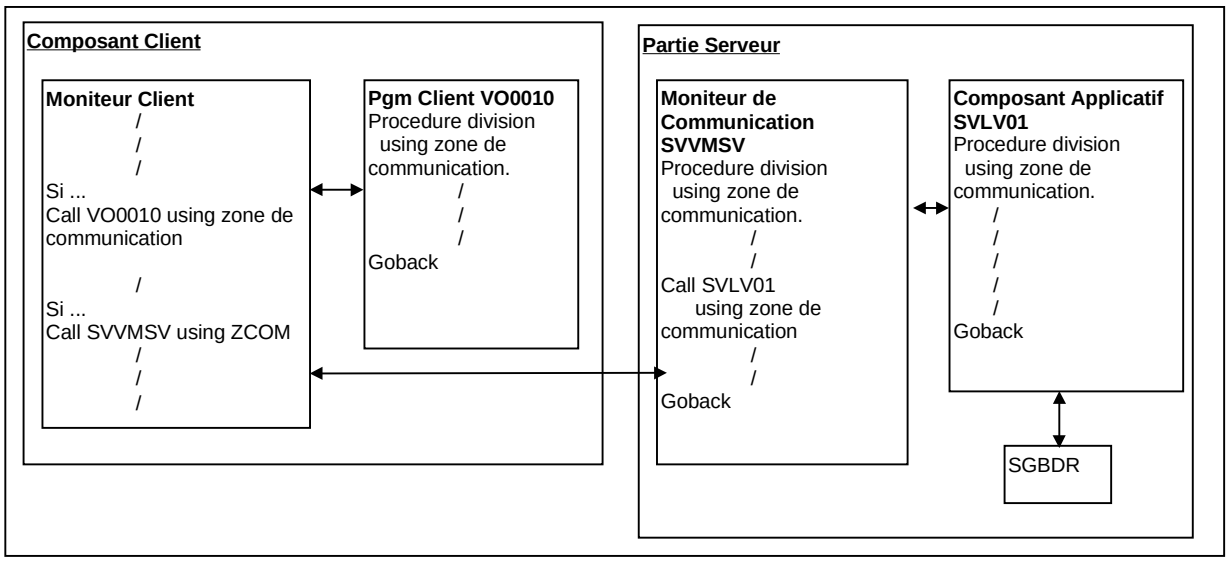
Les **COMMIT** et **ROLLBACK** sont exécutés par des ordres IMS.

6.6.3. Application TUI

Pour des explications sur les zones de **`Working Storage Section`** et la structure intégrale du Composant Client, référez-vous au *Manuel de Référence Dialogue IMS* [REF : DD OIM 000 022 F].

6.6.3.1. Architecture

Les différents composants d'une application TUI sont exécutés au sein d'un même IMS. La partie cliente est composée d'un moniteur et d'un ensemble de programmes gérant l'affichage des écrans passifs. La partie Serveur est composée d'un Moniteur de Communication et de Composants Applicatifs. Les appels de programmes sont de type **`CALL`**.



6.6.3.2. Gestion des `COMMIT/ROLLBACK`

L'application fonctionne en **`COMMIT`** implicite.

En cas de détection d'une erreur bloquante par le Composant Applicatif, aucun traitement de type **`ROLLBACK`** n'est prévu en standard.

6.7. UNISYS 2200

6.7.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs Unisys 2000 sont les suivants :

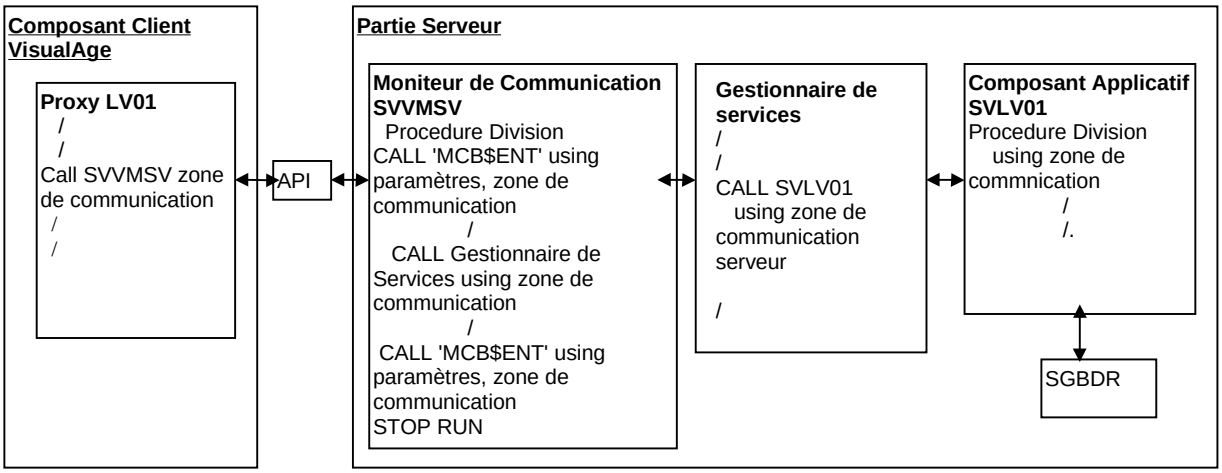
- RDMS/SQL
- Fichiers indexés SFS

6.7.2. Application graphique

6.7.2.1. Fonctions du Moniteur de Communication Unisys 2200 / TCIS

La réception et l'émission des messages effectuées dans le Moniteur de Communications sont basées sur le mode **TCIS** qui consiste à utiliser le module **MCB$ENT** pour les fonctions de réception **(P-TRINIT)** et d'envoi **(P-SENDD)**.

A la réception du message, la zone de communication est initialisée en **WORKING-STORAGE SECTION** puis passée au Composant Applicatif concerné, via le Gestionnaire de Services, par un mécanisme d'appel de type **CALL.**



6.7.2.2. Gestion des COMMIT/ROLLBACK

Chaque erreur détectée par le Composant Applicatif positionne la zone **TECH-COMMIT** à **R**. Cette valeur est ensuite interprétée par le Moniteur de Communication pour appliquer un **ROLLBACK** sur la base de données.

Les **COMMIT** et **ROLLBACK** sont exécutés par des verbes SQL qui dépendent du type de SGBD appelé dans les Composants Applicatifs.

6.8. TANDEM

6.8.1. Gestion des données

Les Gestionnaires de données supportés de manière automatique dans les Composants Applicatifs Tandem sont les suivants :

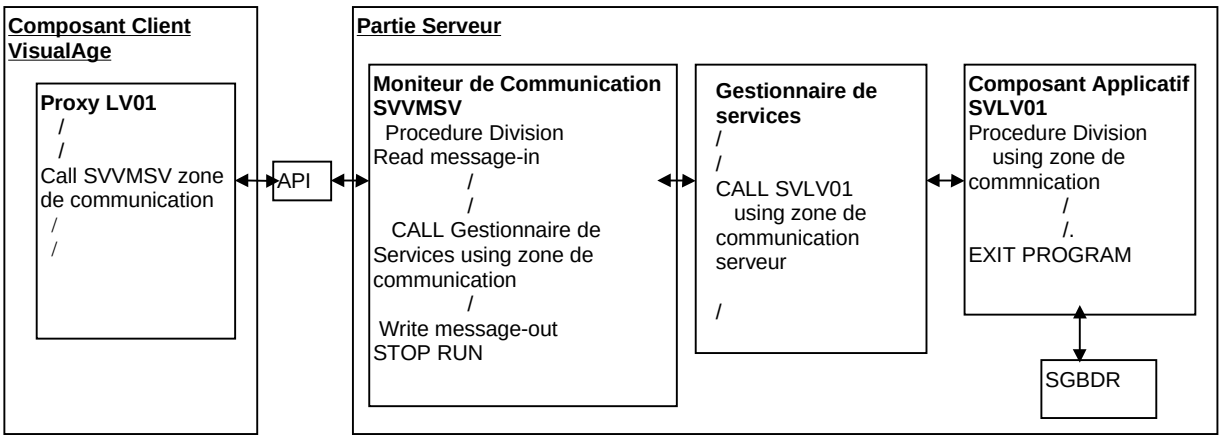
- Non Stop SQL
- Fichiers indexés

6.8.2. Application graphique

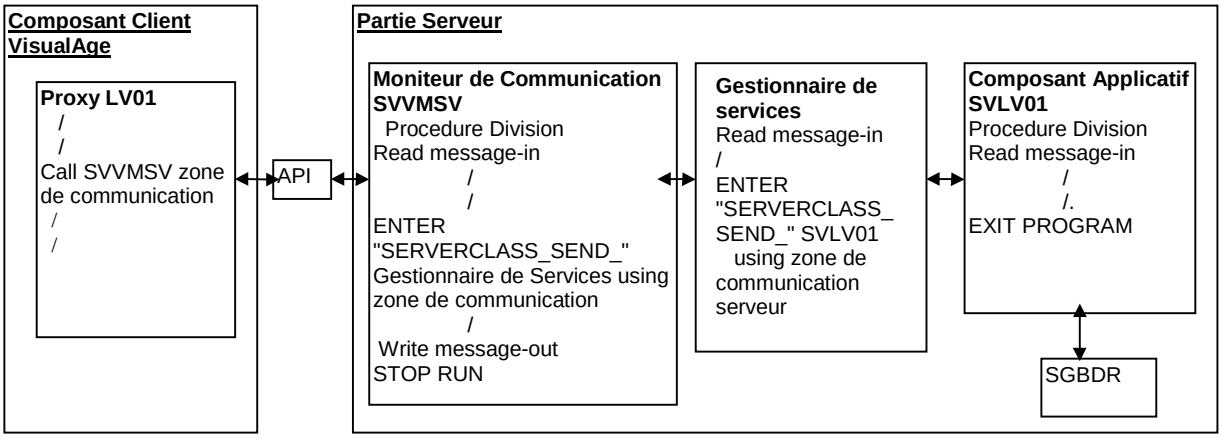
6.8.2.1. TANDEM PATHWAY

6.8.2.1.1.Fonctions du Moniteur de Communication Tandem Pathway / Socket

Pour le protocole **`TCP-IP Socket`**, le Moniteur de Communication n'intègre pas de verbe de communication particulier. Le principe d'échange est basé sur l'écriture d'un message sur un fichier "Message". Un Serveur associé à cette socket est alors exécuté. Celui-ci décode le message et appelle le Moniteur de Communication en lui transmettant le message dans le fichier "Message".



Si vous avez indiqué l'option **CALLTYPE=PATHSEND**, le schéma est le suivant :



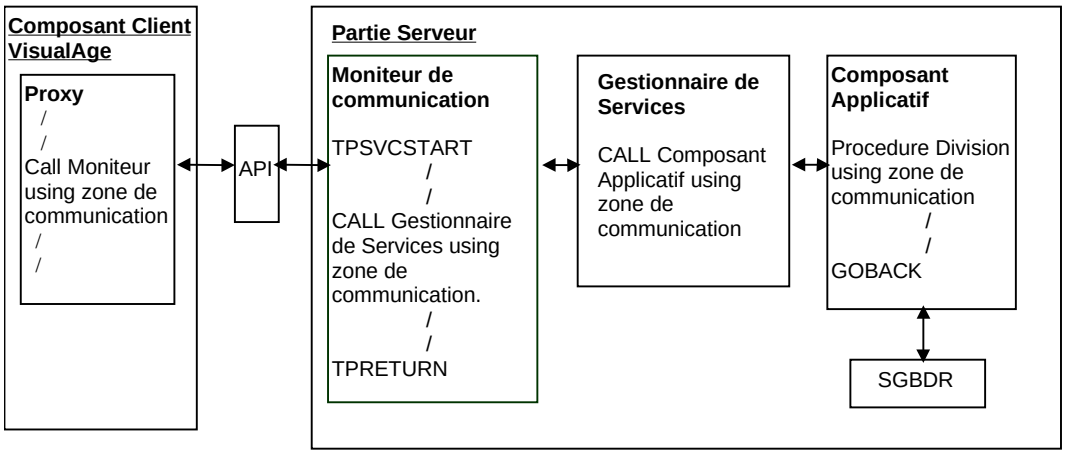
6.8.2.1.2.Gestion des COMMIT/ROLLBACK

La gestion des **COMMIT/ROLLBACK** est assurée par le Moniteur de Communication.

6.8.2.2. TANDEM NonStop TUXEDO NON XA

6.8.2.2.1.Fonctions du Moniteur de Communication

Dans cette architecture, il ne peut y avoir qu'un seul service TUXEDO. En effet, comme une transaction globale n'est pas possible, les Composants Applicatifs ne peuvent pas être des services distincts. La seule possibilité pour valider les mises à jour traitées par plusieurs Composants Applicatifs est de les regrouper au sein du même serveur et même service TUXEDO que le Moniteur de Communication et le Gestionnaire de Services.



6.8.2.2.2.Gestion des COMMIT/ROLLBACK

La responsabilité de l'intégrité de la base de données est à la charge de la partie Serveur. L'ordre de **COMMIT** ou de **ROLLBACK** de la base de données est exécuté par le Moniteur de Communication en fonction du contenu de la rubrique **TECH-COMMIT** (**C** ou **R**). Cette rubrique appartient à la zone de communication qui transite entre les différents composants de la partie serveur. Elle est initialisée par le Composant Applicatif en fonction du déroulement du service demandé.

Après l'exécution d'un **ROLLBACK**, le Moniteur de Communication retourne un message contenant les raisons de l'échec de l'exécution du service à la Proxy.